VISUAL MOTION ESTIMATION AND TRACKING OF RIGID BODIES BY PHYSICAL SIMULATION

by

DAMIEN JADE DUFF

A thesis submitted to
The University of Birmingham
for the degree of
DOCTOR OF PHILOSOPHY

School of Computer Science
College of Engineering and Physical Sciences
The University of Birmingham
August 2010

Abstract

This thesis applies knowledge of the physical dynamics of objects to estimating object motion from vision when estimation from vision alone fails. It differentiates itself from existing physics-based vision by building in robustness to situations where existing visual estimation tends to fail: fast motion, blur, glare, distractors, and partial or full occlusion.

A real-time physics simulator is incorporated into a stochastic framework by adding several different models of how noise is injected into the dynamics. Several different algorithms are proposed and experimentally validated on two problems: motion estimation and object tracking.

The performance of *visual motion estimation* from colour histograms of a ball moving in two dimensions is improved considerably when a physics simulator is integrated into a MAP procedure involving non-linear optimisation and RANSAC-like methods. Process noise or initial condition noise in conjunction with a physics-based dynamics results in improved robustness on hard visual problems.

A particle filter applied to the task of full 6D *visual tracking* of the pose an object being pushed by a robot in a table-top environment is improved on difficult visual problems by incorporating a simulator as a dynamics model and injecting noise as forces into the simulator.

ACKNOWLEDGEMENTS

Dante Rossetti is reported[1] to have said “The worst moment for the atheist is when he is really thankful and has nobody to thank”, a statement rendered falsified by my expression here of my enthusiastic gratitude for chaotic happenstance and the colour red, in a universe that may or may not be deterministic, but is assuredly approximately as far from being encompassed now as it was before the publication of this thesis. I also unreservedly thank my my wonderful wife Esra who fought through this period in our lives with love and courage; my marvellous mother, whose patience and enthusiasm is like marigolds in a swamp; my fine father who with foresight indoctrinated me, without violence, in Science; my delightful daughter who had the good grace to be born whole and happy; my superb supervisors, Jeremy Wyatt and Rustam Stolkin, whose humility is unwarranted; all those fantastic friends, generous and wise colleagues and mentors who kindly distracted me with beautiful banter from the cavernous absence of a written thesis[2]; the celebrated and uncelebrated generations of philosophers and scientists who painstakingly created that beautiful tapestry of knowledge that I herein sully; the British[3], European[4] and New Zealand taxpayers[5]; all of the people exploited to build the present economic system that enables a small privileged unqualified few to fiddle around building up plaque on the fringes of science; and the noble self-sacrifice of the thesis that I originally set out to write.

But, Reader, it was all for you.

[1]Chesterton, 1987.
[2]Particular thanks to Zeyn Saigol & Peter Lewis for carrying this 2 kilogram load the last kilometre.
[3]ORSAS Scholarship and University of Birmingham School of Computer Science Scholarship
[4]EU FP6 CoSy FP6-004250-IP and FP7 CogX FP7-IST215181 projects
[5]StudyLink Student Loan Scheme & Student Allowance

CONTENTS

CHAPTER 1

INTRODUCTION

"I'm just saying I saw it bounce out."
`The Pebble and the Mustard Jar,`
`Joshua Malbin`

The aim of computer vision is to make computers see. It is, though, not always obvious what it means to see.

Clearly, it is possible to see colours, shapes, and light. But then it gets, figuratively speaking, murky. A layperson would be happy stating that it is possible to watch a pebble bounce; it is possible to look at the joy on a child's face; it is possible to see an affair come to an end. These are somewhat more controversial if they are to be considered within the remit of computer vision research.

It is not obvious what divides seeing from other kinds of understanding of a surrounding physical world. These kinds of understanding might commonly be thought to be based on inference from prior knowledge and other modalities as well as vision, but perhaps not vision in and of itself. But, there is prior knowledge involved in the seeing of edges and contours too. And although a non-researcher would admit to seeing edges or contours, they would probably not be the same edges or contours that preoccupy research in computer vision. Although emotion can be inferred from sound, it can be inferred from solely visual information.

So, it might seem that object impact, emotions, the status of relationships and many other topics would properly be in the domain of computer vision. But now the burden on the computer vision researcher is great. So a computer vision researcher probes a small part of this grand problem; chooses to push the boundaries of the discipline or push its capabilities ever so slightly with each new finding: deformable shapes, inference based on function, more robust vision, and so forth.

In this fashion, this thesis considers the inference of the parameters of rigid body motion as part of the vision problem and uses the rigid body models underlying this inference to improve existing vision techniques on marginal vision problems.

To reiterate, because these are the main ideas that this thesis is concerned with, the practical tasks of:

- Inferring the dynamic behaviour of rigid bodies from vision.
- Using rigid body dynamics knowledge to improve motion estimation and tracking performance.

1.1 Problems addressed

In this thesis these practical tasks are addressed by solving concrete problems in computer vision. *Motion estimation* is addressed both in simulation and within a set of real-world scenarios involving inference of fast moving objects from colour, and *object tracking* is addressed in the context of an object being pushed by a robot and its full 6D pose reconstructed over time from object texture. Motion estimation is the problem of reconstructing trajectories, that is sequence of poses, from visual information. Object tracking is the problem of maintaining an estimate of the current state of an object.

The specific problems addressed and where they are found in this thesis, a summary of the techniques employed, and their main findings, are listed in Table 1.1.

A version of the standard continuous-state analog to the Hidden Markov Model is used to incorporate a standard off-the-shelf physics simulator into the estimation problem as a dynamics model.

Chap.	Problem	Features	Framework	Main finding
5	6D motion estimation	Robust estimation & Levenberg-Marquadt	Noisy computer-generated 6D poses	Robust estimation shown necessary to obtain performance in presence of rotation, collisions, outliers.
5	2D motion estimation	Robust estimation & adaptive line-search	Colour histogram template matching	Physics-based dynamics model improves performance when used in a robust framework.
6&7	6D object tracking	Texture edge matching	Particle filtering from texture edges	Physics-based dynamics model improves performance with realistic impulse-based noise model.

Table 1.1: The main problems addressed in this thesis

Despite the extra complexities its introduction creates, the physics knowledge improves performance on the problems analysed here.

To solve the motion estimation problem, a combination of robust estimation algorithm and standard non-linear optimisation algorithms are employed with an off-the-shelf physics simulator employed to reconstruct trajectories from subsets of data, and to add hard and soft constraints to trajectories on the basis of expected physical behaviour.

To solve the visual tracking problem, a particle filter is employed, where the process of sampling candidate solutions forward in time is mediated by an off-the-shelf physics simulator.

1.2 Main ideas & terminology

1.2.1 Simulation & motion estimation

The aim in this thesis is to incorporate physical cognition into visual estimation problems; that is, a mastery of real-world scenarios involving object interaction and motion. The kind of physical cognition employed in this thesis is a very specific kind of physical cognition: *physical simulation.* Within the domain of control and motion planning, this is also known as a *forward model* since it infers state forward in time. In general, physical cognition is a broad term encompassing a number of different abilities - not only forward models (simulators), but inverse models at different levels of abstraction (that is, models of the actions necessary to obtain a given goal in a given state), the ability to estimate physical systems, including estimation from vision, manipulation of objects, describing objects, and more. This thesis is an attempt to find synergies between two of these kinds of physical cognition: estimation of object movement and physical simulation.

In particular, there is a lot of physical knowledge tied up in contemporary physical simulators. While the ability to simulate a physical system is a small subset of the kinds of processes needed for a human-like understanding of physical systems, it still has not been shown that this information can be exploited in any serious way to improve the estimation of the motion of physical systems.

Existing approaches to motion estimation and physical simulation share a lot with each other in terms of the representation of objects and their motion. However, despite this strong commonality, an investigation of the insights at the core of this thesis throws into relief some subtle differences in the form of the physical knowledge used in each task. While a physical simulator contains knowledge about how object state will change from one time-point to the next, motion estimation must make use of knowledge about how object appearance at one time-point relates to possible object state or how uncertain observations of partial object state relate to sets of possible object trajectories.

It is easy to conceive of visual motion estimation or physical simulation of things that are not easily called objects, such as fluids. Even so, the discussion in this thesis is restricted to the more amenable estimation and simulation of objects. Moreover, a specific class of objects is considered - *rigid bodies* - that is, objects that don't change their shape when they move. Furthermore, this thesis restricts its use of physical simulation to pre-existing off-the-shelf rigid body physical simulators - because doing so is the quickest route to an improved estimation system using physics, and will benefit from contemporary advances in simulation technology.

A short summary of the terms used so far is appropriate:

- *Visual motion estimation* is the task of using visual information to accurately reconstruct the movement of objects. Typically this movement is represented as the sequence of poses of each object.

- *Visual tracking* is the task of using visual information to maintain an estimate of the current location and state of target objects. This estimate can include information about uncertainty as well as the dynamic state of the object.
- *Physical simulation* is the task of calculating the evolving state of a set of objects as they move through space and interact with each other. Usually this state is represented as poses as well as dynamical information.

The bulk of the remainder of this document is focused on the elaboration of approaches for doing motion estimation using physical simulation and the results of experiments evaluating them.

1.2.2 Models used

In this thesis, the physical simulator is a strong *model* of the behaviour of objects in the world. When instantiated with a particular scenario (objects and their instantaneous states), it *models* their movement. The physical simulator therefore can model the dynamics of the object; any such model (one that models the movements of objects) is called a *dynamics model*, or a *motion model*[1]. If the dynamics model can represent uncertainty in how the state will change, it can be called *stochastic*. This thesis is particularly concerned with using such a stochastic dynamics model.

In addition to a dynamics model, in computer vision, other kinds of model occur. One is an *object model*. Such a model typically consists of analogs to an object's real-world physical properties - its shape, for example, appearance, such as a texture model, and so

[1]In this thesis, the phrase "dynamics model" is preferred as it implies a range of models considering the dynamic behaviour of objects, including collisions, though motion model is more common in some parts of the literature where issues of dynamics are not a strong consideration.

forth. An object model also might include physical properties like mass[1].

Another kind of model that occurs in computer vision is a *model of image formation*. Such a model can be considered a physical model of the journey of light through different media, and its inversion is particularly important in computer vision. In 3D computer vision one example of such a model is the *projective model* of image formation. Beyond the projection of points in space onto the 2D image plane, it is also possible to model the medium through which light moves, and so forth. In this thesis the focus is on physics in the scene itself, not on the physics of image formation. In probabilistic frameworks, a model of how world state is transformed into observations (images, for example) is called a probabilistic *observation model*.

This thesis does not attempt to build models of human cognition. It is purely grounded in the practical problem of estimation. Human physical cognition seems to depend a lot on context, and the use of explicit universal Newtonian-style models as employed here are almost certainly not applicable to an analysis of physical cognition in humans[2].

It will be useful to build physical cognition into synthetic systems, however. The advantages of doing this are several, particularly in the area of cognitive robotics. Cognitive robotics is of great potential and practical use to the community, with applications to human assistance, retroactive waste management, search and rescue, automated experimentation, environmental monitoring, dynamic industry and more. Vision is arguably the most important and difficult part of the problem; novel and integrative solutions are valuable.

As noted by Hogg (1984) in one of the first works in computer vision based on strong dynamics models, familiarity with an object, including the way it moves, is helpful when the object is far away, fast responses are needed, and so forth. This thesis is a humble contribution to this effort. In this thesis, visual estimation is achieved in the presence of

[1]It is useful to hold a distinction in mind between a dynamics model and an object model, not because they are necessarily mutually exclusive (they are not: an object model can be essential to an accurate dynamics model), but because the term "model" is common to them both but they are very different concepts.

[2]While possible implications for human understanding of physical and visual phenomena will be returned to in the discussion, they are not the core motivation for the work herein.

blur, glare, distractors, occlusion and occlusion-obscured passive physical interaction by using an improved dynamics model.

1.3 The rest of this thesis

The first chapter after the introduction (2) introduces the general stochastic model of the evolution and observation of systems used to frame the remainder of the work. Physical simulation is treated as a parametrised deterministic black-box function and it is shown how such a function can be used in general in motion estimation and object tracking.

Chapter 3 then deals with existing approaches to visual estimation, including motion estimation and recursive estimation, as viewed from the stochastic framework previously introduced. Robust estimation is also discussed for those cases where conventional assumptions break down. This chapter also details the concerns of visual geometry and feature extraction of contemporary approaches to computer vision, visual tracking, visual estimation, motion estimation and structure from motion and motivates algorithms based on the criterion of maximising posterior probability and recursive estimation formulation developed in the previous chapter.

Subsequently, the background basics of modern real-time physical simulation are given and related approaches that incorporate physics information are considered in the areas of object tracking, motion estimation, scene interpretation, control, motion planning and motion synthesis, as well as naive physics (Chapter 4).

Having established a framework for the use of physical dynamics in motion estimation and examined existing approaches, the following chapters present work focusing on specific scenarios in estimation from vision. Some of this experimental work has been published as conference papers (Duff, Mörwald, Stolkin, & Wyatt, 2011; Duff, Wyatt, & Stolkin, 2010).

First, work in simulation is described that validates the basic approach, and motivates the use of the RANSAC algorithm beyond conventional non-linear optimisation approaches that tend to only be capable of refining an estimated trajectory (Chapter 5). Refinement approaches use local cost gradient information to find new solutions but such approaches give rise to problems with local minima. RANSAC makes use of an inverse observation model employing a physics simulator to jump rapidly to approximately correct solutions. Next, a real-world scenario is investigated (again, Chapter 5). The motion of a bouncing ball is estimated from colour histogram features in a handful of image sequences in the presence of blur, glare, fast motion, as well as occlusion and distractors. The motion of the ball is restricted to two dimensions. It is shown how the use of a strong dynamics model based on physical simulation can put enough of a prior expectation on ball motion to estimate trajectories much more accurately than with simple dynamics models, and that simpler dynamics models must be applied only weakly to get adequate results. This improvement is seen when observations are noisy, as is the case with blur, occlusion, and distractors. Simple refinement is again insufficient for adequate estimation and RANSAC is used to get adequate results. Two refinement algorithms are presented, based on two different models of uncertainty (uncertainty in the observations and initial state as well as uncertainty in the process) and two different consequent parametrisations of a trajectory.

Chapter 6 then describes an approach to model-based tracking of the full 6D pose of a textured object from edge features, mainly texture edges, using a particle filtering approach that exploits graphics hardware to do quick testing of generated hypothetical poses. It is shown how a physics-based dynamics model can be used to improve the hypothesis generation part of the algorithm (and consequently the performance of the algorithm) in robot manipulation scenarios involving occlusion, distractors and fast motion, by adding a process noise model to a physical simulator that models force noise through the application of impulses rather than state perturbations. Chapter 7 applies this framework to scenarios involving a robot manipulator pushing an object in a tabletop environment. Prior information about the known trajectory of a robotic manipulator does little to improve this approach on the scenarios examined.

Finally, a summary (Chapter 8) is made of these results, their implications summarised, possible future work speculated on, and the discussion widened slightly.

CHAPTER 2

FOUNDATIONS: STATE ESTIMATION UNDER PARTIALLY OBSERVABLE STOCHASTIC DYNAMICS

"The market is still moving around in a \$90-to-\$100 range and I can't see it moving much past \$100 given the present news."

`Crude Oil Rises After U.S. Supplies Decline More Than Expected,`

`San Francisco Chronicle, 13 July 2011`

The problem that this thesis sets about addressing is a problem of estimation of object state (pose and velocity) from a series of images. The problem of state estimation presupposes a model of known structure in which there are some hidden variables whose values are to be found. This chapter introduces the basic structure of this model, which will be refined in later chapters as new methods are introduced. The basic model introduced here is largely the standard model in probabilistic robotics with some additional detail to allow for transforming deterministic dynamics models into probabilistic ones in ways different to those usually employed; by adding noise to the inputs of the model. The probabilistic machinery for doing this is well-known and basic, but the resulting framework comes in very useful as will be seen in the experimental Chapters 5, 6 and 7.

Since the basic premise of this thesis is that knowledge of the physical dynamics of objects should help with this state estimation problem, the model that is introduced should incorporate dynamics in a useful way. Moreover, it should have a place for deterministic simulation, since the knowledge of physical dynamics mentioned is encapsulated in state of the art real-time simulation technology.

Indeed, the model introduced here is apt not just for estimation, but also for stochastic simulation, control and decision making, and other kinds of probabilistic inference. This is a positive development, since estimation, simulation and control are but three kinds of physical cognition and obtaining a model that can encompass them all and allow for different kinds of inference is desirable for engineering reasons as well as for the potential to lead to insights into the nature of physical cognition.

In this thesis, the conceptual model of the partially observable dynamic world is a Hidden Markov Model (HMM) generalised to apply over continuous variables. To get an intuition for what an HMM is, an unfamiliar reader should look at Figure 2.1; the diagram represents the probabilistic dependences between variables; this can be interpreted as encapsulating knowledge about the causal dependence of variables in the model; however, the form in which they are stated, as joint or conditional probabilities, does not itself imply causation. Slightly more rigorously, an HMM consists of a Markovian dynamics, meaning that the state of the world is probabilistically independent of all states of the world at previous time-steps given the most recent previous state, and a generative observation model[1], meaning that observations at a given time have their only direct probabilistic dependence on the state of the world only at that time.

Although the this is a very general model, in the applications in this thesis, it is refined so that object state expresses linear and angular pose and velocity. The nature of the observations differ with the different computer vision approaches used.

There are some aspects of the model introduced here that are specific to this thesis. These are:

- The model is refined so that the probabilistic dynamics model contains a deterministic component analogous to a deterministic simulator. This allows for the easy integration of an off-the-shelf physics simulator into the estimation framework.

[1]A generative model is a model of how observations are produced from state, often probabilistic. The main alternative is a discriminative model which would model the direct probabilistic dependence of state on observations.

- This deterministic component is converted into a probabilistic model by the introduction of noise to the inputs as well as additive noise to the outputs. This allows for the addition of a novel kind of noise in the particle filter framework of Chapter 6. Converting a deterministic dynamics model to a probabilistic one for use in inference by placing noise over the parameters of the dynamics model is a contribution of this thesis.

- Time elapsed is parametrised in the dynamics model. This has been done many times before (e.g. by Ng, Pfeffer, and Dearden (2005)).

The rest of this chapter fleshes out the details of this model in terms of probability distributions and then moves on to a general discussion of estimation within such a model.

The deterministic dynamics (simulator function) are introduced first and then augmented with stochasticity, the more general form given, and then the observation model is introduced. Finally, the Bayes filter derivation is given, which is essential to the estimation approach used in this thesis.

2.1 State

The development of the model of system dynamics begins most naturally with the definition of system state. The state of a physical system at a given time t_i[1] is denoted $\boldsymbol{x}_{t_i}$[2]. If $t_i = t_{i-1}$ then it is required that $\boldsymbol{x}_{t_i} = \boldsymbol{x}_{t_{i-1}}$.

The state of a rigid object, for instance, might consist of its pose as well as the rate of change of the pose. The state of multiple objects might consist of the pose and rate of change of pose of all of the objects being estimated. For example, in full tracking in 3D space might track translational $\boldsymbol{T}$ and rotational $\boldsymbol{q}$ displacement, and translational $\boldsymbol{v}$ and

[1]The time t is indexed by i since a discrete time-step is generally used and i is used to index over these discrete time-steps, though the time between two consecutive time-steps can can vary (and a varying time-step length is a characteristic of physics simulators since time-steps can coincide with events at arbitrary times, such as collisions)

[2]Although the state can theoretically be any structured representation of a part of the world over which a probability distribution can be placed, in practice it is usually a numeric vector

rotational $\boldsymbol{\omega}$ velocity:

$$x = \begin{bmatrix} \boldsymbol{T} \\ \boldsymbol{q} \\ \boldsymbol{v} \\ \boldsymbol{\omega} \end{bmatrix}$$

A sequence of states $\boldsymbol{x}_{t_i} \cdots \boldsymbol{x}_{t_f}$ in this thesis is termed a *trajectory*. This is intuitive in the case of a single object. When it comes to multiple objects or non-rigid objects, the term trajectory is also used, which is slightly less intuitive, but reflects the simple concept that the dynamical behaviour of any physical system can be viewed as a sequence of states. For example, if two balls bouncing through a scene are modelled as 2D point particles, then the instantaneous location and velocity of both balls together may be considered the state, and the sequence of such states a trajectory.

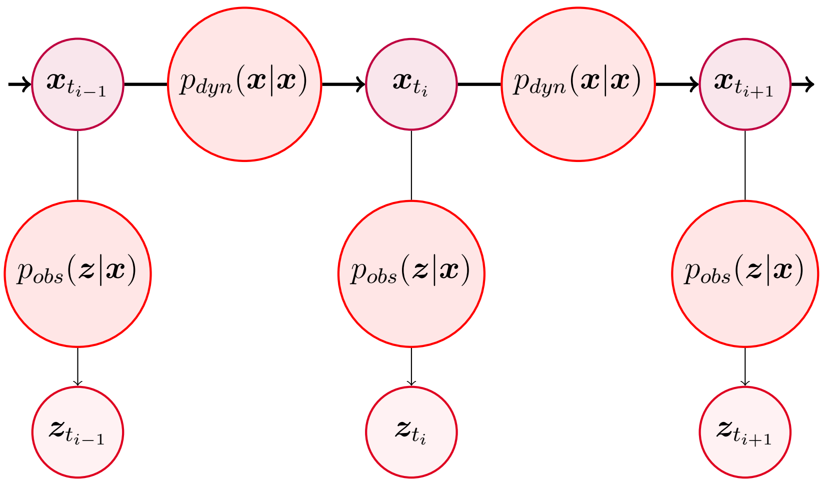

$\boldsymbol{x}_{t_{i-1}}$
$p_{dyn}(\boldsymbol{x}|\boldsymbol{x})$
$\boldsymbol{x}_{t_i}$
$p_{dyn}(\boldsymbol{x}|\boldsymbol{x})$
$\boldsymbol{x}_{t_{i+1}}$
$p_{obs}(\boldsymbol{z}|\boldsymbol{x})$
$p_{obs}(\boldsymbol{z}|\boldsymbol{x})$
$p_{obs}(\boldsymbol{z}|\boldsymbol{x})$
$\boldsymbol{z}_{t_{i-1}}$
$\boldsymbol{z}_{t_i}$
$\boldsymbol{z}_{t_{i+1}}$

Figure 2.1: Schematic of a generalised HMM (Hidden Markov Model). Big circles represent conditional probability distributions and little circles represent random variables over which a probability distribution can be put. Edges reflect dependence relations between distributions; a sequence of edges from random variable A to random variable B means that random variable B is independent of all the random variables with edges to distribution A, given that the distribution over A is known. The diagram shows the conditional independence of a state $\boldsymbol{x}_{t_{i-1}}$ from earlier states such as $\boldsymbol{x}_{t_{i-1}}$, given the immediately preceding state $\boldsymbol{x}_{t_i}$. It also shows the conditional independence of the observations $\boldsymbol{z}_{t_i}$ of all other variables at time t_i given the state $\boldsymbol{x}_{t_i}$ at the same time point.

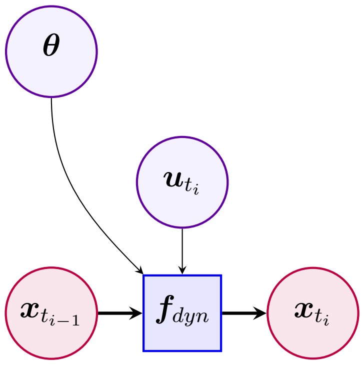


Figure 2.2: The deterministic simulation function, $\boldsymbol{f}_{dyn}()$, transforming state $\boldsymbol{x}_{t_{i-1}}$ at time t_{i-1} to state $\boldsymbol{x}_{t_i}$ at time t_i. Parameters: simulation parameters $\boldsymbol{\theta}$ and input parameters $\boldsymbol{u}$. Time elapsed, Δt_i is not depicted.

2.2 Deterministic dynamics

The model described in this chapter is a dynamic one. Consequently, having defined state, it becomes important to define how state changes over time. As mentioned previously, the model of state and state evolution defined in this chapter is a probabilistic one. However, first, a deterministic model of state evolution is developed, and then that deterministic model is extended into a probabilistic one.

The deterministic model of state evolution is a function over state with its output a state. i.e. a function mapping from the state of a physical system at one time to the state at a future time. This is a discrete time dynamical system. This function is termed the simulation function or the deterministic dynamics function.

In addition, the function is augmented with a time elapsed Δt_i[1]; a time-independent set of parameters $\boldsymbol{\theta}$ (typically a vector); and control or nuisance parameters $\boldsymbol{u}_{t_i}$ (also typically a vector). The function is depicted visually in Figure 2.2 and written as:

$$\boldsymbol{x}_{t_i+\Delta t_i} = \boldsymbol{f}_{dyn}(\boldsymbol{x}_{t_i}, \boldsymbol{\theta}, \boldsymbol{u}_{t_i}, \Delta t_i)$$

The motivation behind Δt_i is to allow variable length time-steps or 0-length time-

[1]The time elapsed is here subscripted with an i since the simulation function must be applied to one of the varying-length intervals between time-steps.

steps. In general $\underset{\Delta t_i \to 0}{lim}\ \boldsymbol{f}_{dyn}(\boldsymbol{x}_{t_i}, \boldsymbol{\theta}, \boldsymbol{u}_{t_i}, \Delta t_i) = \boldsymbol{x}_{t_i}$ (as the time elapsed approaches zero, the new state approaches the old). The vector-valued function $\boldsymbol{f}_{dyn}()$ is not one to one and it is not guaranteed to be onto[1].

The parameter $\boldsymbol{\theta}$ allows for the efficient specification of variables that do not change through time in the system under question, like rigid object shape, mass, restitution and friction coefficients[2]. The time-based input parameters $\boldsymbol{u}_{t_i}$ are useful for inference in the context of control since they can stand for control inputs, but in this thesis it is used as an alternative way of turning the deterministic dynamics described here into a probabilistic one. Rather than add noise to the output of $\boldsymbol{f}_{dyn}()$ to turn it into a probabilistic dynamics, a probability distribution can be placed over $\boldsymbol{u}_{t_i}$. An additional benefit of $\boldsymbol{u}_{t_i}$ is that it can stand for inputs into the system, such as forces on objects at each time-step [3]. Relatedly, it can be used to compensate for failures in the simulation function.

Clearly the simulation function is an abstraction of any conventional discrete-time physical simulator, including the off-the-shelf real-time physics simulator used in this thesis. Although analysis of the internals of this function would be a profitable line of research, in the context of this thesis, the simulator is a black box, with inputs and outputs as described above. The internals of the simulation function are described further in Chapter 4.

As a simple example of this simulator in action, imagine that $\boldsymbol{x}_{t_1}$ is the displacement and velocity of a simulated bouncing ball 33ms before it hits a vertical wall, with $\boldsymbol{T}_{t_1} = \begin{bmatrix} 1m \\ 1m \end{bmatrix}$ and $\boldsymbol{v}_{t_1} = \begin{bmatrix} -1ms^{-1} \\ 0ms^{-1} \end{bmatrix}$. If x_{t_2} represents the the displacement and velocity of

[1]Each state at time-step t_i can have more than one corresponding state at time-step t_{i-1} and not every state at time-step t_i has a corresponding state at time-step t_{i-1}.

[2]It is a somewhat arbitrary decision as to what information belongs in $\boldsymbol{\theta}$ and what belongs in the structure of the function $\boldsymbol{f}_{dyn}()$; the difference being that $\boldsymbol{f}_{dyn}()$ is a black-box in this formulation but $\boldsymbol{\theta}$ can be analysed. The less information that is represented in $\boldsymbol{\theta}$, the more information must be represented in the function. On the other hand, a simple numeric $\boldsymbol{\theta}$ is more amenable to analysis if the function $\boldsymbol{f}_{dyn}()$ remains black-box. The main motivation for the inclusion of $\boldsymbol{\theta}$ is to provide the framework for a search over the physical parameters of the simulation in order to better fit a sequence of data and as such in this work it is generally a vector of numeric values and object arity and shape consequently encoded in the function $\boldsymbol{f}_{dyn}()$.

[3]Injecting theoretical noise into the system is one method of many for partially overcoming the closed world assumption by encapsulating unknown external influences in the injected noise; an approach that should be effective if the parameters and the distribution over them reflect the kind of influence that may affect the system.

the simulated ball 66ms later (i.e. $\Delta t_i = t_2 - t_1 = 0.066s$), then that new state can be written in terms of $\boldsymbol{f}_{dyn}$ as $\boldsymbol{f}_{dyn}(x_{t_1}, \boldsymbol{\theta}, \boldsymbol{u}_{t_1}, 0.066) = \boldsymbol{x}_{t_2}$ and would be $\boldsymbol{T}_{t_1} = [{}^{1m}_{0.98m}]$ and $\boldsymbol{v}_{t_1} = [1ms^{-1} - 0.65ms^{-1}]$. The global simulation parameter $\boldsymbol{\theta}$ might encode the mass of the ball, its coefficient of restitution, a gravity vector, the shape of the ball and so forth. The input parameter $\boldsymbol{u}_{t_1}$ in this case is absent but might be an estimated external force, like the force of an unseen actor or unmodelled surface interactions.

2.3 Stochasticity

The task that this thesis has set out to solve is estimation of object motion. The idea is that physical dynamics as embodied in a physical simulator and available as a function $\boldsymbol{f}_{dyn}()$ can be helpful in solving that task. Unfortunately, such a simulator is seldom particularly accurate. Further, simulators have a trade-off with between accuracy and specificity - a very accurate simulator is generally specific to a small range of situations. With any estimation task, there is the problem that the model of the world dynamics used often is not a good model in that it does not well reflect the behaviour of the world (particularly in task-sensitive ways).

The most common way of dealing with model mismatch like this is to make the simplifying assumption that the model is correct but that the world generates additional random *noise* that creates the mismatch. For an example of such an approach, see Figure 2.3. The advantage of the assumption of model correctness with random added noise is that it is possible to continue to use a deterministic model as-is but still reason with it, taking into account deviation from expected behaviour. Noise consists of perturbation of a value from that which it would have taken were it realised deterministically. These perturbations are drawn from a probability distribution[1].

[1]For an introduction to probability distributions, see DeGroot's *Probability & Statistics* (DeGroot, 1986). Rather than derive it here, a probability distribution is described intuitively as a the shape of the probability of occurrence across all possible values of a random variable (where that variable is generally continuous in nature). In Bayesian probability theory, the *probability distribution* is the first-order object of analysis, as it is in this thesis. The assumption that a probability distribution can be picked to stand

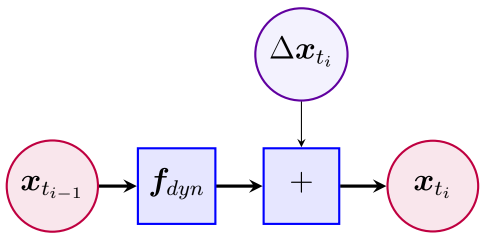
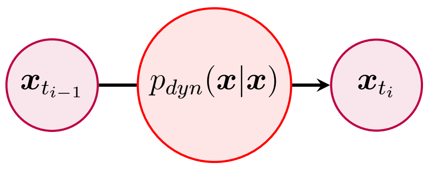


Figure 2.3: How a deterministic dynamics is typically made stochastic, by including additive noise. The stochasticity is introduced by making the variable $\Delta \boldsymbol{x}_{t_i}$ here into a random variable.



$\boldsymbol{x}_{t_{i-1}}$
$p_{dyn}(\boldsymbol{x}|\boldsymbol{x})$
$\boldsymbol{x}_{t_i}$

Figure 2.4: The general probabilistic dynamics model.

Other ways of creating probabilistic motion models exist, and abstractly, such approaches can be viewed as a conditional probability distribution over the state of an object conditional on the state at a previous time step $p_{dyn}(\boldsymbol{x}_{t_i}|\boldsymbol{x}_{t_{i-1}})$. This general model is illustrated in Figure 2.4.

Where no noise is added at all, for a fixed $\boldsymbol{x}_{t_{i-1}}$ the domain of the conditional distribution would contain only one point (it would be the Dirac delta) and would be for all practical purposes equivalent to a deterministic dynamics.

In this thesis, there are three sources of noise considered (these can be seen in Figure 2.4):

- Noise $\Delta \boldsymbol{x}_{t_i}$ added to the output of the simulator $\boldsymbol{f}_{dyn}$[1].

in for unknown aspects of the world is a strong one and presupposes some knowledge of the mechanics of the unknown part of the world. In the simplest case, picking a probability distribution over the likelihood of heads and tails during a coin toss should require some knowledge of the mechanics behind coin tosses, or some experience of them. The beauty of the use of probabilities is that even the barest knowledge is often enough to to get started.

[1]This approach is very common in standard models, since additive noise is conducive to analysis - for example, if a variable is described by a Gaussian, and the noise added to it is Gaussian, the random variable that is the sum of the two is also a Gaussian - this is used by the Kalman filter.

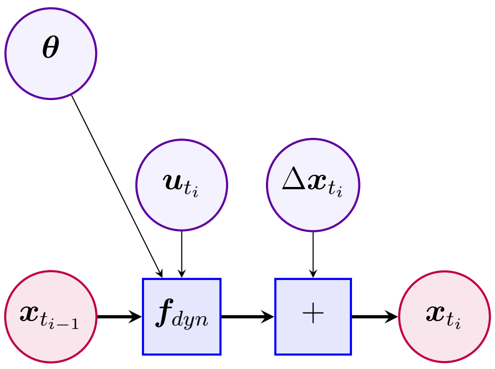


Figure 2.5: The model used in this thesis for obtaining stochasticity.

- Noise in the simulator parameters $\boldsymbol{\theta}$[1].
- Noise in the nuisance input parameters $\boldsymbol{u}_{t_i}$[2].

The probabilistic dynamics model resulting from including all of these sources of noise can be found in Figure 2.5.

2.4 Observations

The previous section described a set of probabilistic "dynamics models", some built on deterministic dynamics models.

Similarly in this thesis, the concept of an "observation model" is used. A generative observation model is a model of how "observations" are formed from events and/or states in the world. An observation is an event in a space of possible observations. So, for instance, if a ball is bouncing in front of a camera, an observation made by that camera might consist of a pixel of the colour of the ball in one place in a captured image. Clearly, if this model is viewed as a mapping from the space of states to the space of possible observations, it is not a one to one mapping (and so not invertible). Also important is

[1]Allowing background parameters to be a random variable is a common practice - for example, in map-building for autonomous robots.

[2]Noise in input parameters to a dynamcis model is a contribution of this thesis, at least in the context of estimation with dynamics models.

that such observations are often called "partial" because they do not reveal the full state of the world (in the case of object tracking or motion estimation, the world mostly consists of the object being tracked) - they generally provide some information about it, however, if interpreted well.

In this thesis, the phrase observation model is applies to probabilistic observation models, which define the probability of each observation given a particular state $p_{obs}(\boldsymbol{z}_{t_i}|\boldsymbol{x}_{t_i})$. If the observation is known but the state allowed to vary, this probability density is called an observation likelihood. If the prior probability of any observation is known, the probability of the state in terms of the observation may be derived from Baye's theorem[1].

It is important to note that often in the estimation task, the generative probabilistic model is not well defined. This is the case for example when heuristic methods are employed and a match score is used, this match score might be assumed to reflect an underlying unnormalised probability.

In this thesis, there are three different observation models used, and each will be discussed in their respective chapters. Not all are well defined. In probabilistic terms, examples include:

- State as a 6D pose and velocity of a rigid object. Pose is considered to be fully observed, perturbed by noise and with outlier noise added (Chapter 5).

- State as a 2D pose and velocity of one or more simple rigid objects. The object induces a colour histogram close to its own colour histogram and close to the true position of the object. Outliers are also described by the model (Chapter 5).

- State as a 6D pose of a rigid object. The object texture and geometry induces edges which are measured in the image. Observed edges are modelled as being likely to have a similar location, orientation and strength as actual edges, but are perturbed by noise (Chapters 6 and 7).

[1]In practice in this thesis there is not much difference between the two approaches since the prior probability of the observation is often superfluous in the final formulation of the estimation approaches used here anyway.

2.5 Partially observable stochastic processes from deterministic dynamics

In order to make more rigorous the probabilistic model above (the generalised HMM), this section gives some more details of the probabilistic framework behind it.

To begin with, two core assumptions are made to justify such a model, but have not so far been spelled out in this chapter. These conditional dependence assumptions are:

- *Observation independence.* The principle of the direct probabilistic dependence of observation on current state. If the current state is known, then no more information can be found about the current observation by looking at other past states or observations.
- *Temporal independence.* The "Markov assumption" of the direct probabilistic dependence of state on previous state: the assumption that all information about previous states and observations is not informative about the current state if the immediately preceding state is known.

These can be written more succinctly in terms of probability distributions:

$$p(\boldsymbol{z}_{t_i}|\boldsymbol{z}_{t_{0:t_{i-1}}}, \boldsymbol{x}_{t_{0:t_i}}) = p_{obs}(\boldsymbol{z}_{t_i}|\boldsymbol{x}_{t_i}) \tag{2.1}$$

$$p(\boldsymbol{x}_{t_i}|\boldsymbol{z}_{t_{0:i-1}}, \boldsymbol{x}_{t_{0:i-1}}) = p_{dyn}(\boldsymbol{x}_{t_i}|\boldsymbol{x}_{t_{i-1}}) \tag{2.2}$$

This relationship is required for the HMM diagram in Figure 2.1 to make sense. In particular, the right hand side in the two equations above, the conditional probabilities, are visible in the diagram. The notation $\boldsymbol{x}_{t_{0:i-1}}$ refers to the full set of states up to time t_{i-1}, and $\boldsymbol{z}_{t_{0:i-1}}$ the set of observations up to time t_{i-1}. These assumptions are important in the derivation of the motion estimation and recursive estimation (filtering) frameworks used in later chapters.

Note again that the right hand side in Equation 2.2 above is in this thesis described in more detail through the use of a deterministic dynamics at its core, so that the following distributions induce[1] a conditional probability distribution $p_{dyn}(\boldsymbol{x}_{t_i}|\boldsymbol{x}_{t_{i-1}}, \boldsymbol{\theta}, \boldsymbol{u}_{t_i}, \Delta t_i)$ over $\boldsymbol{x}_{t_i}$:

$$\begin{aligned} \boldsymbol{x}_{t_i} &= \boldsymbol{f}_{dyn}(\boldsymbol{x}_{t_{i-1}}, \boldsymbol{\theta}, \boldsymbol{u}_{t_i}, \Delta t_i) + \Delta \boldsymbol{x}_{t_i} \qquad (2.3) \\ \boldsymbol{x}_{\Delta} &\sim p_{\Delta \boldsymbol{x}} \\ \boldsymbol{u}_{t_i} &\sim p_{\boldsymbol{u}} \\ \boldsymbol{\theta} &\sim p_{\boldsymbol{\theta}} \end{aligned} \qquad (2.4)$$

This refined dynamics as inserted into the HMM can be seen in the diagram in Figure 2.6 and as a probability distribution, would be written with the extra parameters[2]:

$$p_{dyn}(\boldsymbol{x}_{t_i}|\boldsymbol{x}_{t_{i-1}}, \boldsymbol{\theta}, \boldsymbol{u}_{t_i}, \Delta t_i)$$

This refined dynamics conditional probability distribution is additionally used in parts of this thesis as a part of a multi-distribution combination, such as a mixture, so that, in the case of a mixture:

$$p_M(\boldsymbol{x}_{t_i}|\boldsymbol{x}_{t_{i-1}}) = \sum_k \pi_k p_k(\boldsymbol{x}_{t_i}|\boldsymbol{x}_{t_{i-1}})$$

Where p_k is a probability distribution in the form of that given in Equation 2.3 above, and π_k is a component weight. Combinations of probability distributions are not restricted

[1] $\Delta \boldsymbol{x}_{t_i}$ and $\boldsymbol{u}_{t_i}$ are governed by a constant probability distribution making this a stationary process.

[2] Note that in this thesis in diagrams it is sometimes convenient for brevity to drop the $\boldsymbol{\theta}$, $\boldsymbol{u}_{t_i}$ and Δt_i parameters on which the dynamics distribution p_{dyn} is conditioned.

to mixtures only, however. The mixture distribution is not illustrated here graphically [1][2].

Finally, the problem of estimation in the model outlined here is introduced.

2.6 Estimation with partially observable stochastic dynamics

The above-described general probabilistic model of a partially observable stochastic process is applicable to a variety of computational problems, of which stochastic simulation, control and estimation are three. In this thesis the problem of estimation in such a model is tackled. In particular, motion estimation, the task of estimating the state of an object across time given that all the observations are collected in advance, and recursive estimation, the task of maintaining an estimate of the current state of an object iteratively as new observations arrive. Approaches to both of these estimation problems generally exploit the Markov and observational independence assumptions described above in Equations 2.2 and 2.1.

The estimation problem can be approached according to several different probabilistic criteria, depending on the application. In the first approach, a complete probability distribution over the variables to be estimated might be desired. However, the issue of representation of the distribution often arises, and in some applications a single best-value is desired. Alternatively, in a second approach, an *expectation* of the variables might be desired, where the expectation is the average of the variable's possible values, weighted

[1]Although this model provides several means by which theoretical noise can be introduced into a deterministic system, the assumption still remains that the real structure of the noise is not known, since it originates in physical aspects of the world unknown to the human that designs the distributions of the noise. More generally, it is impossible to put a prior over anything in the system in a disciplined way without knowing everything about every mechanism in the system. It may be possible to make suggestive arguments about what shape the noise should have (for instance using the Central Limit Theorem to suggest that they should be Gaussian, and through appeals to postulated mechanisms). However, the noise model represents that which is not known but makes the assumption that we know the structure of that which we don't know; an assumption that is, in general incorrect, strictly speaking - but nevertheless useful.

[2]To "put a prior" over something is short-hand for completely specifying a "prior" probability distribution over that the values that thing might take, before any further evidence is incorporated - essentially it contains *what is known so far and can be expressed as a probability distribution.*

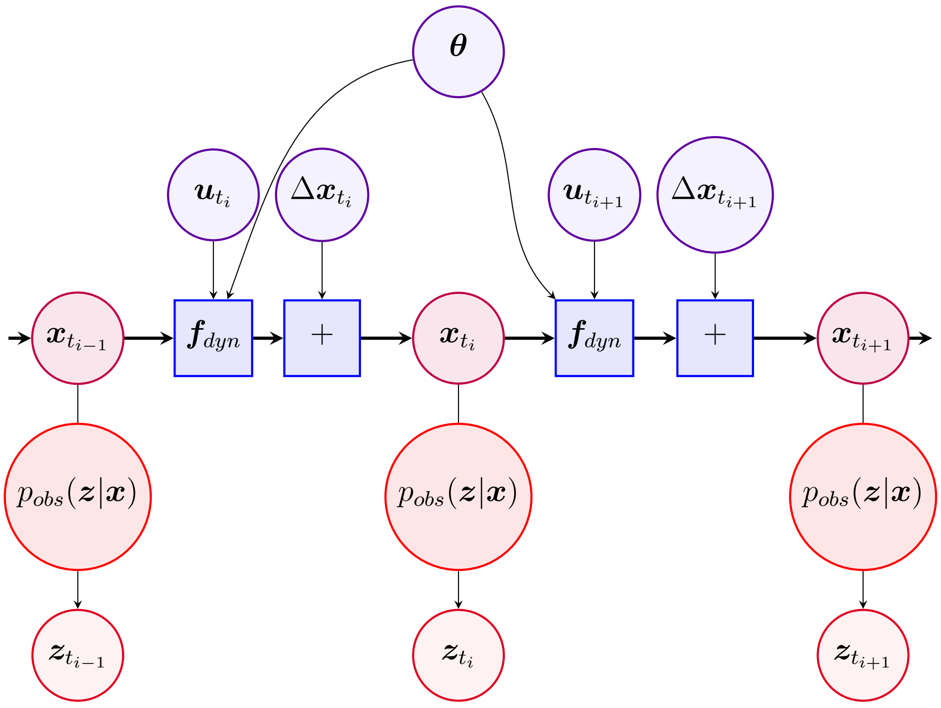

$\boldsymbol{\theta}$
$\boldsymbol{u}_{t_i}$
$\Delta\boldsymbol{x}_{t_i}$
$\boldsymbol{u}_{t_{i+1}}$
$\Delta\boldsymbol{x}_{t_{i+1}}$
$\boldsymbol{x}_{t_{i-1}}$
$\boldsymbol{f}_{dyn}$
$+$
$\boldsymbol{x}_{t_i}$
$\boldsymbol{f}_{dyn}$
$+$
$\boldsymbol{x}_{t_{i+1}}$
$p_{obs}(\boldsymbol{z}|\boldsymbol{x})$
$p_{obs}(\boldsymbol{z}|\boldsymbol{x})$
$p_{obs}(\boldsymbol{z}|\boldsymbol{x})$
$\boldsymbol{z}_{t_{i-1}}$
$\boldsymbol{z}_{t_i}$
$\boldsymbol{z}_{t_{i+1}}$

Figure 2.6: Generalised HMM with the dynamics being a combination of a deterministic simulation function $\boldsymbol{f}_{dyn}()$; stochastic state dispersion $\Delta\boldsymbol{x}$; simulation parameters $\boldsymbol{\theta}$; and input parameters $\boldsymbol{u}$.

by its probability distribution. Since an expectation generally is only useful for unimodal distributions, more commonly a maximum a posteriori (MAP) or maximum likelihood (ML) criterion is applied, and makes up a third approach. The MAP approach involves finding the values of the variables to be estimated that have the maximum probability according to evidence seen so far and any priors placed on unobserved variables in the system. The ML approach involves finding the values of the variables to be estimated that result in the highest probability of the observed variables. MAP and ML are closely related and one can be formulated in terms of the other[1].

In Chapter 5, a MAP or ML approach is detailed to the motion estimation problem, and in Chapters 6 and 7 the recursive estimation problem is tackled, which generally requires some estimate of the full probability distribution to be maintained, since this estimate is re-used in future stages of the estimation. In the remainder of this chapter, it will be shown how the Markov assumption and the assumption of observational independence assumption can be used to simplify the problem of estimation. This is standard and introduced here for the purpose of setting the thesis work in an easily understood probabilistic framework.

2.6.1 MAP trajectory estimation

In the case of MAP trajectory estimation, the quantity of interest is generally:

$$\arg\max_{\boldsymbol{x}_{t_{0:i}}} p(\boldsymbol{x}_{t_{0:i}} | \boldsymbol{z}_{t_{0:i}})$$

i.e. the maximum a posteriori estimate of the full trajectory of states given all observations made to date. Or, taking account of the extra simulation parameters introduced

[1]It should be noted that MAP and expectation are only stand-in measures for a quality of an estimate and generally speaking the relevant criteria is the utility of an estimate in supporting decision making (planning, control, communication and so forth).

in this chapter:

$$\arg\max_{\boldsymbol{x}_{t_{0:i}}} p(\boldsymbol{x}_{t_{0:i}}|\boldsymbol{z}_{t_{0:i}}, \boldsymbol{\theta}, \boldsymbol{u}_{t_{0:i}})$$

Using the two (observation and temporal independence) assumptions given above, the quantity $p(\boldsymbol{x}_{t_{0:i}}|\boldsymbol{z}_{t_{0:i}})$ can be refined to be a product of the conditionals in the generalised HMM - i.e.:

$$\begin{aligned} p(\boldsymbol{x}_{t_{0:i}}|\boldsymbol{z}_{t_{0:i}}) &= \frac{p(\boldsymbol{z}_{t_i}|\boldsymbol{x}_{t_{0:i}}, \boldsymbol{z}_{t_{0:i-1}})p(\boldsymbol{x}_{t_i}|\boldsymbol{x}_{t_{0:i-1}}, \boldsymbol{z}_{t_{0:i-1}})p(\boldsymbol{x}_{t_{0:i-1}}|\boldsymbol{z}_{t_{0:i-1}})}{p(\boldsymbol{z}_{t_{0:i}})} \\ &= \frac{p_{obs}(\boldsymbol{z}_{t_i}|\boldsymbol{x}_{t_i})p_{dyn}(\boldsymbol{x}_{t_i}|\boldsymbol{x}_{t_{i-1}})p(\boldsymbol{x}_{t_{0:i-1}}|\boldsymbol{z}_{t_{0:i-1}})}{p(\boldsymbol{z}_{t_{0:i}})} \\ &= \frac{p_{obs}(\boldsymbol{z}_{t_0}|\boldsymbol{x}_{t_0})p_0(\boldsymbol{x}_{t_0})\prod_{j=1}^{i}[p_{obs}(\boldsymbol{z}_{t_j}|\boldsymbol{x}_{t_j})p_{dyn}(\boldsymbol{x}_{t_j}|\boldsymbol{x}_{t_{j-1}})]}{p(\boldsymbol{z}_{t_{0:i}})} \\ &= \frac{p_0(\boldsymbol{x}_{t_0})\prod_{j=0}^{i}[p_{obs}(\boldsymbol{z}_{t_j}|\boldsymbol{x}_{t_j})]\prod_{j=1}^{i}[p_{dyn}(\boldsymbol{x}_{t_j}|\boldsymbol{x}_{t_{j-1}})]}{p(\boldsymbol{z}_{t_{0:i}})} \end{aligned} \tag{2.5}$$

Note that rather than include the base case (time t_0) as a special case, a small trick could be played by not including the time t_0 base case. However, later approaches to MAP estimation rely rather a lot on boundary conditions.

In this form, the MAP problem is relatively amenable. If all the conditional distributions in the right hand side are defined, and a set of observations acquired, then for any selected trajectory, $\boldsymbol{x}_{t_{0:i}}$ the probability of that trajectory can be calculated from Equation 2.5. Often, the *log* of the function can be taken, yielding sums where before were multiplications, and simplifying some distributions (in particular, Gaussians). Note that in this formulation, a trajectory is scored on the basis of its match to the observations and its local conformance to the dynamics model. The formulation provides a direct way of deciding the probability score of a hypothesised trajectory - but it does not provide a simple way to find the hypothesis in the first place, which is from this perspective an algorithmic question.

However, depending on the shape of the probability distributions, it may be possible to calculate the MAP or the ML value of $\boldsymbol{x}_{t_{0:i}}$ directly, without search. This is often the case in the motion estimation literature - when Gaussians are used, a procedure much like weighted averaging can be applied - this is least-squares estimation (DeGroot, 1986). In this thesis, even though noise is usually modelled as Gaussian, the highly non-linear and bifurcating nature of the dynamics render such approaches difficult to apply and search-based methods and sampling methods are generally used.

Note that if the control parameter $\boldsymbol{u}_{t_i}$ and background parameters $\boldsymbol{\theta}$ are included in the formulation, then the distribution over which the trajectory is to be optimised is written:

$$p(\boldsymbol{x}_{t_{0:i}}|\boldsymbol{z}_{t_{0:i}}, \boldsymbol{\theta}, \boldsymbol{u}_{t_{0:i}}) = \frac{p_0(\boldsymbol{x}_{t_0}) \prod_{j=0}^{i} [p_{obs}(\boldsymbol{z}_{t_j}|\boldsymbol{x}_{t_j})] \prod_{j=1}^{i} [p_{dyn}(\boldsymbol{x}_{t_j}|\boldsymbol{x}_{t_{j-1}}, \boldsymbol{\theta}, \boldsymbol{u}_{t_i}, \Delta t_i)]}{p(\boldsymbol{z}_{t_{0:i}})} \tag{2.6}$$

2.6.2 Recursive estimation

In the case of recursive estimation, the distribution that the estimator is interested in maintaining is:

$$p(\boldsymbol{x}_{t_i}|\boldsymbol{z}_{t_{0:i}}) \tag{2.7}$$

i.e. the current state given all observations made to date.

Or, if the extra simulation parameters are added:

$$p(\boldsymbol{x}_{t_i}|\boldsymbol{z}_{t_{0:i}}, \boldsymbol{\theta}, \boldsymbol{u}_{t_{0:i}}) \tag{2.8}$$

Rather than try to calculate this in terms of all the observations made so far, the Markov assumption allows for all previous observations to be incorporated in the estimate of previous state, $p(\boldsymbol{x}_{t_{i-1}}|\boldsymbol{z}_{t0:i-1})$, so that:

$$\begin{aligned} p(\boldsymbol{x}_{t_i}|\boldsymbol{z}_{t0:i}) &= \frac{p_{obs}(\boldsymbol{z}_{t_i}|\boldsymbol{x}_{t_i})}{p(\boldsymbol{z}_{t_i})} p(\boldsymbol{x}_{t_i}|\boldsymbol{z}_{t0:i-1}) \\ &= \frac{p_{obs}(\boldsymbol{z}_{t_i}|\boldsymbol{x}_{t_i})}{p(\boldsymbol{z}_{t_i})} \int_{\boldsymbol{x}_{t_{i-1}}} p_{dyn}(\boldsymbol{x}_{t_i}|\boldsymbol{x}_{t_{i-1}}) p(\boldsymbol{x}_{t_{i-1}}|\boldsymbol{z}_{t0:i-1}) d\boldsymbol{x}_{t_{i-1}} \end{aligned} \tag{2.9}$$

This reduces the problem to one of taking the previous estimate $p(\boldsymbol{x}_{t_{i-1}}|\boldsymbol{z}_{t0:i-1})$, incorporating the dynamics model $p_{dyn}(\boldsymbol{x}_{t_i}|\boldsymbol{x}_{t_{i-1}})$, observation $\boldsymbol{z}_{t_i}$ and the observation model $p_{obs}(\boldsymbol{z}_{t_i}|\boldsymbol{x}_{t_i})$ and obtaining an updated distribution $p(\boldsymbol{x}_{t_i}|\boldsymbol{z}_{t0:i-1})$.

This idealisation assumes of course that $p(\boldsymbol{x}_{t_{i-1}}|\boldsymbol{z}_{t0:i-1})$ is maintained perfectly, and that all probability distributions can be combined perfectly, assumptions that are only approached if easily analysed models can be found or as the available computational facilities increase.

An example of an easily analysed model is the Kalman filter family of recursive estimators. Here, representing the conditional distributions as Gaussians allows for a quick and accurate update. However, the Kalman filter applies directly only to systems with linear dynamics and Gaussian noise, conditions not fulfilled in this thesis. Extensions exist (see Chapter 3 Section 3.6.2) but the approach taken in this thesis is a Monte Carlo approach, where the distribution is approximated by samples - such a recursive estimator is called a particle filter.

Again, if $\boldsymbol{\theta}$ and $\boldsymbol{u}$ and Δt_i are included in the recursive formulation, the following recursive distribution is obtained:

$$p(\boldsymbol{x}_{t_i}|\boldsymbol{x}_{t_{i-1}}, \boldsymbol{z}_{t0:i}, \boldsymbol{\theta}, \boldsymbol{u}_{t0:i}, \Delta t_i) =$$

$$\frac{p_{obs}(\boldsymbol{z}_{t_i}|\boldsymbol{x}_{t_i})}{p(\boldsymbol{z}_{t_i})} \int_{\boldsymbol{x}_{t_{i-1}}} p_{dyn}(\boldsymbol{x}_{t_i}|\boldsymbol{x}_{t_{i-1}}, \boldsymbol{\theta}, \boldsymbol{u}_{t_i}, \Delta t_i) p(\boldsymbol{x}_{t_{i-1}}|\boldsymbol{z}_{t0:i-1}, \boldsymbol{\theta}, \boldsymbol{u}_{t0:i-1}) d\boldsymbol{x}_{t_{i-1}} \tag{2.10}$$

$$\tag{2.11}$$

2.7 Final chapter notes

Estimation can and should exploit the particular structure of the problem beyond the general probabilistic framework given here. The fact that any two arbitrary objects are usually not interacting can be exploited, for instance, by solving for each object individually most of the time, reducing the dimensionality of the problem. Such considerations are in this thesis dealt with in the context of particular methods.

In the next chapter, the background literature on visual motion estimation as well as on physical simulation is examined, as well as work related to the approach described in this thesis. Then a series of chapters detail approaches to motion estimation and recursive estimation developed for particular scenarios, based on the ideas introduced in this chapter, in particular the incorporation of physical simulation into the estimation problem.

CHAPTER 3

BACKGROUND: VISUAL ESTIMATION IN STOCHASTIC SYSTEMS

"My task which I am trying to achieve is, by the power of the written word, to make you hear, to make you feel — it is, before all, to make you see."

The Nigger of the 'Narcissus'

Joseph Conrad

In this thesis, the core problem addressed is that of visual estimation[1] - in other words, attempting to determine some relevant properties of the world from visual information. This thesis takes the particular step of investigating how an advance knowledge of object dynamics (in the form of a physical simulator) can be used to improve the estimation of object state over time.

As discussed in the previous chapter, in order to account for the inaccuracy of existing models of visual data, and for a changing world, stochastic models are used as the basis of visual estimation. The present chapter discusses how visual estimation generally is done within such a framework.

More rigorously, the problem that this chapter focuses on is that of estimating object state, $\boldsymbol{x}_{t_i}$ over time from observations $\boldsymbol{z}_{t_i}$ received over time in the form of visual information. A variety of methods will be discussed for doing this, many of them applying only

[1]Estimation involves finding parameters of a system from observations of that system, possibly over time. This contrasts with regression that aims to find a relationship between an independent and a dependent variable, though regression is itself an estimation problem as it is the parameters of this relationship that are estimated.

to estimation of state at a single point in time rather than over multiple time-steps (i.e. generally from images taken at one point in time). Note that in the visual estimation literature the main focus of study is estimation of *objects* as tackled in this thesis and estimation of a *scene*[1].

The main distinction maintained in this chapter is between *visual motion estimation* and *visual tracking* or *visual recursive estimation*[2]. The reason that this distinction is maintained here is because the empirical work in this thesis is on two problems - the visual motion estimation problem and the visual tracking problem. The distinction is between taking a sequence of images and batch processing them to produce an output trajectory and taking images in real-time and producing a current estimate of object state (and uncertainty in object state).

What constitutes an object state differs from application to application. At the most extreme, it is possible to consider the state of the whole world as the thing to be estimated (if in a closed-world). But then, since only a small number of things are typically of interest for a particular problem, or can be estimated from information available to the visual apparatus, in practice the state to be estimated is well delineated around one or a small number of objects[3]. The state to be tracked in model-based tracking is generally the pose of the known tracked object (or objects), since this information can be useful in other model-based tasks, such as manipulation or scene understanding[4]. Dynamic properties of objects such as their velocities are sometimes of interest too, when acting[5].

This chapter first deals with the problem of acquiring features and using them to

[1]The tasks of rigid object estimation and scene estimation are, for the most part the same problem. A scene is just an object that usually fills the camera's viewing area, though in the case of multiple objects the problem is a little more complex.

[2]The distinction between visual motion estimation and visual tracking matches the distinction between trajectory estimation and recursive estimation introduced in Chapter 2.

[3]It may sometimes only be necessary to track the pixel location of an object, for instance, if developing a human computer interface, or to track a pixel location and a rotation in the image plane. It may be necessary only to segment the object in the image plane if feeding into an object recogniser. It may be necessary to get the image location as well as depth if doing simple servoing - for face tracking, rotation away from the image plane may be important.

[4]For non-rigid objects, pose is insufficient to describe their state and the estimation problem typically harder; for unknown rigid objects the shape of the object is important to estimate also.

[5]Though in most applications dynamic quantities like velocity are only estimated in order to better estimate pose-related object state.

disambiguate the contents of a static image, followed by a brief discussion of algorithmic methods for making use of those features, and finally a generalisation of this approach to estimation over time.

3.1 The visual apparatus

Exposition of computer vision problems usually begins with a description of the theoretical visual apparatus; a model of how light interacts with objects in the world and reaches the lens and light sensitive surface, ultimately producing pixels. If the reader wishes some background on this, a gentle introduction is given in Appendix A. To summarise this, a model is provided of the projection of light from world-points to pixel coordinates, in terms of a rigid transform from object-centred coordinates to world-centred coordinates, from world-centred coordinates to camera-centred coordinates, projected to the pixel-space and lastly corrected for any non-linear lens distortions due to that linear projection.

The problem of reconstructing a configuration of the world, even given a model of how a world configuration produces an image, is the problem addressed in this thesis. The naive approach from a model-based perspective is of course generate and test (instantiate a possible configuration and test it against retrieved images); indeed the approaches detailed in this thesis follow some variations on the generate and test approach (local search and Monte Carlo). The most common way of reducing the complexity of the problem is by first extracting "features" from the images received. These features are generally extractable from the image without the use of any top-down knowledge, and should be informative about the world state. These features can then be used to solve a simpler or more tractable problem and are the subject of the next section.

In this thesis, colour histograms and texture and geometric edge features are used in different chapters, though the techniques described in this thesis are applicable across a wide range of features. The following section therefore reviews colour histograms and edges.

3.2 Visual features

Visual features are data structures embodying information extracted from images, used to guide processing. The extraction of a feature generally involves some image analysis, after which the extracted feature can be used in further processing[1]. This section describes those visual features employed in this thesis. For a discussion of other kinds of features available, see Appendix B.

3.2.1 Edges

The idea of edges comes from the insight that sharp discontinuities in light in an image, arrayed in a straight line, often signifies a discontinuity in the real-world. Edges in computer vision are just these - long discontinuities in the image - and as such can signify many other things in the world as well - shadows and textures being two such things. There are numerous efficient (and parallelisable) algorithms for detecting and using edges (Canny, 1987; Jain, Kasturi, & Schunck, 1995; Sonka, Hlavac, & Boyle, 1998; Trucco & Verri, 1998).

The first task in finding such edges is calculating the rate of change in brightness or colour across the image in which edges are to be detected (i.e. the image intensity gradient). More generally, this means finding the match in each part of the image to an edge-like quality [2]. This generally involves sampling the discrete neighbourhood around each potential edge point and approximating the degree of match (or the rate of change). After this, some post-processing is often done; for instance, thresholding away low edge response pixels, non-maxima suppression so as not to find extra edges from one real edge, or hysteresis to link edge pixels into plausible sequences. Matching of edges between two images is often done by calculating the degree of overlap in edge pixels, sometimes

[1]A feature is not equivalent to a model of the object being tracked; an object is something in the world being estimated while a feature is a property of an image. However, in some applications simply describing an object in terms of its associated feature or features is sufficient - for example, contour models of objects.

[2]The intensity profile along an edge can differ depending on the kind of edge (for example, a step change, a brightness peak or ridge, etc) (Canny, 1987). The classic edge is the step change.

accounting for the direction and strength of the edge. Edges are used in Chapters 6 and 7 to track object texture (or, in the absence of texture, geometric features).

Calculating the intensity gradient is generally done by applying a discrete filter to the image, so that the pixels in the neighbourhood of each point are summarised into an estimate of the gradient. A large number of such filters exist, with differing theoretical and practical properties. One of the more popular is the first order derivative of a 2D Gaussian. Since this derivative exists in two directions, two such filters might be applied, producing edge responses in two directions (usually vertical and horizontal), which can then be combined into a magnitude that represents the edge-ness of a point (Canny, 1987).

The camera model introduced in Appendix A can be used to transform lines in space as well as points (Hartley & Zisserman, 2000) if edges are processed into lines, though often they are maintained as a pixel map with each pixel representing response or responses of an edge detector at that pixel[1].

One way of further processing edges to provide more information about an object is by attempting to match them into contours, which are parametric descriptions of the flexible visible outline of an object or of parts of an object (Baumberg & Hogg, 1994a; Blake, Curwen, & Zisserman, 1993; Heap & Hogg, 1998; Isard & Blake, 1998a, 1998b; Kakadiaris, Metaxas, & Bajcsy, 1994; MacCormick & Blake, 2000), or with texture edges or edges induced by the known shape of a rigid 3D object (Drummond & Cipolla, 2002; Harris & Stennett, 1990), a task made easier if an approximate location of the contour in the image is known (for example, if the contour is being tracked over time). This locality requirement that has only with recent computational gains been relaxed with multi-hypothesis methods like the particle filter (Klein & D. W. Murray, 2006; Pupilli & Calway, 2006).

The algorithmic expense of tracking of arbitrary 3D models with edges is hidden edge removal and the mapping of edges to model contours and textures. Recently, this task has been made more efficient by recruiting parallel processing on low-cost graphics hardware

[1]In Chapters 6 and 7, for example, the framework used there is one of texture projection, and at one part of the algorithm the texture that is projected is just a pixel edge map.

(Klein & D. W. Murray, 2006; Mörwald, Zillich, & Vincze, 2009).

As compared to feature points[1], edges have the disadvantage that they are not as distinct locally in an image, but they are likely to exist even in plain objects and are quite invariant to a lot of changing conditions (such as illumination) and are more resistant to blur and image noise. Moreover, edges' lack of distinction locally is partially discounted by the distinctiveness of their configurations globally. Of course, it is possible to combine both edge and point features to get a more robust tracker (Vacchetti, Lepetit, & Fua, 2004a; Vacchetti, Lepetit, & Fua, 2004b).

3.2.2 Colour

Most visual features are initially defined on grey-scale video and can be adapted to exploit colour with small extensions. However, there are a class of methods designed to work with colour as the primary source of information. While colour is not by itself strongly indicative of the fine spatial and geometric properties of an object, it is often good at discriminating between classes of objects (Swain & Ballard, 1990). Moreover, when used in combination with other properties, it can be very discriminatory - for example, Wren, Azarbayejani, Darrell, and Pentland (1997) use contours and colour distributions to track human figures, with colour dominating in dynamic situations, but contours more discriminative in more stable situations[2].

Use of colour tends to involve profiling the colour distribution of a target region (since a typical object has a discriminative mixture of colours) as well as the background (Wren, Azarbayejani, et al., 1997). Colour-based methods should not be sensitive to illumination, and as such the representation should be able to capture properties of colour that are not dependent on illumination.

The most popular way of capturing a colour distribution is by putting it into a histogram, where each bin in the histogram is delineated by a particular range of intensities

[1]Feature points are described in more detail in Appendix B.

[2]Region segmentation algorithms (discussed in Appendix B which goes into a wider variety of feature types) can also use colour to produce region features.

of colour from different channels and the value in the bin corresponds to the relative observed number of pixels seen in that range. This method is used in Chapter 5.

In order to overcome the illumination invariance issue, the colour space used is often one that has relatively invariant channels, such as the HSV colour space. The colour histogram method is relatively invariant to object view as the histogram can embody a summary of the colour distribution over a set of views. The more representative the views, and the more the histogram partition is able to conform to colour invariants, the better the representation. As far as this last issue is concerned, it has been noted that a 2D colour space is more difficult to get right, and depends much more on the quantisation of the space (Ennesser & Medioni, 1995). Another possibility for dealing with illumination invariance during object tracking is to adapt the colour histogram over time (Nummiaro, Koller-Meier, & Van Gool, 2002).

Once the colour distribution is captured for an object, it can be matched to regions in a target image. Regions in which colour distributions are measured might be circles or elipses or in the shape of the target object if known; they might also be based on kernels if a gradual, or fuzzy region is desired (Comaniciu, Ramesh, & Meer, 2003; Naeem, Mills, & Pridmore, 2006).

If the scale of the region is not known in advance[1] then searching over all possible scales can be a computationally forbidding task; in which case growing methods could be used, that look for local evidence of a possible match and grow the match region opportunistically (Ennesser & Medioni, 1995). In tracking methods, where images are received one after another, the task of scale-finding is made easier in that the scale needs only to be adapted over time, for instance by looking the area of match within the current region, or a proxy for it (Bradski, 1998). Ennesser and Medioni (1995) also propose including local spatial information to make the histogram method more discriminating, and Bradski (1998) uses the spatial distribution of colour to determine the rotation of a target object in

[1]For the case where the opportunity exists to look for an object at all scales, the reader may find it interesting to consider the relationship between region size and the likelihood of a match with a matching or arbitrary colour histogram, dependent on the quantisation and distribution of background colours.

a scene; Perez, Hue, Vermaak, and Gangnet (2002) use multiple linked object parts each with different colour distributions. Indeed, delineating the boundaries of colour objects is one area where multiple features can be very useful, for example by combining colour cues with edges (Stenger, Thayananthan, Torr, & Cipolla, 2006).

For each possible region to be matched it is possible to calculate a match score (or a match distance[1]) based on the overlap between the histogram of a target object and the histogram of the target region[2]). This matching method can be used to find a known object in an image or to identify an object from a set of objects; when these two tasks are tackled together the complexity does increase (Swain & Ballard, 1990).

There are a number of ways of calculating the match score between histograms. Histogram intersection (Swain & Ballard, 1990) is the most basic method, where the source and target histograms are normalised and the differences in colour intensity in each bin summed over the whole histogram. This technique is fast and simple: $d_{HI}(H^{tar}, H^{reg}) = \sum_j H^{tar}(j) - H^{reg}(j)$, where H^{tar} and H^{reg} are the target and region histograms respectively and j is the bucket number. The target object histogram could be used directly for indexing colours in the image, but more often the values in the target object histogram are first divided by the values in the histogram of some background set, such as the whole image (e.g. $H^{nor}(j) = \frac{H^{tar}(j)}{H^{img}(j)}$) to make the histogram more discriminating of the object (Comaniciu et al., 2003; Swain & Ballard, 1990)[3].

Sometimes, rather than convolving a target with a whole image to get a histogram match at each location, what is desired is to find a target close to a position where it is known to be (as is the case with object tracking where each new image typically comes

[1]From a probabilistic perspective, a match score can be considered a proxy for an unnormalised likelihood.

[2]A full overlap between histograms would constitute a maximum match score and a zero match distance.

[3]In order to make the matching process a lot faster, a multi-channel image can be replaced by an image where each pixel contains an index into a colour histogram rather than containing the original intensity values. This can then be used for building another new image where the value at each pixel is the (possibly normalised) number of entries in the corresponding target histogram bin (Swain & Ballard, 1990). From that new image, any arbitrary region template can be cross correlated with the image to smooth it and produce a match score at each pixel. The pixel with the best match score is considered most likely to be the location of the object.

with some knowledge about the object in the previous time step). In that case, it will be possible to use match scores that take longer to calculate. A very widely used match score in tracking is the Bhattacharyya Coefficient ($C_{Bhatt}(H^{tar}, H^{reg}) = \sum_j \sqrt{H^{tar}(j) \cdot H^{reg}(j)}$) and its counterpart, the Bhattacharyya Distance ($d_{Bhatt} = -log(C_{Bhatt})$) (Comaniciu et al., 2003; Mihaylova, Brasnett, Canagarajah, & Bull, 2007).

Finally, rather than using a histogram to model colour distributions, much success has being observed using parametric distributions - Gaussians (Wren, Azarbayejani, et al., 1997) and mixtures of Gaussians (Magee, 2001).

3.3 General vision solution methods

Having acquired image features, the problem still remains of using them to reconstruct the location and pose of an object[1], particularly as a part of the motion estimation and object tracking problems. As well as techniques designed for these particular problems, there is a large set of techniques designed for general problems of photogrammetry and computer vision, such as reconstruction of a scene from multiple viewpoints, that are directly applicable to the problems.

This chapter will deal with these general techniques at a broad level, though emphasising the MAP problem and related solutions, and will discuss specific work on long sequence motion estimation with simple dynamics models. Finally the work on visual tracking will be addressed. First, the problem of camera calibration is discussed briefly, then photogrammetry introduced.

3.3.1 Calibration

Each camera is different, and each situation that it is used in is different. As discussed in Appendix A regarding the camera apparatus, there are numerous parameters govern-

[1]The problems of recognition and segmentation are not concerns of this thesis. This thesis focuses on two kinds of problem: motion estimation and tracking.

ing the working of a model camera. These parameters might be ignored, they might be estimated at the same time as other parameters (see bundle adjustment and direct linear transform, below), or they can be estimated in advance. The last option is calibration, and is most useful if the parameters are unlikely to change significantly during operation, or if these parameters can be updated asynchronously. By estimating these parameters in advance (or in parallel), the estimation problem can be significantly simplified, and computational complexity reduced. Good calibration is required for model-based techniques that require an accurate metric and do not include estimation of calibration parameters.

The determination of the pose of a fixed camera is sometimes called the calibration of "external parameters" and the determination of focal length, aspect, distortion parameters and so forth called the calibration of "internal parameters"[1] (Hartley & Zisserman, 2000; Karara, 1989). In the work discussed in this thesis, fixed cameras are used, so any calibration done is both internal and external.

In Chapter 5, calibration is entirely manual, and in Chapters 6 and 7, calibration of parameters is done according to the widely employed method of *Open Source Computer Vision Library: Reference Manual* (2001), Zhang (2000), Zhang (1998). This method uses images of a plane from multiple views where points on the plane are of known coordinates in the plane frame; the mapping between such views, without taking into account distortion parameters, is a planar homography. From the homography, camera intrinsic parameters can be calculated algebraically. In order to include distortion parameters and to refine the solution to minimise geometric error, a refinement stage is subsequently carried out using Levenberg Marquadt minimisation. This two-step procedure is common in computer vision and calibration in particular (Karara, 1989; Tsai, 1987). To calculate the extrinsic parameters, only one view of the calibration plane is provided - given a calibrated camera, a small number of known points imaged is sufficient to recover pose.

[1]Internal and external parameters should not be confused with "interior orientation" and "exterior orientation" which deal with the registration of scene objects to a camera centred coordinate system or a world centred coordinate system.

3.3.2 Reconstruction

Reconstruction is the process of turning 2D measurements (such as of images and projections onto plates) into 3D objects. The usual set of techniques involve finding points (or sometimes lines) in multiple 2D images, and from the locations of those features, and constraining equations derived from the model the projection of light[1], solutions are found for camera parameters as well as the 3D location of points (or one or the other). Although closed-forms exist for many of the sub-problems[2], often iterative methods are required to solve such problems, either in order to incorporate a geometric notion of error or to deal with non-linearities in the more sophisticated formulations. Such iterative methods are known as adjustments[3] in photogrammetry[4] for the purposes of efficiency when solving these non-linear systems with large number of parameters, such adjustments typically make use of the sparseness and block structure of matrices describing the linearised systems[5] (Hartley & Zisserman, 2000; Karara, 1989; Triggs, McLauchlan, Hartley, & Fitzgibbon, 2000).

The techniques of traditional reconstruction are relatively susceptible to noisy data as they were developed originally for manual labelled 2D data, and often assume all or most points have reasonable information about their 2D locations and precise details of the camera are known. However, the advent of computer vision problems has brought with it the necessity to use more robust techniques, such as robust estimators[6], and

[1]The basic example of such a constraining equation is the co-linearity equation, which expresses the fact that the projection centre, image point, and originating object point all lie on the same ray; for multiple images of the same point, for example, the resulting system of equations can be solved.

[2]Perhaps the most well-known direct calculation is the original Direct Linear Transform (DLT), where the projection equations are solved for either the entries in the projection matrix (finding the external parameters) or the 3D points of image features - but newer versions of the DLT exist that involve iteratively solving non-linear equations, for finding e.g. radial distortion (Tsai, 1987).

[3]When adjustment of camera parameters and point locations are done simultaneously, this is called bundle adjustment.

[4]Photogrammetry is the science, by decades preceding the relatively young discipline of computer vision. Photogrammetry has its own terminology though shares a lot of methods with computer vision. Photogrammetry is often divided into topographic and non-topographic forms, the former mostly related to remote sensing and the latter devoted to close-range methods - the latter of course most closely related to modern 3D computer vision.

[5] Iterative methods are used in this thesis for solving the motion estimation problem in Chapter 5, though they there employ a simplified model of image formation.

[6]RANSAC (RANdomised SAmple Consensus) is an example of a widely used robust estimation tech-

probabilistically motivated techniques.

If point features are considered, if the 3D location of the feature points ("control points") with respect to an object are known, then four points in general position in one image are sufficient to recover the pose of the object with respect to the camera (Fischler & Bolles, 1981; Hartley & Zisserman, 2000; Karara, 1989). Three points is almost always sufficient, and the task of finding pose from 3 points is called the P3P (pose from 3 points) problem: more generally, this is called the PNP problem. If the 3D position of points is not known in advance, then additional constraints, such as known vanishing points, may be exploited, but with only 2D point matches, one image is not sufficient to obtain their 3D locations - since feature points rarely are complemented with distance information, one image is generally enough only to describe rays from the camera centre along which the points must be. The problem of obtaining both the pose of a scene as well as the location of points or surfaces in it is called structure from motion and is discussed in Appendix C.

There are generalised tools for solving such problems as are posed in this section, and these are discussed next, because such approaches are also valid in the case of motion estimation in long sequences.

3.3.3 Least squares formulation

In this section, the estimation problem is formulated from the preliminaries given in Chapter 2. This formulation is common to many basic methods in 3D computer vision, as well as motion estimation, and is adapted in this thesis in Chapter 5 for use with motion estimation with a physics model. This analysis applies to the structure from motion problem as well as the motion estimation problem and many other estimation problems. Indeed, its application to estimation from one image will be described in this section[1].

nique in computer vision.

[1]The least-squares framework is feature-agnostic given that it measures a distance in observation space but observation space can consist of points, edges, templates, etc., though it does assume an

Recall the MAP formulation for trajectory reconstruction given in Section 2.6.1, ignoring the extra nuisance parameters for now:

$$p(\boldsymbol{x}_{t_{0:i}}|\boldsymbol{z}_{t_{0:i}}) = \frac{p_0(\boldsymbol{x}_{t_0}) \prod_{j=0}^{i}[p_{obs}(\boldsymbol{z}_{t_j}|\boldsymbol{x}_{t_j})] \prod_{j=1}^{i}[p_{dyn}(\boldsymbol{x}_{t_j}|\boldsymbol{x}_{t_{j-1}})]}{p(\boldsymbol{z}_{t_{0:i}})} \tag{3.1}$$

Estimating the whole sequence will be discussed in section 3.5. If the task is to estimate the state at a single time-point, then the probability of observations made at a particular time-point can be written in terms of a set of (independent) observations at that time-point[1].

$$p(\boldsymbol{x}_{t_j}|\boldsymbol{z}_{t_j}) = p(\boldsymbol{z}_{t_j}|\boldsymbol{x}_{t_j})p(\boldsymbol{z}_{t_j}|\boldsymbol{x}_{t_j}) = \frac{p(\boldsymbol{x}_{t_j})}{p(\boldsymbol{z}_{t_j})} \prod_{k=0}^{K_j} p(z_{t_j}^k|\boldsymbol{x}_{t_j}) \tag{3.2}$$

If it is assumed that the priors $p(\boldsymbol{x}_{t_j})$ and $p(\boldsymbol{z}_{t_j})$ are constant, in log form this gets:

$$log\left[p(\boldsymbol{x}_{t_j}|\boldsymbol{z}_{t_j})\right] \sim log\left[p(\boldsymbol{z}_{t_j}|\boldsymbol{x}_{t_j})\right] = \sum_{k=0}^{K} log\left[p(z_{t_j}^k|\boldsymbol{x}_{t_j})\right] \tag{3.3}$$

If every $p(z_{t_j}^k|\boldsymbol{x}_{t_j})$ is a normal distribution and all noise independent and identically distributed, and letting $f_{obs}^k(\boldsymbol{x})$ be a deterministic function picking out the mean of that normal distribution for a given $\boldsymbol{x}$, this log probability can be written as a sum of squares:

$$log[p(\boldsymbol{z}_{t_j}|\boldsymbol{x}_{t_j})] = C_{like} \sum_{k=0}^{K} (z_{t_j}^k - f_{obs}^k(\boldsymbol{x}_{t_j}, \boldsymbol{\theta}))^2 \tag{3.4}$$

Here, C_{like} is a constant. The state $\boldsymbol{x}_{t_j}$ and calibration parameters $\boldsymbol{\theta}$ together combine to produce observations $z_{t_j}^k$ which are then perturbed by Gaussian (normal) noise[2].

If the assumption of identical distribution is relaxed then each term in Equation 3.4

underlying space on which a Gaussian distribution makes sense - unit quaternions, for example, do not admit Gaussians in a natural way since quaternion space wraps around.

[1]Usually observations at a time-point are assumed to be generated by a rigid body transform from the object pose to camera view, but the feature type determines how the transform is implemented, and almost always adds extra details to the transformation model - for example, in modelling the projection of texture edges an important consideration is how the viewpoint dilates or thins the edges.

[2]Equation 3.4 uses subtraction, which not all spaces on which $\boldsymbol{z}$ lives can admit, in which case a general distance $d_{\boldsymbol{z}}$ might sometimes be definable between observations.

is weighted according to its relative variance: $w_k(z^k_{t_j} - f^k_{obs}(\boldsymbol{x}_{t_j}, \boldsymbol{\theta}))^2$. Further, if the assumption of independent noise is relaxed, then more terms are introduced with mixed terms: $w_k(z^k_{t_j} - f^k_{obs}(\boldsymbol{x}_{t_j}, \boldsymbol{\theta}))(z^l_{t_j} - f^l_{obs}(\boldsymbol{x}_{t_j}, \boldsymbol{\theta}))$ where k is not always the same as l[1]. This is called generalised linear least squares. Since this sum of terms is just a sum of 2nd order terms, it is possible to write:

$$log[p(\boldsymbol{z}_{t_j}|\boldsymbol{x}_{t_j})] = C_{like} \left(\boldsymbol{z}_{t_j} - \boldsymbol{f}_{obs}(\boldsymbol{x}_{t_j}, \boldsymbol{\theta})\right)^T S_{obs} \left(\boldsymbol{z}_{t_j} - \boldsymbol{f}_{obs}(\boldsymbol{x}_{t_j}, \boldsymbol{\theta})\right) \tag{3.5}$$

Here, S_{obs} is a matrix of weights. Where the problem is only weighted least squares rather than generalised least squares the matrix S_{obs} is diagonal. Indeed, since this derivation is based on a Gaussian assumption; S_{obs} is the inverse of a covariance matrix.

If the dynamic constraints are ignored momentarily[2], then the log value in function 3.4 or 3.5 represents the sole criterion for the goodness of a selection for parameters in terms of an observed image. A minimum in the log of the probability is a maximum in the probability. The problem to be solved then would be an optimisation problem in the parameters of the function f^k_{obs}; when considered as a function of these parameters it would be called an "objective function". Over multiple frames, the objective function would be the value of this function summed over all frames.

As a typical example of this approach, each $z^k_{t_j}$ might be the image location of a known object point. The function f^k_{obs}[3]. would then be a projection via camera pose $\boldsymbol{x}_{t_j}$[4]. If

[1]This is all a simple consequence of taking the log of the normal distribution, where the quadratic matrix term to which e is exponentiated is pulled out.

[2]When dynamics constraints are not being ignored, the most common constraint is that states between time-points should be rigid body transforms.

[3]In Chapter 5, the function f^k_{obs} is the projection of the location of a ball in a plane parallel to the image plane, to a point in the image plane. Confidence information is also obtained for each match, weighting the components of the observation likelihood mixture in Equation 3.3 and 3.4 according to the strength of histogram match. Chapter 5 further extends this model so that the dynamics consider realistic physics by incorporating a simulator, either by introducing the physics as a constraint on the parameters in the above optimisation problem or by adding a probabilistic dynamics that creates new terms in the above formula. In Chapters 6 and 7, the problem is one of object tracking, so the approach described in Section 2.6.2 where rather than estimating the whole sequence of $\boldsymbol{x}_{t_j}$, only the latest one is estimated, is more applicable. This approach will be discussed late in this chapter in Section 3.6.

[4]If the object point location is not known, f^k_{obs} might be a projection via camera pose $\boldsymbol{x}_{t_j}$ of 3D point locations $\boldsymbol{\theta}^k$.

f^k_{obs} is linear in the parameters to be estimated (and the noise is independent with fixed variance) then the problem is a linear least-squares problem which can be solved easily in a procedure similar to averaging (DeGroot, 1986)[1].

Next, this thesis looks at solution methods for motion estimation or single-frame estimation, and then goes on to describe techniques for dealing with non-normality in the noise models as well as excessive multi-modality of the objective function described.

3.3.4 Non-linear optimisation

While many techniques, such as the original Direct Linear Transform (DLT), exploit important properties of the particular problem, in particular the linearity of equations to be solved, in this thesis the problem posed is far from linear (this is because the dynamics modelled are not linear). As such, rather than dwelling further on those methods that exploit linearity, the discourse here quickly shifts to standard techniques for solving non-linear problems.

The prototypical approach for finding the maxima or minima of such general non-linear functions[2] is Newton's Method, though a plethora of iterative local approaches exist, loosely classified as either trust-region based or line-search based (Fletcher, 2000; Nocedal & S. J. Wright, 1999). With respect to global optimisation, all these methods are in the limit local methods in that they are methods for refining a given solution iteratively towards the optimal solution, though if underlying approximations are met they can jump straight to a solution[3].

Most non-linear optimisation methods rely on the ability to calculate the gradient

[1]The Kalman Filter that will be briefly discussed in the section on recursive estimation below is an elaboration of linear least squares to estimate over time.

[2]Note that discrete optimisation is not considered here, because the problems here, although sometimes having discontinuities in the objective function, are largely continuous in nature. Discrete optimisation problems do come in to computer vision, for instance in graph-cut.) and model selection (deciding which of a set of discrete models fit the data best).

[3]Of course, tailored and closed-form solutions exist for many of the problems discussed here; for example the ability to derive an object or scene pose in almost closed form from four control points, and where such solutions exist they can be used to initialise refinement using such local methods.

or gradient and Hessian[1] of the objective function at a given point. If the gradient is calculated at a particular point, a linear approximation of the objective function can be made at that point. If the Hessian is also calculated, a quadratic approximation can be made, which has the nice property of having well-defined minima or maxima that can allow the method to make steps straight to these approximating minima. Approximations to the Hessian can also be made.

Line search methods start at a previous solution and choose a direction in parameter space to search in; then they attempt to find an optimal or good solution in that direction. In this category is gradient descent, which makes a step in the direction of strongest descent of the gradient. If the Hessian is known, the direction of descent can be chosen as the direction necessary to obtain the minima of the corresponding approximating quadratic function, as with Newton's method. Other common methods include the alternating variables method where each parameter is optimised consecutively, and the conjugate gradient descent which can be thought of as the alternating variables method with the dimensions linearly remapped so as to be independent (Fletcher, 2000; Nocedal & S. J. Wright, 1999).

In contrast to line search methods, trust region methods attempt to define a region within which the polynomial or other approximation holds to within a certain tolerance, and then solve the approximating problem within that region (Fletcher, 2000; Nocedal & S. J. Wright, 1999). Thus, they are a kind of counterpoint to the line search methods in that the problem is reduced to one over a restricted region rather than one over a restricted dimension (the line).

The Levenberg-Marquadt method interpolates between the direction of strongest gradient and the direction specified by the second derivatives of the cost function and, along with the Gauss-Newton approach, is able to approximate the second derivatives by exploiting the structure of least-squares problems; it is generally considered a trust-region

[1]The gradient of an objective function is a vector describing the rate of change of the value of the objective function for each of the parameters of the objective function, while the Hessian is the matrix of the mixed and unmixed second derivatives of the objective function with respect to these parameters.

method (indeed, one of the first), in that the step size is controlled by an adaptive parameter that damps the step size by damping the Gauss-Newton equations that are solved at each step (Fletcher, 2000; Nocedal & S. J. Wright, 1999)[1].

Solving the local sub-problems in both line search and trust region methods generally involve solving quadratic or linear systems (possibly constrained). Therefore, efficiency in solving these sub-problems is all-important. Methods which exploit the sparsity of the corresponding matrices or their block structure can increase efficiency so much as to solve problems with very large numbers of variables (Karara, 1989; Nocedal & S. J. Wright, 1999; Triggs et al., 2000). Such sparsity often occurs in computer vision[2].

Also worthy of mention are the many heuristic methods, many of which are guaranteed to converge to the globally correct answer eventually[3][4] (Glover & Kochenberger, 2003; Michalewicz & Fogel, 2004). A feature that many of these methods share is the maintenance of multiple hypotheses (or other information about the problem beyond a single hypothesis) and non-local search.

3.3.5 Multiple optima, sensitivity and non-normality

As noted above, the objective function may have numerous optima. It may be particularly sensitive in some regions to changes in parameters, and completely insensitive in other regions. The errors may be badly modelled and the possibility of extreme errors may make the underlying distribution of observation error non-normal and so the least-squares likelihood inapplicable. All of these are problems for the standard techniques, and all of these are found in the scenarios investigated in this thesis.

One of the advantages conferred by the heuristic methods discussed briefly above is

[1]In Chapter 5, a simple line search as well as the Levenberg-Marquadt algorithm are used to solve the non-linear optimisation problems posed there, with similar results in this application. This algorithm is described in more detail in Appendix D.

[2]Sparsity can be found, for instance, where structure parameters are mostly independent of each other, as exploited in the bundle adjustment literature.

[3]In practice, the guarantee of global convergence is not as relevant as the practical rate of convergence and rate of success.

[4]There are a host of heuristic approaches, including evolutionary and genetic algorithms, simulated annealing, tabu search, and swarm methods.

their ability to do non-local search for the case where non local optima exist. However, random restart can also confer this advantage to the polynomial-approximation based non-linear optimisation methods described above giving a similar benefit. Moreover, the non-linear optimisation methods are often in practice initialised with a very good guess based on closed-form calculations[1][2]. This is not always possible, particularly without an in-depth analysis of the problem, and some problems are simply resistant to this kind of solution.

Even if the multiple optima problem is addressed successfully, sensitivity and non-normality remain problems. Moreover, randomisation can still miss many good optima in a high dimensional space where many optima can exist and sampling is unlikely to find them[3].

Sensitivity is a problem for all methods, however, unless there is a good way to guide search, as there is usually no way of knowing beforehand what parts of the parameter space need to be searched more carefully. Approaches aiming to overcome the problem of non-normal errors (in particular the existence of extreme errors in the data), as well as often helping to overcome the multiple optima and sensitivity problems (together, these problems constitute the most common kinds of deviation from the typical statement of the estimation problem given above), are called *robust estimation* techniques and they are described next.

[1]Heuristic methods can sometimes outperform local methods with random restart but if a good initial solution is known (as is the case where, for example, search for object pose is initialised based on a closed form PNP - pose from N points - solution, and subsequently refined), the non-linear optimisation methods are generally more desirable as they converge to a good solution very quickly by exploiting the known local structure of the problem - at least, as observed by practitioners in practical non-linear optimisation (Fletcher, 2000; Triggs et al., 2000).

[2]In computer vision, local optima can often be combatted by better posing the problems so as to make them convex so that backtracking is not necessary (Hartley & Kahl, 2009).

[3]In cases with many local optima in high-dimensional search space (the case with model-based human tracking in 3D for example), solutions involving explicitly searching from mode to mode in the objective function have found some success (Sminchisescu & Triggs, 2003, 2005).

3.4 Approaches to robust estimation

Robust estimation is a set of algorithms and techniques that build on conventional estimation techniques in order to address situations where they tend to fail, most importantly non-normality, but also multiple optima and sensitivity[1]. All of these characteristics apply to the data used in this thesis. In particular, this thesis addresses situations where typical approaches fail, and also uses observation and dynamics models that are not always correct, so robustness is a key consideration.

The problem of non-normality in the larger context could be considered as the problem of a mismatch between the model of the probabilistic distribution of data and its empirical distribution, though in practice this match is usually one between short-tailed distributions and long-tailed ones. In the case of long-tailed distributions outliers are more common than are modelled by the normal distribution, for example, which can have the effect of pushing an estimate far away from the actual value. A good practical example of this is the tendency of one or more outliers[2] to disproportionately affect the mean. A simple and well-known example of a robust alternative is the median[3]. Robust estimation applies to a broad set of areas beyond computer vision, including manual interpretation of data where it prescribes methods for outlier removal and dataset cleansing.

Robust estimation can grow out of parametric probabilistic methods, that assume that distributional information is available[4]. As long as enough observations are generated in the expected way, robust methods should continue to work efficiently. In contrast, distribution-free (non-parametric) methods generally do not assume that the probabilistic model is known in advance, or at least do not model the way data is generated using a distribution (Gibbons & Chakraborti, 2010; Hettmansperger & McKean, 2010).

[1]According to Huber (1981), whose work forms the foundation of modern robust estimation, the term "robust" is multifaceted, but takes his domain of consideration to be estimators that are robust to small deviations in assumptions, in particular distributional robustness.

[2]Outliers are observations or data-points that are so radically incorrect as to badly mislead existing estimation techniques.

[3]The median is more robust than the mean because it is not affected by values far from the median as the mean is.

[4]Knowledge of the distributions involved reflects advance meta-knowledge about the system, which can sometimes be violated with decent results still obtained, particularly with robust methods.

In Chapter 5, the focus the least-squares framework used is based on distributional assumptions. RANSAC, the main approach used there for making estimation robust, is a non-parametric approach, needing only the specification of outlier probability and desired probability of success[1]. In Chapters 6 and 7, the non-parametric particle filter is used for recursive estimation, discussed below in Section 3.6.3.

What follows is a quick list of approaches to robust estimation, starting with a discussion of a brief discussion of how robust estimators are analysed.

3.4.1 Analysis of data

A key part of many approaches to robust estimation is estimating the likelihood of any estimate provided by the estimator in use being a bad estimate (being an Outlier). For this purpose, there are several ways of re-sampling the data and recalculating the estimate based on subsampling and comparing the estimates acquired, including bootstrapping, jackknifing and cross validation[2].

3.4.2 Robust cost functions

Huber (1981) divides robust estimators up into 3 types, of which M-estimators are considered here[3]. These are based on a distributional maximum likelihood, or indeed maximum

[1]To clarify, in RANSAC, probability distributions over the continuous reconstruction parameters are not needed or employed. However, reinterpretations and extensions of RANSAC do make use of probability distributions over model parameters.

[2]A commonly used concept in robust estimation is that of "breakdown point" - usually defined as the number of outliers at which normal methods break down. After this point, robust estimators can be used, but they also have a breakdown point. The analysis of any robust technique typically includes a breakdown analysis.

[3]The other two categories of robust estimators described by Huber are:

- R-estimators: Estimators based on selecting data-points from a ranking of some function of the data (called R-estimators because they are based on rankings), which tend to create median-like estimates.
- L-estimators: Estimators based on weighting of sample data based on their ordering in a sample (called L-estimators because they are linear combinations of estimates based on individual samples), such as the Windsorized Mean approach or trimmed mean, where the mean is calculated based on a central subset of the data.

a-posteriori as is used in this thesis[1], such as least squares estimation or robustified least-squares as discussed next.

The M-estimator approach is essentially the least-squares MAP estimator discussed above, though with the observation likelihood differing from a normal distribution, usually with longer tails. Just as with the least-squares approach, which is a simple transform of a MAP distribution, the MAP of a robust probability distribution can usually be transformed into a cost function. Unlike the least squares approach, this will not be a sum of squares cost function, but is usually a sum of increasing functions of squares. M-estimators are essentially MAP estimators maximising formula 2.5; in general in robust estimation these estimators correspond to probability distributions that deviate from the normal, however. In order to deal with non-normality the usual approach is to use the least-squares solution that corresponds to the normal distribution but alter each term by passing it through a monotonic function that climbs slower at higher values (or indeed, redescends at higher values, tending to reject outliers altogether), reflecting the long-tailed nature of the underlying distribution by decreasing the weight of outliers in the cost function[2].

For non-linear functions, the common approach is again analogous to the case where a normal distribution is assumed. At a candidate solution, the gradient and/or Hessian is calculated and the polynomial approximation to the problem solved, and then this step is iterated; except that the gradient and/or Hessian are calculated off the robust cost function rather than the basic least-squares one[3] (Triggs et al., 2000).

Studentized MAP. An example of a long-tailed probability distribution is Student's

[1]M-estimators are thus called because they are based on a Maximum likelihood.

[2]One way of calculating the MAP estimate is to find the points where the MAP distribution is flat (has gradient zero), which are candidate optima. For a linear problem with a normal distribution, such a point is generally unique and can be acquired by setting the derivative of the least squares function to zero and solving (which is solving a linear system). This approach of finding zero-gradient points applies more generally to M-estimators, but only when they are tractable (that is, differentiable and the derivative solvable). If the M-estimator is based on this approach then it is called a ψ-type M-estimator. Otherwise, it is ρ-type.

[3]Interestingly, a quadratic approximation of the cost surface is equivalent to assuming that the underlying probability distribution is unimodally normal, so each step of the search for an optimum is acting as if it is solving the problem non-robustly, but over a number of steps the effect is to alter the direction of search so as to favour outliers less.

T distribution, which assumes that underlying data distribution is normal but compensates for the different distribution of residuals[1]. The transformation of the normal to the T distribution is a process called studentizing and is a first step in applying robustness, but only compensates for the properties of the residual under a normal underlying distribution - if the underlying distribution itself is not normal then studentizing is generally not sufficient[2] (Maronna, Martin, & Yohai, 2006).

Absolute deviation error. Another common approach is to forget the sum of squared error associated with the normal distribution, and use the "least absolute deviations" - that is the sum of absolute deviations between data points and data points predicted from the estimate. This approach is equivalent to the double exponential (Laplace) distribution in the same way that the sum squared error is equivalent to the normal distribution, and equivalent to the l_1 norm on the difference between data and prediction in the same way that the sum squared error is equivalent to the l_2 norm. The double exponential distribution has long tails and the absolute error estimate is generally considered robust because it does not weight the residuals of outliers higher as the sum of squared error approach does. It's main issue is in the that it does not, unlike the sum of squared errors, always induce a single optima in linear problems, and can be numerically difficult as a result.

A similar approach is the generalised spatial median, which minimises the least absolute deviations in one dimension, but in the multivariate case corresponds to an eliptically symmetrical distribution in exchange for the marginal distributions not being double exponential distributions; the corresponding probability distribution has been described and is analogous to the multivariate normal distribution, and a single cost term is the *root* squared error (Blackhall & Rotkowitz, 2008; Plungpongpun & Naik, 2008). This does have

[1]Since the sample mean follows data points and residuals are calculated from the sample mean, the residuals tend to have inflated variance in the area of the sample mean due to the effect of data points far from the sample mean.

[2]One way of forcing the T distribution to be long-tailed is forcing the degrees of freedom on which the distribution is parametrised to a low number: if the degrees of freedom are 1 this results in the Cauchy distribution.

the advantage of making the measure independent of the choice of basis for the data[1]. In applications this means that the one generally would choose to minimise the sum of root sum of squares.

Smooth M-estimators and mixture distributions. It would be useful for the estimator to behave like a least-squares estimator for inliers. As such, smooth cost functions are proposed, two of the most common being Huber's and Tukey's. Huber's behaves like the least squares cost within a defined range and smoothly transitions to the absolute deviations cost outside the defined range. Tukey's, similarly, transitions to a constant function outside of a certain range (Huber, 1981; Maronna et al., 2006).

From these cost functions can be derived corresponding theoretical distributions. A slightly more well motivated cost function is that obtained by assuming that data is derived from a mixture of normal and one other distribution, potentially a normal distribution with much larger variance, or a uniform distribution (Torr, 2002; Torr & Zisserman, 2000). This is called the contaminated normal model. This does not collapse to a simple norm when the log is taken, since a mixture of distributions results in a probability distribution that is the sum of the component distributions, which does not pass easily through a log. However, this distribution is interesting because it models local noise and outlier noise as being completely separate processes and so can be used to analyse their differing impact.

3.4.3 Algorithmic approaches

The above section assumes an algorithmic basis much the same as the standard iterated least-squares approach and alters the cost function to ensure that the cost function is sufficiently robust. However, many algorithmic improvements are possible and can address, not just robustness to data distribution, but also the problems of multiple optima and system sensitivity to parameters.

[1]Though the way that the l_1 norm is tied to the basis is sometimes highly useful in application, particularly if the aim is to minimise the number of non-exact matches (Blackhall & Rotkowitz, 2008; Candes & Tao, 2005).

The simplest approach is to use the basic iterated least squares approach, but at each step calculate which terms are likely to be outliers and re-weights terms in the cost function according to a formula that downweights outliers. This is called "Iterated Weighted Least Squares". The effect is very similar to using a robust cost function with the standard approach, and like that approach in many non-linear problems with multiple optima is sensitive to initial choice of hypothesis. Another approach ("case deletion") is to delete points considered outliers to reduce their effect on the final estimate (Hartley & Zisserman, 2000).

RANdomised SAmple Consensus (RANSAC). RANSAC (Fischler & Bolles, 1981) is the prototypical algorithmic approach to robust estimation. It not only succeeds in the presence of a large number of outliers, but also deals well with cases where cost functions have multiple optima, and handles local parameter sensitivity.

The power of RANSAC lies in its exploitation of a required sub-algorithm for quickly calculating an estimate from a small subsample of the data. So, for example, where the P3P problem can be solved, different subsamples of 3 points can be selected, a pose reconstructed and that pose tested against all of the other data-points. Each subsample of data leads to an estimate and an estimate is chosen that maximises the number of inliers according to some inlier threshold.

Of course, RANSAC needs an algorithm to solve the subproblem based on subsamples of the data so is not applicable to all problems, but this subproblem is often significantly easier to solve than the problem involving all the data. This sub-algorithm can be thought of as a deterministic function from a subset of M observations $\left\{ z_{t_i}^{k_0}, \ldots, z_{t_i}^{k_M} \right\}$ to a solution. So the nth solution is:

$$\boldsymbol{x}_{t_i}^{n} = f_{INST}(\left\{ z_{t_i}^{k_0}, \ldots, z_{t_i}^{k_M} \right\}) \tag{3.6}$$

As hinted above, the RANSAC paradigm was first introduced to deal with recovering object pose from point matches, taking advantage of the geometric solution to P3P or P4P (Fischler & Bolles, 1981), but the paradigm is general, and any feature can be used, such as edges (Armstrong & Zisserman, 1995).

RANSAC also requires the calculation of some parameters: the inlier threshold is very important, and there are also stopping threshold and max iterations. These parameters can be calculated based on a desired rate of success and outlier probability though these values are also sometimes difficult to come by.

As with the technique discussed above of restarting the refinement algorithm with a closed form solution, so does the RANSAC procedure help to defeat the problem of multiple optima, since the algorithm selectively samples across the space of solutions in a way that is likely to find a global optima if one exists and if even a relatively small proportion of data agree with it. After the RANSAC step, a refinement procedure can be applied using the inliers to fit the estimate to the data better.

The breakdown point of RANSAC is very low, since if the algorithm is forced to provide a hypothesis there is no lower limit to the number of inliers required; the hypothesis with the highest number of inliers is chosen and this number can be a small proportion of the full dataset.

Least median squares (LMS). RANSAC instantiates estimates based on subsamples of the data and evaluates them according to number of inliers. The inlier count can be thought of as a cost function, but it contributes only one bit of information for each data point.

A potential alternative approach is to score candidate hypotheses according to the residual of a data-point. This is what Least Median Squares (LMS) does (Hartley & Zisserman, 2000), by scoring a hypothesis by the median residual. It has a breakdown point of 50% since 50% of the data must be closer to the estimate than the median. As with the inlier count score, this approach has large discontinuities on the cost surface.

MLESAC and MAPSAC. Not only do RANSAC and LMS have nonsmooth cost functions, but they also throw away a lot of information about the relative benefit of different estimates. Because the cost functions are not derived explicitly from probability distributions over data, it is also difficult to include other sources of information when evaluating estimates generated during RANSAC.

A more flexible approach would be to score each estimate according to one of the robust cost functions previously discussed, so that outliers continue to score very low but the score discriminates between different inliers. Torr (2002), Torr and Zisserman (2000) follow this approach for estimating scene geometry with MLESAC and MAPSAC, whose motivation is Maximum Likelihood Estimation and Maximum A Posteriori estimation (hence the new acronyms). They use the contaminated normal distribution with a uniform outlier distribution. In this case the relative proportion of outliers and inliers can be set in advance or it too can be estimated. Generally the quickest way to do this is to find the estimate for which the mixing coefficient is maximised, though Expectation Maximisation can be used to marginalise it - which is to say find the solution that is good for the largest majority of probable values of the mixing coefficient.

Any robust function could be used in this context as is done by Fontanelli, Ricciato, and Soatto (2007), though the mixture of inliers and outliers distribution is well-motivated in view of the RANSAC algorithm's key motivation, which is to exclude outliers from the estimation calculation.

Efficiencies & improvements. RANSAC like algorithms are very efficient but for large datasets or large subproblems sampling the dataset cleverly for subsamples can be an important part of the process. Such approaches include checking candidate hypotheses against bigger subsamples rather than the whole dataset (Chum & Matas, 2002). Subsamples that are used to generate hypotheses can be generated in a smart fashion, so that once a first data-point is chosen, the subsequent data-points are chosen to be more likely to come up with a good solution - for instance, by selecting points nearby the first point, and therefore more likely to be in the category of inlier if the original point is one (Myatt, Torr, Nasuto, Bishop, & Craddock, 2002), or, conversely, by sampling points from separate buckets so that hypotheses are more likely to be well located.

Sometimes the MAPSAC and MLESAC approaches continue to throw away information; that is information about the reliability of data-points. If an image point, for example, is strongly matched to another image point, and that information is made available by the matching algorithm, the information about the match, at least about the strength of the match, should be used. The way that Tordoff and D. W. Murray (2005) use this information is in estimating the probability that each data-point is an inlier (rather than estimating it as discussed above in the context of MLESAC). This alters the impact that each data-point has on the cost function and incorporates this information from the pre-processing, which can be quite predictive of the quality of a match. See Chapter 5 for an implementation of a similar approach.

Having discussed a number of typical approaches to estimation and robust estimation in vision, the application of these approaches to sequence estimation is discussed, before the different problem of recursive estimation, or tracking, is tackled.

3.5 Sequence estimation from vision

This chapter dealt first with kinds of visual information in the form of visual features and then discussed a number of typical techniques for estimating objects and parameters in computer vision from those features. This section refines that exposition by considering the problem of sequence estimation in particular.

The problem of sequence estimation from vision was defined in Chapter 2 and is essentially the problem of deriving a sequence of states from a sequence of observations or sets of observations. The way in which this problem is tackled depends on the features chosen to be extracted from the data, the constraints placed on the motion, and the algorithmic techniques chosen for reconciling this information into a hypothesis or a set of hypotheses.

Further developing the formalism given in Chapter 2, and the least-squares approach discussed in Section 3.3.3, if an uninformative dynamics is again assumed (and maintaining the assumptions of conditional independence of observations and Markovian dynamics) then the posterior distribution over the trajectory becomes the product of the normalised distributions over each frame.

$$p(\boldsymbol{x}_{t_{0:i}}|\boldsymbol{z}_{t_{0:i}}) = \frac{p_0(\boldsymbol{x}_{t_0})\prod_{j=0}^{i}[p_{obs}(\boldsymbol{z}_{t_j}|\boldsymbol{x}_{t_j})]\prod_{j=1}^{i}[p_{dyn}(\boldsymbol{x}_{t_j}|\boldsymbol{x}_{t_{j-1}})]}{p(\boldsymbol{z}_{t_{0:i}})} \tag{3.7}$$

$$= \frac{\prod_{j=0}^{i}[p(\boldsymbol{z}_{t_j}|\boldsymbol{x}_{t_j})]}{p(\boldsymbol{z}_{t_{0:i}})} \tag{3.8}$$

Taking the log of that and assuming Gaussian observations, K independent and identically distributed observations per frame, and an uninformative observation prior[1], the following least squares formulation is obtained for trajectories:

$$log[p(\boldsymbol{x}_{t_{0:i}}|\boldsymbol{z}_{t_{0:i}})] = \sum_{j=0}^{i} log[p(\boldsymbol{z}_{t_j}|\boldsymbol{x}_{t_j})] \tag{3.9}$$

$$= \sum_{j=0}^{i} C_{like}\sum_{k=0}^{K}(z_{t_j}^{k} - f_{obs}^{k}(\boldsymbol{x}_{t_j},\boldsymbol{\theta}))^2 \tag{3.10}$$

$$= C_{traj}\sum_{j=0}^{i}\sum_{k=0}^{K}(z_{t_j}^{k} - f_{obs}^{k}(\boldsymbol{x}_{t_j},\boldsymbol{\theta}))^2 \tag{3.11}$$

$$\tag{3.12}$$

Again, it is possible to generalise this to have weighted and cross-weighted observation terms by relaxing the independent and identically distributed assumptions of the observation distributions.

This least squares form is the form of most approaches to sequence estimation from vision; another set of approaches solve equations in the parameters of interest; such approaches tend to minimise a slightly different implicit error based on the closest algebraic solution but can be faster and simpler, though some extra work generally needs to be done to ensure that constraints are satisfied (Hartley, 1998).

[1]Note that even were the observation prior informative, it would not change the least squares formulation since the term is constant in terms of the state; it is just extracted early here to simplify the derivation.

3.5.1 Motion models

Constraints may be simply a co-linearity constraint or a rigid body constraint, through to strong motion models[1]. Indeed, rigidity may only be a soft constraint (Ullman, 1983). As an example of a strong motion model, consider a constant translational velocity, no acceleration model (Young, Chellappa, & Wu, 1991). This approach works on rolling objects in particular, and objects in free flight if the mass distribution of the object is balanced so as not to precess.

This dynamics model can be generalised to consider constant acceleration in the translation (Hu & Ahuja, 1992a; Young & Chellappa, 1990), which can account for gravity in flight, or even a Taylor expansion of the motion to an arbitrary degree (Broida & Chellappa, 1991; Hu & Ahuja, 1992b; Iu & Wohn, 1990) which, though arbitrarily general, does not match the kinds of motion generally found in nature, such as bouncing, so in order to accommodate a significantly larger category of motions, a much larger set of parameters would need to be estimated.

The rotation can similarly be modelled as a constant rate of rotation with a fixed axis (Broida & Chellappa, 1991; Young, Chellappa, & Wu, 1991) as might be seen in free-flight or rolling of a balanced object. Or it could be further generalised so that rotation has a constant acceleration with a fixed axis of rotation (Hu & Ahuja, 1992a; Young & Chellappa, 1990) - there are a handful of cases where this might apply, such as the use of a motor. Rotation can also be generalised according to a Taylor expansion with a fixed axis of rotation (Hu & Ahuja, 1992b) or a varying axis of rotation to capture precession, usually obtained by considering the rate of change of all the rotation parameters (Iu & Wohn, 1990; Weng, Huang, & Ahuja, 1987), though the same issues regarding the range of motions that this naturally parametrises apply as with Taylor series translations.

When it comes to motion estimation for long image sequences with strong motion models the particular challenges are choosing motion models that best reflect the kind of

[1]In this thesis, the phrase "dynamics model" is preferred as it implies a range of models considering the dynamic behaviour of objects, including collisions, though motion model is more common in some parts of the literature where issues of dynamics are not a strong consideration.

motion found in the real world, while remaining as general as possible. Motion estimation with weak motion models on the other hand can use motion models that use no rigid constraints at all (e.g. just co-linearity of points). Such approaches can find the motion of individual points only in relation to each other, unless a fixed or known set of camera poses is used or known camera pose is used. The addition of rigid constraints simplifies the problem considerably and enables the camera motion to be estimated too with respect to the object or scene. Estimation of multiple object motions can be done with respect to the camera frame of reference or a world frame of reference if that too is known or can be estimated. Even if the structure parameters (e.g. 3D point locations) are unknown, the motion can be solved for (Hu & Ahuja, 1992a; Hu & Ahuja, 1992b).

3.5.2 Estimation of motion

Once a motion model is selected, the motion must be fit to the data. The simple rigid body and general motion models were discussed briefly in the section above on structure and motion. The following discussion applies to the strong motion models just mentioned. Note that some of the following approaches fit just motion parameters (Broida & Chellappa, 1991), while others also estimate structure (Hu & Ahuja, 1992a; Hu & Ahuja, 1992b; Iu & Wohn, 1990; Weng et al., 1987; Young & Chellappa, 1990; Young, Chellappa, & Wu, 1991).

One approach is to treat the problem as recursive estimation - that is, given an estimate of the start state, incorporate subsequent measurements in the sequence one at a time[1]. Basic techniques for this are the Kalman filter or Extended Kalman Filter (Iu & Wohn, 1990; Young & Chellappa, 1990). Filtering can be extended to “smoothing”, which, rather than simply propagating an estimate forward in time, propagates it first forward and then backward in time to adjust the earlier estimates retrospectively (Triggs et al., 2000; Young & Chellappa, 1990). For sequence estimation, this technique has the weakness that the

[1]For more information about recursive estimation, see the section on tracking and recursive estimation below.

initial estimate must be a good one otherwise the state may not converge to the correct solution, particularly if the motion and observation models are non-linear.

If the model is a polynomial type model, only the initial parameters are needed to recover the full motion solution (Hu & Ahuja, 1992a). The simpler the model is (i.e. the less expansions in the Taylor series), the less parameters are needed. If these polynomial type motion models are used with a fixed rotation axis then the resulting equations can generally be solved for the initial parameters algebraically (Hu & Ahuja, 1992a; Hu & Ahuja, 1992b), though rotation tends to bring out non-linear equations that are solvable using e.g. the non-linear least squares discussed above (Hu & Ahuja, 1992a; Hu & Ahuja, 1992b). Indeed, rotation, translation and structure parameters can often be solved independently or consecutively (Hu & Ahuja, 1992a; Hu & Ahuja, 1992b). It is possible to estimate all the parameters simultaneously, typically resulting in a batch optimisation problem as discussed above with with conjugate gradient descent being a popular solution procedure (Broida & Chellappa, 1991; Young, Chellappa, & Wu, 1991). Also, if the sequence of two-view problems can be solved first, the non-linear parameter estimation problem can be significantly simplified (Hu & Ahuja, 1992a; Hu & Ahuja, 1992b; Weng et al., 1987). This is analogous to a two-step process where the first step is segmented regression to local motion models and the second step is an attempt to fit the global motion model. The second step in such a case must be robust to problems arising in the first; the presence of bad fits in some of the subproblems.

Most of these approaches have been tested on simulated data (Iu & Wohn, 1990; Weng et al., 1987; Young & Chellappa, 1990; Young, Chellappa, & Wu, 1991) or hand labelled images with hand labelled motion or in very controlled conditions (Broida & Chellappa, 1991; Hu & Ahuja, 1992a; Hu & Ahuja, 1992b; Weng et al., 1987). For this reason, the above solutions are unfortunately not designed with robustness to real data in mind (whereas the current thesis uses a pre-collected data-set consisting of videos of real objects moving under real dynamics).

These approaches do not consider the problem of multiple objects explicitly, though this has been tackled with weaker motion models in the structure and motion literature (Hartley & Zisserman, 2000). In the tracking literature, the problem of associating measurements with objects is called data association and is discussed below.

All of the approaches discussed so far have made use of point features, probably because the geometrical analysis is simplified compared to denser methods. However, optic flow is also possible. In the absence of a strong motion model, optic flow methods typically proceed by taking the 2D optic flow constraints and applying additional constraints to the corresponding flat patches in an object; the resulting problem is generally solved in the same general way as the point match approaches (Aggarwal & Nandhakumar, 1988).

3.5.3 Model selection

If there are multiple motion models to choose between, the fit residual of each model is not sufficient to decide on which model to use, since the most general model (e.g. the polynomial expansion with the most terms) will always have the lowest residual, and so the data will always be overfit.

Hu and Ahuja (1992b), for instance, need to require the improvement in fit of the model from using a more detailed motion model to be over a predefined threshold before accepting a more detailed motion model. There is room to use modern model selection techniques on this problem. Moreover, since the benefit of polynomial models is not proven, a wider range of motion models can be possible; for example, the models used in this thesis.

As far as model selection is concerned, the literature on this continues to evolve. Traditionally, a statistical test can be used to test whether the ratio in the residual is large enough to allow the more refined model to be used, though for this to work the width of the confidence interval needs to be selected appropriately - this can be estimated from data or chosen as a “magic number” but generally it would be better to be more sensitive to context. It would be useful to prefer less parameters but again magic numbers or prior evidence are required to come up with a good answer. The Bayesian approach takes the likelihood of each model and marginalises the parameters of each model - this has the effect of producing different answers depending on the number of parameters of the model as would be desired. To calculate this naively involves integrating over these parameters for each model, which is for the most part prohibitive, so different methods exist depending on what approximations they make to produce this marginalised probability (Torr, 2002).

Within the area of motion estimation, the number of objects in the scene can be considered a model selection problem, however, associating different salient data in the scene with different objects is a difficult problem. This problem is well-studied in the context of object tracking, as discussed below, and is traditionally called data association. In motion estimation, it is most well studied in analysing surveillance footage and sports. One approach is to utilise an object classifier and use spatial constraints to reduce the space of possible object matches (Bennett, Magee, Cohn, & Hogg, 2004). Another approach is to discretise the space into an occupancy map and optimise with respect to the number of moves objects can make between locations in the map between frames (Berclaz, Fleuret, & Fua, 2009). These approaches usually explicitly assume a dynamics model involving little motion.

3.6 Recursive estimation from vision

So far this chapter has discussed the details of models used in computer vision, the visual features that can be used in reconstruction from vision, algorithmic approaches to this reconstruction, including robust approaches, and has focused in particular on estimation of long sequences of motion[1]. The last piece in the literature review in vision is to discuss the literature around recursive estimation and object tracking of objects from vision, the subject of this section.

[1]The previous section is background for the contents of Chapter 5 which deals with motion estimation, corresponding to the framework MAP sequence estimation model introduced in Chapter 2. This thesis also solves the problem of object tracking via recursive estimation, a framework for which was also introduced in the framework chapter, and which underlies the novel work described in Chapter 6.

Recall from Equation 2.9 in Chapter 2 that the task for recursive estimation is the task of finding a distribution over the current state in terms of a distribution over the previous state and new observations. A likelihood relating state to observation is generally known, along with a probabilistic dynamics relating state to successor state (again, the nuisance and simulation parameters u and $\boldsymbol{\theta}$ are ignored):

$$p(\boldsymbol{x}_{t_i}|\boldsymbol{x}_{t_{i-1}}, \boldsymbol{z}_{t_{0:i}}) = \frac{p_{obs}(\boldsymbol{z}_{t_i}|\boldsymbol{x}_{t_i})}{p(\boldsymbol{z}_{t_i})} \int_{\boldsymbol{x}_{t_{i-1}}} p_{dyn}(\boldsymbol{x}_{t_i}|\boldsymbol{x}_{t_{i-1}}) p(\boldsymbol{x}_{t_{i-1}}|\boldsymbol{z}_{t_{0:i-1}}) d\boldsymbol{x}_{t_{i-1}} \tag{3.13}$$

When choosing between different recursive estimation mechanisms, the choice is typically one of how the posterior $p(\boldsymbol{x}_{t_i}|\boldsymbol{x}_{t_{i-1}}, \boldsymbol{z}_{t_{0:i}})$ is represented and updated from dynamics and observation information.

In tracking applications, the pose, motion, general state of one or more objects is the subject of the recursion. However, in the problem of Simultaneous Localisation and Mapping (SLAM), it is both a maintenance of the pose of the observer and the structure of the environment (usually in terms of the location of landmarks) that are posed as problems. This can be considered equivalent to a single-object tracking problem with the necessity to simultaneously recover structure. Recursive estimation is generally the framework of choice for generating and analysing solutions to this problem. Note that, like tracking, SLAM is not restricted to just vision; any sensor can and usually is used, but very good vision based solutions to SLAM exist (Davison, 2003; Davison, Reid, Molton, & Stasse, 2007; Pupilli & Calway, 2005).

Recursive estimators differ depending on the nature of the observation likelihood, which is related to the features used, the model used of dynamics, how these various probability distributions are represented or approximated, and how they are updated. The structure of the current chapter will focus on the representation of probability distributions and their updating rules.

3.6.1 Propagating the best estimate

Before getting into probabilistic models of recursive estimation, it should be noted that there is a large literature of methods that do not involve propagating full probability distributions, rather taking a single estimate from a previous frame and updating that estimate in the new frame. Such methods can be powerful and take less computational power; are effectively a single frame estimation problem (that can be solved with the standard methods above or with ad-hoc methods[1]), with information about the previous estimate guiding estimation (Masson, Dhome, & Jurie, 2004; Sminchisescu & Triggs, 2003, 2005). The implicit dynamics assumption for such methods is a constant pose model with a posterior approximated by one sample.

Mean shift is a common technique in colour trackers. Given a likelihood distribution generated by the observations, mean shift is an efficient algorithm for finding the nearest mode in that distribution from a seed point (Bradski, 1998; Comaniciu et al., 2003; Naeem et al., 2006). The search for the mode typically begins close to the previously found object location or predicted one[2].

Of course, in the best of possible worlds, it should be possible to track most of the time on the basis of information extracted from individual images, only subsequently solving for continuity; only in extreme situations would more sophisticated reasoning over multiple time-steps be necessary. It turns out that this approach is successful in the presence of features that don't need to be matched locally, such as feature points (Lepetit & Fua, 2005; Özuysal, Lepetit, Fleuret, & Fua, 2006), and can be made more to exploit continuity information by updating the representation of the object to be matched depending on recent views of it (Vacchetti et al., 2004a; Vacchetti et al., 2004b). In such approaches, the dynamics is not used to guide sampling, but could in theory be used to

[1]The challenges with such methods are the same as single-frame tracking, with the additional requirement to incorporate knowledge about the position of the object at the last time point. In the latter case, the basic approach is to use the previous position as a starting point for the search for new features (Harris & Stennett, 1990).

[2]Mean shift in tracking works in the observation space, so the distribution being analysed typically does not include any information about dynamics.

later select between different possible sequences of pose estimates based on pose estimates at each frame.

Below are described methods that use the probabilistic formulation given in Chapter 2, the Gaussian based and sequential Monte Carlo approaches[1].

3.6.2 Gaussian models

The prototypical recursive filter is the Kalman filter (Baumberg & Hogg, 1994a; Kalman, 1960; Sorenson, 1970). The Kalman filter is an extension of the solution to the the linear least squares problem to an evolving probability distribution. The probability distribution over the state at time t, $p(\boldsymbol{x}_{t_i}|\boldsymbol{z}_{t_{0:i}})$ is assumed to be a Gaussian with mean $\hat{\boldsymbol{x}_{t_i}}$ and covariance Σ_{t_i}. A Gaussian probability distribution can be represented in terms of a state mean and the covariance of the state variables. The dynamics, $p_{dyn}(\boldsymbol{x}_{t_i}|\boldsymbol{x}_{t_{i-1}})$, is a Gaussian probability distribution with known covariance of a linear function F_{dyn} over state, and observations are assumed to similarly be constituted by a Gaussian distribution with known covariance of a linear function F_{obs} over state, $p_{obs}(\boldsymbol{z}_{t_i}|\boldsymbol{x}_{t_i})$. These three distributions are represented in terms of normal (Gaussian) distributions below:

$$p(\boldsymbol{x}_{t_i}|\boldsymbol{z}_{t_{0:i}}) = \frac{1}{(2\pi)^{N/2}\left|\Sigma_{t_i}^{1/2}\right|} e^{\left(\boldsymbol{x}_{t_i}-\hat{\boldsymbol{x}_{t_i}}\right)\Sigma_{t_i}^{-1}\left(\boldsymbol{x}_{t_i}-\hat{\boldsymbol{x}_{t_i}}\right)^T} \tag{3.14}$$

$$p_{dyn}(\boldsymbol{x}_{t_i}|\boldsymbol{x}_{t_{i-1}}) = \frac{1}{(2\pi)^{N/2}\left|\Sigma_{dyn}^{1/2}\right|} e^{\left(\boldsymbol{x}_{t_i}-F_{dyn}\boldsymbol{x}_{t_{i-1}}\right)^T\Sigma_{dyn}^{-1}\left(\boldsymbol{x}_{t_i}-F_{dyn}\boldsymbol{x}_{t_{i-1}}\right)} \tag{3.15}$$

$$p_{obs}(\boldsymbol{z}_{t_i}|\boldsymbol{x}_{t_i}) = \frac{1}{(2\pi)^{N/2}\left|\Sigma_{obs}^{1/2}\right|} e^{\left(\boldsymbol{z}_{t_i}-F_{obs}\boldsymbol{x}_{t_i}\right)^T\Sigma_{obs}^{-1}\left(\boldsymbol{z}_{t_i}-F_{obs}\boldsymbol{x}_{t_i}\right)} \tag{3.16}$$

Given a distribution over the current state, the observations can be incorporated in a method analogous to the weighted linear least squares problem to produce an updated estimate of state[2].

[1]In addition to the Kalman and particle filters, rather than sampling from Gaussians or using Monte Carlo sampling, grid-based representations of probability distributions have proved useful - indeed, these can be made efficient by approximating them with trees (Stenger et al., 2006).

[2]One difference with the least-squares approach is that the covariance is maintained and when new

Before update there is a simpler prediction step which predicts the distribution over current state in terms of the distribution over the previous state given the known dynamics[1]. Since everything is linear, Gaussians passed through these functions remain Gaussians. The effect of the dynamics is to propagate previous estimates forward in time[2]. The Kalman filter procedure is not described here.

The traditional Kalman filter of course only works for discrete time systems with linear observations and linear dynamics, and only with the normal (Gaussian) distribution. Non-linear functions distort the probability distribution passed through them so as to make them no longer Gaussian[3].

The basic approach to dealing with non-linearity is to use an extended Kalman filter (EKF), which is to say that the gradients of all non-linear functions are calculated and the normal linear filter applied; this is in many applications the default approach in visual tracking where the model is non-linear but the tracking wants to be smoother and temporary distractors have less effect (Davison, 2003; Davison et al., 2007; Magee, 2001; Wren & Pentland, 1998). In many cases this approach works well, but this depends on the dynamics and observation models used. Also, since the size of the matrix representing the covariance of the distribution increases in the square of variables maintained, either the number of variables needs to be kept low by judicious use of features and parametrisation (Davison et al., 2007), or by appropriate factoring of independent parts of the problem (Montemerlo, Thrun, Koller, & Wegbreit, 2002; Sim, Elinas, & Little, 2007).

Theoretically and experimentally, depending on the problem, there are many reasons to seek an improvement over the EKF for propagating Gaussians non-linearly: not only

observations are incorporated the covariance is decreased by an amount calculated according to the known covariance of observations. The way that observations are incorporated is analogous to averaging, with the addition of a linear function that allows the averaging process to be easily propagated from observed variables to state, even if the dimensionality of these spaces differ.

[1]Since the dynamics introduces noise, the calculated covariance will increase as the state estimate is passed through the dynamics.

[2]Note that the covariance over the state is never calculated from the data in the traditional Kalman filter since this is calculated from the covariances over motion and observations that are assumed known.

[3]The system described in the framework chapter of this thesis can in theory incorporate both non-linear dynamics and observations; indeed the observation models in Chapters 5 and 6 incorporate non-linear dynamics.

the shape but the variance and location of distributions can be badly estimated when the propagating function is significantly curved, and these effects can compound in the absence of continual good measurements. As such, many alternative approaches to propagating Gaussians non-linearly exist. An alternative approach is to propagate the Gaussian through non-linear dynamics and measurements by passing a small set of well-chosen "sigma" points around the Gaussian mean through these equations (Julier & Uhlmann, 1997). It turns out that to accurately model the covariance of a Gaussian, the number of points required is linear in the number of dimensions. The sigma point approach (also called an "unscented" Kalman filter - UKF - to contrast it with the less lauded EKF) may be the best trade-off for dealing with non-linearity if the distribution remains unimodal and if the short-tailed Gaussian distribution is assumed. Using judicious sampling rather than representing the covariance matrix directly can also be of help in reducing the size of the problem (Evensen, 2003). More generally, the problem of propagating a Gaussian through a non-linear function can be considered as one of integrating the gaussian with the function, an approach that can be solved with other integral approximations (Ito & Xiong, 2000).

Despite the effectiveness of the above methods, non-linear functions do tend to cause bifurcations and the single Gaussian approach does not allow for the multiple modes that would be necessary to express multiple hypotheses. An alternative is to express the state itself as a mixture of Gaussians, which both solves the problem of non-linearity and of multiple modality (Alspach & Sorenson, 1972; Ito & Xiong, 2000; Sorenson & Alspach, 1971). A mixture of Gaussians can in theory approximate any distribution arbitrarily well. More robust distributions can be dealt with by using contaminated normal models made from multiple Gaussians, or mixtures approximating more robust distributions (Blackhall & Rotkowitz, 2008), and non-linearity can be dealt with by passing each Gaussian in the mixture in the mixture through an EKF.

Another common extension is to add an iterative component to the Kalman filter in order to further refine the estimate at each time point. When the basic filter is an

Extended Kalman filter, this is called the Iterated Extended Kalman Filter. Since the update phase of a Kalman filter is a linear least squares problem, this iteration phase is essentially non-linear minimisation in the same vein as Newton's method described above in the section on non-linear estimation (Geeter, Brussel, Schutter, & Decreton, 1997; Koller, Daniilidis, & Nagel, 1993), though can be done in conjunction, for example, with an sigma point update (Cao, Ohnishi, Takeuchi, Matsumoto, & Kudo, 2007). This iterative procedure is not theoretically optimal but can further help overcome non-linearities in the EKF. Another use of iteration is successive relaxation of hard constraints (Decarlo & Metaxas, 2000).

Although any distribution can in theory be approximated by mixtures of Gaussians, the shape of a finite mixture is typically characterised by the smooth local shape of each Gaussian. However, there may be some rigid constraints on the state space of the propagation. In order to comply with constraints, the mean of each Gaussian can be moved into the part of the state space[1] where the constraint is satisfied. This is easy for linear constraints. Non-linear constraints can be linearised but it is better to apply them weakly and repeatedly in order to reduce the buildup of error (Geeter et al., 1997). If a sigma point filter is used, each sigma point can be constrained, which is better because this forces more of the Gaussian inside the constrained area (Kandepu, Foss, & Imsland, 2008). An alternative approach is to use truncated Gaussians, which are Gaussians with parts of state space set to zero probability arbitrarily (Brankart, 2006). In this way, constraints can be dealt with by truncating the probability distribution off from the forbidden part of the state space. Unfortunately, only linear or very well defined constraints are easily dealt with by this method since propagation of non-linear constraints through a non-linear state evolution function quickly can make them intractable. As will be seen in the background chapter on physical simulation 4 below, physical models to tend to employ constraints; for example interpenetration and energy conservation constraints, so dealing with these

[1] "State space" being the theoretical continuous and discrete space containing all possible states; the state space is the same across all time-steps but at any given time the system only inhabits one of these states - note that this is the dynamical systems definition of a state space rather than the pure non-stochastic probability definition

is important.

Rather than assuming a discrete stochastic system, the continuous time Kalman filter is based on differential equations governing the state evolution, thereby requiring the state and observations to be expressible as functions with a continuous domain of time (Blake et al., 1993; Metaxas & Terzopoulos, 1993). The methods used to solve the resulting differential equations are very similar to those used in simulation as will be discussed in Chapter 4; indeed, this overlap has been exploited previously in physical estimation problems (Metaxas & Terzopoulos, 1993). The additional element to normal integration of differential equations is the inclusion of noise and measurement terms in the differential equations and a higher order matrix differential equation governing the state covariance. The basic continuous time Kalman filter is based on a linear differential equation in the state but non-linear approximations are generally handled as with the extended Kalman filter, and simplifications such as state decoupling generally made for efficient integration (Metaxas, 1997; Metaxas & Terzopoulos, 1993).

Because of its relative simplicity in application, its computational efficiency, and age, the Kalman filter is traditionally the first method applied to any tracking problem, and a Kalman filter exists for all of the visual features discussed above and their many combinations; for example, contours (Baumberg & Hogg, 1994a; Kakadiaris et al., 1994), blobs (Magee, 2001; Wren & Pentland, 1998), edge segments (Chan, Metaxas, & Dickinson, 1994; Koller et al., 1993), optical flow (Decarlo & Metaxas, 2000), and point features and template matching guided by the filter predictions (Davison et al., 2007). Because of its ubiquity, a Kalman filter is often applied as a last step to smooth a solution.

3.6.3 Particle filter

The use of Gaussians is powerful because the parametric Gaussian distribution allows for a compact representation of a probability distribution, and the Gaussian is more tractable to analysis than many of its peers. Chapter 6 describes a specific implementation of the particle filter in more detail.

However, because it is a parametric distribution, not all dynamics and observation functions can easily be dealt with; in particular, non-linearity is something that is dealt with through extensions to mixtures and deterministic sampling of the Gaussians as discussed above. An infinite number of hypotheses can be dealt with with parametric approaches, but, particularly in high dimensions, there is a trade-off between accuracy of the estimate and the coverage of the space of hypotheses[1]. Mixtures can help solve this but the hypothesis structure is typically dependent on the design of the mixture model[2].

Another approach to dealing with non-linearity, and also for a natural way of handling multiple hypotheses, is to approximate probability distributions by sets of random samples. This is the approach taken by Monte Carlo methods. In Monte Carlo methods of estimation, a set of discrete samples is evaluated on known probability distributions. Sequential Monte Carlo is the set of methods where a sequence of such sample sets is generated, as is the case in the tracking or motion estimation problems where the probability distribution changes with time.

Such approaches are also known by the term “particle filter”, since they are recursive estimators like the Kalman filter, though here the probability distributions are approximated by samples rather than parametric distributions, and those samples are called particles. Particles can be simply hypotheses, but they can carry other information - in particular, weights corresponding to the point mass of the probability distribution. Although the term “sequential Monte Carlo” describes more precisely the concept behind the method, particle filter is the more common term.

As with all recursive filters, a particle filter takes a distribution over the current state, approximated by a set of, possibly weighted, particles (hypotheses), and processes it along with any new observations and the dynamics model to create a posterior distribution over the subsequent state, again approximated by a set of particles. Usually this follows the

[1]Multiple object tracking in particular can benefit a lot from proper handling of multiple hypotheses. See below for a more extensive discussion of multiple hypothesis tracking.

[2]Though methods exist for fitting mixtures automatically.

predict-update cycle, as does the Kalman filter, where the dynamics model is used to propagate hypothesis particles individually to create a prior distribution (approximated by an intermediate set of hypothesis particles), which is used as a proposal distribution and then this processed by using the observations and likelihood model to re-weight or re-sample the hypothesis particles, usually one by one. However, this proposal distribution does not need to be the result of dynamics model prediction; in fact the best proposal distribution would be the same as the posterior, taking into account both dynamics and observations; however, this distribution usually cannot be sampled from directly; however, it is still very useful to use proposal distributions closer to the posterior so that parts of the state space with low prior probability but high likelihood are sampled from (Pitt & Shephard, 1999; van der Merwe, Doucet, Freitas, & Wan, 2001).

In order to convert the proposal distribution to the posterior, one approach is to weight the particles according to observation likelihood. This set of weighted particles together can represent the posterior, but they are typically re-sampled so that the discrete approximating domain of the posterior more evenly covers the mass of the distribution. Alternative approaches are to accept or reject generated particles with probability related to the likelihood (Pitt & Shephard, 1999).

Because of its generate-and-test approach, the only problem-specific information required by the basic particle filter is generally an arbitrary dynamics model and likelihood. This generality means that basic particle filters are pairable with any of the features discussed above, such as contours (Heap & Hogg, 1998; Isard & Blake, 1998a, 1998b; MacCormick & Blake, 2000), edges, textures and junctions (Klein & D. W. Murray, 2006; Mörwald, Zillich, et al., 2009; Pupilli & Calway, 2006), point features (Pupilli & Calway, 2005), colour (Nummiaro et al., 2002), combinations of texture and colour (Mihaylova et al., 2007), and so forth. As such, particle filters are quickly coming to rival Kalman filters in popularity in the areas of robotics and computer vision.

As with many approaches, the particle filter does not scale well with dimensionality. As such, the most common set of extensions to the particle filter revolve around analysis of tractable parts of the problem to reduce the dimensionality of the particle filter part of the solution, by exploiting independence between different parts of the state (MacCormick & Blake, 2000; Tweed & Calway, 2002), or factoring out Kalman-type processes for part of the problem (Montemerlo et al., 2002; Sim, Elinas, Griffin, & Little, 2005; Sim, Elinas, & Little, 2007).

The particle filter also has a problem that if the dynamics or observations are not modelled as noisy enough, the finite number of particles will converge on a single hypothesis, which is a situation that is difficult to recover from. A common approach is to add artificial process noise, but that has its own problems[1]. A variety of methods exist for keeping the particle filter multimodal, again generally by using the particle filter in conjunction with other methods, for example using a mixture model to ensure that there are multiple modes in the distribution and re-sampling within each component (Vermaak, Doucet, & Perez, 2003), or by using mode-finding[2] to find modes in the posterior or likelihood distributions and so concentrate the particles where they are the most beneficial (Chang, Ansari, & Khokhar, 2005).

Indeed, this last approach, which involves using a kernel to approximate the distribution so that mode-finding methods can work, involves dispensing, at least during part of the process, with the theoretical finite domain particle filter and treating the distribution as parametric. Such approaches can also serve to keep the proposal distribution as close to the posterior as possible, or at least to reflect the likelihood, so that the sampling process will place samples in areas where the posterior has mass. A common approach is to use parametric methods to incorporate observations efficiently into the proposal, by approximating the particle distribution at the previous time-point by a parametric Gaussian distribution, incorporating both dynamics and observations into that distribution and using it as a proposal distribution for the particle filter (Cao et al., 2007; van der Merwe, Doucet, et al., 2001). This can be extended to mixture distributions to

[1]Apart from the theoretical difficulties, artificial process noise might be an issue, for example, when trying to exploit a good dynamics model as is done in Chapter 6.

[2]Mean-shift is the the typical algorithm for mode-finding, though to use it the discrete particle distribution must be approximated with a continuous one.

deal better with non-linearity and non-Gaussian distributions (van der Merwe & Wan, 2003). Another advantage of augmenting particles with distributions is that parts of the state space outside of the finite distribution are covered (van der Merwe, Doucet, et al., 2001). Conversely, rather than using a Gaussian update to help with the propagation of the particles, it has been proposed that a Gaussian filter use the sequential Monte Carlo method to propagate the parametric distribution by sampling from the distribution at the previous time step and propagating the samples through the dynamics and observations and then reconstructing the Gaussian from this data (Djuric & Kotecha, 2003; Kotecha & Djurić, 2003).

Just as Kalman filters have iterative extensions to deal with non-linearity in the observation model, so too do particle filters have iterative extensions to deal with the likely possibility that the first set of particles projected at a given time step do not optimally fit the likelihood (or posterior) modes. It has been proposed to re-iterate the re-weighting/re-sampling step (Deutscher, Blake, & Reid, 2000; Mörwald, Zillich, et al., 2009), do an additional local optimisation step on some particles (Heap & Hogg, 1998), or to use a Markov transition kernel[1] to better approximate the posterior after re-sampling[2] (van der Merwe, Doucet, et al., 2001).

As with Kalman filtering, there is a continuous time version of the particle filter, mostly to deal with the problem of asynchronous measurements (Ng et al., 2005); because the form of the covariance need not be analysed this becomes a lot simpler. Since arbitrary functions and distributions are by default available to the particle filter, the problem of continuous time is not so pronounced since the functions and distributions themselves can model the time element.

An important observation is that constraints are dealt with quite naturally by particle filters; states outside of the constrained areas will simply not receive samples[3]. As well as

[1]A Markov transition kernel is a kind of probabilistic search guaranteed to maintain the existing the distribution by virtue of appropriate selection of the transition probabilities.

[2]If parametric approximations are made then the parametric process can be repeated too, as with the Iterated Kalman filter and relatives (Cao et al., 2007).

[3]The observation that constrained areas are not sampled in a particle filter does not apply when parametric approximations are employed.

their natural application to non-linearity, multiple objects, and non-Gaussian processes, this makes the particle well-suited to complex physical-simulation as is used in this thesis, with the added advantage that little analysis of the dynamics is required. In the long term, however, the plain particle filter suffers from problems with dimensionality - as the number of dimensions in the state to be tracked increase, the number of samples required to cover it as well increases non-linearly.

3.6.4 Moving horizon estimation

The recursive estimation and motion estimation problems can be considered sub-types of the moving horizon estimation problem; as is guessable from the title, moving horizon estimation tries to solve the motion estimation problem for the last n frames, with the help of additional information carried over from the previous limited horizon problem, analogous to the distribution carried forward by recursive estimation. A moving horizon estimation (MHE) framework is a good framework for adapting solutions of the problem of motion estimation to recursive estimation.

3.6.5 Simpler motion models

Although this thesis focuses on the integration of a sophisticated physics predictive dynamics model into the tracking task, various dynamics or motion models are indeed implicit in existing tracking work. Mean-shift tracking, for example, is based on the assumption that the mode of the observation likelihood does not move too far from frame to frame (Bradski, 1998; Comaniciu et al., 2003; Naeem et al., 2006); an assumption effectively of zero displacement with noise. In the Kalman filter, a linear motion model must be specified. This is typically specified as the identity function, again assuming zero displacement with noise (Baumberg & Hogg, 1994a; Davison et al., 2007). This can be averaged with a constant velocity predictor if velocity is maintained or if two frames are used to predict velocity[1]. Although particle filters can in theory embody any arbitrary motion model, in practice they tend to also assume constant displacement, with a normal uniform or long-tailed distribution around it (Isard & Blake, 1998a, 1998b; MacCormick & Blake, 2000; Pupilli & Calway, 2005). In the motion estimation literature (see Section 3.5) constant velocity or acceleration type motion models are sometimes employed, as well as low-velocity (constant displacement) motion models. Learned motion models are also common (Baumberg & Hogg, 1994b; Heap & Hogg, 1998; Magee, 2001; Mörwald, Kopicki, et al., 2011; Tweed & Calway, 2002) and more sophisticated models, including those based on physics understanding, will be discussed in Section 4.2.

3.6.6 Multiple objects

Tracking multiple objects brings with it an additional set of problems not seen with single-object tracking. At the simplest level, objects generate observations but it if observations are ambiguous it is not always clear which object (or background noise) is generating them - for example, an edge may belong to any of the target objects or the background. This is the problem of data assocation. Data association is less of a problem in vision than other tracking mobilities since in vision objects often have distinct signatures - however, it is not always simple to process these signatures adequately, and many real-world situations do exist where objects are ambiguous. For example, when using colour cues, multiple objects might have the same colour.

Multiple objects also occlude each other. Occlusion is a problem for single object tracking too, where it is other scene objects that occlude the target object. In visual tracking, partial occlusion can occur, and objects can overlap. Using models of objects almost always helps in these cases, giving the ability to reason in terms of occluding contours and so forth.

Also, the dynamics of each object can be affected by other objects.

[1]In general a combination of velocity and displacement motion models might best be modelled as a mixture of these two models rather than an average, since dynamics would generally be one or the other rather than both simultaneously, in which case the basic Kalman filter is not easily applicable.

The classic approaches[1]. to dealing with multiple objects include probabilistic data association and multiple hypothesis tracking (Blackman, 2004).

Multiple hypothesis tracking involves forming a hypothesis for every possible combination of matches between observations and objects and maintaining these new hypotheses over time (with appropriate pruning to reduce the set of hypotheses to the most likely). For objects that produce a very few number of observations this can be tractable, though in computer vision often a large amount of data needs to be dealt with; some features can be reduced to such a small set but not all (Blackman, 2004).

Probabilistic data association divides the effect of multiple observations between targets depending on the closeness of the observation to each target. However, it does deal with each target separately and this has the effect of bringing all objects closer to ambiguous and false observations. Joint probabilistic data association tries to remedy this by considering the space of joint associations (so that the filter takes account of the fact that an observation is not generally generated by two targets). Properties of visual features can be exploited directly in the observation space, for example by reasoning over the kinds of occlusions expected, and known physical properties of objects, for example, their jointed nature, can be exploited to reduce the space of possibilities (Rasmussen & Hager, 1998).

Sometimes it is possible to say that part of the problem can be handled jointly but part of the problem can not. In computer vision, models of image formation typically show a lot of interdependence - for example, even if only two objects are in a scene, in real situations they are often very likely to overlap. On the other hand, because object dynamics are relatively decoupled, or at least not well known, it can be much more efficient to reason about the movement of objects independently, though reason about their appearance jointly (Lanz, 2006). The desire to move beyond this and start to reason about joint physical constraints of objects is one of the motivations behind the current

[1]Of course, the most straightforward and computationally simple approach to data association is to greedily select associations between observations and objects, for example with a nearest-neighbour search. Clearly this can quickly lead to degenerative behaviour.

thesis[1].

Both of these approaches are mostly applicable to Kalman filter style methods, though can be altered to deal with sampling methods and multiple modalities (Rasmussen & Hager, 1998). Particle filters have two natural ways of dealing with multiple objects: either extending the state space to make a joint tracker, or assuming each object independent except where necessary. The latter approach tends to reduce computational complexity but the former approach is much simpler analytically speaking (Chang et al., 2005; MacCormick & Blake, 2000; Tweed & Calway, 2002; Vermaak et al., 2003).

3.7 Evaluating motion estimation and tracking

When evaluating estimation algorithms it is always useful to know the ground truth. For experiments in simulation this is not a problem as the true trajectory is well known (Iu & Wohn, 1990; Weng et al., 1987; Young & Chellappa, 1990; Young, Chellappa, & Wu, 1991). Experiments can be done in real situations where the scenario is manipulated by hand (Broida & Chellappa, 1991; Hu & Ahuja, 1992a), or automatically (Stolkin, Greig, & Gilby, 2005). Perhaps the most flexible approach to obtaining ground truth is to instrument the scene, for example with motion sensors, highly calibrated multiple cameras, gyros, and so forth (Balan, Sigal, & Black, 2005).

However, this information is not always available in real-world scenarios, or where the equipment is not available. This is particularly relevant in the work described in this thesis where the aim is to evaluate the performance of motion estimation and tracking algorithms on real object motion.

A commonly used alternative approach is simply to run the algorithm and check by eye for major tracking failures. For a quantitative comparison it may be possible to count the frame at which the tracker fails or count the number of failed videos. For the most

[1]Efficiently reasoning about joint physical constraints is not done in a general way in this thesis; it is further work. The point being made is that the work in this thesis is an foray in that direction.

part this approach works well, though when trying to deal with numerical differences in performance other methods are necessary (Needham & Boyle, 2003).

To get numerical results in sampling trackers, one approach is to run the tracker a lot and assert that the resulting estimate is close enough to the real estimate to make for a good comparison[1] (Mörwald, Kopicki, et al., 2011); this is a problem for monocular vision though as some parts of the estimate (e.g. depth) are inherently very variable. To solve this, a rig can be set up with multiple cameras and hand-assisted bundle adjustments run to find the best parameters; such an approach is probably the best that can be done in such a case. Where time or space do not allow for this setup, the author has found it mostly sufficient to compare the image projections of estimated object location to the hand-labelled image projection, thus doing the comparison in image space (Duff, Mörwald, et al., 2011). This approach does tend not to assign a large weight to errors in naturally ambiguous dimensions (such as depth); this is both a benefit (since monocular algorithms are not expected to do so well there) and a problem (since it does not challenge them to do better). Indeed, it is also common to cross-check results from one approach or modality with results from other modalities, sometimes human-assisted (Erdem, Tekalp, & Sankur, 2001; Erdem, Sankur, & Tekalp, 2004).

How distance-from-match should be evaluated numerically is also a cause of some discussion, with considerations focusing on which elements of tracking failure should be emphasised and what happens when part of an object is occluded (Erdem, Tekalp, et al., 2001; Erdem, Sankur, et al., 2004). In this thesis, when ground truth is known, for 3D poses, the continuous space of point mismatches within an object is integrated over; for a set of 2D points, a simple sum of squares error is sufficient (Duff, Wyatt, et al., 2010), and for image projections of 3D objects, squared distance in image space is added over a set of distinctive points (Duff, Mörwald, et al., 2011). A plethora of other methods exist that deal with the situation where an object is not in the scene and many different

[1]The idea behind running a sampling-based tracker a lot is that it is possible to be reasonably certain that most of the state-space has been explored; as a consequences, such approaches are really tests of the sampling strategy more than anything else.

statistics should be collected to characterise performance in different ways - including frames properly tracked, false track, track failure, track fragmentation, latency, total overlap, target swapping, track closeness and match (Yin, Makris, & Velastin, 2007); a lot of the counting methods use thresholds for example to determine whether a frame is tracking or not; it is not always clear what this threshold should be so either an ROC style analysis needs to be done or pure continuous measurement approaches used (not always possible if a target really does disappear, though even invisible targets could be tracked to a degree). Since point trajectories can be similar but displaced in time or space, it may be possible to match the absolute time and distance displacement between trajectories, or even to ignore the time element altogether, though how rotating objects might be dealt with is further work (Needham & Boyle, 2003).

CHAPTER 4

BACKGROUND: PHYSICAL SIMULATION AND ESTIMATION

"What happens if a big asteroid hits Earth? Judging from realistic simulations involving a sledge hammer and a common laboratory frog, we can assume it will be pretty bad."

Ascribed to Dave Barry

The previous chapter dealt with the background of techniques for motion estimation and computer vision. This thesis is about addressing the same problems with the added help of physical cognition in the form of off-the-shelf physical simulation.

This chapter will proceed in two parts. It will first deal with the theory and practice of modern real-time rigid body physical simulation focus on that used in games and demos, as well as a brief discussion of alternatives. Then it will go on to discuss other approaches to visual estimation using physics.

4.1 Physical simulation

Physical simulation is the task of modelling the change of physical systems over time; given a full description of the system at a certain time and a model of its dynamics, predicting its evolution over time. Actually, many simulations do not use prediction as a criterion for success since the aim is to appear realistic to humans, in which case plausibility is the more often used criterion. This is true of the simulator exploited in this thesis, PhysX[1] ("PhysX Features," 2010), based on the presumption that if a physical simulator embodies some information about a system, it should be possible to exploit that information to do better tracking.

Recall that in Chapter 2, the deterministic simulation function $\boldsymbol{f}_{dyn}()$ was defined as:

$$\begin{aligned} \boldsymbol{x}_{t_{i+1}} &= \boldsymbol{x}_{t_i+\Delta t_i} && (4.1) \\ &= \boldsymbol{f}_{dyn}(\boldsymbol{x}_{t_i}, \boldsymbol{\theta}, \boldsymbol{u}_{t_i}, \Delta t_i) && (4.2) \end{aligned}$$

For simple systems, such as point masses moving in the presence of gravity, it is often sufficient to specify the equations of motion $\boldsymbol{f}_{dyn}()$ and initial state $\boldsymbol{x}_{t_i}$ and solve them analytically for $\boldsymbol{x}_{t_{i+1}}$. For more general simulators, however, including collisions and motions specified as difficult differential equations, possibly with constraints, generally what is required is a mixture of collision detection and numerical integration.

In summary what happens is this: The equations of motion are integrated forward in time, with constraints applied if objects are touching or for joints and so forth, and at each time-step the system is checked for discrete changes in configuration - in particular, collisions. If a change is detected then the integrator is updated, for example, with the introduction of new constraints, forces, and impulses based on object contact and run again (Witkin, Baraff, & Kass, 1997).

Since collision detection is performed at discrete intervals, clearly this is a kind of approximation, but it works if objects are not moving too fast[2]. If objects are moving too fast then collision detection must be solved across the time as well as spatial domain and

[1]If the reader is wondering why PhysX (previously NovodeX) was chosen as the simulator of choice in this work, particularly in light of its closed sourced nature where many open source alternatives exist, both used in games and in research and education ("Bullet Physics Library," 2011; "CRISIS Physics Library," 2011; "Newton Game Dynamics," 2011; "Open Dynamics Engine Manual," 2011; "PhysBAM," 2011), that reader can be assured that the choice is arbitrary. PhysX is indeed fast, robust and easy to learn, though other simulators share these characteristics; it is because it was already in use by colleagues of the author. The use of another physics engine is not anticipated to result in significantly better performance.

[2]Too fast in this context is a velocity that would make an object cross over several different collision configurations in one time-step.

so becomes more complex, to implement and computationally. This is called continuous collision detection (or CCD). In most typical applications this is not required.

Once collision detection finds a contact, it may be that objects are already penetrating. At this point it could be possible to wind back time to close to the point of collision but again that is a hard problem to get exactly right, and deferred recovery over time can lead to visible jitter, so objects are often allowed to interpenetrate to various degrees.

There is a lot of variety in the way that these steps are carried out and the different approaches will be discussed below. But first, a word about representation of the state that is being propagated forward in time here, then a discussion of the mechanics behind the integration of physical systems, and collisions. Ample tutorials exist already about the basics of real-time physical simulation (Baraff, 1993; Catto, 2005; Eberly, 2010; Lengyel, 2003; McShaffry, 2009; Witkin, Baraff, & Kass, 1997) and robotic mechanics (R. M. Murray, Li, & Sastry, 1994), so it will only be covered in as much detail here as is required to elucidate the peculiarities of the PhysX physical simulation engine.

4.1.1 Representation

Because the scenarios addressed in this thesis generally fit the assumptions of rigid body motion, the focus in this section is on rigid body motion. A rigid transform translates and rotates a body (for a tutorial in the context of the visual apparatus see appendix A). A rigid body is a body whose component displacements are only ever in terms of the rigid transform governing the whole body - that is, parts of the body do not move independently[1]. These constraints allow the pose of the whole object to be represented by a small number of variables. A rigid transform is typically defined in terms of the transform of the object reference frame; the pose of an object is the transform of this frame with respect to the world reference frame. The centre of the coordinate system in

[1]The rigid constraint can be defined in a number of ways, for example, by requiring that the distances between particles in the body remain fixed (R. M. Murray et al., 1994). This constraint is sufficient if it is assumed that the particles making up the body can only move continuously in 3D space; alternatively, a handedness preserving criterion could be used, such as conservation of cross product relationships.

the object reference frame is arbitrary as long as it is treated consistently, though the centre of mass is a good choice because in free-flight motion it is about the centre of mass that the object will rotate (R. M. Murray et al., 1994; Witkin & Baraff, 1997).

The pose of a rigid object can be approximated by 6 numbers (3 for 3D location, and 3 for rotation in Euler angle or axis-angle format), though the easiest representation to apply to points contains a full rotation matrix representing rotation $R_{O/W}$ together with a linear translation $\boldsymbol{T}_{O/W}$ as these can be directly applied to coordinates as transforms. Axis-angle is possibly the most intuitive representation as it defines the axis through which the rotation occurs and the amount of the rotation and can be represented as one 3D vector with the length encoding the angle of rotation (called exponential coordinates)[1]. To obtain a rotation matrix from these representations involves sine and cosine functions, which is not always optimal; and for this reason quaternions are often used instead; unit quaternions are like angle-axis representation with the sines and cosines already taken, and can for this reason be applied directly to coordinates to transform them[2]. Quaternions are common to use in the internals of physical simulation and also estimation (Catto, 2005; Metaxas, 1997; Popovic, 2001; Witkin & Baraff, 1997); though full rotation matrices can often be used just as efficiently, they can easily drift from pure rotations[3] (normalising quaternions is less costly computationally). If an object's translation is $\boldsymbol{T}$ and quaternion $\boldsymbol{q}$, its pose $\boldsymbol{x}$ can therefore be written as a 7 dimensional quantity $\boldsymbol{x} = \left[\begin{smallmatrix}\boldsymbol{T}\\ \boldsymbol{q}\end{smallmatrix}\right]$.

Clearly the pose of an object is alone not sufficient to represent its dynamic state, which should include its instantaneous velocity too[4]. The translational velocity, that is the rate of change of the object's local coordinate system, can be represented as the set of rates of change of each of the basis vectors in the global coordinate system. However, angular velocity is not so straightforward. At each time point it is possible to define the

[1]To obtain three number rotations, Euler numbers (3 subsequent rotations along separate axes) are possible but these have problems with singularities and are not always straightforward to understand.

[2]Quaternions are used as a representation for rotation in the estimation part of Chapter 6.

[3]Moreover, in the context of estimation, there are too many degrees of freedom in a rotation matrix; this parameter drift can dominate the process and stop all progress.

[4]An alternative representation involves using linear and angular momentum instead of linear and angular velocity (Witkin & Baraff, 1997); each approach has consequences.

location of any point on the object in terms of the current rotation of the object frame with respect to the world frame and its linear displacement ($\boldsymbol{Y}_{Wld} = \boldsymbol{T}_{O/W} + R_{O/W}\boldsymbol{Y}_{Obj}$). If the rate of change due to the translation in the object coordinate frame with respect to the world frame is subtracted, the movement that remains to a particle is to orbit at a fixed distance around the centre of the object coordinate system (since distances are fixed in rigid bodies). Therefore, the points' motion must be perpendicular to the vector reaching it from the origin of the coordinate frame. So for all particles in the object there is a unique vector perpendicular to both of these vectors (and parallel to the rotation axis) that can be used to define the movement of all of the particles[1] (R. M. Murray et al., 1994; Witkin & Baraff, 1997).

Defining this angular velocity vector as $\boldsymbol{\omega}$ and the translation rate of change (velocity) as $\boldsymbol{v}$, the instantaneous state of a fully dynamic object might be written as $\boldsymbol{x} = \begin{bmatrix} \boldsymbol{T} \\ \boldsymbol{q} \\ \boldsymbol{v} \\ \boldsymbol{\omega} \end{bmatrix}$, 13 numbers[2].

Just as the rigid constraint allows the motion of all the particles to be written in a very compressed way, so too does it allow the mass distribution and momentum of all the particles to be written compactly. The translational momentum ρ of an object is the sum of the translational momentum of the particles aligned with the movement of the object centre, and the angular momentum L is what is left over (and subsequently each quantity is conserved in the absence of external forces). The translational momentum carries the relationship with mass and velocity as individual particles: $\rho = m\boldsymbol{v}$. Since the angular momentum is therefore related only to the angular velocity and mass of all of the particles, a relationship can be found between angular momentum and velocity in terms of the mass distribution over the object. It turns out that this relationship can be found by integrating the mass distribution over the object and obtaining an inertia tensor, here

[1]Note that the rate of change of the rotation matrix can be easily calculated by calculating what happens to the basis vectors making up the rotation matrix (R. M. Murray et al., 1994; Witkin & Baraff, 1997).

[2]If a more compact representation were chosen, with the rotation in angle-axis (exponential) coordinates the full state might be $\boldsymbol{x} = \begin{bmatrix} \boldsymbol{T} \\ \boldsymbol{\theta} \\ \boldsymbol{v} \\ \boldsymbol{\omega} \end{bmatrix}$ 12 numbers, or if the full rotation matrix were used, it might be $\boldsymbol{x} = \begin{bmatrix} \boldsymbol{T} \\ \hat{R} \\ \boldsymbol{v} \\ \boldsymbol{\omega} \end{bmatrix}$ 18 numbers.

called I, such that $\boldsymbol{L} = I\boldsymbol{\omega}$ (R. M. Murray et al., 1994; Witkin & Baraff, 1997). With respect to the object's own frame of reference this remains constant, but to be useful it is typically calculated in the world frame of reference and hence can change as the object rotates. The relationship between angular velocity and angular momentum (also typically calculated in world coordinate frame) therefore can change; since the momentum of all the individual particles must be conserved, the angular momentum of the whole object is conserved, and the angular velocity can oscillate - this is called precession. Of course, this effect is small and so it is possible to ignore this precession and get reasonable looking results in terrestrial situations - indeed, experiments show that the physics engine used in this thesis - PhysX - does make the assumption that translational velocity rather than momentum is conserved. Considering that simulation typically involves linearisation of velocity while solving for motion (rather than momentum) parameters, clearly having a constant velocity can lead to more accuracy and stability in the simulation (Catto, 2005; Duff, 2011a).

The shape of the body can be approximated by a set of parametric surfaces, a composition of polygonal primitives, a polyhedral mesh, or, as is done in this thesis, by an arbitrary trimesh[1]. Convexity of the trimesh or shape is often invoked to reduce the complexity of collision detection and resolution[2]. Since conducting intersection searches for planes or surfaces from two objects can be quite expensive computationally, often the object is associated with an approximating shape, typically a box. This box can be axis aligned (AABB) or it can be oriented with the body (OBB). The axis aligned bounding box box can be used for such computations as an initial check for the possibility of intersection of two objects (see the collision detection section below).

The characteristics of the surfaces of the body are also important in simulation. This is typically boiled down to a few properties of the material from which each facet of the body is made; friction (static and dynamic) as per the Coulomb model of friction,

[1]Such a trimesh is represented as a set of vertex coordinates and a list of triangles each with three indexes into the list of vertices.

[2]Concave objects are typically represented as composed of multiple convex objects.

coefficient of restitution and skin width are the most common.

Friction refers to the force on objects coming from the interaction of surfaces - static friction the force that helps maintain objects in static contact and dynamic friction that slows objects moving over each other. The friction forces in the Coloumb model are calculated as proportional to friction coefficients that are properties of the material and the force pushing the objects together (generally that force aligned with the contact normal). The Coulomb model is very well supported experimentally, and its discontinuous nature difficult to do away with, without getting implausible physical behaviour (Stewart, 2000). For real-time simulation the Coloumb model is not the most robust model to implement and so approximations are typical; for example, this can be as extreme as eliminating the effect of the size of perpendicular forces (Catto, 2005); PhysX is more realistic than that but there are similar efficiencies built in (Duff, 2011b).

The coefficient of restitution describes the proportion of momentum left in two bodies after colliding impact (as opposed to resting contact, where the coefficient can be considered zero). This is generally considered a body-level property, but when two objects collide it is not always clear how the coefficients of the two bodies should be combined; indeed, physics engines typically allow for a number of different approaches depending on the application. Again, this parameter is an approximation and it turns out that factors such as the shape of the object influence the restitution coefficient through multiple microcollisions, vibrations set up within the object and other factors (Stewart, 2000; Stoianovici & Hurmuzlu, 1996).

Skin width is not really a property of the material; it is a parameter that is used to determine what degree of interpenetration is allowable between resting objects, and as such less related to the physical model than an indication of how it is dealt with procedurally by the simulator (and not every simulator will use it). To understand why interpenetration might be useful in a physics engine, consider first that the problem of exactly determining the state of resting contact so that all contact points align and are not moving is a difficult one (Witkin & Baraff, 1997); rather than solve it completely, it is frequently easier to apply an additional soft constraint that pushes objects together again. A small restoring force can be used to reduce penetration over multiple solver iterations, but that can have the visual artifact of objects settling slowly into place or jitter. Indeed, if penetration is too deep then it can become unstable and bounce. So rather than applying restoring forces, sometimes it is better just to leave the object where it is. This is not always possible and for deep penetrations, one trick (used by PhysX) is to forget the restoring velocity after each time step (D. Black, 2011). The effect of a small amount of interpenetration is typically not visible, so it is plausible after all for applications in which it is the human eye that is important or where exact conformance to a physical model is not necessary.

4.1.2 Time integration and imposition of constraints

As mentioned above, the core task of physical simulation is integrating the equations of motion over time. It is the consideration of the rate of change of the system, its dependence on system state and effect on the time evolution of the system that makes this a dynamic approach - as opposed to kinematics, where the task is one of determining paths through state space without these considerations. In order to integrate the equations of motion over time, the rate of change of the system is required. In the following example, the rate of change of translation, rotation (in quaternion coordinates), linear and angular velocity is the vector that represents the rate of change of the system with time:

$$\frac{d}{dt}\boldsymbol{x}_t = \frac{d}{dt}\begin{bmatrix}\boldsymbol{T}\\ \boldsymbol{q}\\ \boldsymbol{v}\\ \boldsymbol{\omega}\end{bmatrix} = \begin{bmatrix}\boldsymbol{v}\\ \frac{d\boldsymbol{q}}{dt}\\ \frac{d\boldsymbol{v}}{dt}\\ \frac{d\boldsymbol{\omega}}{dt}\end{bmatrix} \tag{4.3}$$

The calculation of these components depends on the current state of the system in a number of ways: inertia, but also impulses applied to objects from collisions with each other, constraints on the object motion from for example joints or resting contact between surfaces, and also external forces such as gravity or forcefields.

One approach to dealing with contacts is to take each contact or constraint in isolation and calculate an impulse based on the current state (Mirtich & Canny, 1995), though in order to create an efficient and more stable simulator it is typically better to deal with all the constraints simultaneously (Catto, 2005; Witkin, Baraff, & Kass, 1997) since for multiple contacts an impulse based method resolves multiple contacts over time which can lead to jitter or instability as different local restoring impulses compete; this could alternatively be resolved by computing those impulses iteratively before advancing the simulation (Bender & Schmitt, 2006). A typical constraint based approach would solve for internal forces in terms of current state and external forces and resolve those forces into $\frac{d}{dt}\boldsymbol{x}_t$. From this rate-of-change the aim is to integrate forward in time to produce the next state:

$$\boldsymbol{x}_{t_{i+1}} = \boldsymbol{f}_{dyn}(\boldsymbol{x}_{t_i}, \boldsymbol{\theta}, \boldsymbol{u}_{t_i}, \Delta t_i) \tag{4.4}$$

$$= \boldsymbol{x}_{t_i} + \int_{t_{i-1}}^{t_i} \frac{d}{dt}\boldsymbol{x}_t dt \tag{4.5}$$

The most basic way of integrating Equation 4.5 is by using Euler's method. Euler's method simply uses the velocity at each time step to calculate what the state would be at the subsequent time step, assuming that the state only changes linearly. If these steps are made small enough, then the result converges to the true result:

$$\boldsymbol{x}_{t_{i+1}} \approx \boldsymbol{x}_{t_i} + \Delta t_i \frac{d}{dt}\boldsymbol{x}_t \tag{4.6}$$

However, the system has a tendency to drift because of the inbuilt approximation, particularly in the presence of "stiff" problems (so called because of such a systems analog to a stiff spring) - that is, problems with $\frac{d}{dt}\boldsymbol{x}_t$ very sensitive to state parameters. Such problems defeat the linearisation of the problem. Actually, the term "drift" can be an understatement. If the step size is too large for the problem, error can propagate so quickly that the system becomes unstable and rather than drifting, "explodes".

For very realistic or at least sophisticated offline models of the world, in theory these numerical steps could be run at very high temporal resolutions, but even for some very stiff offline systems this is impractical and the way the problem is posed is important. As such, in real-time physics, many extra approximations are usually done to the model to ensure that the numerical results not diverge too drastically from the estimated; some of these have already been mentioned above: maintaining a constant angular velocity when solving for velocity, for example. An additional such approximation done in PhysX is to scale down the constraint forces acting on a body by the number of constraints on the body - this helps stability in resting stacks, for example (D. Black, 2011).

Apart from reposing the problem, the Euler method can be updated to use different ways of solving the integration problem. Rather than using the velocity calculated at time-point t_i, the Runge-Kutta method works by running a full Euler step but then taking from that and from successive calculations different pieces of information about the expected state change at different time points during the time step and calculating the new derivative from that; ultimately an estimate of the average change over the whole time-step is acquired (Shabana, 2001).

Constraints can be dealt with in a number of ways, including introducing theoretical new force variables into the problem that serve to maintain the constraints, or indeed reducing the number of variables by taking into account the reduction in degrees of freedom and mapping system variables to a smaller space (R. B. Gillespie, Patoğlu, Hussein, & Westervelt, 2005). The former approach is in generally easier to apply and so dominates real-time physical simulation (Catto, 2005).

An orthogonal improvement to the Runge-Kutta method is to pose the problem of time-stepping as an "implicit" time integration problem. The aim here is, rather than calculating the state at time t_{i+1} based on information of the state of the system at time t_i, a sub-problem is solved where the constraints at time t_{i+1} are taken into account and the rate of change at t_{i+1} calculated. Such an approach has the effect of better accounting for constraints because constraints must hold at the resultant time point. The resulting system of equations that make up the sub-problem can be difficult to solve

(Witkin & Baraff, 1997); however, in order to solve the resulting sub-problem the motion of the system can be linearised and most benefits still obtained; also, an iterative solver can be used to approximate the solution to a chosen degree (Catto, 2005; Moravánszky & Terdiman, 2004); iterative solutions can be effective when exploiting solutions from previous time steps - the tendency of solutions to problems to be similar over time is called "coherence". Such systems can also be quite sparse[1] because constraints can be mostly local - such sparseness in the sub-problem can be exploited when solving[2].

4.1.3 Collision detection and resolution

The previous section discusses the process of integrating constraints and forces over time; however, dealing with object surface interaction in such a framework would be computationally and practically difficult[3]. Rather than attempt to represent such issues in the equations of time evolution, modern real-time physics engines use collision detection routines to detect when objects are in contact and then resolve them by introducing new forces or constraints into the equations of time evolution.

According to Witkin and Baraff (1997), there are two main kinds of contact:

- Colliding contact where objects are unlikely to spend much time in contact, where the collision is resolved by adding impulses (momentum or velocity changes) to the colliding parties.
- Resting contact where objects may be moving slightly with respect to each other but are maintaining a contact, resolved by introducing new constraints.

Before "near phase" contacts are considered, for the sake of efficiency a "broad phase" collision test is typically carried out where the (usually axis aligned) bounding boxes are

[1]A sparse linear system is one where many of the terms in the system have coefficient zero; such a system is computationally cheaper to solve with sparse methods that exploit the sparse system matrix layout in memory and the ability to effectively ignore much of the matrix

[2]Rather than using sparse matrices, the representation may ignore matrices altogether - and this is what PhysX does in its iterative solver (D. Black, 2011).

[3]The fact that the nature of surface interaction produces largely discontinuous state evolution would mean that any continuous approximation would be exceedingly stiff, and very difficult to solve.

checked for overlaps to efficiently exclude most potential contacts from going through the more complex near phase routine. Such broad phase sweeps are typically done by sorting the boxes and sweeping over them or by partitioning the space efficiently. This broad phase is particularly important when there are a lot of objects in a scene[1] (Lin & Gottschalk, 1998; Witkin & Baraff, 1997). In PhysX the broad phase is done using sweep and prune which is the same as sort and sweep, though re-using the sorted list and list of overlaps between time points (D. Black, 2011; Lin & Gottschalk, 1998; Witkin & Baraff, 1997); PhysX also has a mid-phase pruning algorithm that uses object aligned bounding boxes in the case of triangular meshes (D. Black, 2011; Gottschalk, Lin, & Manocha, 1996; Lin & Gottschalk, 1998; Terdiman, 2001).

For near phase contact routines, the aim is first to determine if there is a contact (the collision detection part) and if so to find some key information about the contact - an estimate of the location of the contact point, the relative velocity of the objects at the contact point, and face normals or vectors perpendicular to touching edges. This last part is called contact generation. Contact generation is a key place where efficiencies can be made in physics engines since eliminating contacts early reduces processing time later and reduces the likelihood of overconstrained systems later when too many contact points introduce too many constraints; approaches to reducing the number of contacts generated like this include ignoring contacts between impossible edges vertices or planes and clustering related contacts; the distribution of the contact points generated is important in stability and realism, however, so general desiderata include widely spaced points[2] (Moravánszky & Terdiman, 2004). In general such approaches make use of the separating axis theorem which states that a plane can be found between non-intersecting convex objects[3] (Lin & Gottschalk, 1998; Witkin & Baraff, 1997).

[1]An additional efficiency improvement is that unmoving objects that are in stable configurations get put to sleep.

[2]As with caching of solutions to constrained systems described above, caching contact points across time-steps can allow the physics engine to analyse the contacts for drift as well as to speed up the search for contact points (Moravánszky & Terdiman, 2004).

[3]The nature of the routine for determining contact and finding contact points depends on the colliding shapes, and since a lot of different kinds of component shapes exist in PhysX, there are a lot of such routines (D. Black, 2011). In this thesis, however, only tri-meshes are employed.

Once the contact point information is extracted, if the relative velocity of the objects at the contact point is high enough, a colliding contact is assumed. Based on the location of the contact point with respect to each of the centres of mass, the linear and angular momentums of the bodies, a force is applied at the contact point, which affects both the linear and angular momentum and velocity of each object. The amount by which energy is conserved in the collision is determined by the coefficient of restitution of the materials making up the individual bodies[1].

If the relative velocity of the objects towards each other is low enough at the point of collision (this threshold can depend on the coefficient of static friction), then a sliding or resting contact is assumed. In this case the resulting constraint typically forces the relative velocity of the objects to zero, and the sliding velocity can depend on the coefficient of dynamic friction. Imposing a velocity constraint rather than a constraint on the state is more likely to create linear constraints, with less likely the need for later physically implausible imposition of constraints and less likelihood that constraint based solvers be inaccurate, though as noted above, drift may need to be addressed under such a scheme (R. B. Gillespie et al., 2005). This is the idealised case and, again, efficiency improvements often trump devotion to any model of physics. For the purposes of visual tracking this forces the visual tracker to use the physics model in a robust way; this is essential at any rate because the model can always be wrong.

4.1.4 Stochastic simulators

In this thesis, the deterministic simulator is used as the backbone of a stochastic simulator by sampling from noise that is added to the simulator. However, a field of simulation already exists called stochastic simulation which deals with the numerical simulation of stochastic processes. The basic approach to simulation of such a process typically involves approximating the time evolution by sampling from the variance and adding

[1]Kinematic-only actors are treated as having infinite mass, which means that all of the resultant force is applied to the other object.

the rate of change multiplied by a very small Δt_i (D. T. Gillespie, 1992). This is clearly essentially a stochastic version of the Euler method described above. Indeed, the discrete and continuous extended Kalman filter can be considered a kind of stochastic simulation when carried out in the absence of observations, the same linearisation process occuring locally in the discrete version; the particle filter too does Monte Carlo simulation of a process.

4.1.5 Learned simulators

The numerical problems posed above are complex, and their efficient real-time calculation the subject of a large literature. Rather than attempt to solve them directly, the problem can be bypassed by re-representing it in a less explicit but algorithmically simpler way.

In the Neuroanimator system, Grzeszczuk, Terzopoulos, and Hinton (1998) use a physical simulator to train neural networks to do the same job that the physical simulator does. The neural network has the advantage of being compact and easy to evaluate online. The technique is used to get realistic animations and its smoothness allows controllers to be trained on the learned simulator. On the other hand, it requires transformation into an object based frame, normalisation, regularisation with some Euler steps, and other tweaks; it doesn't handle collisions, and stiff systems can be a problem. However, for some graphics applications, it proved useful[1].

Indeed, in robotics learning, predictive models (forward models), sometimes predictive of state, sometimes of sensor input, are a common precursor to learning or deriving controllers (inverse models), though these predictive models are themselves often learnt from data (Jakobi, 1997; Jordan & Rumelhart, 1992; Kopicki, Wyatt, & Stolkin, 2009; Todorov, 2011; Wolpert & Kawato, 1998; Zagal, Delpiano, & Ruiz-del-Solar, 2009; Ziemke, Jirenhed, & Hesslow, 2005). These predictive models are a form of simulator; the difficulty

[1]Reissell and Pai (2001) also present an approach to learning simulations from data, in the form of autoregressive dynamical systems or different exemplar-based motions with Markov Chain switching to handle discontinuities, though the dimensionalities of the systems simulated do not yet approach that of Neuroanimator. Reissell and Pai (2001) add stochasticity to their simulation through the Markov Chain or by adding noise based on the standard deviations of the learned residual.

is in learning them and applying them to the control problem. In the present thesis, the simulator is applied to the estimation problem, but many of the same concerns remain in the application phase; that is, dealing with inaccuracies in the prediction. Whether the simulators are learnt from data or are designed, this problem applies: all models are simplifications and only approximate reality according to some broad criteria. Jakobi (1997) notes that there are aspects of the simulation implementation that are important for learning a controller (or in our case, for employing it in an estimator), and there are aspects that are irrelevant or misleading. His solution is to add a lot of noise to the simulator, particularly in areas that are noisy in reality or that would mislead the controller, whilst learning the controller so that the controller only exploits aspects of the simulation that are robust to change under noise, and as a result under model failure. In the context of estimation, this addition of noise to the model is essentially what is done in the particle filtering case in practice, where process noise is added to account both for model failure and loss of track. There are many ways that noise can be added to a simulator, as discussed in this thesis, including adding noise to the simulator parameters, adding dispersion noise, and impulse noise. Wolpert and Kawato (1998) suggest that context-dependent predictive models and controllers should account for a large range of human motor learning and adaptation capabilities, the presence of the predictive model assisting in context-switching and driving learning of the controller.

4.1.6 Qualitative simulation

Rather than simulate the exact estimated state of the system or its numerical uncertainty, the qualitative approach to simulation propagates symbolic descriptions or intervals[1](Davis, 2008; De Kleer & J. S. Brown, 1984; Kuipers, 2001). For example, Nielsen (1988) simulate a clock qualitatively by working in a configuration space made up of in-

[1]Qualitative descriptions might include intervals and directions of change, or in the “semi-qualitative” case, exact numeric propagation of intervals according to “qualitative calculus”: constraints and descriptions of the evolution of a system which may be as broad as the direction of change of a variable or as specific as a differential equation on the upper or lower bound of an interval (De Kleer & J. S. Brown, 1984; Kuipers, 2001).

teracting components[1], but the simulations include a range of behaviours from perfectly working clocks to clocks broken in a number of interesting ways[2].

The main potential benefit of qualitative approaches would seem to be the application of known reasoning methods to new problems[3]. Another insight is that a good discrete abstraction is more amenable to computation than the original continuous problem one, and that mechanisms for reasoning over qualitative representations should generalise easily to different situations. However, the imprecise nature of the qualitative approach forces it to consider a large set of possible system behaviours and so suffer from a problem of combinatorial explosion; moreover, the promise of generalisability has not been met in practice and each new system needs to be studied in terms of representation and algorithm (Cellier & Roddier, 1991).

It has been formally shown that a qualitative simulation can be guaranteed to simulate the true behaviour of the system (providing the model supplied is correct), but not that the qualitative simulation can be guaranteed to find only the true behaviour of the system (Say & Akın, 2003). The upshot is that in order to be able to perform as well as quantitative simulation in terms of accuracy, qualitative simulation may have to become much more quantitative.

However, useful hybrids exist. A hybrid qualitative-quantitative approach is described by (R. B. Gillespie et al., 2005) where constraints are embedded into systems of equations, and their degrees of freedom reduced, using algebraic reasoning online [4]. Moreover, automatically abstracting qualitative models from quantitative ones and from data is an active

[1]One group of qualitative abstractions of physical systems involves dividing the world up into interacting components and describing the interactions, others include discrete processes, constraints, actions and events (Davis, 2008; De Kleer & J. S. Brown, 1984; Kuipers, 2001).

[2]For an example within the domain of this thesis, the idea might be to produce a system that reasons "the object is falling", "the object has bounced (or might still be falling", "the object has bounced twice (or might have bounced only once)", and so forth.

[3]To be more specific, dependent on finding the right abstraction, tools exist so that in each domain simulation, estimation, planning, counterfactual reasoning, initial value problem, and other tasks are possible with minimal set-up. For example, in the domain of manipulation planning, dividing the kinetic configuration space up into connected components corresponding to grasps and placements can speed up the search for a good manipulation plan (Jiménez, 2008; LaValle, 2011a, 2011b).

[4]Obvious links exists between stochastic approaches to simulation and discrete qualitative reasoning; qualitative reasoning being a form of stochastic reasoning with interval-based probability distributions.

line of research (Alur, Henzinger, Lafferriere, & Pappas, 2000; Hau, Coiera, & Muggleton, 1997; Liu, 2008; Loscalzo & R. Wright, 2010; Sachenbacher & Struss, 2005; Struss, 2002; Wang & Cellier, 1990; Zabkar, Mozina, Bratko, & Demsar, 2011).

Qualitative simulation is part of a broader approach to tackling computer understanding of physical phenomena via reasoning, called qualitative physics (Davis, 2008; De Kleer & J. S. Brown, 1984). Qualitative physics itself has its roots in naive physics - attempts to create computer models of theories for how humans reason about physics(Hayes, 1979 as cited by De Kleer and J. S. Brown, 1984; B. Smith and Casati, 1994). The main critique levelled against these approaches by psychologists is that although explicitly focusing on human-level or human-like descriptions of the world, the are generally too tangled in the preoccupations of computer science and formal ontology (B. Smith & Casati, 1994)[1]. Of course, more broadly, it is necessary to be skeptical of any approach that attempts to reconstruct a system for reasoning motivated primarily by introspection. Indeed, humans are generally wrong about the mechanisms of their reasoning in physical systems (Schwartz, 1999).

Further regarding the input of psychology into this area, there is some work in mental models of phenomena that is relevant. There is a school of thought that a lot of what people do in the way of reasoning about physical situations can be considered mental simulation of physical models. Hegarty (2004) collects evidence of characteristics of human physical reasoning that seem to share some properties with simulation, but might not fit so well with alternative explanations such as mental imagery and inference from setential descriptions.

For example, the amount of time it takes to solve some tasks offline relates proportionally to the amount of time it would take to solve the same task by simulating the task in reality - mental rotation being the classic example, but also simulating interacting

[1]One way that this shows itself is that the reasoning mechanisms for inference of time course in a qualitative simulation can often be quite a convoluted logical train and far from the kind of reasoning a human might describe. One approach to dealing with this is to provide a post hoc causal explanation; that is in terms of a train of events each of which has a local cause (De Kleer & J. S. Brown, 1984). In order to do this a partial ordered "mythical time" is invoked; such a local-cause approach is reminiscent of the impulse trains applied by Bender and Schmitt (2006) in order to maintain constraints.

cogs (Schwartz & J. Black, 1996). On the other hand, time taken to simulate complex physical systems goes up in the complexity of the component interactions as well, the process carried out by participants seems to be a propagation of effects through components. Other evidence found in individual differences in spatial reasoning ability and dual-task studies which ties the theory to neuroscience suggests that language brain centres are not exploited (though this is not true when the tasks can be solved using learned rules). Of particular interest is that some experiments involving imagining the task of pouring of liquids were difficult to do unless the body was oriented appropriately for the task. An interesting list of such probes and more can be found in Hegarty (2004)[1].

According to Gentner (2002), human mental models (a word that encompasses mental simulation of physical models as well as more rarefied models such as economic models) are frequently wrong in interesting ways - for example, the way a ball rolls off a table may be thought to involve some continued horizontal movement before it drops, and missiles flung in a circle may be thought to continue this curvature in their free flight. Moreover, people hold conflicting mental models but only apply them in relatively restricted situations; moreover, models are frequently re-used in analogy to earlier models.

One implication of this approach with respect to this thesis is the assumption that simulation can underly physical inference; this may be so as illustrated in this thesis, but the converse can still also apply; simulation is just one kind of physical cognition among many, and while the ability to solve one kind of physical inference (for example, control, planning, estimation) could easily share mechanisms, the inference tasks are still very different and different representations required for each one; neither is it clear that

[1]Pylyshyn (2003) argues that it is the subject and nature of the task requested of participants in investigations probing the use of internal models, rather than the nature of the underlying cognition that leads to image-like behaviour of cognition under the circumstances reported in the visual imagery literature - if a person is asked to perform a task in an image-like way then that is what they will do; the same argument can be applied to simulation type tasks. In the experiments by Schwartz and J. Black (1996) participants showed a variety of approaches to solving the problem of inferring the future state of a set of two gears, depending on task setup such as level of detail provided, whether visualisation or analytic approaches were requested, arguing that this casts light on the mechanism involved with each approach and its primes. Schwartz (1999) notes that inferences based on explicit belief are often wrong compared to when participants are required to visualise, implying that there are at least distinct mechanisms at play.

simulation is anyway the primordial such task; rather it would seem that possible that control is the more primitive task (Clark & Grush, 1999)

4.2 Dynamics in vision

Physics simulators as discussed in the previous section embody one kind of physical cognition. There are other kinds of physical cognition, some of which can be captured in the framework given in Chapter 2. It is estimation in such a framework that is of interest in this thesis, by exploiting knowledge of the dynamics p_{dyn}. Others have proposed using a physics model of dynamics to improve computer vision, or more broadly, have proposed combining the two disciplines together in some way to some end. In this section, some this work is reviewed - this work can be seen as working in parallel to this thesis in that the tasks tackled are similar to the tasks tackled herein.

4.2.1 Deformable simulation and continuous Kalman filtering

Chan et al. (1994), Metaxas (1997), Metaxas and Terzopoulos (1993), Metaxas and Kakadiaris (2002) use parametric deformable shapes and finite element surface models as a model of full 3D objects during motion estimation and tracking. Observations are collected in the form of 3D points that induce forces in the object model, deforming and pulling the corresponding object point towards the observation as the model is simulated over time using Euler-style integration methods as discussed above in Section 4.1.2. The simulation variables are generalised coordinates representing object deformation shape and pose. Constraints similar to those discussed above are used for binding compound objects.

So estimation and simulation are tackled within one approach. The authors used continuous Kalman filter methods for dealing better with the noise in observations; interestingly, it turns out that the covariance in the Kalman filter can be thought of as transformation of the forces. Metaxas and Terzopoulos (1993) apply the model to synthesised and motion-captured 3D data points. Chan et al. (1994) apply a version of the model to tracking rigid objects in stereo images by allowing an image potential derived from the smoothed edge filtered image to be projected back into 3D space as forces. Metaxas and Kakadiaris (2002) track deformable objects and estimate their elastic properties online in the case of 2D contours from medical images, as well as 3D head models from 3D point data.

Deformable models are highly useful because many objects in the real-world are deformable, not rigid. Moreover, even if the objects being tracked are rigid, the extra degrees of freedom can make up for noise in the estimation system, particularly if the deformations are posed appropriately (Masson, Dhome, & Jurie, 2005; Metaxas, 1997). For rigid objects on the other hand it is often better to regularise this problem using the known patterns of projection of 3D objects (Chan et al., 1994; Marchand, Bouthemy, Chaumette, & Moreau, 1999). Deformable models are also useful for finding new shapes in static or moving scenes (Metaxas, 1997) or inferring dynamic properties of objects, such as the existence of multiple parts (Kakadiaris et al., 1994).

For a truly general vision system, handling deformable models will be necessary; sometimes it can simplify the vision problem. A lot of success has been obtained with 2D contour tracking (Isard & Blake, 1998b; Metaxas & Kakadiaris, 2002) and 3D problems with controlled conditions such as medical imaging (Metaxas & Kakadiaris, 2002). Face tracking is particularly applicable for deformable model based methods because a lot of the movements of faces can be captured as general deformations, rather than the specific kinds of deformations found in joints for example (Decarlo & Metaxas, 2000). Deformable object tracking is a very popular area and is becoming more robust over time to re-initialisation, occlusion, and false positive data[1].

[1]Although this thesis does not go so far as to consider explicitly deformable objects, it manages to distinguish itself by focusing on the problem of robustness to very bad observations and also a degree of model failure.

4.2.2 Explanation mediated vision

Explanation mediated vision shares a history and many techniques with qualitative physics. In addition to simulating or making inferences in systems, explanation mediated vision intends to understand scenes described in images or sequences of images in qualitative terms by operating on vision-extracted information.

Stark, Hoover, Goldgof, and Bowyer (1993) have an active sensing system in which the potential function of static objects found with a range-finder is analysed in order to determine the identity of the object and further prompt scans of the object. The approach is an expert system with functional primitives like adjacent free-space and output like “it must be a chair”. Brand (1997), Brand, Cooper, and Birnbaum (1995) analyse static images of unknown gear trains by passing them through a domain-specific qualitative simulator and testing and propagating consistency of constraints around the network of joints formed by the gears and checking that the network is in equilibrium, and consequently inferring hidden structure; further visual operations (local region growing to find new parts) are seeded by the reasoner to disambiguate the structure on the basis of violated constraints, and related information. A similar system is presented by Birnbaum, Brand, and Cooper (1993) in the context of checking stacks of blocks for stability. These systems tend to drive the visual system around the scene looking for new blocks, particularly in areas where apparent inconsistency exists. Note the relationship of this approach with the mental simulation work by Hegarty (2004) discussed above in Section 4.1.6 where it was noted that it has been found that people tend to reason around links in diagrams when doing mental simulation[1]. Additionally, it seems that in humans, support relations are processed even in infants, and that support and other physical relations are made use of early in processing in adults (Baillargeon & Hanko-Summers, 1990; Biederman, Mezzanotte, & Rabinowitz, 1982). Brand (1997), Brand et al. (1995) also implement a

[1]This qualitative propagation of constraints, related to the causal approach to qualitative physics seems similar, not only to the tendency of people to propagate effects around a scene, but also to to constraint propagation in quantitative simulation. There seems to be something about the nature of this problem that encourages such an approach.

robotic solution to a cup-lifting use-case to illustrate the benefits of being able to do causal physical inference from vision. Here, the vision server is used to make camera motion and do visual routines, while a separate reasoner sends queries and requests and receives 2D scene geometry.

Siskind (2003) analyses images for support relationships and stability; polygons are pre-segmented from images and each other on the basis of cues such as colour cues, and assigned to layers, allocated joints with other objects, and considered grounded or free on the basis of logical criteria looking for a quasi-static stable and non-interpenetrating scene, with the fewest such assertions[1]. Finding the optimal cost of a particular interpretation for a frame can be specified as a linear programming problem in the feasible gravity-induced velocities of objects; friction is not only useful to model physical behaviour, but to reduce numerical instability. Using the force-object relations extracted from a movie, system behaviour specified at a high level can be inferred (that is, event recognition) - for example a stack can be described in terms of pick ups and put downs wich involve sequences of force-relations between an inferred hand, object and table.

Rather than use a kinematic and stability analysis, R. Mann and Jepson (1998) analyse scenes in terms of their dynamic properties, but suffer from noise in inferred acceleration and velocity so that inferences in attachment, propulsion, etc are often spurious. The use of forces handles non-passive forcing by inferring hidden "motors" in the scene; different objects have different kinds of force-effects. Collision inference is done by looking for any discontinuity in motion. The kinematic approach has these problems too, but not as clearcut. Studies in humans suggest that depending on situation, information about forces, gravity, viscosity, etc can be very important in physical simulation inferences and tend to supplant kinematic information (Schwartz, 1999).

Bennett et al. (2004) do motion estimation by reasoning logically about object entries and exits, occlusions, and so forth, with the assumption that objects should not move far

[1]The use of such criteria as reducing the number of required assertions in ordering logical interpretations is called circumscription, and it is done as a later step over sequences of frames to obtain a most consistent interpretation of a sequence by preferring less changes of state (particularly with respect to grounding) in objects.

from time-point to time-point. D. Brown (2008) combines physics simulation and video for use in teaching. A part of this package is Autotracker that uses template matches to find well defined objects in a video stream. This supplements a manual labeling process for comparing physics models to real-world data [1].

All of these qualitative approaches tend to be done in 2D or 2½D, based on simple visual region cues, and tend not to be particularly robust, operating under controlled conditions and making assumptions of little or well-defined data noise; this is perhaps a consequence of the need to obtain compact high-level descriptions as soon as possible in the processing pipeline, in order to conduct qualitative reasoning - there tends to be a difficult trade-off there between the compactness and tractability of the representation and representing uncertainty in it.

It should be apparent that simulation based approaches have the potential to be explanatory as well, insofar as impulse trains, temporary constraints[2], contact points, and so forth can be transformed into meaningful explanations; this does not go as far as explanation *mediated* approaches in not making direct use of the explanation, but it is, at any rate, still not clear that qualitative explanations in logical or linguistic form are workable intermediate representations in these operations[3]; progress is being made on both fronts.

To compare qualitative approaches to quantitative briefly with respect to application to computer vision, the main benefit of qualitative approaches is in providing constraints on visual inference problems, potentially regularising them and hopefully reducing computational complexity. Though computer vision techniques do require a strong coherence between the estimated world state and the observed video stream; that is to say numerical

[1]Understanding events from video is a particularly large field but this review has concentrated on those that use physical properties of objects to infer events.

[2]Recall that constraints in physical simulation can sometimes be interpreted as impulses and visa versa just as with the cause/constraint dichotomy found in qualitative physics - compare Bender and Schmitt (2006) to De Kleer and J. S. Brown (1984).

[3]For example, in order to produce satisfactory causal explanations in a qualitative framework, De Kleer and J. S. Brown (1984) were forced to produce post-hoc explanations of the output of their reasoning process. As noted earlier, this may not be a failure of the system, but rather of our expectation that such explanations should play a deep role in reasoning.

accuracy is important and numerical approaches have the advantage there in that they provide exact predictions[1]. Davis (2008) notes that researchers in physical reasoning and knowledge representation agree that vision is in principle within the scope of physical reasoning, though the kind of computation used in computer vision has not in practice turned out to be generalisable using qualitative methods and that attempts to use knowledge representation and qualitative reasoning techniques to constrain visual processing have been rare[2].

4.2.3 Physics of the human body

Some of the most exciting work in the use of physical simulation in computer vision is in the area of model-based tracking of the human body. This is an area where a large number of parameters need to be inferred sometimes from very little data. Physics should help to regularise the problem; it should reduce the space of hypotheses to help speed up inference and make it more accurate.

Use of dynamic motion predictors in the human body goes back a long way; Hogg (1984) does both detection and tracking of full jointed models of human bodies; in order to do tracking a walking cycle is employed and hard constraints applied to the range of velocities and limits of body joints and a local generate and test employed around a predicted pose and potential poses scored on the basis of edge matches with model contours not accounting for self occlusion. The search is factored for efficiency by treating different limb parts separately conditioned on a torso position.

At present, the most common approach to regularising the human body tracking problem is by learning models of human body motion from motion sensors. Indeed, many computer vision approaches use learning to find and exploit related appearances, for example tracking a hand on the basis of a factored set of basis appearances (Heap

[1]The need for numerical accuracy is particularly true for a peaky likelihood formulation like that used in Chapter 6 and 7.

[2]In this thesis a more bottom-up approach is attempted in that the point of generalisation between physical and visual processes is in the use of a stochastic model, rather than qualitative descriptions. Qualitative approaches can be considered a subset of probabilistic approaches.

& Hogg, 1998) or learning a probabilistic partitioning of the state and trajectory space based on appearances (Stenger et al., 2006). Beyond this, learned priors are frequently placed over articulation and limb movement since these movements are constrained and tend to follow stereotyped trajectories (Stenger et al., 2006).

A recent approach and perhaps the most similar to the method attempted in this thesis adds physics knowledge into human body tracking (Vondrak, Sigal, & Jenkins, 2008), where a full dynamic rag-doll model of human motion is used as the basis of an object tracker. Additionally, the rag-doll is given a predictive motion planner trained on 3D human motion data and a custom plausible controller separate from the simulator. The tracker uses a particle filter with pose noise in conjunction with silhouette and edge features. The passive rag-doll model does not appear significantly better than a constant velocity dynamics; this is thought to be because the constant velocity dynamics approximates the rag-doll model well most of the time. However, when given a motion planner trained on data from motion sensors and a controller, the new method well outperforms prediction free and constant velocity approaches as well as an annealed particle filter, particularly in monocular tracking where constraints of penetration and joint angles so forth (but also known motion patterns) help constrain search priors in each scene (and the annealed particle filter, that normally does very well, tends to overfit). In order to truly know whether the improvement was due to the framework within which the learned data was expressed, rather than by fitting to a motion prior fit directly to learned data, more experimentation would be required, however.

Similarly, Wren and Pentland (1998) use control patterns to create a potential field that is added to a dynamic model of human motion, particularly enforcing joint limits. They use a Kalman filter to propagate estimates based on blobs using this controller and simulator; they choose the currently active controller from a set of them on the basis of the residual of the Kalman filter on each controller.

4.2.4 Learning & control

Álvarez, Luengo, and Lawrence (2009) use a dynamical model of a body and arm to infer the hidden (latent) forces and hyper-parameters in the model based on a Gaussian process model and an analytically integrated Lagrangian dynamics. They successfully train a model on motion capture of two different movements and predict a third movement by conditioning the prediction on one of the joints. Álvarez, Peters, Schölkopf, and Lawrence (2011) extend this approach to a switching model to deal with discontinuities, making this approach more applicable to impacts, for example. They use this model to learn a model of robot movement from demonstration. In other recent work, again in the context of a Gaussian process model, Nguyen-Tuong and Peters (2010) combine a rigid dynamics model of a robot arm with machine learning to augment the rigid dynamics model for use in tracing a provided trajectory. The combination of rigid body simulator and machine learning proved more successful than either combined. Three kinds of learning were done with the simulator: learning the inertial matrix, mass and centre of mass parameters of the arm segments, learning a non-linear additive corrector to the simulator's prediction, and using the simulator to structure the learning space by providing a kernel. The latter approach is not as easily brought into the model developed in Chapter 2 as the simulator is not directly used as a model for time evolution of the state but is rather used to alter the topology of the learning problem.

4.2.5 Motion synthesis

Motion synthesis is the task of synthesising realistic motions of any kind (deformable body, rigid body, character) to match some desiderata; this differs from simulation in that although the motion will ultimately be simulated, the parameters of the simulation need to be generated to create a desired effect. For example, it may be desirous to have a hat twist through the air and land on a coathook (Popovic, 2001; Popovic, Seitz, Erdmann, Popovic, & Witkin, 2000).

Although the task is different from that of motion estimation, some of the problem is shared - in motion estimation using simulation, soft constraints exist on the simulation, just as with motion synthesis. The main difference is on the need for robustness and autonomy in computer vision as well as the overarching concern with how to translate images into observations; motion synthesis is usually human-driven so does not have these concerns.

However, some of the motivation for the concept in this thesis came from motion synthesis approaches like that of Popovic (2001), Popovic et al. (2000), where a simulator is used and a Jacobian[1] calculated across collisions with respect to simulation parameters such as friction of certain facets, collision normals and boundary conditions in order to fit control points or control poses. Indeed, a similar approach is used by Bhat, Seitz, Popovic, and Khosla (2002) in the absence of collisions to infer free flight trajectories from silhouettes. Similarly, Chenney and Forsyth (2000) sample from possible motions online using Markov Chain Monte Carlo methods based on a probabilistic model accounting both for plausibility of collisions (noise in the collision normal) and maintenance of the constraints of the desired animation[2].

Clearly all of these problems are also related to control, where the task is to achieve a certain trajectory, or at least a trajectory with certain properties, and so can be considered related problems. This can be seen clearly in the problem of getting n pucks to target positions by determining appropriate initial velocities, where some pucks are constrained to have zero velocity (Tang & Ngo, 1995).

[1]The Jacobian is the matrix of partial derivatives of a multivalued and multivariate function. The number of rows is the number of values output by the function and the number of columns is the number of variables input.

[2]Note that by selecting parameters to vary that do not greatly affect the realism of the simulation, in effect the simulator is being treated as stochastic with these parameters considered the most noisy; compare this to the proposal discussed above of Jakobi (1997) that simulations be applied to real-world tasks (in their case, the learning of controllers for use in the real-world) by adding noise to aspects of the simulation. Additionally, as noted by Chenney and Forsyth (2000), the existence of some randomness in simulations, e.g. resulting from collisions, is more realistic.

4.2.6 Computer vision and control

This last section reviews a smattering of work that tends to cross the boundary between vision and control or planning via physics.

As should be clear from this and the previous chapter, there are numerous overlaps between the tasks of motion control, motion planning, motion synthesis, and motion estimation, particularly in the model-based case. A small number of representations and algorithms are common across all these disciplines. This enables the vision and planning/control to easily swap data about objects and so gain synergies. It is very common for vision and sensing to produce or match 3D models of objects; these estimates of shape, category and pose can be input directly into robotic manipulation routines - e.g. Klank, Pangercic, Rusu, and Beetz (2009), Kragic, Miller, and Allen (2001), Krainin, Henry, Ren, and Fox (2010), and parameters of controlled systems such as robot joints directly estimated from data (Nickels & Hutchinson, 1998). Such approaches can make use of the same visualisation routines in both tasks, can call the visual routine online from the grasper, plan a grasp on the basis of that data (planning grasps typically uses sophisticated kinematic or dynamics-based simulators and motion planners) initiate or integrate online visual servoing for placement and re-plan when necessary on the basis of visual input. Motion planning and control make use of simulators to aid in this planning (Ettlin, Buchler, & Bleuler, 2005).

As well as common object representations, the use of a probabilistic framework allows synergies between disciplines too. Fu, Srinivasa, Pollard, and Nabbe (2007) tackle a 2D pinball-like problem in simulation where the shape of the ball is unknown and the camera is noisy. Noise is introduced by collisions including the planned action of hitting with the bat under the agent's control. They tackle both planning and estimation in a probabilistic framework, estimating shape pose and velocity of the object with a particle filter, and acting, first to gain information and second, to make a hit towards the goal.

Just as estimation is a process for inferring states, actions or other variables from incomplete observations, simulation is a process for inferring states from actions and other variables. Furthermore, as was noted above, planning and motion synthesis can be thought of as processes for inferring states, actions and other variables from goal states or constraints. Particularly in the latter problem, the goals and constraints involved in motion planning or synthesis could be thought of as observations, in which case the flow of influence in the literature might soon be seen going from work on estimation back into work on control and synthesis. The strong links between estimation, simulation and planning, particularly within a probabilistic inference framework, are in the process of filtering through into the cognitive psychology literature, where the common framework is providing a vision for understanding human behaviour, cognition and neuronal makeup (Clark, 2011; Todorov, 2009; Toussaint, 2009), in particular the idea that cognitive systems are essentially predictive - of perceptual errors, or as forward models in control (Clark, 2011; Clark & Grush, 1999; Grush, 2004; Ziemke et al., 2005). According to Toussaint (2009) the idea of using the same inference process for at least both estimation and control is compelling, though it is admitted that in practice the problems are different enough that contemporary approaches must be significantly differentiated, an observation that has so far been borne out by this thesis. In particular, the need for robustness and the incompleteness of observations tends to make the estimation problem harder. In the future, robust estimation techniques (for example, as are used in this thesis), may find applicability in motion planning, where trajectories goals and constraints may be only loosely specified. Indeed, Todorov (2009) points out that the typical motion planning problem is equivalent to a hard estimation problem, i.e. having a small amount of guiding information. Similarly, a typical estimation problem is a hard motion planning problem because of the large number of soft constraints involved.

CHAPTER 5

APPROACH AND EXPERIMENTS: MOTION ESTIMATION OF BOUNCING OBJECTS

"Then, in the 89th minute, Katiego Mphela muscled on to the end of a long hopeful punt upfield from his goalkeeper Itumeleng Khune and trickled his left foot shot goalbound, only to languish in agony as he saw it bounce apologetically off the post."

`South Africa 1 Mexico 1: match report,`

`Ian Chadband, The Telegraph, 11 June 2010`

This chapter tackles the motion estimation problem. This problem will be tackled by building dynamics and physical simulation into the solution. This builds bridges with the motion planning and motion synthesis problems which by their nature deal with the kinematics and, sometimes, the dynamics of *trajectories*[1], whereas the other experimental chapters focus on recursive estimation of state.

However, the main scientific aim of this chapter, as with all of the experimental chapters in this thesis, is to put to the test in a small way the following statement:

Inclusion of a physics simulator in the estimation process can be done in such a way as to increase its performance across scenarios.

Thus, this chapter is both about the possibility of such an augmented estimation

[1]Motion synthesis approaches attack a similar problem to that of motion estimation, often opening up the simulator and making it stochastic or using its Jacobian[2] with respect to simulator parameters, but do not deal with issues of robustness. Motion planning must be robust to possible plan failure, but goals are well-defined, unlike the visual information available in visual motion estimation.

process, and about the technology proposed to get there.

The scenarios considered in this chapter are ones where background physical knowledge may be of some use in motion estimation; where objects are fast moving, with blur and glare, where extreme noise like distractors and partial and complete occlusion exist, and other solutions are expected to have trouble. The scenarios are drawn from cases where such noise exists, and where objects are bouncing through a scene. There are two sets of scenarios.

The first set is a large number of simulated bouncing trajectories with added Gaussian and outlier noise, where the experiments described here determine the best algorithm to use in trajectory reconstruction and it is shown how a refinement approach is employed to reconstruct trajectories, as well as a RANSAC approach; these approaches are pitted against each other.

The second set of scenarios is a small set of four actual bouncing trajectories of a ball bouncing in the image plane with occlusion, blur, glare and severe distractors. Experiments are presented determining the best dynamics model when reconstructing the trajectory from colour histograms. Five different dynamics models are compared. This includes two novel physics-based models, one where the whole trajectory is parametrised by initial state, and one where state along the whole trajectory is parametrised, and additive displacement and velocity noise assumed. These novel approaches show that a physics-based dynamics can solve the motion estimation problem in the presence of bad or absent visual data where more naive dynamics models are unable to.

Both sets of scenarios are summarised in Table 5.1. The first section in this chapter deals with experiments carried out on around a simulated bouncing object and the second section deals with the experiments around the bouncing ball estimated from colour. The story told by the first section is that the estimation problem is non-linear and RANSAC fixes that as well as excessive noise. The second section illustrates the benefits of the use of a physics model across the scenarios described here, while more naive dynamics models tend to fail. First, however, the background of this chapter is refreshed briefly, and the

MAP formalism introduced in Chapter 2 developed further to incorporate the dynamics into an optimisation framework[1].

5.1 Background

Table 5.2 summarises where in this thesis to find background on the topics discussed in this chapter.

To summarise; in visual tracking, sophisticated and robust methods do exist for tracking from colour, often with dynamic (blob) shapes (Comaniciu et al., 2003; Magee, 2001; Wren, Azarbayejani, et al., 1997) but those methods do not make use of sophisticated dynamics models. In human tracking, there is work on tracking using simulation (Hogg, 1984; Vondrak et al., 2008; Wren & Pentland, 1998); in the present chapter, the physical simulation is by itself sufficient for improvement, rather than relying on being driven by a controller learnt from motion capture data. The work on tracking deformable objects from physics (Chan et al., 1994; Decarlo & Metaxas, 2000; Kakadiaris, Metaxas, & Bajcsy, 1997; Marchand et al., 1999; Masson et al., 2005; Metaxas, 1997; Metaxas & Terzopoulos, 1993) focus on the hard problem of tracking deforming objects but not of robustness to very bad visual information as is investigated in this chapter.

Rather than modelling object physics, blur and glare can be dealt with by modelling the blur and glare processes themselves. Occlusion can be dealt with by reasoning about possible occluding objects. Distractors can be dealt with using data association or model selection techniques. In the present chapter it is shown that the physical simulator combined with RANSAC is enough in the current set of scenarios for dealing with all of these problems.

[1]Many of the results described in this chapter have been published in the IEEE International Conference on Robotics and Automation (Duff, Wyatt, et al., 2010).

Property	Bouncing object in simulation	Bouncing ball from colour
Source	Simulation	USB Camera
Output Dim	6D Pose × length	2D Pose × length
Input features	Noise perturbed 6D pose	Colour histogram match score map
# data-points	50 trajectories per noise condition	4 trajectories
Scenario	Simulated object bouncing with precession, added Gaussian & uniform noise.	A ball bouncing in the presence of severe distractors, occlusion, blur, & glare.
Experiment probes	Choice of fitting algorithm	Choice of dynamics model
Conditions	Refinement vs RANSAC Amount & quality of noise	No dynamics Constant displacement Constant velocity Globally parametrised physics Locally parametrised physics

Table 5.1: **Scenarios used in experiments in this chapter**

Subject	**Found where**	**Details**
Probabilistic formulation	Chapter 2	Puts forward a probabilistic formulation for estimation from trajectories.
Trajectory estimation	Section 2.6.1	Specialises in MAP trajectory estimation, as is done in the present chapter.
Computer vision	Chapter 3	Discusses the background of motion estimation and recursive estimation in computer vision.
Colour histograms	Section 3.2.2	Discusses the use of colour histograms..
Visual motion estimation	Section 3.5	Discusses long sequence motion estimation, the problem approached in this chapter. That work incorporates physics models beyond polynomial approximations, or robustness.
Physical simulation	Chapter 4	Discusses the fundamentals of physical simulation.
Physics based estimation	Section 4.2	Looks at previous applications of physical simulation to computer vision, particularly human and deformable object tracking.

Table 5.2: **Where to find background information for this chapter.**

5.2 Parametrisation of MAP trajectory estimation

Recall that in the framework Chapter 2, Section 2.6.1 the MAP trajectory estimation problem was introduced. That is:

$$\begin{aligned}&\arg\max_{\boldsymbol{x}_{t_{0:i}}} p(\boldsymbol{x}_{t_{0:i}}|\boldsymbol{z}_{t_{0:i}}, \boldsymbol{\theta}, \boldsymbol{u}_{t_{0:i}}) \\ &\qquad = \arg\max_{\boldsymbol{x}_{t_{0:i}}} \frac{p_0(\boldsymbol{x}_{t_0}) \prod_{j=0}^{i}[p_{obs}(\boldsymbol{z}_{t_j}|\boldsymbol{x}_{t_j})] \prod_{j=1}^{i}[p_{dyn}(\boldsymbol{x}_{t_j}|\boldsymbol{x}_{t_{j-1}}, \boldsymbol{\theta}, \boldsymbol{u}_{t_i}, \Delta t_i)]}{p(\boldsymbol{z}_{t_{0:i}})}\end{aligned} \tag{5.1}$$

The vector sequence $x_{t_{0:i}}$ is the sequence of states making up the trajectory. The problem is to find these states to maximise the above probability.

In the context of introducing typical methods to estimation in computer vision, in Chapter 3 Section 3.3.3, the log of the observation probability distribution was taken, allowing for a generalised least squares single frame estimation assuming that the observation distributions were Gaussian. In Section 3.5 the least squares approach was extended to trajectory estimation based on the assumption of a non-informative dynamics.

Here, two formulations are introduced that incorporate a dynamics model: In the first, it is assumed that the dynamics is always correct and the only noise exists in the observations and an unknown initial state. This is called a "globally parametrised" trajectory since in order to represent a trajectory only the boundary conditions (initial state) are required and will be used in the experiments in simulation as well as the real-world experiments. In the second formulation, the dynamics themselves can be noisy. Because of this, each state in the trajectory is estimated, and so the trajectory is called "locally parametrised". This is used only in the real-world experiments.

5.2.1 Globally parametrised dynamics

The globally parametrised formulation of the trajectory uses a deterministic dynamics. The simulator is assumed to correctly model the world in which the estimation is being done. Simulation parameters $\boldsymbol{\theta}$ and a nuisance/control signal $\boldsymbol{u}_{t_i}$ are included in the model. Thus, each state $\boldsymbol{x}_{t_i}$ is completely constrained by the previous state $\boldsymbol{x}_{t_{i-1}}$. This means that the trajectory estimation problem is then only to find the initial state $\boldsymbol{x}_0$

In order for the trajectory probability to be non-zero, the following constraint must be realised[1]:

$$\forall i > 0 : \boldsymbol{x}_{t_i} = \boldsymbol{f}_{dyn}(\boldsymbol{x}_{t_{i-1}}, \boldsymbol{\theta}, \boldsymbol{u}_{t_i}, \Delta t_i) \tag{5.2}$$

Here $\boldsymbol{f}_{dyn}()$ is again the deterministic simulation function. That is, each state is fully determined by the dynamics model and the previous state. Having established this constraint, the probability of each dynamics distribution $p_{dyn}(\boldsymbol{x}_{t_i}|\boldsymbol{x}_{t_{i-1}}, \boldsymbol{\theta}, \boldsymbol{u}_{t_i}, \Delta t_i)$ must be 1. Therefore, the MAP problem becomes:

$$\arg\max_{\boldsymbol{x}_{t_{0:i}}} p(\boldsymbol{x}_{t_{0:i}}|\boldsymbol{z}_{t_{0:i}}, \boldsymbol{\theta}, \boldsymbol{u}_{t_{0:i}}) = \arg\max_{\boldsymbol{x}_{t_{0:i}}} \frac{p_0(\boldsymbol{x}_{t_0}) \prod_{j=0}^{i} [p_{obs}(\boldsymbol{z}_{t_j}|\boldsymbol{x}_{t_j})]}{p(\boldsymbol{z}_{t_{0:i}})} \tag{5.3}$$

Equation 5.2 is recursive with a base case of $\boldsymbol{x}_{t_0}$. As such, $\boldsymbol{x}_{t_i}$ can be expressed directly in terms of $\boldsymbol{x}_{t_0}$:

$$\boldsymbol{x}_{t_i} = \boldsymbol{f}_{dyn}(\boldsymbol{x}_{t_0}, \boldsymbol{\theta}, \boldsymbol{u}_{t_{0:i}}, \Delta t_{0:i}) \tag{5.4}$$

Note that this function is a composition of individual per-time-step $\boldsymbol{f}_{dyn}()$ functions. Given this constraint the MAP problem can be written:

$$\begin{aligned} &\arg\max_{\boldsymbol{x}_{t_{0:i}}} p(\boldsymbol{x}_{t_{0:i}}|\boldsymbol{z}_{t_{0:i}}, \boldsymbol{\theta}, \boldsymbol{u}_{t_{0:i}}) \\ &\qquad = \arg\max_{\boldsymbol{x}_{t_{0:i}}} \frac{p_0(\boldsymbol{x}_{t_0}) \prod_{j=0}^{i} [p_{obs}(\boldsymbol{z}_{t_j}|\boldsymbol{f}_{dyn}(\boldsymbol{x}_{t_0}, \boldsymbol{\theta}, \boldsymbol{u}_{t_{0:j}}, \Delta t_{0:j})]}{p(\boldsymbol{z}_{t_{0:i}})} \end{aligned} \tag{5.5}$$

Taking the log of this, assuming a uniform distribution over $\boldsymbol{x}_{t_0}$ over the region of

[1]For the globally parametrised dynamics, because there is only one possible successor to a given state, $p_{dyn}()$ can be considered to be a delta function in the state $\boldsymbol{x}_{t_i}$.

interest, the following cost function to be minimised is arrived at:

$$\begin{aligned} Cost(\boldsymbol{x}_{t_0}) &= log[p(\boldsymbol{x}_{t_0}|\boldsymbol{z}_{t_{0:i}})] \\ &= \sum_{j=0}^{i} log[p(\boldsymbol{z}_{t_j}|\boldsymbol{x}_{t_j})] - log[p(\boldsymbol{z}_{t_{0:i}})] \\ &= \sum_{j=0}^{i} ObsCost(\boldsymbol{z}_{t_j}, \boldsymbol{f}_{dyn}(\boldsymbol{x}_{t_0}, \boldsymbol{\theta}, \boldsymbol{u}_{t_{0:j}}, \Delta t_{0:j})) + Const \end{aligned} \tag{5.6}$$

The problem then is one of finding an $\boldsymbol{x}_{t_0}$ that minimises the global cost function in Equation 5.6.

If the observations exist in a space that permits a Gaussian, the observation probability can be considered a Gaussian and a generalised least squares problem is arrived at:

$$\begin{aligned} &ObsCost(\boldsymbol{z}_{t_i}, \boldsymbol{f}_{dyn}(\boldsymbol{x}_{t_0}, \boldsymbol{\theta}, \boldsymbol{u}_{t_{0:i}}, \Delta t_{0:i})) \\ &\qquad = ObsCost(\boldsymbol{z}_{t_i}, \boldsymbol{x}_{t_i}) \\ &\qquad = (\boldsymbol{z}_{t_i} - \boldsymbol{f}_{obs}(\boldsymbol{x}_{t_i}))^T S_{obs} (\boldsymbol{z}_{t_i} - \boldsymbol{f}_{obs}(\boldsymbol{x}_{t_i})) && (5.7) \\ &\qquad = (\boldsymbol{z}_{t_i} - \boldsymbol{f}_{obs}(\boldsymbol{f}_{dyn}(\boldsymbol{x}_{t_0}, \boldsymbol{\theta}, \boldsymbol{u}_{t_{0:i}}, \Delta t_{0:i})))^T S_{obs} (\boldsymbol{z}_{t_i} - \boldsymbol{f}_{obs}(\boldsymbol{f}_{dyn}(\boldsymbol{x}_{t_0}, \boldsymbol{\theta}, \boldsymbol{u}_{t_{0:i}}, \Delta t_{0:i}))) && (5.8) \end{aligned}$$

Even if the observations are linear normal in the state, this is not a *linear* least squares problem, however, unless the dynamics $\boldsymbol{f}_{dyn}()$ are linear (which they, in general, are not). The weight matrix S_{obs} can be used for weighting different observations according to their informativeness.

If the observations include rotations, it is straightforward to put a Gaussian over the space of rotations, but not obvious how to do it well; representation becomes very important in such a situation. This issue is dealt with in this chapter by putting a theoretical Gaussian over the individual points in the object. This induces an easy and natural distribution over rotations. This solution will be discussed in Section 5.3.4.

5.2.2 Locally parametrised dynamics

In the second formulation of the trajectory estimation problem, the dynamics, while considered informative, are not considered deterministic, so that a cost/objective function is derived on the basis of both an observation and a dynamics cost. Again, the problem is a cost minimisation problem, but with a larger parameter space $\boldsymbol{x}_{t_{0:i}}$ and dynamics cost terms as well as observation[1]. Simulation parameters $\boldsymbol{\theta}$ and a nuisance signal $\boldsymbol{u}_{t_i}$ are again included.

$$\begin{aligned}
Cost(\boldsymbol{x}_{t_{0:i}}) &= log[p(\boldsymbol{x}_{t_{0:i}}|\boldsymbol{z}_{t_{0:i}})] \\
&= log\left[\frac{p_0(\boldsymbol{x}_{t_0})\prod_{j=0}^{i}[p_{obs}(\boldsymbol{z}_{t_j}|\boldsymbol{x}_{t_j})]\prod_{j=1}^{i}[p_{dyn}(\boldsymbol{x}_{t_j}|\boldsymbol{x}_{t_{j-1}},\boldsymbol{\theta},\boldsymbol{u}_{t_j},\Delta t_j)]}{p(\boldsymbol{z}_{t_{0:i}})}\right] \\
&= \sum_{j=0}^{i} log[p(\boldsymbol{z}_{t_j}|\boldsymbol{x}_{t_j})] + \sum_{j=1}^{i}\big(log[p(\boldsymbol{x}_{t_j}|\boldsymbol{x}_{t_{j-1}},\boldsymbol{\theta},\boldsymbol{u}_{t_j},\Delta t_j)]\big) - log[p(\boldsymbol{z}_{t_{0:i}})] \\
&= \sum_{j=0}^{i} ObsCost(\boldsymbol{z}_{t_j},\boldsymbol{x}_{t_j}) + \sum_{j=1}^{i} DynCost(\boldsymbol{x}_{t_j},\boldsymbol{x}_{t_{j-1}},j) + Const \qquad (5.9)
\end{aligned}$$

If Gaussian assumptions are made about the distribution of observations with respect to state $p_{obs}(\boldsymbol{z}_{t_i}|\boldsymbol{x}_{t_i})$ and about the dynamics $\boldsymbol{p}_{dyn}(\boldsymbol{x}_{t_{i-1}}|\boldsymbol{x}_{t_{i-1}},\boldsymbol{\theta},\boldsymbol{u}_{t_i},\Delta t_i))$ then the observation and dynamics cost terms are least squares terms. So they would be written:

$$ObsCost(\boldsymbol{z}_{t_i},\boldsymbol{x}_{t_i}) = (\boldsymbol{z}_{t_i} - \boldsymbol{f}_{obs}(\boldsymbol{x}_{t_i}))^T S_{obs}(\boldsymbol{z}_{t_i} - \boldsymbol{f}_{obs}(\boldsymbol{x}_{t_i})) \qquad (5.10)$$

$$DynCost(\boldsymbol{x}_{t_i},\boldsymbol{x}_{t_{i-1}},i) = (\boldsymbol{x}_{t_i} - \boldsymbol{f}_{dyn}(\boldsymbol{x}_{t_{i-1}},\boldsymbol{\theta},\boldsymbol{u}_{t_i},\Delta t_i))^T S_{dyn}(\boldsymbol{x}_{t_i} - \boldsymbol{f}_{dyn}(\boldsymbol{x}_{t_{i-1}},\boldsymbol{\theta},\boldsymbol{u}_{t_i},\Delta t_i)) \qquad (5.11)$$

The observation weight matrix S_{obs} is again found here. Additionally, S_{dyn} is a cost weight matrix for the dynamics terms. The relative size of the norms of these two matrices determines the relative input of the dynamics and observation models in determining the optimal solution. If S_{dyn} became zero, for instance, then the dynamics would be ignored

[1]To put it another way: In the locally parametrised dynamics, all of the states in the trajectory are available to be varied to find a solution, whereas in the globally parametrised dynamics only the initial state can be varied.

and only observations considered in the trajectory optimisation problem. Conversely, if S_{obs} were zero, any arbitrary trajectory would only be weighted by its compliance with the laws of physics as specified by the dynamics model. Given the derivation of this optimisation problem in MAP terms, then these weights reflect the assigned uncertainty (variance) in these sources of information.

Velocity and translation term weighting. S_{dyn} needs discussing here because the state vector $\boldsymbol{x}_{t_i}$ usually contains a mixture of pose and velocity terms. The question of how to reconcile those terms against each other in this approach is the question of how to constitute the weight matrix S_{dyn}. One assumption involves making the matrix diagonal - essentially assuming noise in each state variable is independent in the dynamics model. Given such a weighting, other questions to consider are how to weight velocity against displacement terms and how to weight rotational against translational terms.

In this thesis the relative weight between velocity and displacement terms is calculated by making the assumption that the expected translational error accumulated by a given velocity error over Δt_i should be of the same order as the expected translational error between the two time-steps (that is, velocity error is only considered as a kind of translational error). For this argument to be made a little bit more clear it is assumed here that the state contains just velocity v and translation $\boldsymbol{T}$ (though relative rotational terms can be calculated using a similar argument):

$$\Delta \boldsymbol{x}_{t_i} = \left[\begin{smallmatrix} \boldsymbol{T}_{t_i} \\ \boldsymbol{v}_{t_i} \end{smallmatrix}\right] - \boldsymbol{f}_{dyn}(\boldsymbol{x}_{t_{i-1}}, \boldsymbol{\theta}, \boldsymbol{u}_{t_i}, \Delta t_i) = \left[\begin{smallmatrix} \Delta \boldsymbol{T}_{t_i} \\ \Delta \boldsymbol{v}_{t_i} \end{smallmatrix}\right] \tag{5.12}$$

Here, $\Delta \boldsymbol{x}_{t_i}$ is the difference between mean expected value of $\boldsymbol{x}_{t_i}$ (given the state at $\boldsymbol{x}_{t_{i-1}}$) and the actual value of $\boldsymbol{x}_{t_i}$. Then the argument would be that $\Delta \boldsymbol{T}_{t_i} \approx \Delta t_i \Delta \boldsymbol{v}_{t_i}$; i.e. that $\Delta \boldsymbol{T}_{t_i}$ and $\Delta t_i \Delta \boldsymbol{v}_{t_i}$ should be on the same order. Then the weights for the velocity $w_{\boldsymbol{v}}$ can be expressed in terms of the translational weight $w_{\boldsymbol{T}}$:

$$w_{\boldsymbol{v}} = \Delta t_i w_{\boldsymbol{T}} \tag{5.13}$$

Having discussed probabilistic formulations of the MAP problem tackled here, the exposition moves on to a discussion of the individual experimental scenarios.

5.3 Experiment: Simulated bouncing object

In this section, a set of experiments are described in which an object is simulated in tumbling motion, either in free-flight or bouncing. The primary aim of these experiments is to determine a preferred algorithmic approach to trajectory estimation from noisy data with a physics model included in the solution. The trajectory estimator is provided noise-contaminated observations of the full object pose and is tasked with reconstructing the full translational and rotational trajectory of the object[1].

The simulation of the object is a simple simulation; enough to capture free-flight in the presence of gravity, rotation with precession, and simple collisions with non-unity restitution. The collision detection and resolution were implemented only for sphere-plane collisions[2]. Translational motion between collisions is simply the solution of the laws of kinetic motion, and the rotational motion is simulated by numerical integration using the Runge-Kutta method.

The instantaneous state of an object is represented in terms of pose, with rotation a quaternion, and linear and angular velocity:

$$\boldsymbol{x}_{t_i} = \begin{bmatrix} T_i \\ q_i \\ v_i \\ \omega_i \end{bmatrix} \tag{5.14}$$

The same simulator is used to generate test trajectories and for fitting trajectories to them.

Once generated, noise is added to each pose in the trajectory to make an observation

[1]Such a scenario is analogous to the scenario that would be encountered if a pose recovery algorithm were run independently on each frame of a video and the best guess returned for that frame.

[2]A ball bouncing on a clear flat surface can be modelled adequately with a sphere-plane simulator: collision locations are always the same distance from the object centre and the collision normal always points towards the object centre.

of pose; that is, there is no model of process noise in this system; all the noise is in the observations. Therefore, the natural solution for fitting them is the globally parametrised cost function.

Two kinds of noise are introduced. The first kind of noise is Gaussian dispersion in both rotation and translation space. The second kind of noise is a uniform background noise over a domain of the observation space; outlier noise. An example of a simulated trajectory without Gaussian and outlier noise can be seen at the top of Figure 5.1 (green). An example of a simulated trajectory with noise added can be seen at the middle of Figure 5.1 (red). These two trajectories are superimposed at the top of Figure 5.2 (green and red).

5.3.1 Fitting algorithms

Two fitting algorithms were employed to fit trajectories to noisy sequences of pose observations. The first is the classic Levenberg-Marquadt non-linear optimisation technique using finite differences in the cost function. Because of the strongly non-linear, multi-optima nature of the problem, this algorithm only serves to find local solutions, so is called the "refinement" algorithm in this chapter. In order to fit the data robustly and to do global rather than local optimisation, the second fitting algoirthm is RANSAC, used in conjunction with the refinement optimisation.

5.3.2 Refinement algorithm

In order to find the initial state that best explains the observations, a Levenberg-Marquadt algorithm is used to refine the solution to a nearby optimum in the cost function. The details of the Levenberg-Marquadt algorithm as applied to trajectories can be found in Appendix D. The Levenberg-Marquadt algorithm iteratively calculates the value of the function to be optimised as well as its derivative, locally approximates the function by a polynomial model, and solves that polynomial. It does this until it reaches an optimum.

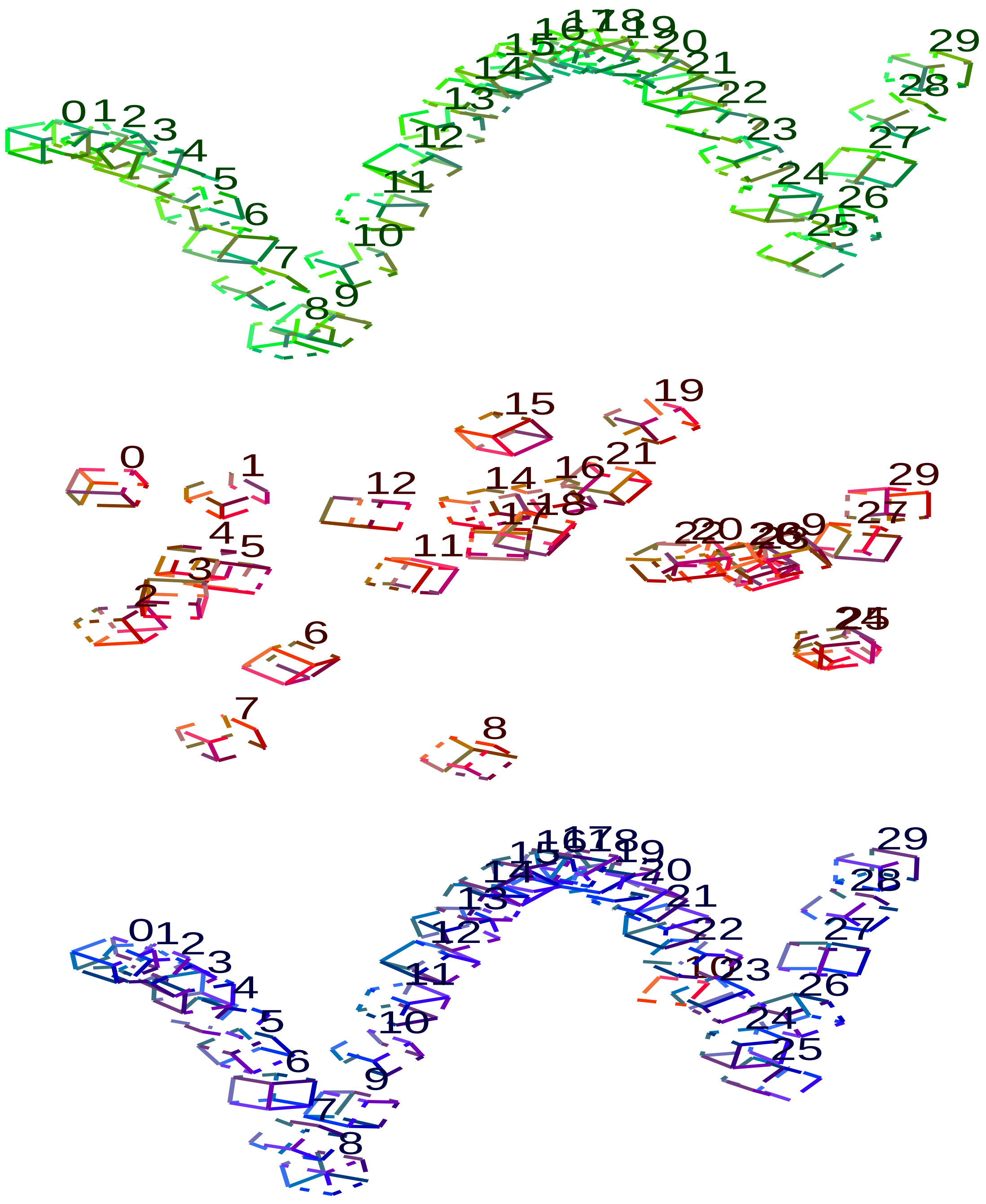


Figure 5.1: **Top** (green): A simulated trajectory. **Middle** (red): A simulated trajectory with added noise (Gaussian noise deviation $\sigma_T = 50cm$, outlier noise rate $\rho = 0.1$). **Bottom** (blue): A trajectory fitted using RANSAC and refinement with parameters reported in Table 5.4.

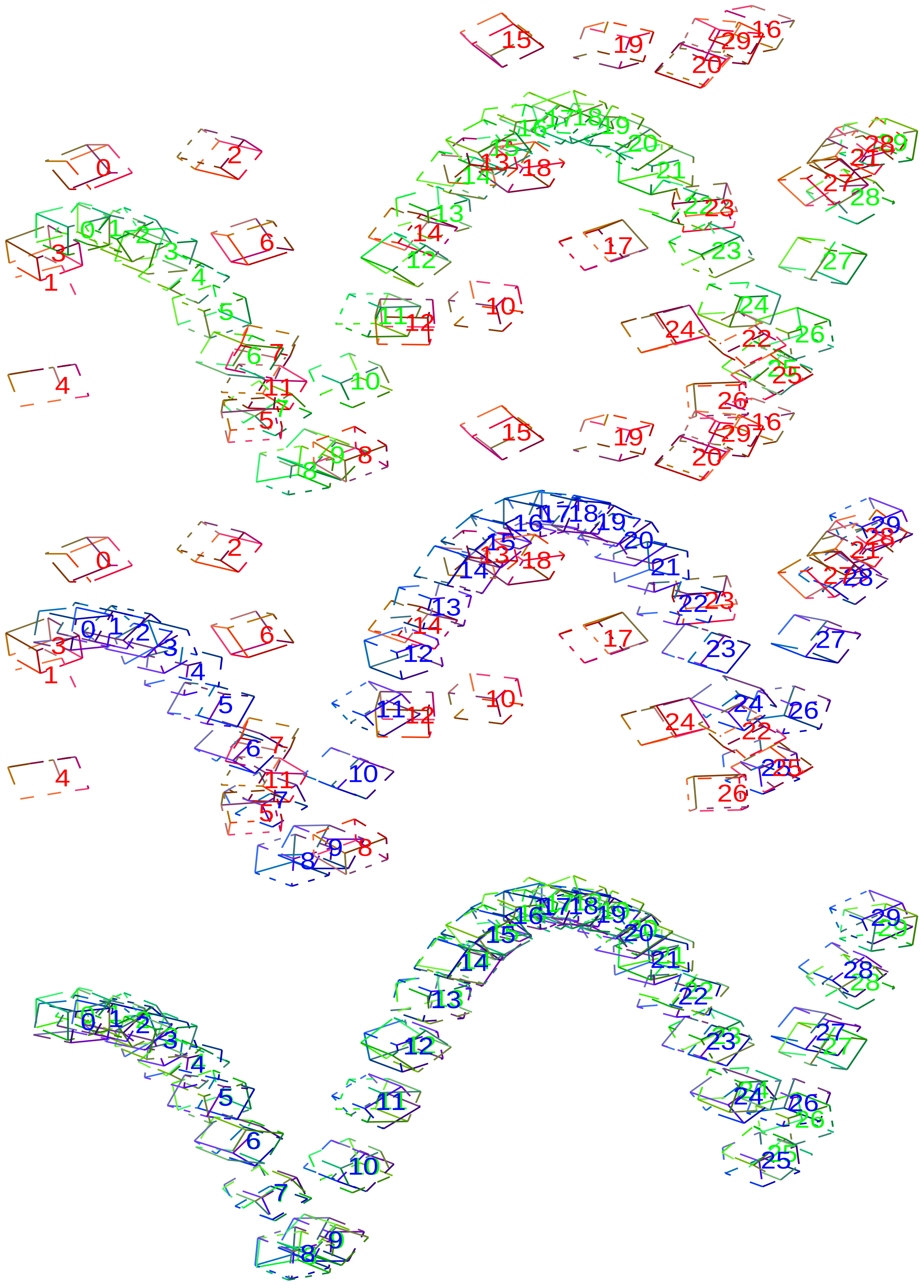


Figure 5.2: **Top**: Simulated deterministic trajectory (green) vs simulated trajectory with added noise (red). **Middle**: Simulated trajectory with added noise (red) vs fitted trajectory (blue).**Bottom**: Original simulated deterministic trajectory (green) vs fitted trajectory (blue). Where noise is added it includes Gaussian noise with a deviation of $\sigma_{\boldsymbol{T}} = 50cm$ and outlier noise with a rate of $\rho = 0.1$.

In the work here, the simulator underlying the dynamics function $\boldsymbol{f}_{dyn}()$ does not easily admit having a derivative calculated. Indeed, it is non-smooth wherever there is a transition in the number of bounces. Rather than analyse the function, the approach in this chapter to estimation is to use finite differences to approximate the Jacobian $\frac{d}{d\boldsymbol{x}_{t_0}}\boldsymbol{f}_{obs}(\boldsymbol{f}^i_{dyn}(\boldsymbol{x}^{j-1}_{t_0}))$. Finite difference simply approximates the derivative numerically by calculating the function along the axes of the parameter $\boldsymbol{x}_{t_0}$ and dividing into the step size. In this work, a fixed step size is used [1].

To evaluate a potential $\boldsymbol{x}_{t_0}$, the simulator is run to generate the full set of states $\boldsymbol{x}_{t_{0:i}}$[2]. The observations $\boldsymbol{z}_{t_i}$ are simply the truncation of the state vectors $\boldsymbol{x}_{t_i}$ to contain only the pose.

Given the simple simulator used in this section, the rotational and translational parameters are divorceable so rotation and translation can be estimated separately; this is the extreme case of exploiting the equation structure at each time point. This is in general a rather problematic assumption but it does reflect the fact that a lot of the time these variables are divorceable, as exploited in the long sequence motion estimation literature (see Section 3.5 for details).

5.3.3 RANSAC algorithm

The previous section described a refinement approach based on a non-linear optimisation algorithm that quickly finds an optimum; it is not guaranteed to find a global optima, however, and the least-squares cost function associated with it is not robust to outliers; that is, as estimated trajectories predict observations further from the observed location of an observation, the cost of an observation goes up quadratically; so outliers have a proportionally bigger effect on the solution in least-squares formulations - this is undesirable

[1]Practically speaking, using finite differences in this work means that for each parameter of the previous best solution $\boldsymbol{x}^{j-1}_{t_0}$ (of which there are 13; pose consisting of 7 and linear and angular velocity consisting of 6), that parameter is varied twice around the value in $\boldsymbol{x}^{j-1}_{t_0}$. This means that, apart from evaluating the original $\boldsymbol{f}_{obs}(\boldsymbol{f}^i_{dyn}(\boldsymbol{x}^{j-1}_{t_0}))$, 22 additional evaluations are done.

[2]Note that noise is not added to the simulator because noise is dealt with by attempting to minimise the distance between the deterministically reconstructed trajectory and the observed noisy trajectory.

and potential outliers should have a small contribution.

The RANSAC (RANdomised SAmple Consensus) algorithm works by instantiating possible solutions from *subsets* of data and then checking these solutions for fit against the rest of the data; with the aim being to find solutions that contain the maximum possible number of inliers. An inlier is defined as being an observation that, given a particular parameter estimate, is not further than a given threshold (*inlier threshold*) from the predicted observation. In order to calculate the possible solution an algorithm is required; here the algorithm is encapsulated by a function $\boldsymbol{f}_{INST}()$ from data-point sets to solutions. Here the solution is represented as a boundary condition and observations can be from throughout the sequence (from any time index i_m and any observation at that time with index k_m):

$$\boldsymbol{x}_{t_0}^n = \boldsymbol{f}_{INST}(\left\{ \boldsymbol{z}_{t_{i_1}}^{k_0}, \ldots, \boldsymbol{z}_{t_{i_m}}^{k_m}, \ldots, \boldsymbol{z}_{t_{i_M}}^{k_M} \right\}) \tag{5.15}$$

RANSAC instantiates a number of solutions using the supplied $\boldsymbol{f}_{INST}()$ and chooses the solution that induces the most number of inliers according to an inlier threshold C_{inl}[1]. The solution therefore satisfies:

$$\boldsymbol{x}_{t_0}^{RAN} = \arg\max_{\boldsymbol{x}_{t_0}^n} \sum_n inl\left(z_{t_{i_1}}^{k_0} - f_{obs}^k(\boldsymbol{f}_{dyn}^i(\boldsymbol{x}_{t_0}^n)) \right) \tag{5.16}$$

$$inl(A) = \begin{cases} 1 \text{ if } A \leq C_{inl} \\ 0 \text{ if } A > C_{inl} \end{cases}$$

Here, f_{dyn}^i is

In this experiment, $\boldsymbol{f}_{INST}()$ takes a sample of two poses and solves for velocity assuming free flight with gravity kinematic equations. Since this solution is almost always wrong when a collision is involved, the second sample is taken from an exponential distri-

[1]In the typical RANSAC formulation, from the expected inlier rate of a correct solution and a desired probability of finding a match, stopping conditions can be determined: the inlier threshold mentioned above, a maximum number of iterations to try, and a minimum number of inliers considered sufficient for termination. In the current experiment, the first and last parameter are not employed, so that a clear comparison can be made between the iterations required of RANSAC and that of refinement.

bution around the time-point of the first (rounded to the nearest time-point containing an observation). The number of rotations that occur between observations is also not derivable from the observations, so this integer is similarly sampled from an exponential distribution around zero. Therefore, the RANSAC instantiation function $\boldsymbol{f}_{INST}()$ is not completely deterministic.

Once the velocities are obtained, because the simulation is simple, it can be reversed (the coefficient of restitution and velocities are merely inverted; the gravity force remains the same), and the initial parameters of the guess trajectory $\boldsymbol{x}_{t_0}$ obtained as well as the full trajectory of states $\boldsymbol{x}_{t_{0:i}}$. These states can be truncated to obtain observations $\boldsymbol{z}_{t_{0:i}}$ and the number of inliers counted using the function $inl()$ described above.

Generally, the RANSAC result is followed by an extra refinement stage that only considers the inliers; this refinement stage can add more inliers online if the changing fit brings the trajectory closer to the observations.

5.3.4 Measuring error

A well-motivated way of reconciling translational and rotational components of pose error is derived in Appendix E and is repeated here:

$$E_{\boldsymbol{T},R_{\theta,\boldsymbol{\Omega}}} = M\left|\boldsymbol{T}\right|^2 + 2(1-\cos(\theta))\boldsymbol{\Omega}^T\mathcal{I}\boldsymbol{\Omega} \tag{5.17}$$

This error measure is motivated by the assumption that the pose error in an object is the sum of the translational errors of the particles or points from which the object is made.

In this section, this is the error measure that is used in fitting trajectories to initial parameters; it is summed across the sequence of poses. It is also possible to use it to obtain a measure of fit to ground truth, though in this section it is opted to present translational and rotational results separately.

Varying parameter	Possible values
Fitting algorithm	Refinement Only RANSAC + Refinement
Ground plane	No ground plane (free flight) Ground plane (bounces)
Sequence length	From 2 frames Step 15 frames To 72 frames
Gaussian noise $\sigma_{\boldsymbol{T}}$	From $0cm$ Step $15cm$ To $160cm$
Outlier rate ρ	From 0.0 Step 0.1 frames To 1.0

Table 5.3: This table reports the conditions that are compared in the first experiment reported in this chapter, comparing algorithms on the problem of estimating motion of a simulated bouncing object.

5.3.5 Experimental setup

In these experiments, each of the two algorithms (the refinement algorithm and RANSAC and refinement combination algorithm discussed above) were used on two different conditions - with and without the presence of a ground plane to induce bounces. For each condition a range of different noise conditions (Gaussian noise and outlier noise) were added to the simulator to examine how the different algorithms fared under increasing noise. When the noise parameters are not being varied, a small amount of Gaussian and outlier noise is added. Also a variety of different sequence lengths were examined. The experimental conditions are summarised in Table 5.3.

Table 5.4 gives the values of the parameters of the simulator, as well as of the two fitting algorithms used.

5.3.6 Results

Figure 5.3 contains the results of the experiment detailed here on the simulated problem; of varying sequence length, Gaussian noise, outlier noise as well as the effect of adding

Simulation parameters		
Parameter	How obtained	Value
Object radius	Arbitrary	$50cm$
Maximum sequence distance	Arbitrary	$1000cm$

Refinement algorithm parameters		
Parameter	How obtained	Value
Maximum iterations	Set conservatively	100
Finite differences step	Set conservatively	10^{-3}
Stopping tolerance	Set conservatively	10^{-7}

RANSAC algorithm parameters		
Parameter	How obtained	Value
Maximum iterations	From experience	100
Inlier threshold	Informal experiments	140
Minimum inlier count	Informal experiments	$\frac{(t_f - t_i)}{5t_s}$

Table 5.4: Parameters of simulation and motion estimation algorithms in MATLAB simulation scenario

bounces and the effect of algorithm choice.

Given the simple simulation used here, the performance of the rotational component is independent of the presence of bounces.

Translation. With respect to translation, the refinement algorithm degrades in the presence of bounces; conversely the RANSAC algorithm does not degrade in the presence of bounces. In extremely high noise (outlier rate greater than $\rho = 0.8$ and Gaussian noise greater than $\sigma_{\boldsymbol{T}} = 120cm$), RANSAC degrades, but for lower levels of noise, RANSAC outperforms refinement alone. Outliers affect both techniques worse than Gaussian noise, though RANSAC degrades worse than refinement alone with very high levels of Gaussian noise. There is an interaction between Gaussian noise and the presence of bounces that affects the refinement algorithm negatively. Long sequences decrease the average error across the sequence for both algorithms, probably due to the larger amount of information allowing for improved discrimination. With these results, note that there is a background outlier rate of $\rho = 0.1$; without outliers, the refinement algorithm is just as powerful as RANSAC on the translational results.

Rotations. The RANSAC algorithm also handles rotations better; in the presence of Gaussian and outlier noise it eventually degrades to the performance of the refinement algorithm, but this performance is already almost the worst possible performance (there is an upper limit on the rotation error). Long sequences again assist in the accuracy of the resulting estimate. Reducing the outlier rate to zero does not fix the refinement algorithm.

5.3.7 Experiment discussion

The performance advantage of RANSAC is due to its robustness to outliers as well as its effectiveness in finding the right (global) optimum. Since RANSAC will drop some points from the refinement stage at high levels of Gaussian noise, it loses some information and refinement can perform better in high levels of Gaussian noise, but only for free-flight where there are not large local optima to cause even more trouble for the refinement algorithm.

The negative influence on translation error of the interaction between Gaussian noise and the existence of bounces for the refinement algorithm is interesting. Theoretically, as the time-step between observations goes to zero there are no non-global local optima in the cost function even with bounces; these local optima occur only when there are very few observations. This can be seen in the results: with zero outliers, the refinement algorithm is as good as RANSAC, even with bounces. This implies that RANSAC's ability to beat global optima is not by itself the cause of its success in tracking object translation; it is its ability to reject outliers appropriately.

In rotation error the refinement algorithm fails to do well even without outliers added. This can be attributed to local optima. Even with an infinity of observations, local optima can exist in the cost function. To see this, imagine that the rotation axis is fixed and the original rotation amount is known to be zero so that the rotation error is $1-\cos(t\theta_0^{guess}-t\theta_0)$. Integrating that with respect to t gives a function with many optima

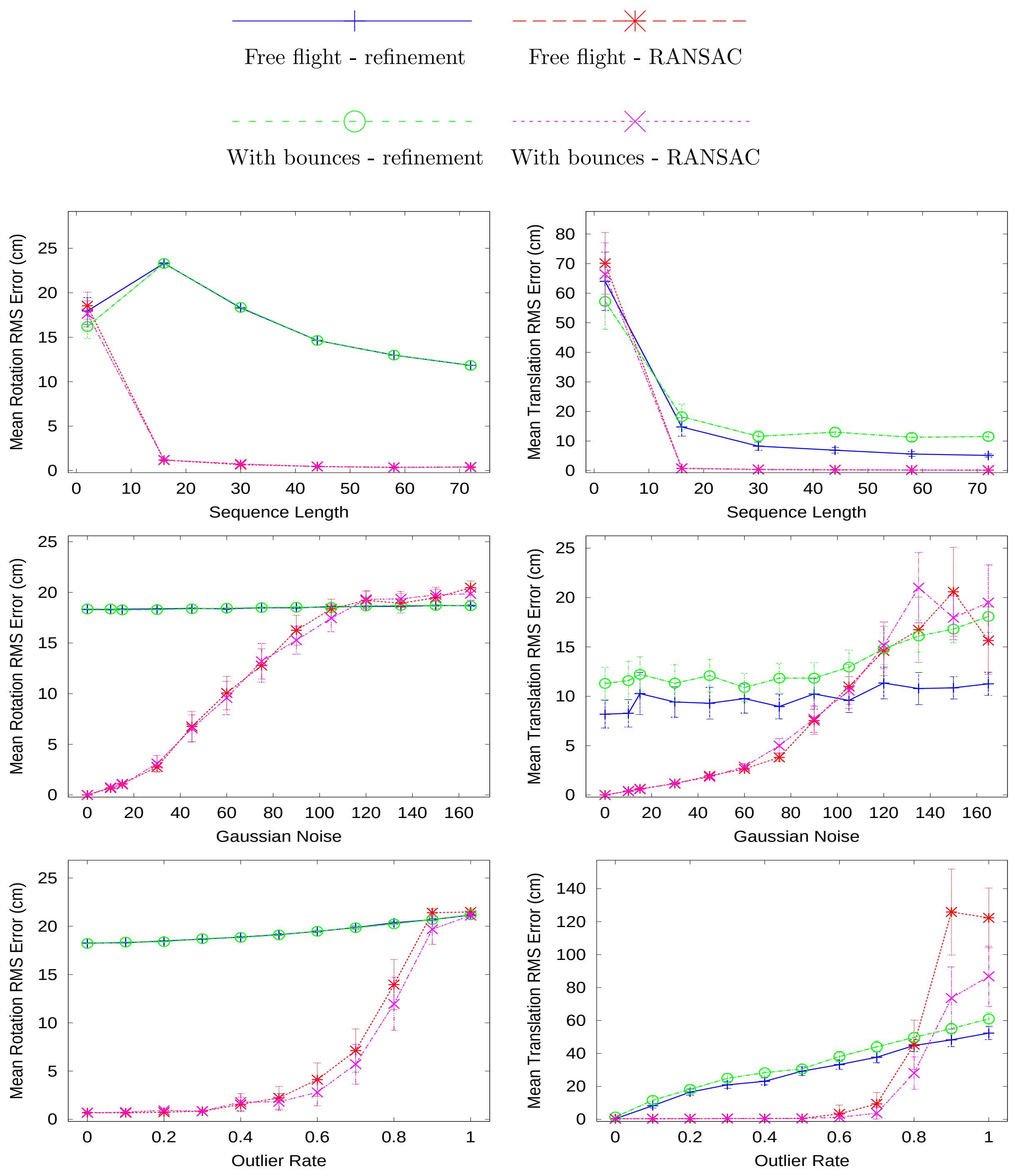


Figure 5.3: Effect of fitting algorithm and effect of bounces against varying noise levels and sequence length. Two fitting algorithms (refinement and RANSAC) are tested on two different kinds of trajectory (free flight and with-bounce). Each of the above charts contains a line for each combination of algorithm and trajectory kind - they key is found at the top. **Left:** Charts on the left report the rotation error. **Right:** Charts on the right report the translation error. **Top:** The top charts report algorithm performance as sequence length is increased. **Middle:** The middle charts report algorithm performance as Gaussian noise is added. **Bottom:** The bottom charts report algorithm performance as outlier noise is added. When not being varied, the sequence length is 30 frames, the isotropic Gaussian noise deviation $\sigma_T = 10cm$, and the outlier rate $\rho = 0.1$. As the sequence length increases, additional bounces occur sporadically. Each data-point contains 50 algorithmic runs. RMS error is the average fitting error per time-step. The translation and rotation error are as given in the second term of Equation E.16, with a constant density assumed, making the error in units of distance.

$$\begin{aligned}E(t, \theta_0^{guess}, \theta_0) &= 1 - \cos(t\theta_0^{guess} - t\theta_0)\\ E(t_l, \theta_0^{guess}, \theta_0) &= \int_t^{t_l} \left[1 - \cos(t\theta_0^{guess} - t\theta_0)\right] dt\\ &= t_l - \frac{\sin(t_l(\theta_0^{guess} - \theta_0))}{t_l(\theta_0^{guess} - \theta_0)}\end{aligned} \tag{5.18}$$

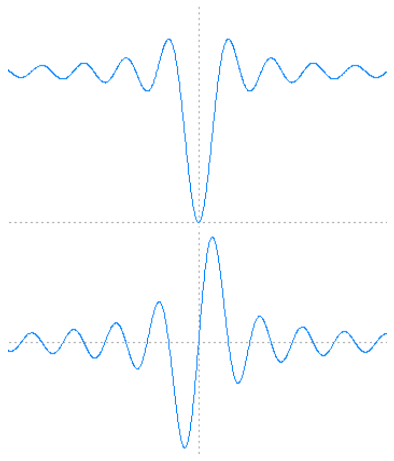


Figure 5.4: An illustration of local optima in fit to a rotation, assuming a fixed rotation axis and translational error, as well as known initial rotation. **Top:** The error measure per observation at time t (given a true rotation speed θ_0 and a guess rotation speed θ_0^{guess}, and the error measure assuming infinite observations are known across the sequence length t_l. The derivative of this with respect to θ_0^{guess} can be examined for zeroes to get optima; it is not shown here. **Bottom:** An illustration by chart of how the cost function with respect to θ_0^{guess} (top) and its derivative with respect to θ_0^{guess} given a t_l of 1 and an θ_0 of 0 (bottom).

as can be seen by differentiating it again with respect to θ_0^{guess}. Figure 5.4 illustrates that this cost function has minima, at least when restricted to this dimension.

Finally, before moving on to the next experiment: Although it is feasible that a real-world problem could involve obtaining noisy poses (for example, using a tracking-by-detection algorithm to obtain these poses) as is done in this experiment, the physics model involved in this experiment is only really applicable to smooth spheres moving in a vacuum[1]. The remaining experiments in this thesis use the more fully-featured PhysX physics engine ("PhysX Features," 2010).

[1]It is an open question as to whether better modelling of the physics is necessary in the real-world.

5.4 Experiment: Bouncing ball from colour

The previous section described the results of experiments with a simulated bouncing object. This section takes that paradigm and applies it to objects bouncing in the real-world and compares its performance using different dynamics models.

The constant displacement and constant velocity dynamics models, as well as a no-dynamics condition, are compared to physics based dynamics models - with global parametrisation and local parametrisation.

In this section, the motion of objects is estimated by exploiting knowledge of their dynamics, as well as observations of object location from colour. Object motion is restricted to a plane parallel to the image plane, thereby significantly simplifying the problem; the problem is again made complex due to the complex physical motion of real-physical objects being inferred. Motion estimation with and without physics is applied to the four target videos illustrated in Figure 5.5.

Since the trajectories being recovered in this experiment occur in a plane parallel to the image plane, and object rotation is ignored, only four parameters are required to describe the state $\boldsymbol{x}_{t_i}$ of the target object at any point in time: the two-dimensional location $\boldsymbol{T}_i$ and the two-dimensional velocity $\boldsymbol{v}_i$:

$$\boldsymbol{x}_{t_i} = \begin{bmatrix} T_i^1 \\ T_i^2 \\ v_i^1 \\ v_i^2 \end{bmatrix} \tag{5.19}$$

The locally parametrised dynamics derived in Section 5.2.2 are incorporated into the refinement solution in order to deal better with model mismatch. RANSAC is always used, but there is no corresponding RANSAC solution utilising theoretical process noise (i.e. the locally parametrised approach).

In this section, first the observation model is described, then the algorithms and dynamics models, experimental setup and finally results.

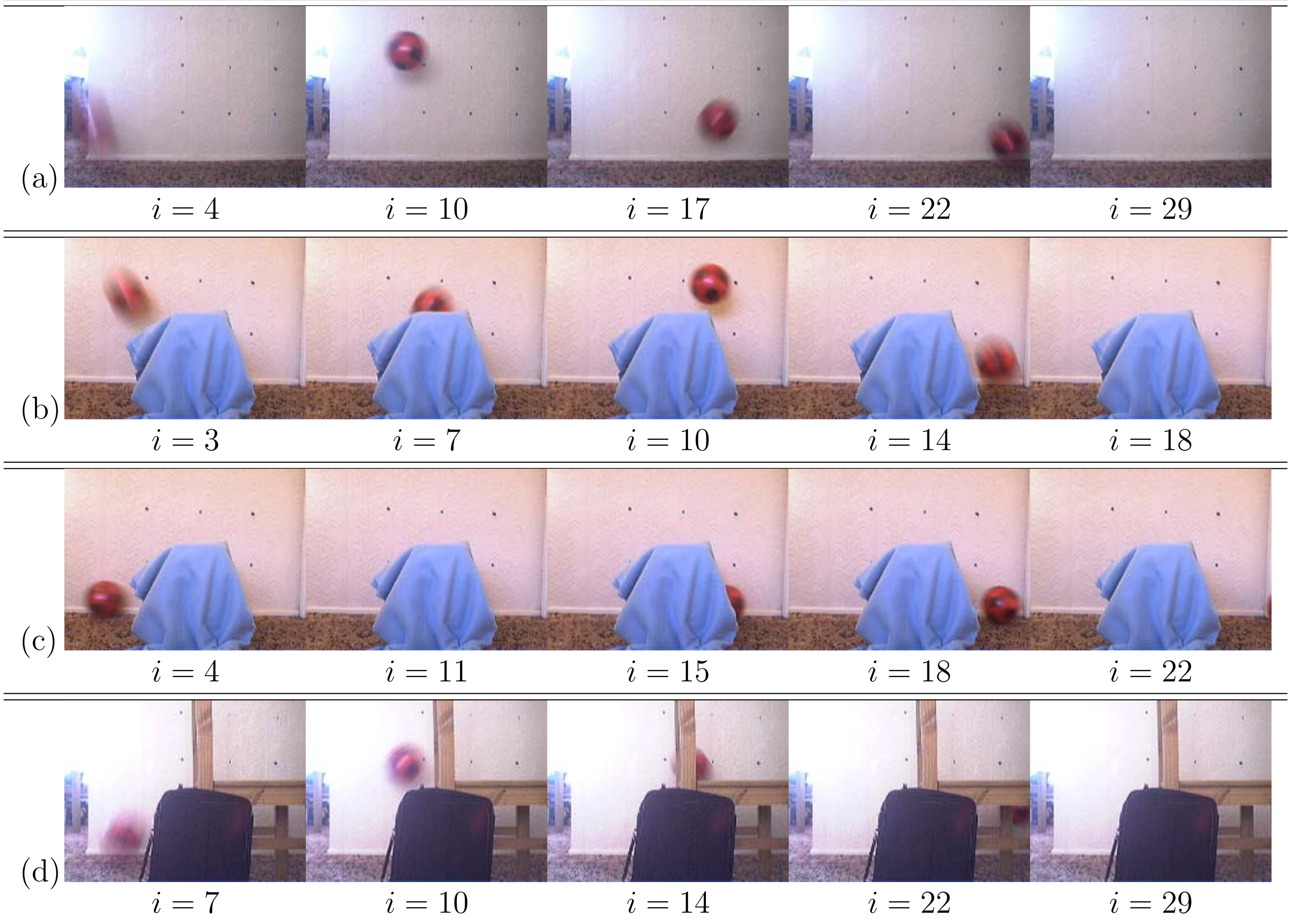


Figure 5.5: Sample frames from the 4 videos tested in this chapter. Each frame is labelled by its frame number. **First row:** A ball bouncing through the frame; some blur and glare in the frame and the ball spends some time off-frame. **Second row:** A ball bouncing behind an occluder. **Third row:** A ball rolling behind an occluder. **Fourth row:** A ball bouncing behind some occluders, with a distractor in the middle of the frame, and blur and glare.

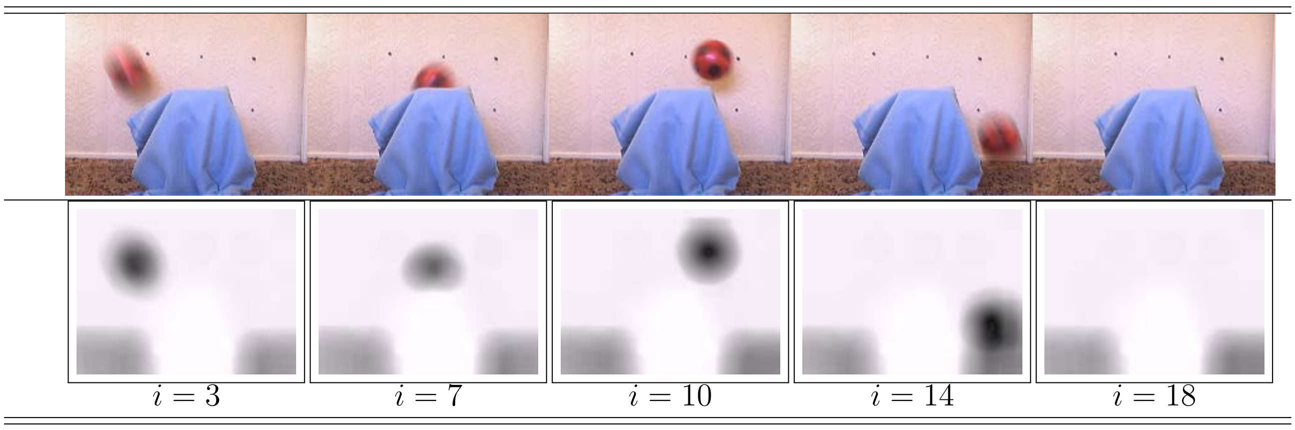


Figure 5.6: Sample frames from the bounce with occlusion (b) video tested in this chapter before and after processing by the template matching routine to find a match score and subsequent smoothing of the match image. **Top:** The original video frames. **Bottom:** The match map. The intensity of each pixel matches the match score (darker is a better match). Each frame is labelled by its frame number.

5.4.1 Observations

Each pixel location in each image in the image sequence has a match score associated with it according to how well the local colour histogram matches the colour histogram of the object template, normalised by the background histogram. This forms a match score map of each image. Figure 5.6 illustrates the match score map on one of the videos.

Observations consist of possible object locations, extracted by finding match optima. There can be multiple possible locations in each frame, but each observation has an associated weight according to the score of the match with the object.

5.4.2 Calculating the match score

The match score between a pixel in the image being processed and the target object is based on the colour histogram in the target object template and a fixed-radius circular region around the pixel to be matched. The histograms are collected in normalised red/green space, and the target histogram bins divided by the corresponding bins in the background histogram to create a histogram weighted away from background distractors in the image. Histograms are subsequently normalised.

The brightness normalised image is calculated as follows, where y is the pixel location, I_r is the normalised red channel, I_g is the normalised green channel and I_R,I_G and I_B are the original RGB channels[1]:

$$\begin{aligned} I_r(\boldsymbol{y}) &= \frac{I_R(\boldsymbol{y})}{I_R(\boldsymbol{y}) + I_G(\boldsymbol{y}) + I_B(\boldsymbol{y})} \\ I_g(\boldsymbol{y}) &= \frac{I_G(\boldsymbol{y})}{I_R(\boldsymbol{y}) + I_G(\boldsymbol{y}) + I_B(\boldsymbol{y})} \end{aligned} \tag{5.20}$$

For a given pixel $\boldsymbol{y}$ in a given image I, the histogram bin indices (b_R, b_G) of the colour in that pixel can be found by applying the following indicator function (B_G and B_Y are the number of bins for the red and green components of the colour space, so that the total number of bins is $B_G \cdot B_R$):

$$InBin(b_R, b_G, \boldsymbol{y}, I) = \begin{cases} 1 \text{ if } \substack{I_g(\boldsymbol{y}) \geq \frac{b_G}{B_G}, I_g(\boldsymbol{y}) < \frac{b_G+1}{B_G}, \\ I_r(\boldsymbol{y}) \geq \frac{b_R}{B_R}, I_r(\boldsymbol{y}) < \frac{b_R+1}{B_R}} \\ 0 \text{ otherwise} \end{cases} \tag{5.21}$$

Since the colour channels are normalised, $I_r(\boldsymbol{y})$ and $I_g(\boldsymbol{y})$ range between 0 and 1. It is simple after this to represent a histogram $H_{r,y,I}()$ of the colour in a radius r around a point y in an image I in terms of these values and the bin address (b_R, b_G). $\boldsymbol{Y}_I$ is the set of pixel centres in the image.

$$H_{r,\boldsymbol{y},I}(b_R, b_G) = \sum_{\substack{\boldsymbol{y}_j \text{ s.t.} \\ \boldsymbol{y}_j \in \boldsymbol{Y}_I \text{ and} \\ |\boldsymbol{y}_j - \boldsymbol{y}| < r}} InBin(b_R, b_G, \boldsymbol{y}_j, I) \tag{5.22}$$

This histogram is normalised so that the sum of bin values is 1:

$$H^*_{r,\boldsymbol{y},I}(b_R, b_G) = \frac{H_{r,\boldsymbol{y},I}(b_R, b_G)}{\sum_{\beta_r,\beta_g} H_{r,\boldsymbol{y},I}(\beta_r, \beta_g)} \tag{5.23}$$

[1]In other applications luminance is often calculated as a weighted sum of the RGB components in order to reconstruct subjective brightness for humans, though for this application there is no expected gain from weighting the components differently.

In order to acquire a histogram template of an object H_{obj}, this histogram is accumulated over a small number of frames I_j where the object location $\boldsymbol{y}_j$ is known (r_{ob} is the object's radius).

$$H^*_{obj}(b_R, b_G) = \frac{\sum_j H^*_{r_{ob},\boldsymbol{y}_j,I_j}(b_R, b_G)}{\sum_j 1} \tag{5.24}$$

In order to obtain a match score for each pixel in an image that might contain the object, the standard measure of histogram similarity, the Bhattacharyya distance is used between the histogram calculated from the neighbourhood of that pixel and the histogram of the target object template. The Bhattacharyya distance is based on the sum of root products of the magnitude of the paired bins in the histograms:

$$d_{Bhatt}(H^*_1, H^*_2) = -log\sqrt{\sum_{(b_R,b_G)} \sqrt{(H^*_1(b_R, b_G)H^*_2((b_R, b_G)))}} \tag{5.25}$$

The pixel match score at each point y for image I (with object radius r implicit) is therefore:

$$I_{match}(\boldsymbol{y}, I) = d_{Bhatt}(H^*_{obj}, H^*_{r,\boldsymbol{y},I}) \tag{5.26}$$

5.4.3 Creating an observation sequence

After calculating a match map, it is smoothed and then the best points can be chosen greedily, suppressing points within r distance from the best points already chosen from a frame.

In the present experiment, only one point is obtained from each frame. Once the best point for each frame is chosen according to the Bhattacharyya distance, as discussed in the previous section, a weight is assigned based on the match distance.

So the output of the image processing is a sequence of sets of potential image locations and associated weights. In this experiment there is only one image location $\boldsymbol{z}_{t_i}$ and associated weight w_{t_i} returned per time-point t_i, so:

$$\boldsymbol{z}_{t_i} = \arg\max_{\boldsymbol{y}} I_{match}(\boldsymbol{y}, I_i) \tag{5.27}$$

$$w_{t_i} = I_{match}(\boldsymbol{z}_{t_i}, I_i) \tag{5.28}$$

When compared to a formulation that uses the match map directly as a cost function (so that rather than there being a discrete number of observations, the likelihood of each possible state is derived directly from the histogram match), this formulation based on image distance to discrete observations offers a significantly smoother and less peaky likelihood function, appropriate for optimisation methods based on exploiting the local shape of the function, as are used here. Multiple objects and data association can easily be obtained by considering multiple observations and weights per frame and considering observation/object associations.

The observation associated with a state $\boldsymbol{x}_{t_i}$ is the projection of the 2D object coordinates $\left[\begin{smallmatrix} T^1 \\ T^2 \end{smallmatrix}\right]$ according to focal length and unit transform scaling factor f and optical centre $\left[\begin{smallmatrix} y_1^C \\ y_1^C \end{smallmatrix}\right]$ that make up the observation function $\boldsymbol{f}_{obs}$:

$$\begin{aligned} \boldsymbol{f}_{obs}(\boldsymbol{x}_{t_i}) &= \boldsymbol{f}_{obs}\left(\begin{bmatrix} T_i^1 \\ T_i^2 \\ v_i^1 \\ v_i^2 \end{bmatrix}\right) \\ &= \left[\begin{smallmatrix} fT_i^1 + y_1^C \\ fT_i^2 + y_2^C \end{smallmatrix}\right] \end{aligned} \tag{5.29}$$

5.4.4 Fitting algorithms

In this experiment, RANSAC and a subsequent refinement algorithm are employed to fit a trajectory to observations. There are two variants of this procedure according to the parametrisation of the trajectory, and different dynamics models are compared; physics-based models and models more commonly used (the assumption of constant pose with dispersion noise, and constant velocity with dispersion noise).

Dynamics	Parametrisation
No dynamics (observations assumed correct)	Local parametrisation
Constant displacement dynamics	Local parametrisation
Constant velocity dynamics	Local parametrisation
Physics-based collision dynamics	Global parametrisation
Physics-based collision dynamics	Local parametrisation

Table 5.5: This table reports the conditions that are compared in the first experiment reported in this chapter, comparing algorithms on the problem of estimating motion of a simulated bouncing object.

The parametrisation of the trajectory is either "global" or "local" as explained in Sections 5.2.1 and 5.2.2[1].

Since the naive dynamics models are seldom sufficient to model the full trajectory of an object through a scene, this experiment only attempts to pair the simple dynamics models (used for comparison with the physics-based dynamics model introduced here) with a local parametrisation of the trajectory. These experimental conditions are listed in Table 5.5.

The physics simulator encapsulated in the dynamics $\boldsymbol{f}_{dyn}$ is PhysX ("PhysX Features," 2010). The simulator includes a model of the balls in the scene constrained to be in the plain parallel to the image plane[2], as well as a ground plane. The coefficients of restitution and friction and linear damping were chosen from a small number of manual fits on a different video using a graphical tool. They are given in Table 5.5.

5.4.5 RANSAC

RANSAC on this problem proceeds much as described for the previous experiment in Section 5.3, except that because the physical simulator used here is not completely reversible,

[1]In the global case the whole trajectory is parametrised in terms of the initial conditions (the state at time $t_i = 0$) and the full trajectory can subsequently be reconstructed using the deterministic dynamics; that is, it is assumed that there is no dynamics noise at all. In the local case, the trajectory is parametrised in terms of the state of the object at each time-point. Therefore, the estimation can admit trajectories that are a certain distance from valid according to the dynamics model.

[2]In the case of multiple balls, multiple layers can exist, with each layer having a different depth coordinate.

reconstruction of reverse-time trajectories from samples is a bit more complicated. Further, observation weights are made use of here.

In brief, RANSAC works by reconstructing trajectories on the basis of subsamples of the data and then checking those trajectories against the rest of the data.The most important thing to do in order to use RANSAC, therefore, is to define the function $\boldsymbol{f}_{INST}$ that instantiates a trajectory from a subset of data. In the case of constant velocity dynamics, this is clearly to fit a linear trajectory to observations of the ball, $\boldsymbol{z}_{t_i}$. In the case of constant displacement dynamics, the result turns out to be the same between the two observations, but before the first observation and after the last observation the target object is assumed to remain stationary (and velocities are not instantiated). The number of observations used is itself sampled so that it it is sometimes more than 2, and multiple segments are fit.

For physics dynamics involving collisions, the assumption is made that the two observations are part of the same free-flight period, so no collision occurs (and so that only two observations are necessary to reconstruct the instantaneous parameters of the object). This does mean that observations far apart in time are less likely to result in accurate trajectory reconstructions, so observations are sampled in such a way that they are likely to be close to each other in time. This is similar to the observation made by Myatt et al. (2002) that in general, sampling sets of data-points close to each other in the dimension of interest is more likely to come up with sets of inliers. From the assumption of free-flight, it is easy to find the 2D pose and velocity of the object at one of the time points.

From this instantaneous estimate $\boldsymbol{x}_{t_a}$, it is possible using the simulation function $\boldsymbol{f}_{dyn}$ to reconstruct the full trajectory from that time-point onward. States ahead in time of t_a need to be obtained by simulating with $\boldsymbol{f}_{dyn}$ and states previous in time with respect to t_a need to be obtained by simulating with a time-reversed simulation function $\boldsymbol{f}_{-dyn}$.

For states before the instantaneous estimate $\boldsymbol{x}_{t_a}$ the problem is not so simple as $\boldsymbol{f}_{dyn}$ is not easily invertible in its previous-state parameter $\boldsymbol{x}_{t_{i-1}}$. The theoretical reason for this is that damping and friction can lead to a loss of information, particularly static friction -

for example, a stationary object could have come to rest from any direction. In the work here, an approximate solution is used, where the simulator is run with reversed velocity and restitution coefficient, no linear damping and no friction.

With this approach, the resulting trajectory can be parametrised by the state at a single time point, but that state is not necessarily the state $\boldsymbol{x}_{t_0}$ at time t_0. As such, if the trajectory is to be globally parameterised, the time point $\boldsymbol{x}_{t_a}$ is used to parametrise it. This point of parametrisation can be moved during the refinement phase if the move does not negatively affect the cost of the trajectory and/or the number of inliers.

Having defined $\boldsymbol{f}_{INST}$ it is a matter of testing each instansation for inlier fit; that is the number of observations that fit the trajectory. In this procedure, the inlier threshold is used to determine whether an observation is an inlier with respect to a trajectory, and each inlier observation counted once to determine the number of inliers. Using the weights w_{t_i}, this procedure is modified slightly in two ways:

- The inlier threshold distance is scaled by the weight, so that trajectories close to more highly weighted observations are more likely to contain them as inliers.
- The number of observations that each inlier observation counts for is scaled by the weight, so that more highly weighted observations contribute more to the inlier count.

Each modification has a slightly different effect. The scaling of the inlier threshold decreases the necessary accuracy of observations if there is good evidence of the target object nearby. This is a somewhat unconventional approach but it does allow the trajectory estimate to be more inexact under controlled conditions. The scaling of the inlier count is more conventional and its motivation is to give higher weight to trajectories that contain good observations.

5.4.6 Refinement

Once RANSAC has initialised a trajectory, it can be improved by attempting to further optimise the trajectory to minimise a cost function based on observation error and deviation from dynamics.

Trajectory parametrisation. Two different cost functions are used depending on how the trajectory is parametrised. The general form of these cost functions are described in Sections 5.2.1 and 5.2.2.

If the trajectory is globally parametrised, there is no dynamics cost since the trajectory is constrained to fit the dynamics:

$$Cost(\boldsymbol{x}_{t_a}) = \sum_{j=0}^{i} ObsCost(\boldsymbol{z}_{t_j}, \boldsymbol{x}_{t_j}) \tag{5.30}$$

$$\boldsymbol{x}_{t_j} = \begin{cases} \boldsymbol{f}_{dyn}(\boldsymbol{x}_{t_a}, \boldsymbol{\theta}, \boldsymbol{u}_{t_{a:j}}, \Delta t_{a:j}) \text{ if } j > a \\ \boldsymbol{f}_{-dyn}(\boldsymbol{x}_{t_a}, \boldsymbol{\theta}, \boldsymbol{u}_{t_{j:a}}, \Delta t_{j:a}) \text{ if } j < a \\ \boldsymbol{x}_{t_a} \text{ otherwise} \end{cases} \tag{5.31}$$

The observation cost of each frame is the match-score weighted distance between the orthographic projection of the object location and the best object match:

$$ObsCost(\boldsymbol{z}_{t_i}, \boldsymbol{x}_{t_i}) = (\boldsymbol{z}_{t_i} - \boldsymbol{f}_{obs}(\boldsymbol{x}_{t_i}))^T \begin{bmatrix} w_{t_i} & 0 \\ 0 & w_{t_i} \end{bmatrix} (\boldsymbol{z}_{t_i} - \boldsymbol{f}_{obs}(\boldsymbol{x}_{t_i})) \tag{5.32}$$

Thus the match-score derived weight of the observation reflects the supposed variance in its distribution and consequently alters its impact on the overall cost score[1].

If the trajectory is locally parametrised then the cost consists of a weighted sum of

[1]The assumption that the theoretical variance in object location matches its match score is more than tenuous, but it works in practice to tie together the weighting of observations with the MAP formalism.

observation and dynamics costs with no constraints:

$$Cost(\boldsymbol{x}_{t_{0:i}}) = \sum_{j=0}^{i} ObsCost(\boldsymbol{z}_{t_j}, \boldsymbol{x}_{t_j}) + \sum_{j=1}^{i} DynCost(\boldsymbol{x}_{t_j}, \boldsymbol{x}_{t_{j-1}}, j) \tag{5.33}$$

The dynamics cost at time t_i is the distance between the expected state at time t_i given the estimated state at time t_{i-i}, $\boldsymbol{f}_{dyn}(\boldsymbol{x}_{t_{i-1}}, \boldsymbol{\theta}, \boldsymbol{u}_{t_i}, \Delta t_i))$, and the estimated state at time t_{i-1}:

$$\begin{aligned} &DynCost(\boldsymbol{x}_{t_i}, \boldsymbol{x}_{t_{i-1}}, i) \\ &\qquad = (\boldsymbol{x}_{t_i} - \boldsymbol{f}_{dyn}(\boldsymbol{x}_{t_{i-1}}, \boldsymbol{\theta}, \boldsymbol{u}_{t_i}, \Delta t_i))^T S_{dyn} (\boldsymbol{x}_{t_i} - \boldsymbol{f}_{dyn}(\boldsymbol{x}_{t_{i-1}}, \boldsymbol{\theta}, \boldsymbol{u}_{t_i}, \Delta t_i)) \end{aligned} \tag{5.34}$$

The weight S_{dyn} needs to mediate between the weight of location $\begin{bmatrix} T_i^1 \\ T_i^2 \end{bmatrix}$ and velocity $\begin{bmatrix} v_i^1 \\ v_i^2 \end{bmatrix}$ and also to mediate between the weight given to observations and that given to dynamics. In the latter case, since the translational residual in dynamics and the translational residual in observations exist in the same unit space, the natural approach is to weight them equally. However, they do need to be transferred into the same units[1].

Regarding velocity, the approach taken is to weight a velocity residual according to the number of time-steps of translational information that would be required to calculate the velocity; since 2 time-steps are generally sufficient, the weight is 2. In this experiment, the velocity weight is multiplied by the inverse of the frame-rate $\frac{1}{\Delta t_i}$, that would by itself obtain a velocity cost equivalent to the expected accumulation of translation cost over one second, which in this experiment equates to a multiplier of $15\times$ since the videos in this experiment were taken with a USB camera at 15 frames per second. This choice is relatively ad-hoc and was done because the velocity part of the estimation problem was deemed to be less constrained by observations and so ought to be more constrained by the dynamics[2]. The multipliers used in experiments are described in Table 5.6.

[1]In this experiment dynamics is multiplied into image coordinates by multiplying by the square of the pixels per unit metre (square because the weight is multiplied into a squared error).

[2]The practical effect of different velocity multipliers should rather be experimentally determined.

So, in general in this experiment:

$$S_{dyn} = \begin{bmatrix} \frac{Dyn}{Obs} & 0 & 0 & 0 \\ 0 & \frac{Dyn}{Obs} & 0 & 0 \\ 0 & 0 & \frac{Dyn}{Obs} \times \frac{Vel}{Disp} & 0 \\ 0 & 0 & 0 & \frac{Dyn}{Obs} \times \frac{Vel}{Disp} \end{bmatrix} \tag{5.35}$$

Although rotation of the object around an axis parallel to the camera axis in the simulation $\boldsymbol{f}_{dyn}$ is allowed in order to better account for the behaviour of a freely rotating object undergoing contact, the rotational state of the object is not considered in the estimation problem and its rotational velocity effectively set to zero before any simulation.

Having defined the cost function, the problem of minimising it is faced. As with the previous experiment, the form of the cost function is a sum of squared differences. Thus, the Levenberg Marquadt and similar algorithms are applicable as a refinement approach (see Section 5.3.2 and Appendix D for details of this approach). To reiterate, these algorithms iteratively find local optima by locally linearising the observation and dynamics models and then either jumping to the corresponding optima, or moving in the direction of the gradient of the square (or some mixture of the two). In this experiment, no difference in performance was noticed between the use of a line-search algorithm performed in the direction of the gradient and the Levenberg Marquadt algorithm.

Both algorithms are applicable for both cost functions described above (the one based on the global parametrisation and the one based on the local parametrisation); in the case of the local parametrisation, the dynamics terms are also sums of squared differences so fit into the same solution framework. In the case of the global parametrisation, the number of variables to be optimised is significantly decreased, though no constraint needs to be applied since all expected states and observations are calculated directly from the parametrised state. If during refinement, observations become close enough to the trajectory to be considered inliers, they are dynamically included in the cost function (or dropped if they fall out of range).

Simulation parameters		
Parameter	How obtained	Value
Coefficient of restitution	Guess	0.8
Coefficient of static friction	Guess	2.0
Coefficient of dynamic friction	Guess	0.1
Linear damping	Guess	0.2
Ball mass[1]	Measured with kitchen scales	$110g$
Ball radius	Approximated with flat ruler	$10.5cm$

Cost parameters		
Parameter	How obtained	Value
Dynamics/observation weight ratio	Calculated	$350 \times 350 pixels^2/m^2$
Velocity/displacement weight ratio	Calculated	$15 \times 15 s^{-2}$

Refinement algorithm parameters		
Parameter	How obtained	Value
Maximum iterations	Set conservatively	400
Finite differences step	Informal experiments	10^{-5}
Stopping tolerance	Set conservatively	3×10^{-6}

RANSAC algorithm parameters		
Parameter	How obtained	Value
Maximum iterations	Calculated	400
Inlier threshold	Guessed	$40 pixels$
Minimum inlier count	Guessed	5

Video capture information		
Parameter	How obtained	Value
Camera	Commonly available	Logitech QuickCam Pro 4000
Resolution	Maximum capability	640×480
Frame rate	Maximum at resolution	$15s^{-1}$

Table 5.6: Parameters of physics simulator, of motion estimation algorithms and visual apparatus in the 2D ball bouncing scenario.

As in the simulation case, finite-differencing is done to calculate the gradient of the cost function with respect to the parameters, or of the squared terms with respect to the parameters.

5.4.7 Results

Figure 5.7 shows the error achieved per video per algorithmic condition. The error is measured in root mean squared error in object location averaged across all frames in the video in which a human was able to determine object location. While the velocity is reconstructed by some algorithms, it is the translational state that is of interest since it is more easily measured, and is the target of most motion estimation algorithms.

Because the RANSAC step using the naive dynamics models (constant displacement and constant velocity dynamics as well as no dynamics) almost always produces very bad estimates on the difficult data-set used here, a second set of experiments is run in which the RANSAC step is handled by the superior collision dynamics model, and performance of different dynamics models compared on the refinement stage.

Figure 5.8 shows examples of failure and success conditions for each video and algorithmic condition.

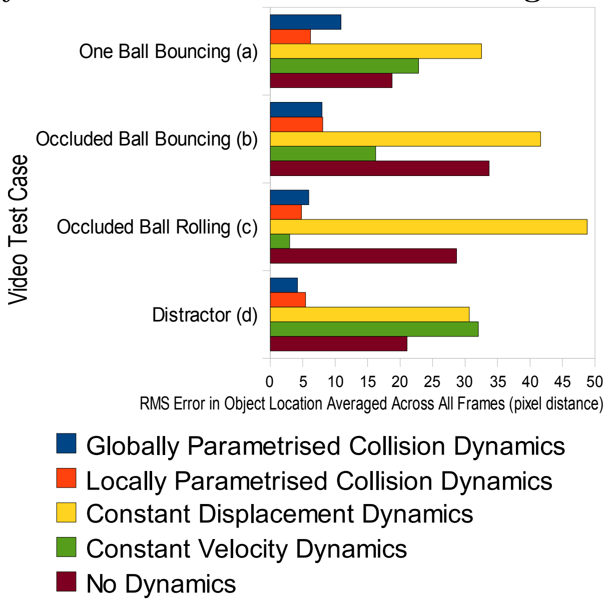
When considering their use in RANSAC, the no dynamics, constant velocity dynamics, and constant displacement dynamics fail quite badly, except in the rolling ball case (c) where the constant velocity dynamics performs well as would be expected since the object is moving with nearly constant velocity. The collision based dynamics, though relatively successful, performs slightly worse here than constant velocity because the model of the ground plane is not well calibrated to the ground-plane in the images.

In the bouncing ball with glare and blur video (a), constant displacement and constant velocity dynamics fail to initialise a good trajectory, mostly fitting well only to a single free-flight segment of the trajectory. In contrast, the collision based physics dynamics is able to find a near enough bouncing trajectory. In the bouncing ball under occlusion video (b), the bouncing in the object is again missed by the naive dynamics models and the occlusion mid-trajectory can means that the RANSAC-generated trajectory does not even pass behind the occluding object. Again, the collision-based dynamics are able to initialise an approximately correct trajectory. In the case of the video ball bouncing with occlusion and distractors (d), the naive dynamics models again perform terribly in the RANSAC phase, generally coming up with highly implausible trajectories missing most of the trajectory of the object. Because the colour histograms are normalised, in video (d) the black bag provides a distractor. Normalisation helps reduce the effect of luminance, but cannot reason about illumination levels in the scene; naive dynamics models are more likely to include the distractor in estimated trajectories, and omit the ball when it is occluded.

When considering only their performance in refinement (by initialising the refinement with a solution derived from RANSAC with a collision-based dynamics model), the naive dynamics models are more often able to show some success, but the collision-based dynamics still mostly do better. The naive dynamics models are still not able to handle the bouncing behind occlusion (b), mostly inferring trajectories that circumnavigate the occluder, and cannot handle the distractor case well (d.i and d.ii), only able to fit to one segment of the trajectory. In general, bouncing is inaccurately smoothed; the sharp discontinuities in velocity and the sharp displacement profile at a collision do not match the assumptions of the naive dynamics models.

When comparing the collision dynamics with global parametrisation to collision dynamics with local parametrisation, it is seen that the latter typically obtains a marginal improvement over the former, mostly to do with integrating more visual information into the more flexible dynamics model; however, it can do worse where the visual information is particularly bad, as with video (d).

RANSAC + Refinement error results per dynamics model
(The same dynamics model used with both algorithmic phases)



Refinement error results per dynamics model
(physics based dynamics used for initialisation)

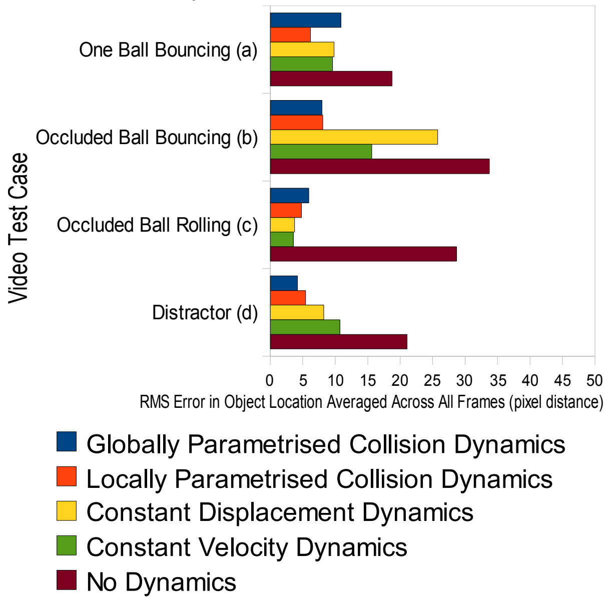


Figure 5.7: Numerical error results for each dynamics model on each of the four ball scenarios. Performance measured in RMS error in pixel distance. Ground truth labelled by hand. Ground truth exists even for occluded objects. The top chart contains a clear comparison of dynamics models as used in the 2 phase algorithm (RANSAC + refinement). The bottom chart contains a comparison of dynamics models against the refinement algorithm; because it performs best in the RANSAC phase, the physics based dynamics were used with RANSAC to initialise the refinement search.

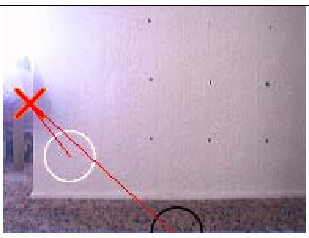
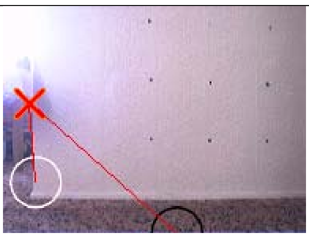
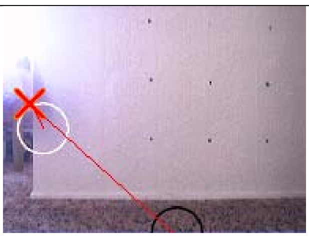
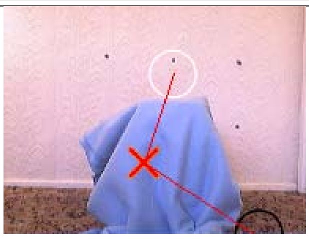
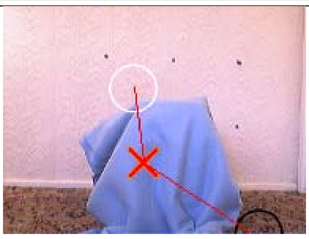
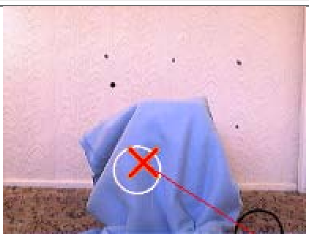
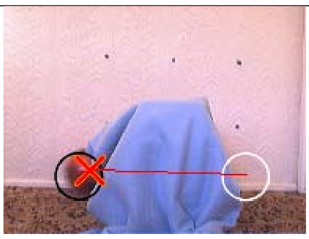
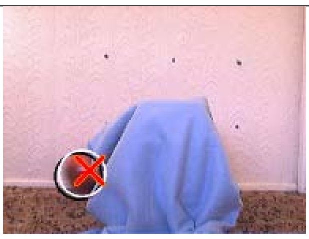
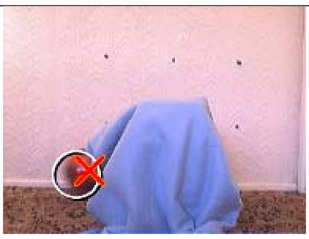
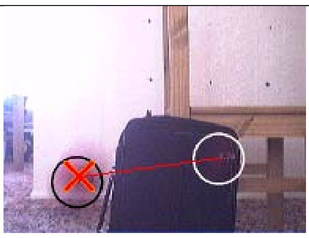
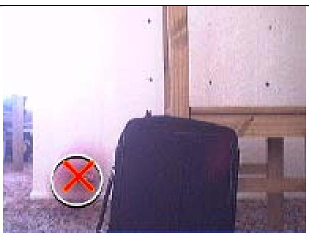
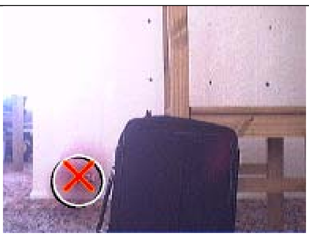
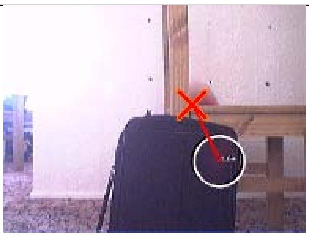
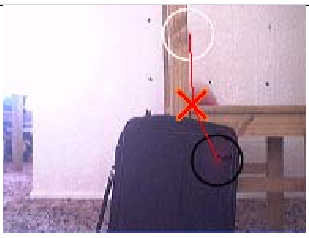
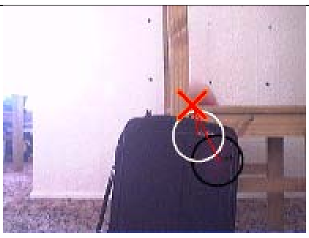
	Constant Displacement Dynamics	Constant Velocity Dynamics	Collision Dynamics (Local)
(a)			
		$i = 3$	
(b)			
		$i = 5$	
(c)			
		$i = 6$	
(d).i.			
		$i = 7$	
(d).ii.			
		$i = 15$	

Figure 5.8: Successes and failures on the 4 videos analysed in this chapter. **First row:** Frame 3 from the video with a ball bouncing across the scene with some glare and blur. **Second row:** Frame 5 from the video with a ball bouncing behind an occluder. **Third row:** Frame 6 from the video with a ball rolling behind an occluder. **Fourth row:** Frame 7 from the video with a ball bouncing behind a distractor and occluder in the presence of blur and glare. **Fourth row:** Frame 15 from the same video. **Left:** Failure and success situations for the constant displacement dynamics based algorithm. **Middle:** Failure and success situations for the constant velocity dynamics based algorithm. **Left:** Failure and success situations for the physics based dynamics model based algorithm. The black circle is the observation-only (without dynamics) estimate, the white circle is the estimate based on the current dynamics model, and the red cross is the human-labelled ground truth estimate.

5.4.8 Experiment discussion

The results do not contradict the hypothesis that the inclusion of a physics-based dynamics, accounting for collisions, can improve estimation performance on real-world data - such performance is observed at any rate on the videos analysed with the normalised colour histogram template visual feature used, motion parallel to the image plane, and a single simple object. The physics-based dynamics is particularly useful when observations are missing or distracting. Whereas naive dynamics models work some of the time and each can be applicable to certain scenarios, when visual information is lacking they are insufficient to cover a wide variety of object behaviour.

As discussed in Chapter 3, and noted in the previous experiment, robustification with algorithms like RANSAC is particularly important where noise is not simple dispersion noise, as is the case with many of the videos addressed here. Moreover, it can help with the search for good trajectory candidates, helping to overcome the problems of non-linearity and multiple optima. This is shown in the experiments in this chapter, with robust initialisation an essential component in the performance of the algorithms used here.

One important consideration when considering the robustness of physics-based estimation is how robust the estimation algorithms are to model mismatch, which is fated to occur. Additional experimentation suggests that the performance of RANSAC is more sensitive to inlier threshold than to the simulation parameters like the coefficient of restitution. This is good news as it suggests that this method can be robust to model inaccuracies. However, when extended to multiple, possibly interacting, balls, the problem becomes more sensitive to the physics model and more methods are necessary for dealing with that. Possible approaches are discussed in the further work section.

5.5 Contributions

There has been little work on incorporating anything other than polynomial models of dynamics into motion estimation. On the other hand, in control, motion planning, and motion synthesis, sophisticated physics models are employed, but, there, robustness to noisy data is not a consideration. In human tracking, some good work exists but again robustness is not the main consideration, and the benefits of the more sophisticated models are not clear. The current chapter shows an improvement in motion estimation in the presence of occlusion, fast object motion, glare, blur and distractors, incorporates physics into the estimation process in a new way, and offers and experimental validation of the use of off-the-shelf real-time physics in motion estimation.

The two physics-based approaches presented in particular showed a marked improvement on the real-world motion estimation from colour problem, irrespective of whether process noise was incorporated in the refinement step.

The remainder of the chapter will discuss the work that remains to this program of research.

5.6 Further work

Further work mostly fits into two categories:

- Obtaining an improvement from a physics model while integrating better visual features.
- Obtaining an improvement from a physics model while expanding the range of real-world scenarios handled.

The former category revolves around showing that a dynamics model can improve performance even when state of the art computer vision techniques are applied. The latter revolves around showing its applicability in a wider range of situations including progressively more complex and real-world situations.

5.6.1 Improved visual features

Estimation performance on the real-world scenario considered here may be improved by improving the use of features rather than the dynamics model. As noted by Ennesser and Medioni (1995) and mentioned in the literature review, the use of a 2D colour space can be difficult to get right. This might contribute to the difficulty of the vision problem, thereby giving the physics an opportunity to do much better by compensating for the weaknesses in the vision.

On the other hand, the visual problem here has been made artificially easy in the scenarios experimented on in this chapter, by using a colour template of known image radius and relatively controlled visual conditions. Occlusion and blur are common in real-world vision problems, and they are a prominent feature of the scenarios here.

An empirical question that remains to be answered then is, with a better colour space quantisation, as well as a model of object occlusion, perhaps using adaptive blob type features as the primary feature rather than fixed-size templates, multiple feature types, and possible blur correction techniques, whether the use of physics still an improvement.

The physics-based approach should still yield improvements with state of the art computer vision techniques if it is to have long-term worth. At the very least there will always be corner-cases where it is useful (that, in specific applications, can be very important). As an example, visual features, particularly monocular ones, are poor at discriminating the depth of even known objects. For example, the radius of a ball is a relatively weak indicator of its distance. Object physics can be used for discriminating very well objects relative distances based on their interactions. Objects interacting under occlusion, and the use of containers to transport objects are also scenarios where this approach should work well even when visual cues are indirect (the trajectory of objects emerging from occlusion, or the trajectory of the container).

5.6.2 More scenario variety

In order to investigate the applicability of this kind of method it is necessary to expand the number and type of real-world scenarios handled. These new scenarios will induce the necessity of the application of new methods.

The obvious next step is to address scenarios with multiple, possible interacting balls. The most improvement would be expected where the balls have ambiguous visual appearance. Beyond that, further steps involve moving to full unconstrained movement in 6D pose, different kinds of object, and less advance specification of scene properties.

5.6.3 Robustness to model inaccuracy

Although the approach given is robust to model inaccuracy on the scenarios described here, there are easily describable scenarios where the approach given would fail in the presence of even slight model inaccuracy. This is because dynamical systems can be particularly sensitive to initial conditions and as such reconstructing one part of a trajectory on the basis of another part can depend on small changes in the first: an example is given in Figure 5.9(a). This is something that naive RANSAC and related methods will not always be able to handle as they would rely on the dynamics model to reconstruct parts of the trajectory from other parts and small noise in one part may lead to incorrect inferences about the other.

Moreover, a bad or inaccurate model of the world can lead to similar problems, as illustrated in Figure 5.9(b). Again, RANSAC relies on the correctness of the dynamics model to infer parts of the trajectory from distant parts so problems like this may lead to RANSAC producing trajectories that cannot be corrected in the refinement stage.

The proposed solution is to introduce process noise in the RANSAC stage, equivalent to the implicit noise model that underlies the cost function used in the refinement algorithm. Thus, during RANSAC, not only are observations sampled from, but also process noise is added. This leads to a much greater sample space but it remains to empirical in-

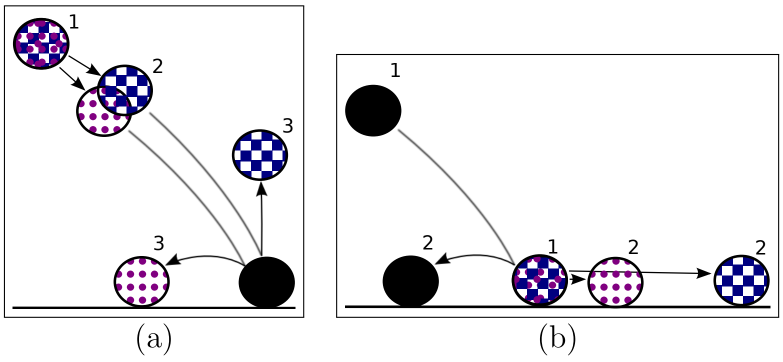


Figure 5.9: **Illustration (a)** illustrates how nonlinearity combined with some permuting noise can lead to large changes in reconstructed trajectory due to the nonlinearity of the dynamics model. If observations at time points 1 and 2 are used to calculate the trajectory of a ball and some permuting noise added to the observation at time 2 different resulting positions at time point 3 and later can be found. **Illustration (b)** shows one of many scenarios where a bad model fit can lead to incorrect reconstructed trajectory. If a trajectory is reconstructed incident on a stationary ball, that ball may roll different distances, or indeed may roll back the way it came, depending on restitution, friction, exact ball shape, surface properties, and other unmodelled or incorrectly modelled physical phenomena.

vestigation whether it can lead to an improvement. To reduce the sample space, it would be possible to detect collisions and only add process noise in the event of a collision, as is common in motion synthesis to produce realistic looking behaviour, and would focus the noise sampling on particularly sensitive parts of the dynamics.

A likely improvement to computational complexity can be obtained by reconstructing sub-trajectories and then combine them. This is currently how the naive dynamics models in the experiment in Section 5.4 are handled during the RANSAC stage.

In order to handle the case in Figure 5.9(b), it is also possible to sample from the simulation parameters $\boldsymbol{\theta}$ such as restitution, linear damping and coefficients of friction. In general, increasing the number of parameters open to inference can move this work to overlap more with structure from motion where traditional shape parameters, for instance, can be inferred as well the physical object and surface parameters.

The MAP problem addressed in this chapter involves the function $p(\boldsymbol{x}_{t_{0:i}}|\boldsymbol{z}_{t_{0:i}}, \boldsymbol{\theta}, \boldsymbol{u}_{t_{0:i}})$ but this can be rearranged to find $p(\boldsymbol{x}_{t_{0:i}}, \boldsymbol{\theta}|\boldsymbol{z}_{t_{0:i}}, \boldsymbol{u}_{t_{0:i}})$ instead, which would introduce a prior $p_{\boldsymbol{\theta}}(\boldsymbol{\theta})$ into the cost, which can be used to constrain the search close to plausible physical setups. An alternative approach would be to marginalise $\boldsymbol{\theta}$ out of this distribution altogether in order to find $p(\boldsymbol{x}_{t_{0:i}}|\boldsymbol{z}_{t_{0:i}}, \boldsymbol{u}_{t_{0:i}}) = \int_{\boldsymbol{\theta}} p(\boldsymbol{x}_{t_{0:i}}, \boldsymbol{\theta}|\boldsymbol{z}_{t_{0:i}}, \boldsymbol{u}_{t_{0:i}})d\boldsymbol{\theta}$, but this is analytically more difficult to achieve and it is unclear what gain it would obtain in this context.

The parameters $\boldsymbol{u}_{\boldsymbol{t_i}}$ can also be descended on using the same cost function and refinement algorithms. In this way, rather than sampling from dispersion in state space, other more physically appropriate sampling methods can be chosen such as sampling of collision normals or object-incident forces. The latter approach is pursued in Chapter 6.

5.6.4 Multiple ambiguous objects

In order to handle multiple ambiguous objects, the data association problem must be solved; that is, at each time step it must be decided which observation corresponds to which object. For small numbers of objects this association can be randomised and each randomised hypothesis checked against a criteria, though the complexity of this approach grows quickly and more sophisticated data association approaches may become necessary.

Any approach must handle weak dynamics models; that is, dynamics models that are not expected to predict the trajectory well and so must have lots of implicit noise, such as constant displacement dynamics and constant velocity dynamics. RANSAC can instantiate trajectories based on such weak models by, for example, randomising elements of the reconstruction, or reconstructing multiple segments.

When assigning observations to objects, it is necessary to check the level of feasibility of the trajectory with respect to the dynamics model. The present approach is to use inlier count as the criteria for checking possible RANSAC solutions. However, with this approach, it is too easy for RANSAC to accept trajectories that involve implausible jumping around the scene as data association is randomly reversed between objects, particularly as the scheme introduced in the previous section whereby sub-trajectories are sampled and combined produces a wider variety of behaviours.

Rather than try to combine a dynamics feasibility check with an inlier count, the obvious approach is to use a cost function that combines the inlier-seeking criteria of inlier count and the dynamics-constraining *DynCost* cost function, and score any potential trajectory on that basis.

The use of a cost-function as used here for refinement and the use of RANSAC are combined in the protocol of MLESAC, MAPSAC and related approaches (Torr, 2002; Torr & Zisserman, 2000), where a cost function is defined based on long-tailed probabilities that contribute very small amounts to the cost for elements that might be outliers. The RANSAC algorithm in its core is preserved, but rather than using inlier count to rank hypotheses, this robust cost function is used. The robust cost function serves the function of inlier count as well as offering more refined discerning based on the gradations of cost.

By adding dynamics terms to this cost function, it is proposed that a MLESAC/MAPSAC approach is particularly applicable to the problem presented here, particularly in ranking reconstructed hypotheses according to dynamics function fit while simultaneously rejecting outliers and considering observation cost. It may be possible to evaluate inliers and dynamics cost separately but at a broader level of analysis, a single cost-function is a way of incorporating different sources of information in a disciplined way[1].

In this paradigm there is also some work deriving weights from visual information and using them in a more disciplined way. In Guided-MLESAC Tordoff and D. W. Murray (2005) a more rigorous approach is taken to making a prior (which would in the case of this chapter be a cost weighting) from a match score.

5.6.5 Evolving algorithm paradigm

As more parameters are sampled from as discussed in the other further work sections ($\boldsymbol{\theta}$,$\boldsymbol{u}$), there is an explosion in the size of the space that needs to be sampled from.

[1]Multiple source of evidence as multiple terms in the cost function, some of them having robust long tails, can be combined sensibly. MAPSAC and MLESAC also do not need an inlier threshold to be specified as it is inferred online. Indeed, there is no clear motivation for using inlier count in RANSAC over a suitable robust cost function with trajectory instantiation from random subsamples, as is done in MLESAC and MAPSAC. This is also argued by Torr (2002), Torr and Zisserman (2000).

The use of the same cost function for refinement and MLESAC/MAPSAC also provides a synthesis with the refinement search in that in both cases the problem is posed as a search across a cost-surface. In the MLESAC/MAPSAC case, this is a heuristic led sampling search, and in the refinement search it is a local approximation and descent search. These approaches may need to be generalised further in the search across the space of efficient algorithms for dealing with this problem.

For example, it would be very easy to design hybrid MAPSAC/EA algorithms. In particular, an initial vision is one where the crossover operation is exploited to splice candidate sub-trajectories, in conjunction with a dispersion mutation operation to explore the general space of nearby trajectories. Alternatively, segments of the trajectory independently, saving good segments for possible inclusion in the final solution. Indeed, in such an algorithm, both dynamics-based and observation-based heuristics can be used to mutate candidate trajectories. Initialisation RANSAC-style is easily incorporated into this kind of approach, as well as non-linear optimisation style operations as used in the refinement approach in this chapter. Efficient methods for traversing complicated cost functions, such as those described by Sminchisescu and Triggs (2005) could come in useful in such problems. Beyond this, a Markov Chain Monte Carlo approach could yield similar benefits, while creating opportunities to again generalise the RANSAC paradigms, and an Expectation-Maximization approach may help to make the algorithm more efficient[1].

5.6.6 Alternative simulation approaches

The use of a black-box simulator here is both a boon, since it simplifies the problem, but also removes some of the flexibility that can be obtained from being able to alter the form of the simulation or to integrate the simulation more closely with the estimation problem as well as other kinds of understanding of the physical dynamics under question.

One of the preoccupations of researchers in robot control is the problem of inverse dy-

[1]The Expectation-Maximization (EM) algorithm is analogous to coordinate-descent except that rather than descending on individual coordinates alternately, the algorithm solves for two different groups of variables alternatively: hidden variables, and optimising variables.

namics; that is, efficiently finding a model of the simulation that also embodies the inverse dynamics allowing the force or impulse train necessary to obtain a trajectory calculated; such simulations are "invertible" (Todorov, 2011). Such an approach is potentially applicable to the problem presented in this chapter in that for an observed trajectory, a dynamical model fit to the trajectory by the calculation of the forces necessary to make the necessary motion. Such an approach needs to be developed to be robust, however, since the existing approaches fit a trajectory directly to a sequence of states. Some approaches already exist in combining the two in clever ways, for example the work discussed in the literature review of Metaxas (1997), Metaxas and Terzopoulos (1993).

Rather than inverting the simulator ("forward dynamics") with respect to the control parameters ("inverse dynamics") as done for example by Todorov (2011), it is found here that more correct ways of inverting the simulator with respect to the previous state parameter $\boldsymbol{x}_{t_{i-1}}$, that is, reversing the simulation in time, would come in useful for reconstructing trajectories when information is known about the trajectory at time t_i but the trajectory needs to be constructed at time t_j, $j < i$. It is possible to define a reversible simulation by allowing information that is normally lost in the simulation (such as velocity direction and size when an object rolls to a stop and is caught by static friction) by keeping oscillations in the system forever (the ball never really stops); such an approach would for the most part induce a continuous simulation function also. However, even if such oscillations were preserved, they are unlikely to match any small-scale oscillations observed unless an immense amount of effort were expended in making the simulation correct; moreover, they are unlikely to be discernible by any visual apparatus and so cannot be used in estimation anyway. Such an approach would be orthogonal to existing work in reversible simulation, which focuses on numerical invertibility (Hairer & Söderlind, 2005), or how probability distributions change in reverse-time (Peters, Janzing, & Schölkopf, 2010).

An alternative approach to solving this problem is defining the simulation from the ground-up stochastic, so that simulating backwards in time involves sampling from the set of possible previous states.

5.6.7 Multiple models

One possible alternative to the use of physics is the use of multiple simple dynamics models. Since one model might apply part of the time and another model part of the time, it may be possible to partially overcome the need for sophisticated physics (though such an approach will sometimes miss cases where the physics approach can shine, for example multiple bounces behind an object). This can also be useful when combining multiple physics-based models when none of them model the world perfectly.

Factored, mixture and boosting approaches are common in the literature and so many techniques exist for making this happen. A variation on this idea is attempted with little success in the next chapter in the context of full 6D object tracking, but that approach is relatively naive and needs some improvement.

CHAPTER 6

APPROACH: RECURSIVE TEXTURE TRACKING OF 6D OBJECT POSE

"Thousands of Egyptian civilians, including protesters who helped topple the authoritarian regime of president Hosni Mubarak, have been tried in military courts without due process."

`Egypt: After Mubarak, the military fist,`

`Inter Press Service, 6 May 2011`

This chapter describes the improvement of an existing real-time 3D texture-based object tracker using a particle filter to propagate hypotheses over time (Mörwald, Zillich, et al., 2009)[1] to take advantage of a physical simulator as part of a dynamics model acting as a proposal distribution for the particle filter. In order to achieve this improvement a novel way of adding noise to the simulator is required.

The scenarios and experiments previously described in this thesis addressed the problem of motion estimation. In the scenario described in this chapter, the emphasis is rather on object tracking, viewed through the lens of recursive estimation of object state. The focus of object tracking is on efficient techniques that enable real-time or near-real-time estimation of object behaviour, while recursive estimation is about maintaining estimates of state across time. It is a probabilistic framework through which object tracking can be understood.

[1]The tracker used here is part of a recent trend to 3D model-based particle-filters, described in Chapter 3, Section 3.2.1 and 3.6.3, including edge-based approaches(Klein & D. W. Murray, 2006; Mörwald, Zillich, et al., 2009; Pupilli & Calway, 2005).

Real-time object tracking is an important goal in cognitive robotics, including robot manipulation, object learning and human-robot interaction. The ability to track objects in space and time opens up a world of possibility; allowing a robot to plan its movements according to the current state of the world, allowing it to learn about and check its understanding of the behaviour of objects and the world by observing their state directly, allowing it to reason jointly with other physically present agents, and more.

In this chapter the target object has a full 6 dimensions of freedom to move in 3D space. The practical limitation in experiments in this chapter is the assumption that the pose of stationary surfaces are known with respect to the camera [1]. In addition the tracking algorithm has a surface and texture model for the tracked object [2].

The main hypothesis with respect to this work is that it should be possible to leverage physical simulation to improve the quality of object tracking, particularly in the presence of occlusions and fast motion, such as when a human hand passes in front of an object, or the robot's own effector, in the presence of clutter, or when objects tumble. It is shown here that this is the case in the videos used here to test this prediction; the improvement is obtained with a particular way of incorporating dynamics noise that inserts the noise as perturbing forces in the simulation.

Table 6.1 lists the background information earlier in this thesis related to this chapter[3].

6.1 Problem formulation

In the scenario described in this chapter, a camera captures a sequence of images of a textured object moving on a tabletop. All of the test videos involve the object being

[1]In principle the pose of a ground plane with respect to the camera can be estimated online, and the algorithms for doing this should be relatively robust (if such a ground plane exists).

[2]The texture estimation has already been incorporated into this tracker (Mörwald, Zillich, et al., 2009), but scene and target structure can also in principle be estimated (Ozden, Schindler, & Van Gool, 2010; Richtsfeld, Mörwald, Zillich, & Vincze, 2010), a task itself potentially made easier with knowledge of how objects are constrained to move.

[3]Many of the ideas described in this chapter have been published in the IEEE International Conference on Robotics and Automation (Duff, Mörwald, et al., 2011).

Subject	Where to find	Details
Probabilistic formulation	Chapter 2	Puts forward a probabilistic formulation for estimation from trajectories.
Recursive estimation	Section 2.6.2	Specialises in recursive estimation, as is done in the present chapter.
Computer vision	Chapter 3	Discusses the background of motion estimation and recursive estimation in computer vision.
Visual tracking	Section 3.6	Focuses on recursive estimation/tracking from vision, the background for the algorithmic techniques described in this chapter.
Physical simulation	Chapter 4	Discusses the fundamentals of physical simulation.
Physics based estimation	Section 4.2	Looks at previous applications of physical simulation to computer vision.

Table 6.1: **Chapter background - where to find background information for this chapter.**

pushed by a robot finger mounted on a five degree-of-freedom (5-DOF) robot arm.

The aim is to take a textured model of the object, illustrated in the top left of Figure 6.1 and find a pose for that object in each of a sequence of images, illustrated in the bottom right of the same figure. The same figure also illustrates the use of texture edge matching in tracking the object (read Section 6.4 below for more details on the texture matching).

The problem is to track the 6-DOF pose of that object. The tracking problem is made more difficult by introducing occlusion and distractors. Also, when the object tips under the force of gravity, it can rotate too fast for a normal object tracker to maintain track. Example image sequences can be seen in Figures 7.2, 7.3, 7.4, 7.5, 7.6 and 7.7 in Chapter 7 where the approach detailed here is applied to real videos.

A 3D shape model of the target object is already provided in the form of a textured polygon mesh. The polygon mesh is designed by hand and the texture acquired during previous tracking[1] (Mörwald, Zillich, et al., 2009). Physical information about world objects is also provided, consisting of the ground plane pose and the physical parameters of the target object, such as mass and friction coefficients.

[1]In order to acquire the polygon mesh, geometric edges alone are used to track the object well enough until the texture can be acquired.

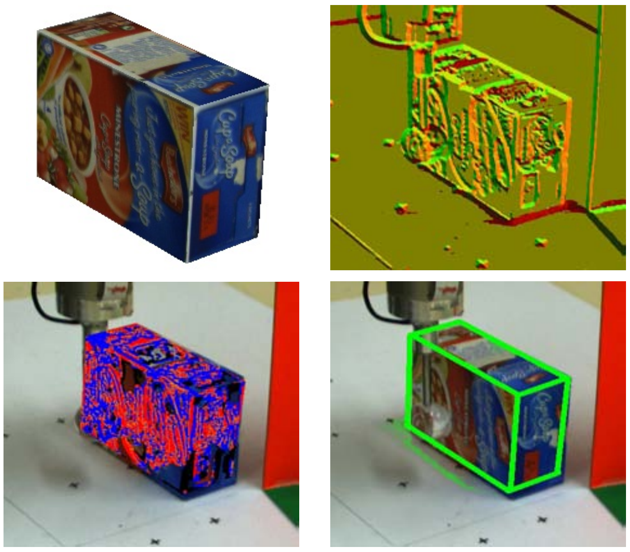


Figure 6.1: **Top left**: A view of the textured model tracked in experiments. **Top right**: A view of the edge image extracted from a candidate image. **Bottom left**: A projection of the edges of the textured model onto the image (obtained by first projecting the texture from a key pose, extracting edges, projecting the result onto the model surface and then re-projecting the edges onto the image). **Bottom right**: A reconstruction of the pose of the object with respect to the camera.

In the rest of this chapter a description of the particle filtering framework and likelihood calculation used is given, followed by a description of various dynamics noise models employed as a part of the particle filter.

In Chapter 7, results are presented that bear on the original prediction that physical simulation should improve the accuracy of tracking. The results also clarify the relationship between simulation and estimation, as well as shedding light on the properties of the probabilistic models and algorithms used in this research.

6.2 Particle Filtering

The approach used here is a direct extension of an existing model-based tracker that uses a particle filter to propagate hypotheses about the pose of an object forward in time and uses graphics card accelerated texture calculations to evaluate the image likelihood of a given pose (Mörwald, Zillich, et al., 2009). The particle filtering framework used there is described first. Particle filtering is an approximate approach to recursive estimation[1].

Representation. In particle filtering a probability distribution p_J is approximated by a set of weighted state hypotheses or particles $J = \{...\langle \boldsymbol{x}[j], w[j]\rangle ...\}$ such that:

$$p_J(\boldsymbol{x}) \approx \sum_{\langle \boldsymbol{x}[j], w[j]\rangle \in J} w[j] \cdot K(\boldsymbol{x} - \boldsymbol{x}[j]) \tag{6.1}$$

Here K is a kernel function[2], usually (and in the case of the approach considered here) taken to be the Dirac delta $K(\boldsymbol{a}) = \delta(\boldsymbol{a})$[3]. This means that the approximated probability distribution has finite support[4].

Figure 6.2 contains a pictorial representation of such a set of particles.

Thus, the probability distribution over the current state given all observations to date $p(\boldsymbol{x}_{t_i} | \boldsymbol{z}_{t_{0:i}}, \boldsymbol{\theta}, \boldsymbol{u}_{t_{0:i}}, \Delta t_{1:i})$ can be approximated by a set of particles J^i_{post} such that $p^i_{post}(\boldsymbol{x}_{t_i} | \boldsymbol{z}_{t_{0:i}}, \boldsymbol{\theta}, \boldsymbol{u}_{t_{0:i}}, \Delta t_{1:i}) \approx p_{J^i_{post}}(\boldsymbol{x}_{t_i})$. These particles are propagated from one time

[1]Recall that recursive estimation is the problem of maintaining a probability distribution, $p(\boldsymbol{x}_{t_i} | \boldsymbol{z}_{t_{0:i}}, \boldsymbol{\theta}, \boldsymbol{u}_{t_{0:i}}, \Delta t_{1:i})$, over an observed system's state, $\boldsymbol{x}_{t_i}$ (in the present chapter, the state of the tracked object), given all observations seen so far, $\boldsymbol{z}_{t_{0:i}}$, by incorporating previous estimates of the object state, $p(\boldsymbol{x}_{t_{i-1}} | \boldsymbol{z}_{t_{0:i-1}}, \boldsymbol{\theta})$, and any new observations $\boldsymbol{z}_{t_i}$ and control inputs $\boldsymbol{u}_{t_i}$. $\Delta t_{1:i}$ is the set of time differences between time-steps. Theta, $\boldsymbol{\theta}$, represents background parameters that do not change from one time-step to the next.

[2]This representation using kernels is on a continuum with the Mixture of Gaussians representation introduced by Sorenson and Alspach (1971) and Alspach and Sorenson (1972) except in the representation given here there is no allowance for maintaining the covariance of a hypothesis; the range of possibilities for representing sampling filters are broader yet - see Chapter 3.

[3]The use of the Dirac delta allows one to conveniently write down the marginalisation of $p(\boldsymbol{x}_{t_i}, \boldsymbol{x}_{t_{i-1}})$ from $p(\boldsymbol{x}_{t_{i-1}})$ defined in terms of these particles if $p(\boldsymbol{x}_{t_i})$ is expressed as an integral over the joint probability's continuous domain. While not rigorously defined here, in practice the following property of the Dirac Delta when used in the place of a function are important: $\int f(\boldsymbol{a} + \boldsymbol{b})\delta(\boldsymbol{a})d\boldsymbol{a} = f(\boldsymbol{b})$ - i.e. the delta function puts all of the probability at discrete points in the domain making the probability density act like probability mass. The Dirac delta can be used in place of a function much of the time but is better described as the limit of a sequence of functions (so that the function in which it is used as a term might itself be considered the limit of a sequence of functions), with properties of a discrete probability described in the language of continuous probability density functions.

[4]The consequences of the assumption that the kernel function used while interpreting particles as a probability distribution is a Dirac delta function is, first, that many things become more tractable, since only those parts of the state-space in which there are samples need be considered (the probability distribution has finite support - in these circumstances the continuous domain may be treated in many ways as if it were discrete). On the other hand, this assumption is effectively the assumption that those parts of the state space where there are no samples are actually impossible for the object to be in (they have probability of 0). This leads generally in practice to the need to compensate by having a dynamics model whose noise is not just dynamics noise, but also noise added for the practical purpose of better exploring the state space around existing samples. This is certainly the case in the particle filtering tracker upon which this work is built.

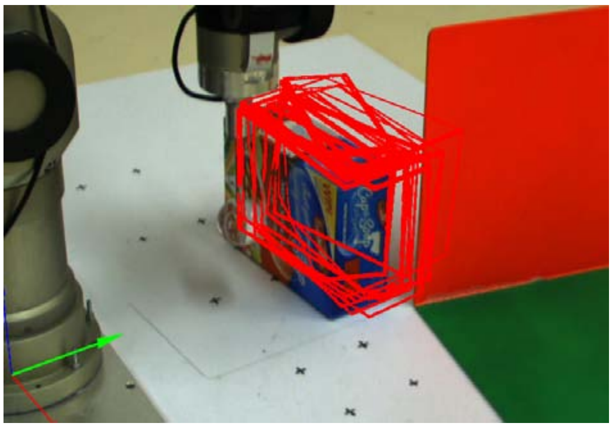


Figure 6.2: A pictorial representation of 20 particles representing 20 hypothesis poses.

step to the next in an attempt to track the probability distribution over current state by a process of sampling, re-sampling and re-weighting of particles.

Having described the representation approximating the distribution over state, in order to define the recursive estimator, it must also be shown how to update this from time-point to time-point.

Propagation. The most efficient way of propagating particles from the posterior at time t_{i-1} (that is, $p(\boldsymbol{x}_{t_{i-1}}|\boldsymbol{z}_{t_{0:i-1}}, \boldsymbol{\theta}, \boldsymbol{u}_{t_{0:i-1}}, \Delta t_{1:i}) \approx p_{J^{i-1}_{post}}(\boldsymbol{x}_{t_{i-1}})$), to the posterior at the later time t_i (that is, $p(\boldsymbol{x}_{t_i}|\boldsymbol{z}_{t_{0:i}}, \boldsymbol{\theta}, \boldsymbol{u}_{t_{0:i}}, \Delta t_{1:i}) \approx p_{J^{i}_{post}}(\boldsymbol{x}_{t_i})$), would be to sample new particles directly from this new posterior probability distribution using the previous set of particles J^{i-1}_{post}. This probability distribution is at least theoretically known, given a dynamics model and observation model, up to scale[1].

If this distribution could be sampled from directly, then the set of new particles produced by sampling from it would represent the distribution over the new state, incorporating the previous estimate as well as all knowledge about state evolution and observation generation. Generally, however, this distribution cannot be sampled from directly. Rather it is sampled from indirectly by first sampling directly from a different distribution (the "proposal distribution") to generate new particles, and those particles weighted so that their weights ("importance weights") reflect the relative probability of each particle. To

[1]If the prior probability over the observations is also known, there will be no scale factor.

keep the set of particles from becoming dominated by zero-weight particles[1], these particles can be re-sampled so that highly-weighted particles spawn more children.

In many cases, and in the case of the particle filter employed here, the proposal distribution is chosen to reflect the distribution over the new state before observations are incorporated (the dynamics model), $p_{dyn}(\boldsymbol{x}_{t_i}|\boldsymbol{x}_{t_{i-1}}, \boldsymbol{\theta}, \boldsymbol{u}_{t_i}, \Delta t_i)$ and the particles weighted according to the probability of the observed output of the hypothesised states (the observation model), $p_{obs}(\boldsymbol{z}_{t_i}|\boldsymbol{x}_{t_i})$[2].

The way that the recursive relationship between a probability distribution over state $\boldsymbol{x}_{t_{i-1}}$ at time-step t_{i-1}, a new observation $\boldsymbol{z}_{t_i}$, and an updated probability distribution over the state $\boldsymbol{x}_{t_i}$ at new time-step t_i can be factored into the dynamics model and the observation likelihood is well-known, and easily derivable from the Markov property and direct dependence of observations on current state. It was given in Chapter 2[3].

In particular, the probability distribution over current state is factored into a part involving the current image probability given a state hypothesis $p_{obs}(\boldsymbol{z}_{t_i}|\boldsymbol{x}_{t_i})$ and an estimate of the current state in terms of previous observations, $p(\boldsymbol{x}_{t_i}|\boldsymbol{z}_{t_{0:i-1}}, \boldsymbol{\theta}, \boldsymbol{u}_{t_{0:i-1}}, \Delta t_{1:i-1})$, expressed in terms of the previous state estimate $p(\boldsymbol{x}_{t_{i-1}}|\boldsymbol{z}_{t_{0:i-1}}, \boldsymbol{\theta}, \boldsymbol{u}_{t_{0:i-1}}, \Delta t_{1:i-1})$ and the probabilistic dynamics model $p_{dyn}(\boldsymbol{x}_{t_i}|\boldsymbol{x}_{t_{i-1}}, \boldsymbol{\theta}, \boldsymbol{u}_{t_i}, \Delta t_i)$. In most particle filters, the representation of probability distribution is frequently re-normalised, so the term $p_{con}(\boldsymbol{z}_{t_i})$, upon which the relative probability of any given state hypothesis $p^i_{post}(\boldsymbol{x}_{t_i}|\boldsymbol{z}_{t_{0:i}}, \boldsymbol{\theta}, \boldsymbol{u}_{t_{0:i}}, \Delta t_{1:i})$ does not depend, may be ignored.

[1]Particles close to zero weight generally become zero weight particles due to the practical issue of number representation.

[2]The former distribution $p_{dyn}(\boldsymbol{x}_{t_i}|\boldsymbol{x}_{t_{i-1}}, \boldsymbol{\theta}, \boldsymbol{u}_{t_i}, \Delta t_i)$ is dubbed the probabilistic "dynamics model" because if any knowledge of observations is left out, the probability distribution can only depend on knowledge about the state at the previous time-step and the probability distribution over the evolution of the true state - the state dynamics. The latter distribution $p_{obs}(\boldsymbol{z}_{t_i}|\boldsymbol{x}_{t_i})$ is dubbed the observation likelihood because it describes the dependence of observations on the state, is recalculated for each state hypothesis, and is only required up to scale because it can be re-normalised.

[3]The recursive estimation relationship is given again here:

$$\begin{aligned} p(\boldsymbol{x}_{t_i}|\boldsymbol{z}_{t_{0:i}}, \boldsymbol{\theta}, \boldsymbol{u}_{t_{0:i}}, \Delta t_{1:i}) &= \frac{p_{obs}(\boldsymbol{z}_{t_i}|\boldsymbol{x}_{t_i})}{p(\boldsymbol{z}_{t_i})} p(\boldsymbol{x}_{t_i}|\boldsymbol{z}_{t_{0:i-1}}, \boldsymbol{\theta}, \boldsymbol{u}_{t_{0:i}}, \Delta t_{1:i}) \\ &= \frac{p(\boldsymbol{z}_{t_i}|\boldsymbol{x}_{t_i})}{p(\boldsymbol{z}_{t_i})} \int_{\boldsymbol{x}_{t_{i-1}}} p_{dyn}(\boldsymbol{x}_{t_i}|\boldsymbol{x}_{t_{i-1}}, \boldsymbol{\theta}, \boldsymbol{u}_{t_i}, \Delta t_i) p(\boldsymbol{x}_{t_{i-1}}|\boldsymbol{z}_{t_{0:i-1}}, \boldsymbol{\theta}, \boldsymbol{u}_{t_{0:i-1}}, \Delta t_{1:i-1}) d\boldsymbol{x}_{t_{i-1}} \end{aligned}$$

This offers a simple and common way of developing a particle filter for use in object tracking. Given a set of particles J^{i-1}_{post} whose distribution represents a probability density over the previous object pose $p(\boldsymbol{x}_{t_{i-1}}|\boldsymbol{z}_{t_{0:i-1}}, \boldsymbol{u}_{t_{0:i}}, \Delta t_{1:i}) \approx p_{J^{i-1}_{post}}(\boldsymbol{x}_{t_{i-1}})$, the dynamics probability model $p_{dyn}(\boldsymbol{x}_{t_i}|\boldsymbol{x}_{t_{i-1}}, \boldsymbol{\theta}, \boldsymbol{u}_{t_i}, \Delta t_i)$ is used to sample a distribution over the current pose $p(\boldsymbol{x}_{t_i}|\boldsymbol{z}_{t_{0:i}}, \boldsymbol{\theta}, \boldsymbol{u}_{t_{0:i-1}}, \Delta t_{1:i-1}) \approx p_{J^{i}_{pri}}(\boldsymbol{x}_{t_i})$ which is then used as a proposal distribution for an importance re-sampling step where particle weight is derived from the likelihood $p_{obs}(\boldsymbol{z}_{t_i}|\boldsymbol{x}_{t_i})$.

The particle filtering algorithm described here is Algorithm 1. It works by propagating particles through the dynamics distribution (described in Algorithm 2) to obtain a prior probability over the current state. Particles are then re-weighted according to the observation likelihood (described in Algorithm 3) and finally they are re-sampled according to this calculated weight (described in Algorithm 4). Note that the particle filtering algorithm described in Algorithm 1 is a simplified *annealed particle filter*, as will now be described.

Annealed particle filter. In particle filtering, in the presence of outliers, the likelihood distribution is very uneven, leading to the necessity of a large amount of sampling, a phenomenon that also occurs when the likelihood distribution is particularly peaky with respect to the state prior (Pitt & Shephard, 1999). In an annealed particle filter, the idea is that the state of an object can continue to be processed by the estimator as long as there is time remaining, so the particle filter noising, re-weighting and importance sampling recursion is repeated multiple times at one time-point.

This paradigm was introduced by Deutscher et al. (2000), and has also been used by Pupilli and Calway (2005). A similar approach was used by Mörwald, Zillich, et al. (2009), where it was called a "recursive particle filter". There, the whole particle population was re-processed, whereas in the annealed particle filter a smaller subset of particles is generally re-processed. Also, in the annealed particle filter the likelihood function was gradually made more peaky (hence the annealing label) over further recursions, while with the approach of Mörwald, Zillich, et al. (2009), the small number of extra recursions only serve to carry out more search of the likelihood. It is the scheme of Mörwald, Zillich, et al. (2009) that is employed in this chapter.

Both filters can be worse than a normal particle filter if the likelihood functions do not contain any information about object location, as when the objects is occluded[1]. Another approach that has the same intent is the addition of a Monte Carlo search after applying the likelihood model (van der Merwe, Doucet, et al., 2001).

In the annealed particle filter of Mörwald, Zillich, et al. (2009), because the particle filter tracker there was developed without the emphasis that knowledge of an object's dynamics could be useful in restricting the probability distribution over the state of the object, rerunning the dynamics propagation state was not a problem at all since the dynamics model in that work served only to disperse the poses of particles. However, to allow this method to generalise to the case where the dynamics model is sensitive to time-elapsed, the dynamics model in the present chapter is parametrised by the time elapsed since the last run of the particle filter Δt_i. This altered formalism allows for specifying an annealed particle filter that can deal with dynamics models that are sensitive to the amount of time that passes.

6.3 Object state representation

When representing the state of an object, only the attributes of the object that are subject to change generally need to be explicitly represented as a part of the state. The rest of the information about the object can be considered background information - which exists in the observation $p_{obs}(\boldsymbol{z}_{t_i}|\boldsymbol{x}_{t_i})$ and dynamics $p_{dyn}(\boldsymbol{x}_{t_i}|\boldsymbol{x}_{t_{i-1}}, \boldsymbol{\theta}, \boldsymbol{u}_{t_i}, \Delta t_i)$ models, as well as background parameters, $\boldsymbol{\theta}$[2].

The object model used here and in most model based vision applications is based

[1]Complete occlusion does not often happen when it is a complete scene being tracked, but for tracking individual objects, it is a very common problem.

[2]The dynamics model must encapsulate any information supplied about the world in which the object can move and the observation model encapsulates how the object will appear when in different states.

Algorithm 1 Particle filter. c is the number of recursions at the one time-step; the filter is iterated until the time remaining will not allow for another recursion. $Time()$ gives the current time which can be compared to the time at which the last observation was acquired t_i.

```
procedure ParticleFilter($J^{-1}_{post}, t_{-1}$)
    for $i \in \{0...\}$ do
        $\langle t_i, \boldsymbol{z}_{t_i} \rangle \leftarrow AcquireObservation(i)$
        $\Delta t_i \leftarrow t_i - t_{i-1}$
        $c \leftarrow 0$
        repeat
            if c=1 then
                $J^i_{pre} \leftarrow Dynamics(J^{i-1}_{post}, \boldsymbol{\theta}, \Delta t_i)$
            else
                $J^i_{pre} \leftarrow Dynamics(J^{i-1}_{post}, \boldsymbol{\theta}, 0)$
            end if
            $J^i_{res} \leftarrow Resample(J^i_{pre})$
            $J^i_{post} \leftarrow Reweight(J^i_{res}, \boldsymbol{z}_{t_i})$
            $c \leftarrow c + 1$
        until $\frac{(c+1)\cdot(Time()-t_i)}{c} \leq \Delta t_i$
    end for
end procedure
```

Algorithm 2 Propagating a set of particles J_{ini} through a dynamics model p_{dyn} with simulation parameters $\boldsymbol{\theta}$ and an elapsed time Δt_i. The tilde $\sim$ represents sampling from a probability distribution.

```
procedure Dynamics($J_{ini}, \boldsymbol{\theta}, \Delta t_i$)
    for $j \in \{1...|J_{ini}|\}$ do
        $\langle \boldsymbol{x}_{ini}[j], w[j] \rangle \leftarrow J_{ini}[j]$
        $\boldsymbol{x}_{fin}[j] \sim p_{dyn}(x_{fin}|\boldsymbol{x}_{ini}[j], \boldsymbol{\theta}, \boldsymbol{u}_{t_i}, \Delta t_i)$
    end for
    Return $\bigcup_{j=0}^{|J_{ini}|} \{\langle \boldsymbol{x}_{fin}[j], w[j] \rangle\}$
end procedure
```

Algorithm 3 Re-weight a set of particles J_{old} with an observation model p_{obs} with observation z

```
procedure Reweight($z, J_{old}$)
    for $j \in \{1...|J_{old}|\}$ do
        $\langle \boldsymbol{x}[j], w_{old}[j] \rangle \leftarrow J_{old}[j]$
        $w_{new}[j] = w_{old}[j] \cdot p_{obs}(\boldsymbol{z}|\boldsymbol{x}[j])$
    end for
    Return $\bigcup_{j=0}^{|J_{old}|} \{\langle \boldsymbol{x}[j], w_{new}[j] \rangle\}$
end procedure
```

Algorithm 4 Re-sampling an array of particles J_{old}

```
procedure RESAMPLE($J_{old}$)
    $w_{SUM} \leftarrow \sum_{\langle \boldsymbol{x},w\rangle \in J} w$
    for $j \in \{1...|J_{old}|\}$ do
        $\langle \boldsymbol{x}[j], w[j]\rangle \leftarrow J_{old}[j]$
        $n[j] \leftarrow \left\lfloor \frac{w[j]}{w_{SUM}} |J_{old}| \right\rfloor$
        $J^j_{new} \leftarrow \bigcup_0^{n[j]} \{\langle \boldsymbol{x}[j], 1\rangle\}$
    end for
    Return $\bigcup_{j=0}^{|J_{old}|} J^j_{new}$
end procedure
```

on a simple textured polygon mesh. Here, this representation is extended to include dynamical parameters such as mass, friction and restitution coefficients, and rotational inertia parameters. The world that this object lives in contains only a simple ground plane, but could in principle contain anything that can be modelled in a physics simulator. In principle, any number of objects can be modelled though in practice extra objects would require alterations to the basic particle filter to allow their joint estimation efficiently.

Pose. There are a large number of possible ways of representing the state of a physical object, though the rigidity of a target object simplifies matters a lot. This is because the rigidity constrains the state of all of the points of the object to lie on a single 6-DOF manifold[1].

In this chapter, the pose is described by a (3D) translation vector and a (4D) unit quaternion. These specify the translation of the object coordinate frame from the origin of the world coordinate frame and the rotation of the object coordinate frame from a base world coordinate frame.

Rates. It turns out that the pose as well as linear and angular velocities are together sufficient to describe the instantaneous state of an ideal rigid object. Taking the state of the object to be its pose in addition to the rate of change of its pose, the instantaneous time-evolution of an object in a completely modelled ideal 3D world can be calculated[2].

[1]Though any number of dimensions is possible for describing the object state, the lowest number of dimensions possible for any parametrisation is six (so it is said that the manifold on which these object poses live has six degrees of freedom). Such a low dimensional specification of a rigid object constitutes the pose. For more details, see the discussions of rigidity in Chapters 3 and 4.

[2]For more details of dynamical modelling see Chapter 4.

Note, however, that the method that is built upon here (Mörwald, Zillich, et al., 2009) has no notion of the rate of change of object pose as it evolves through time, though the authors note the potential desirability of including and exploiting such information. Rather, object state consists only of object pose $\boldsymbol{x} = \begin{bmatrix} \boldsymbol{T} \\ \boldsymbol{q} \end{bmatrix}$, where $\boldsymbol{T}$ is the translation vector, and $\boldsymbol{q}$ the rotation unit quaternion. In that work, the state is evolved by passing candidate poses through a dispersion noise model $p_{dyn} = p_O$ explained below in Equation 6.2. That dispersion noise model is used as the dynamics model of the particle filter.

In the approach described in this chapter, the object state is extended to include rate-of-change parameters, $\boldsymbol{x} = \begin{bmatrix} \boldsymbol{T} \\ \boldsymbol{q} \\ \boldsymbol{v} \\ \boldsymbol{\omega} \end{bmatrix}$ where $\boldsymbol{v}$ is the 3-DOF linear velocity vector and $\boldsymbol{\omega}$ the 3-DOF angular rotation in axis-angle form. However, in order to keep the particle filter low dimensional, in the extension described in this thesis, the particle filter propagates only the pose $\boldsymbol{T}$ and $\boldsymbol{q}$, and the rate parameters $\boldsymbol{v}$ and $\boldsymbol{\omega}$ are reset to zero when the object state is passed through the particle filter dynamics model. So while the full 12-DOF state is modelled, the state estimated and recursed upon is only the 6-DOF pose - rate of change parameters are reset to zero every time the particle filter recurses. If the state stored by the particle filter is $\boldsymbol{x}_{t_{i-1}}[j] = \begin{bmatrix} \boldsymbol{T}_{t_{i-1}}[j] \\ \boldsymbol{q}_{t_{i-1}}[j] \end{bmatrix}$ then that used by the simulator is $\begin{bmatrix} \boldsymbol{T} \\ \boldsymbol{q} \\ \boldsymbol{v} \\ \boldsymbol{\omega} \end{bmatrix} = \begin{bmatrix} \boldsymbol{T}_{t_{i-1}}[j] \\ \boldsymbol{q}_{t_{i-1}}[j] \\ 0 \\ 0 \end{bmatrix}$.

6.4 Observation Model

For the particle filtering algorithm described above to be effective, reasonable probabilistic dynamics and observation models must be provided. The observation model, of the form $p_{obs}(\boldsymbol{z}_{t_i}|\boldsymbol{x}_{t_i})$, describes how observations are generated probabilistically from a given state [1]. In the particle filter upon which this work is based, a routine is given for calculating an un-normalised likelihood of a state given an observation, which provides the observation model.

The full details of that process are given in Appendix F. The outline is given here.

[1]Thus, any state hypothesis is implicitly associated with a probability distribution over observations, and any given observation is implicitly associated with a likelihood over different states.

Given a state hypothesis $\boldsymbol{x}_{t_i}$, the task is to calculate a number which is indicative of the probability $p_{obs}(\boldsymbol{z}_{t_i}|\boldsymbol{x}_{t_i})$ (the likelihood) of the current observation $\boldsymbol{z}_{t_i}$ given that state[1]. The procedure in practice is not probabilistically derived but may be regarded as a heuristic.

First, $\boldsymbol{x}^{\mu}_{t_i}$, a mean pose of the best hypothesis poses is calculated. This is used, along with the camera calibration parameters, to project the object colour texture into the image plane, where edge detection is performed. This "edge texture" is then re-projected onto the object. The edge detection is not done directly on object texture as projecting a texture to a different pose results in edge thinning and dilation. Once the edge texture is obtained, for each pose $\left[\begin{smallmatrix}\boldsymbol{T}\\ \boldsymbol{q}\end{smallmatrix}\right]$, the edge texture is projected again into image space. At this point, the edge texture for the pose can be compared pixelwise to an edge map of the observed image. In order to obtain a finer match, edge strength and direction are considered in the comparison. From the comparison a number is obtained that reflects the similarity of the texture projection related to the state and the observed image. This number (once normalised) is taken to be the image likelihood. The processes of projection, re-projection and pixelwise comparison are accelerated using graphics hardware.

One important consequence of this process is that the use of $\boldsymbol{x}^{\mu}_{t_i}$ means that poses far away from that mean pose are penalised with respect to their likelihood calculations. This naturally discourages multi-modality in the tracker.

For more details of this process see Appendix F.

6.5 Dynamics Models

Having defined the observation model used to weight particles, it is important to define the dynamics model used to generate them. This is the main business of the present chapter, and its primary novelty, where several different dynamics models are presented.

[1]The number need only be provided up to scale since the particle distribution is frequently renormalised.

They are compared in Chapter 7.

Most dynamics models used in practice tend to be relatively simple (see Chapter 3 and particularly Section 3.6.5 in Chapter 4 for examples). In theory, dynamics models reflect the evolution of a system over a discrete time-step. In practice, they are often used mostly to generate new hypotheses to deal with a changing world, as well as being used as a heuristic search to overcome the limitations of sampling and to recover track after it is lost[1].

It is possible to alter the dynamics model to create a better effect in a non-heuristic fashion (i.e. in a probabilistically more well-motivated way) by making the dynamics model more closely reflect the evolution dynamics of the system being tracked. That is what is being done here[2].

The new dynamics model being used is similar to that used previous Chapters 2 and 5, in that it is a probability distribution over the state $\boldsymbol{x}_{t_i}$ at a given time-step t_i conditional on the immediately preceding state $\boldsymbol{x}_{t_{i-1}}$, along with various global parameters $\boldsymbol{\theta}$, a control or nuisance vector $\boldsymbol{u}_{t_i}$ and a time step size Δt_i. The general functional form of the probability distribution is therefore $p_{dyn}(\boldsymbol{x}_{t_i}|\boldsymbol{x}_{t_{i-1}}, \boldsymbol{\theta}, \boldsymbol{u}_{t_i}, \Delta t_i)$. The second important continuity is that this dynamics model is further defined by including the same deterministic simulator $\boldsymbol{f}_{dyn}(\boldsymbol{x}_{t_{i-1}}, \boldsymbol{\theta}, \boldsymbol{u}_i, \Delta t_i)$ with noise added. One of this chapter's novelties is the unique way in which noise is added to the deterministic model, as force noise.

Also, in order to capture multiple sources of knowledge about the movement of the object, the main dynamics model p_{dyn} can be considered to be a combination (via a mixture distribution or heuristic rank selection) of different dynamics models p_1,p_2,...

[1]The kind of simple changes modelled by most dynamics models in use encompassed by the phrase *things don't tend to move very fast*, or, in more sophisticated implementations, *things don't often accelerate*. It is also common to have dynamics models learnt from data - see Section 3.6.5 for more details (Isard & Blake, 1998a, 1998b; MacCormick & Blake, 2000; Mihaylova et al., 2007; Mörwald, Zillich, et al., 2009; Pupilli & Calway, 2005; Richtsfeld et al., 2010; Tweed & Calway, 2002; Vlassis, Terwijn, & Krose, 2002).

[2]There are many other ways it is possible to do better; many heuristics that can be employed to increase the accuracy and efficiency of particle filters by altering the way dynamics are used, such as making the dynamics noise dependent on likelihood peakiness, making the particle count adaptive, manipulating the shape of the noise distribution, and doing multiple recursions at each time-step.

In contrast to the case in Chapter 5, the dynamics model here is only required to simulate forward in time [1]. Also in contrast to the motion estimation Chapter 5, it is difficult to use a noise-less dynamics model in a particle filter, since a particle filter relies on process noise to produce variety in hypothesis states generated at each time-step[2].

In the rest of the chapter, the various dynamics models are described, their rationale given, before experiments exploring the effect of various parameters are described. They are:

- Comparison case: Constant displacement with dispersion - Section 6.5.1.
- Multiple models I: Rank selection - Section 6.5.2.
- Simulation-based dynamics I: Simulation with dispersion noise - Section 6.5.3.
- Simulation-based dynamics II: Simulation with force noise - Section 6.5.4.
- Simulation-based dynamics III: Known finger location - Section 6.5.5.
- Multiple models II: Mixture selection - Section 6.5.6.

The diagram in Figure 6.3 summarises the comparison case and basic approaches to incorporating simulation into the dynamics.

6.5.1 Comparison case: Constant displacement model

As with Chapter 5, the basic dynamics model used as a control for comparison of various more fully featured dynamics model is a constant displacement dynamics model, where

[1]Backwards simulation is unnecessary in a recursive filter as it is assumed that the maintained representation of the probability distribution at the current time is sufficient to encapsulate all previous observations and states. This assumption is the main source of efficiency of recursive filtering, but can be tested when sampling is insufficient to cover important parts of the sample-space (true hypotheses are not generated. This will be seen in the experiments below in Chapter 7, where the introduction of an occlusion can force a naive (non-physics-based) filter to focus all of the probability distribution on incorrect poses.

[2]In order to do without process noise in the particle filter context, it might be possible to, with some analysis, search the prior or likelihood distributions directly, but this would require significantly more analysis because the generate (dynamics) and test (image) paradigm is not available in such a case.

it is assumed that objects are stationary but merely undergo random permutation from simple noise.

The (O)riginal constant displacement model, as used in Mörwald, Zillich, et al. (2009) is:

$$p_O(\boldsymbol{x}_{t_i}|\boldsymbol{x}_{t_{i-1}}, \boldsymbol{\theta}, \boldsymbol{u}_{t_i}, \Delta t_i) = \mathcal{N}(\boldsymbol{x}_{t_{i-1}}, \Sigma_{\boldsymbol{\theta}}) \tag{6.2}$$

Here, the probability distribution over the pose $\boldsymbol{x}_{t_i}$ of the object at time t_i is modelled as a Gaussian distribution around the the previous pose of the object $\boldsymbol{x}_{t_{i-1}}$, with covariance given by the matrix $\Sigma_{\boldsymbol{\theta}}$. The translation component of pose is a simple vector in Euclidean space so a Gaussian distribution over this part of the pose is well-established. However, the Gaussian is also placed over the vector[1] component of the quaternion that defines the rotation part of the pose of the target object. This at least works in practice[2].

The magnitude of the noise used has been tuned from previous experimentation, as this tracker is in use in other applications (Mörwald, Prankl, Richtsfeld, Zillich, & Vincze, 2010). See Table 7.5 for more details of the parameters of this and the other models below.

6.5.2 Multiple dynamics models I: Rank selection

In day-to-day operation of the tracker, a heuristic was found to work in practice.This heuristic ranks all the particles at time step t_{i-1} according to their likelihood scores and ensures that one copy of each of the top ranked particles is added unchanged, without added noise, to the set of particles at the following time-step t_i.

The reason why this heuristic works in practice is that objects are frequently stable or slow-moving and in such cases the heuristic will concentrate sampling very close to previously found likelihood maxima, helping to retain and improve track with a minimal number of samples in such cases. As such, this heuristic can be considered an algorithmic

[1]The vector of a unit quaternion can alone represent the rotation operation and can be parametrised with the three non-scalar components of the quaternion (Ibáñez, 2001).

[2]In previous work, the noise level in the dynamics model, $\Sigma_{\boldsymbol{\theta}}$ has depended also on the quality of the match scores of the best particles, thus increasing the sampling range when a track cannot be obtained. However, this heuristic has little influence on the results presented here.

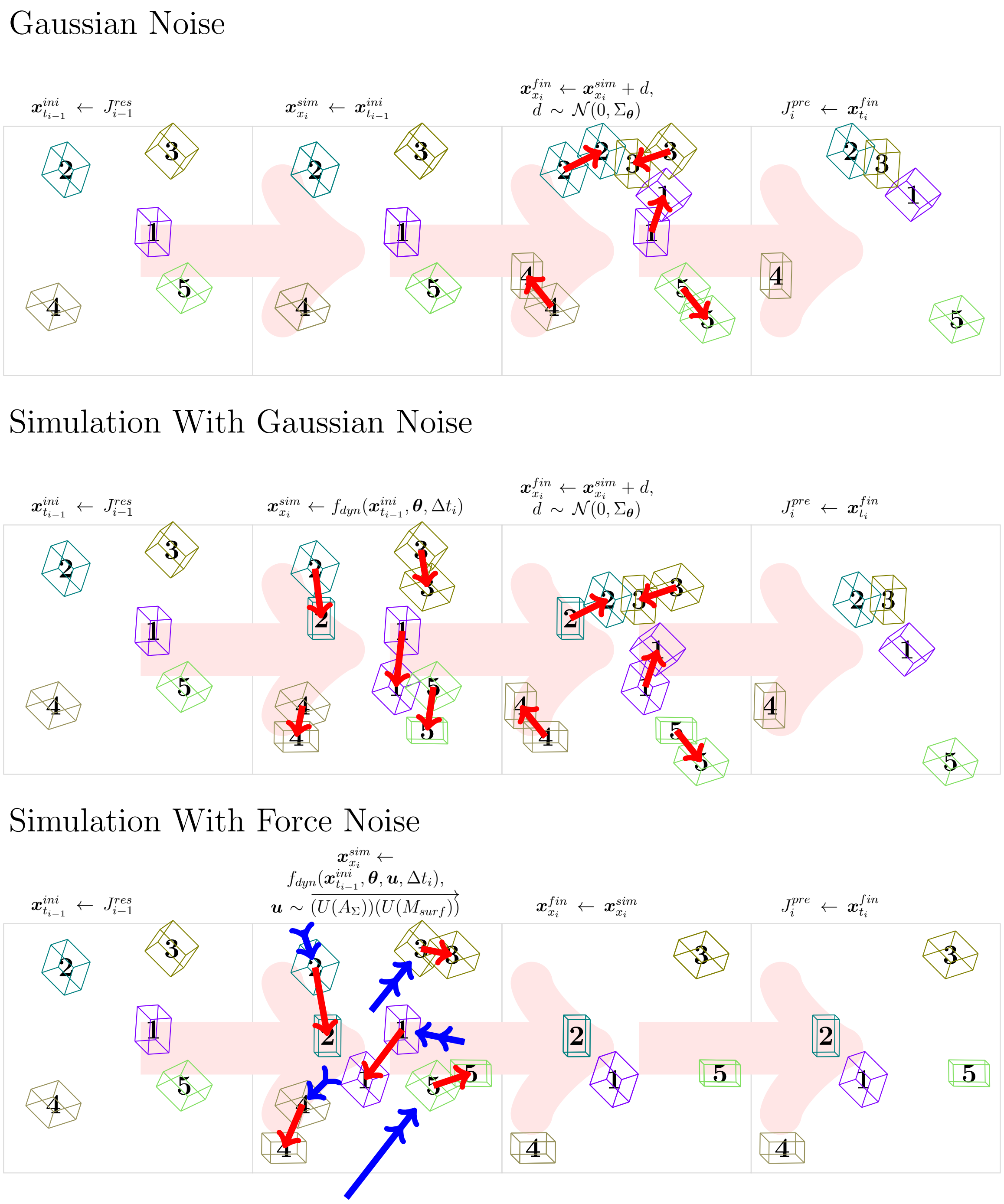

Gaussian Noise
$\boldsymbol{x}^{ini}_{t_{i-1}} \leftarrow J^{res}_{i-1}$
$\boldsymbol{x}^{sim}_{x_i} \leftarrow \boldsymbol{x}^{ini}_{t_{i-1}}$
$\boldsymbol{x}^{fin}_{x_i} \leftarrow \boldsymbol{x}^{sim}_{x_i} + d,$
$d \sim \mathcal{N}(0, \Sigma_{\boldsymbol{\theta}})$
$J^{pre}_i \leftarrow \boldsymbol{x}^{fin}_{t_i}$
Simulation With Gaussian Noise
$\boldsymbol{x}^{ini}_{t_{i-1}} \leftarrow J^{res}_{i-1}$
$\boldsymbol{x}^{sim}_{x_i} \leftarrow f_{dyn}(\boldsymbol{x}^{ini}_{t_{i-1}}, \boldsymbol{\theta}, \Delta t_i)$
$\boldsymbol{x}^{fin}_{x_i} \leftarrow \boldsymbol{x}^{sim}_{x_i} + d,$
$d \sim \mathcal{N}(0, \Sigma_{\boldsymbol{\theta}})$
$J^{pre}_i \leftarrow \boldsymbol{x}^{fin}_{t_i}$
Simulation With Force Noise
$\boldsymbol{x}^{ini}_{t_{i-1}} \leftarrow J^{res}_{i-1}$
$\boldsymbol{x}^{sim}_{x_i} \leftarrow$
$f_{dyn}(\boldsymbol{x}^{ini}_{t_{i-1}}, \boldsymbol{\theta}, \boldsymbol{u}, \Delta t_i),$
$\boldsymbol{u} \sim \overrightarrow{(U(A_\Sigma))(U(M_{surf}))}$
$\boldsymbol{x}^{fin}_{x_i} \leftarrow \boldsymbol{x}^{sim}_{x_i}$
$J^{pre}_i \leftarrow \boldsymbol{x}^{fin}_{t_i}$
1
2
3
4
5

Figure 6.3: The process of three different approaches to incorporating simulation into a sampling dynamics model as used in the particle filter described in this thesis. Each box represents a particle pose hypothesis. **Top:** The basic method that is extended in this thesis; no simulation occurs but particles are dispersed randomly around the previous time pose according to a Gaussian distribution $\mathcal{N}$ with variance $\Sigma_{\boldsymbol{\theta}}$. **Middle:** The basic method extended with a simulation step before the Gaussian dispersion. **Bottom:** Simulation occurs with noise in the form of a force with incidence point sampled from a uniform distribution over the surface of the object $\mathcal{U}(M_{surf})$ and direction and size from a uniform distribution over a sphere $\mathcal{U}(A_\Sigma)$. The diagram illustrates the progression of the sampling procedure from left to right. **Left:** Drawing the pose from the particle set for time t_{i-1}. **Left-centre:** Applying the simulation (with or without force noise). **Right-centre:** Applying the dispersion noise. **Right:** Adding the particle to the particle set at time t_i.

improvement tailored to certain expected common dynamics.

The carrying of particles unchanged from time-step to time-step can be written probabilistically in terms of the Dirac delta distribution:

$$p_\delta(\boldsymbol{x}_{t_i}|\boldsymbol{x}_{t_{i-1}}, \boldsymbol{\theta}, \boldsymbol{u}_{t_i}, \Delta t_i) = \delta(||\boldsymbol{x}_{t_i} - \boldsymbol{x}_{t_{i-1}}||) \tag{6.3}$$

This distribution can be combined with any other distribution, such as p_O described in Equation 6.2 above, by specifying the rank j_R at which a particle is carried over unchanged. Algorithmically, this heuristic requires a change in the Dynamics Algorithm 2. The altered algorithm is given in Algorithm listing 5. There is now a sorting step to rank the particles. The probability distribution actually applied to each particle is now also parametrised by the rank j of the current particle, and which of several identical particles it is n_{same} (since each unique particle needs only to be saved once).

Algorithm 5 A dynamics propagation routine altered to account for the particle ranking multiple-dynamics approach. This dynamics, as with Algorithm 2, propagates a set of particles J_{ini} through a dynamics model p_{dyn} with simulation parameters $\boldsymbol{\theta}$ and an elapsed time Δt_i. This version has the dynamics probability distribution parametrised by the particle rank j

```
procedure DYNAMICS($J_{ini}, \boldsymbol{\theta}, \Delta t_i$)
    $J_{sor} \leftarrow Sort_w\,(J_{ini})$
    for $j \in \{1...|J_{sor}|\}$ do
        $\langle \boldsymbol{x}_{ini}[j], w[j] \rangle \leftarrow J_{sor}[j]$
        if $\boldsymbol{x}_{ini}[j] = \boldsymbol{x}_{last}$ then
            $n_{same} \leftarrow n_{same} + 1$
        else
            $n_{same} \leftarrow 1$
        end if
        $\boldsymbol{x}_{last} \leftarrow \boldsymbol{x}_{ini}[j]$
        $\boldsymbol{x}_{fin}[j] \sim p_{dyn}[j, n_{same}](x_{fin}|\boldsymbol{x}_{ini}[j], \boldsymbol{\theta}, \boldsymbol{u}_{t_i}, \Delta t_i)$
    end for
    Return $\bigcup_{j=0}^{|J_{ini}|} \{\langle \boldsymbol{x}_{fin}[j], w[j] \rangle\}$
end procedure
```

The probability distribution used by this updated algorithm for ranking is written here[1]:

$$p_R[p_A, p_B, j_R][j](\boldsymbol{x}_{t_i}|\boldsymbol{x}_{t_{i-1}}, \boldsymbol{\theta}, \boldsymbol{u}_{t_i}, \Delta t_i) = \begin{cases} p_A(\boldsymbol{x}_{t_i}|\boldsymbol{x}_{t_{i-1}}, \boldsymbol{\theta}, \boldsymbol{u}_{t_i}, \Delta t_i) \text{ if } j < j_R, n_{same} = 1 \\ p_B(\boldsymbol{x}_{t_i}|\boldsymbol{x}_{t_{i-1}}, \boldsymbol{\theta}, \boldsymbol{u}_{t_i}, \Delta t_i) \text{ otherwise} \end{cases} \tag{6.4}$$

This picks from two probability distributions p_A and p_B and returns a different probability distribution for each particle in the ranked set of re-sampled particles if the rank of the particle j is less than the threshold rank j_R and the particle has not been duplicated already ($n_{same} = 1$).

The distributions combined in practice are the delta distribution d_δ and the original dispersion dynamics p_O. The state-of-the-art particle filter before the inclusion of simulator-based dynamics therefore uses the following dynamics model[2]:

$$p_{O/R} = p_R[p_\delta, p_O] \tag{6.5}$$

6.5.3 Simulation-based dynamics I: Simulation with dispersion noise

In the first extension to the basic dispersion dynamics described in Equation 6.2, a simulator is used to introduce a non-linear deterministic dynamics before the dispersion[3].

The simulator is provided by the off-the-shelf PhysX physics API ("PhysX Features," 2010), but could as easily be provided by any reasonably featured physics simulator("Bullet Physics Library," 2011; "Havok Physics," 2011; "Open Dynamics Engine Manual," 2011; "What is Newton Physics Engine?" 2011). The textured object model is augmented with a mass value, coefficients of friction and restitution, an inertial matrix, and a ground

[1]The updated algorithm is strictly more general than Algorithm 2 and reduces to it.

[2]Incorporating something like the ranking heuristic in a more disciplined fashion so that the dynamics model depends on previous observations would possibly involve maintaining a changing distribution over currently effective dynamics models.

[3]The main motivation behind the use of particles in this object tracker is in dealing with the generative and difficult-to-analyse observation model. However, the step of incorporating the non-linear dynamics incorporated in the simulation based dynamics model is itself enough to render a typical Kalman filter highly problematic (Alspach & Sorenson, 1972; Julier & Uhlmann, 1997; van der Merwe & Wan, 2003).

plane. In order to map between the pose of the target object with respect to the camera and with respect to the physical world (in particular, the ground plane), the pose of the camera with respect to the ground plane is pre-calibrated(*Open Source Computer Vision Library: Reference Manual*, 2001; Zhang, 2000; Zhang, 1998).

Whenever it is required that the dynamics $p_{dyn}(\boldsymbol{x}_{t_i}|\boldsymbol{x}_{t_{i-1}}, \boldsymbol{\theta}, \boldsymbol{u}_{t_i}, \Delta t_i)$ be sampled from, an object with the pose defined by the particle representing the object state at $\boldsymbol{x}_{t_{i-1}}$ at time t_{i-1} is constructed and passed through the simulator $\boldsymbol{f}_{dyn}$[1]. See Chapters 2 and 4 for more details about the simulation function $\boldsymbol{f}_{dyn}()$.

After the simulator has deterministically simulated the object motion, the resultant pose undergoes the same Gaussian noise dispersion described for the non-physics tracker above. It makes more sense to conduct this dispersion after the dynamics have been applied so that the likelihood function can be explored more thoroughly, since the effect of simulation is generally to collapse the distribution of poses towards low-energy manifolds, which are unlikely to coincide with observation likelihood peaks[2].

So the probabilistic dynamics model for simulation with (D)ispersion noise p_D can be written:

$$p_D(\boldsymbol{x}_{t_i}|\boldsymbol{x}_{t_{i-1}}, \boldsymbol{\theta}, \boldsymbol{u}_{t_i}, t_\Delta) = \mathcal{N}(\boldsymbol{f}_{dyn}(\boldsymbol{x}_{t_{i-1}}, \boldsymbol{\theta}, \Delta t_i), \Sigma_{\boldsymbol{\theta}}) \tag{6.6}$$

It would be predicted that using this dynamics should improve tracking as a result of increasing sampling in physically plausible parts of the state-space. However, the dispersion noise that needs to be subsequently added to enable the particle filter to sample the space of observations causes particles to be placed in physically implausible states, even if their earlier simulation is plausible. If a sample is drawn so that objects intersect

[1]As mentioned above, since the particle filter in its present state does not account for velocity parameters, the object state is augmented by a zero linear and angular velocity. Adding the velocity parameters is of course further work, but it is unclear what advantages it will bring with a normal $30fps$ camera, particularly in the scenarios tested here, which can mostly be modelled quasi-statically. In a tabletop environment, at least, most movements can be thus modelled; however, there are many cases where this assumption does not hold e.g. when trying to track projectiles and objects with high energy - see Chapter 5.

[2]For example, imagine shaking a box of dice (adding dispersion randomness). After the shaking is finished and a small passage of time has passed, most of those dice will be in the bottom of the box. What is wanted here, however, is to let the dice fall and then disperse them, because it is the dispersion that is important in exploring the image likelihood fully.

(in particular, the target object and the ground plane), the behaviour of the object in the next time-step is not well defined[1]. As a consequence of these facts, the main effect of the simulation can be described as driving the particle set towards low-energy states and states close to plausible ones. The action of this dynamics model is compared graphically to the original method without simulation in Figure 6.3.

In experiments described in the next chapter, simulation parameters are chosen from cursory tuning (see Table 7.5) but the tracking behaviour is insensitive to their value - see Section 7.9.

6.5.4 Simulation-based dynamics II: Simulation with force noise

In order to address the difficulties just mentioned in the simulation-based dynamics with Gaussian dispersion noise, and in order to introduce another approach to integrating simulation into the dynamics model, a slightly different approach is proposed.

Rather than simulating the object and then adding dispersion noise to the pose, noise is added into the simulation itself in the form of a random force applied to the object at a random location on the object surface. These random forces are intended to model the net force applied to the object at each time step, which is a hidden variable as this force is unobserved.

Even if the object is picked up and carried by a human, if the hand of the carrier is not modelled in the model-based system, the force of the hand on the object cannot be directly observed. In the absence of a more specific model of hand and world, in order to account for these hidden forces, they are sampled from directly. The same observation is true of pushes by a robot finger; in the approach presented in this section, the robot finger is not modelled and so any force applied by it must be estimated or treated as a nuisance variable. So the net force acting on the object is sampled at each time point from a distribution of plausible forces.

[1]One fix to the simulation with dispersion noise strategy is to run the collision detector separately for each dispersed particle and sample with rejection. Such an approach has not been attempted here given the success obtained in the next section, but is also a promising line of enquiry.

As with most particle filters, it is unlikely that incorporating a noise model of a hidden variable is useful only for accommodating the effect of that variable, but can be used for overcoming other shortcomings in the model and algorithm; in particular the inability of the filter to sample all plausible poses in the image.

The main motivation for this particular method of adding noise to the simulation is that the resulting dynamics model is more likely to be true to the dynamics as it is modelled by the simulator, restricting and guiding the sampling of new poses even further.

That is the motivation. Here is the formal definition of the probability distribution equivalent to the use of this (F)orce noise method:

$$p_F(\boldsymbol{x}_{t_i}|\boldsymbol{x}_{t_{i-1}}, \boldsymbol{\theta}, \Delta t_i) = \boldsymbol{f}_{dyn}(\boldsymbol{x}_{t_{i-1}}, \boldsymbol{\theta}, \boldsymbol{u}_i, \Delta t_i) \tag{6.7}$$

$$\boldsymbol{u}_i = \boldsymbol{u}_i^{head}\boldsymbol{u}_i^{tail} \tag{6.8}$$

$$\boldsymbol{u}_i^{tail} \sim \mathcal{U}(M_{surf}), \boldsymbol{u}_i^{head} \sim \mathcal{U}(A_\Sigma) \tag{6.9}$$

Here, the force is applied at point $\boldsymbol{u}_i^{head}$ which is sampled from a uniform distribution over the tri-mesh surface of the object[1], $\mathcal{U}(M_{surf})$. The direction and magnitude of the force $\boldsymbol{u}_i^{tail}$ is uniformly sampled from a uniform distribution $\mathcal{U}(A_{\Sigma_{tra}\cdot\Sigma_F})$ over an ellipsoid $A_{\Sigma_{tra}\cdot\Sigma_F}$ whose dimensions are parametrised by the same covariance parameters as the Gaussian translation dispersion noise, Σ_{tra}, adjusted by multiplier Σ_F[2]. The force noise $\boldsymbol{u}_i$ is therefore a point, the point of incidence of the force, and a vector whose magnitude represents the magnitude of the force, and whose direction represents the direction of the force.

As can be inferred from its units, what is actually applied to the object is not a simple force, as the action of a force on an object occurs over time, but a set of impulses applied over each iteration of the rigid body solver underlying the simulation[3]. So the force seems

[1]Sampling over the surface of the tri-mesh is done by first randomly choosing a triangular face on the object, with the probability of choosing each face proportional to the area of the face, and then sampling uniformly from that triangle.

[2]The force noise multiplier Σ_F is required because the units of force noise, $kgms^{-1}$ are different from that of the dispersion translation noise, m.

[3]The parameter `NX_SMOOTH_IMPULSE` in the PhysX API is used to apply force noise to the target object.

to act almost continuously on the object over the whole period Δt_i.

Although the simulation with Gaussian dispersion probability distribution p_D is itself far from being linear, for a fixed and known $\boldsymbol{x}_{t_{i-1}}$, the distribution over $\boldsymbol{x}_{t_i}$ is shaped like a Gaussian. However, any distribution over $\boldsymbol{x}_{t_{i-1}}$ does not make it through the simulation function without its shape being distorted, and neither does the noise added here to the parameter $\boldsymbol{u}_{t_i}$. Some parts of the distribution over state $\boldsymbol{x}_{t_i}$ are collapsed into a lower dimensional manifold (such as when the object can slide over another object), parts that bifurcate into multiple modes (such as between tipping over or tipping back), and parts highly distorted (such as when an object bounces off something). In the particle filtering framework this is not necessarily a problem as arbitrary probability distributions can be sampled from, as long as enough samples are generated in the correct poses. Moreover, it is a closer representation of the shape of probability distributions in the domain of rigid body motion if arbitrary probability distributions are not added to the pose. The addition of force noise does not alter this plausibility as the noise is in an expected input to the simulator rather than second-guessing its output.

Experiments comparing the effect of adding the different approaches to noise are described Chapter 7. The action of this dynamics model is compared pictorially to the original method without simulation and to the simulation with Gaussian noise model in Figure 6.3. In experiments, the parameters of the simulator are chosen the same as with the simulation with dispersion noise condition, and the force noise multiplier Σ_F chosen through cursory tuning to produce similar behaviour to the original tracker.

6.5.5 Simulation-based dynamics III: Known finger location

The context in which the framework being developed in this chapter is applied is one of tabletop interaction between cognitive robotics and humans. Where a robot is the agent interacting with the tracked object, more information is known that should be incorporated into the tracker's estimate of object location; information about the motor commands sent to the effector.

The above simulator based dynamics models do not incorporate knowledge about the robot's own movements, which is a particularly striking absence given that the experiments detailed below in Chapter 7 revolve around tracking an object pushed by a robot. Therefore it is reasonable to attempt to incorporate efferent information in order to deal with this class of situation better[1]. With this in mind, the previous simulator based dynamics are augmented with knowledge about the location of the robot finger. This knowledge is known since the robot controller is a kinematic controller that solves for joint displacements on the basis of a target in operational space. In other words, the robot is commanded to move its finger to a certain location; hence that location is known to within an error.

This knowledge is incorporated into the simulator by introducing an object with infinite mass whose behaviour is determined prior to resolution of the dynamic behaviour of objects in the scene[2]. This object is a sphere whose location is set according to the control signal sent to the robot controller. As such, the location of the finger sphere is not modelled as a random variable and is rather a predetermined variable in the simulator, as with the ground plane[3]. This is recognizable as a control signal, and is incorporated into $\boldsymbol{u}_{t_i}$ as the location of the finger of known size $\boldsymbol{u}_{t_i}^{finger}$.

This extra information can be incorporated into the simulation with Gaussian noise condition or simulation with force noise condition. In the former case, the resulting probability distribution is denoted $p_{D(K)}$ since the finger location is (K)nown. In the latter case, the resulting probability distribution is denoted $p_{F(K)}$.

[1]Indeed, it is likely that humans incorporate efferent information in some form into their estimation, considering that much of the proximal world of a human is subject to the control of her body. The approach here would be compatible with various motor concepts with their root in the concept of efference copy, wherein a self-generated movement is modelled for the purpose of disambiguating sensory signals. Of course, the way that the movement is modelled is particular to the system being studied here and its algorithmic approach and has not, nor will be, validated against humans.

[2]Objects whose behaviour is defined prior to the resolution of the dynamics of a scene are known in real-time simulation as "kinematic" objects.

[3]If force torque information were available, this could be incorporated into the estimation directly as a bias on the forces experienced by objects in the region of the finger; in this scenario, however, only the location of the finger given and forces with the target object are consequences of their interaction in the physics simulator.

6.5.6 Multiple dynamics models II: Mixture model

A number of new probabilistic dynamics models have been introduced now. It is likely that some models will be more accurate in some situations and others will shine in other situations. As such, a way of combining these models to get the best of each should be considered.

While the rank-retention heuristic above does combine multiple dynamics models, it is a heuristic targeted at certain situations and so not satisfactory as a general means to include multiple sources of information[1].

The simplest approach to combining multiple sampling distributions to produce a new sampling distribution is the mixture model, wherein each component distribution is assigned a discrete probability such that a sample is drawn from each component probability distribution with some its probability; these probabilities are usually denoted π and are called component weights[2]. The shape of the final probability distribution is a weighted sum of the component distributions. This is what is done here:

$$p_M[p_1, \pi_1, ..., p_\Pi, \pi_\Pi](\boldsymbol{x}_{t_i} | \boldsymbol{x}_{t_{i-1}}, \boldsymbol{\theta}, \boldsymbol{u}_{t_i}, \Delta t_i) = \sum_{k=1}^{\Pi} \pi_k \cdot p_\pi(\boldsymbol{x}_{t_i} | \boldsymbol{x}_{t_{i-1}}, \boldsymbol{\theta}, \boldsymbol{u}_{t_i}, \Delta t_i) \quad (6.10)$$

Here, the mixture distribution p_M is defined as the weighted sum of the component distributions p_k with weights π_k.

In the worst case, this distribution will show worse behaviour than all its component distributions and in the best case better behaviour than all its component distributions. The advantage of this approach is that sampling should be guided to plausible parts of the state space according to all dynamics models. The disadvantage is that parts of the state space that should not receive samples at all (e.g. where distractors create a higher likelihood of inadmissible poses) may receive them due to the activity of one of the less accurate component dynamics models. Moreover, each particle can oscillate between

[1]Also, the reason why that model was introduced earlier is because it constitutes a part of the state-of-the-art tracker against which all the subsequent dynamics approaches are compared.

[2]In this formulation the mixture weights π_j need to be picked a priori.

dynamics models arbitrarily over time.

The mixture model formulation can be also used as a retention scheme, to select some particles to be kept as-is from time-step to time-step. Recall from above that a rank retention scheme is used to retain particles. In a mixture retention scheme, particles would be assigned to multiple dynamics models, one of which is the no-movement model p_δ. This differs practically from the rank retention scheme in that particles are selected randomly to be retained, rather than on the basis of their likelihood rank. It has the main advantage of being easily describable probabilistically, rather than according to a heuristic algorithmic tweak as with the rank retention scheme.

Having described the new dynamics models tested as a part of this thesis, the following chapter describes some experiments on those dynamics models.

CHAPTER 7

EXPERIMENTS: RECURSIVE TEXTURE TRACKING OF OBJECT 6D POSE

"The investment fund manager from New York who handed us the video showing Tomlinson being pushed to the ground by a police officer, had been reading various stories online, and felt compelled to share his information with us."

`How the crowd changed everything, Sustainability Report 2011,`

`The Guardian, 25 July 2011`

This chapter describes experiments done using the visual object tracker described in Chapter 6, particularly examining the effect of different ways of introducing simulator-based dynamics models[1]. The results suggest that physics-based approaches can show a great improvement over non-physics approaches, but that improvement is dependant on a realistic way of injecting noise; in particular the force-noise approach described in Chapter 6 results in the greatest improvment, including tracking when an object is tipped whilst almost fully occluded. Including information about finger location provides only marginal improvement, and the mixture model approach to combining dynamics does not work well.

The experiments detailed here revolve around tracking the pose of the target object in the video sequences shown in Figures 7.2, 7.3, 7.4, 7.5, 7.6 and 7.7, where the object is pushed around by a robot finger.

[1]Some of the results described in this chapter have been published in the IEEE International Conference on Robotics and Automation (Duff, Mörwald, et al., 2011).

The target object in all experiments is the same coloured and textured box that can be seen in Figure 7.1. A 3D textured tri-mesh of the model had been previously acquired according to the method described in Mörwald, Zillich, et al. (2009). A Point Grey Flea 2 camera ("Flea 2 Specifications," 2009) is used to acquire 800×600 resolution images at $30fps$ of the object.

The first results presented compare the success of all different approaches on each scenario (Section 7.4); subsequently the effect of dynamics models on the success of the algorithm is shown (Section 7.5), followed by the effect of adding a robot finger (Section 7.6) and of using mixtures of dynamics (Section 7.7).

After the main body of results a small number of extra experiments are shown, evaluating the robustness of the proposed physics-based approach to badly modeled scenes (Section 7.8), changes in simulation parameters, and in quantity of noise injection (Section 7.9).

This object is pushed by a robot finger executing a pre-programmed random smooth trajectory within a space of similar pushes. The finger is a fabricated $10cm$ diameter sphere on the end of a solid rod, affixed to a 5-DOF Neuronics Katana 320 arm with position control[1]. In practice in the experiments described here, the arm is generally

[1]The model of Katana arm used is no longer manufactured, but newer versions are similar ("Neuronics - Katana," 2011)

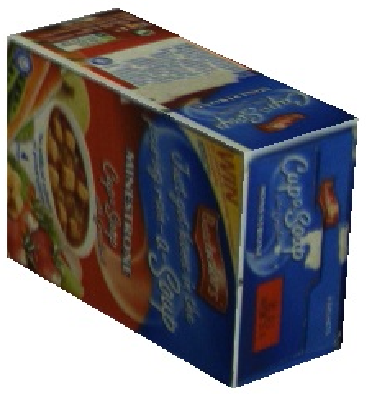


Figure 7.1: A view of the textured model tracked in experiments.

controllable to a precision of $0.1mm$[1] ("Neuronics - Katana," 2011).

The task of the tracker is to track the object while it is being pushed by the finger. High-level results comparing all conditions can be found in Section 7.4. In addition to the hypothesis that the use of physical simulation in the dynamics phase of the particle filter should be able to improve object tracking performance (tested in Section 7.5), the hypothesis is tested that including information about the location of the robot finger should also be able to improve tracking performance[2] (Section 7.6). The effect of using mixture models is subsequently investigated (Section 7.7, and what happens when the assumptions of the dynamics break down (Section 7.8).

The quantitative experiments examine sets of key frames of which representatives are shown in Figures 7.2 to 7.7. The key frames sets are sets of frames where tracker behaviour is compared numerically between different conditions, so that each important phase of a video has a number like X.x[3]. All of the key frame sets are shown in Table 7.6. These key frames were chosen in advance to reflect important points during tracking, such as full occlusion and post-tip. Small sets of frames are sufficient to characterise behaviour over an important phase.

7.1 Test videos

The first video in Figure 7.2 (simple push) is the easiest case. This video is not expected to be a challenge for any existing method.

The second video in Figure 7.3 (push and rotate) is a test of the robustness in tracking likelihood peaks. As noted above, it is not always straightforward to get the tracker stable

[1]No matter what its precision, the accuracy of an arm is only ever as good as its calibration, and in the current application, the camera calibration is the more important variable in accuracy of mapping finger position into visual space.

[2]As such, position commands sent to the Katana arm were recorded, and in some tracking conditions those commands were used to define the location of the finger in the simulation.

[3]Not every frame shown is a key frame but the ones with numbers such as X.x in the figures are key frames of interest belonging to the set designated by that number. Each set has multiple frames so that the numerical results are averaged over the multiple frames that represent a certain phase of tracking.

as object faces disappear and appear.

The third video in Figure 7.4 (tipping) tests the robustness of the tracker to fast motion. As noted earlier, trackers have a tendency to fail when objects move too fast, because of blur and because many trackers are local search methods that can fail if a target object moves too fast away from the target search area.

The next video in Figure 7.5 (colour distractor occlusion) tests the robustness of the tracker to occlusion by a coloured object with strong distracting edges.

The penultimate video in Figure 7.6 (occlusion, recovery and tipping) has an object undergoing occlusion by a human hand; after the object is revealed it is ultimately pushed over. So the case includes both occlusion and fast movement and allows for an analysis of post-occlusion behaviour.

The last video in Figure 7.7 (tipping while occluded) is the hardest of all and combines occlusion with strong distractors with tipping. A human can see what is going on in these scenes so an object tracker should be tested on these scenarios also.

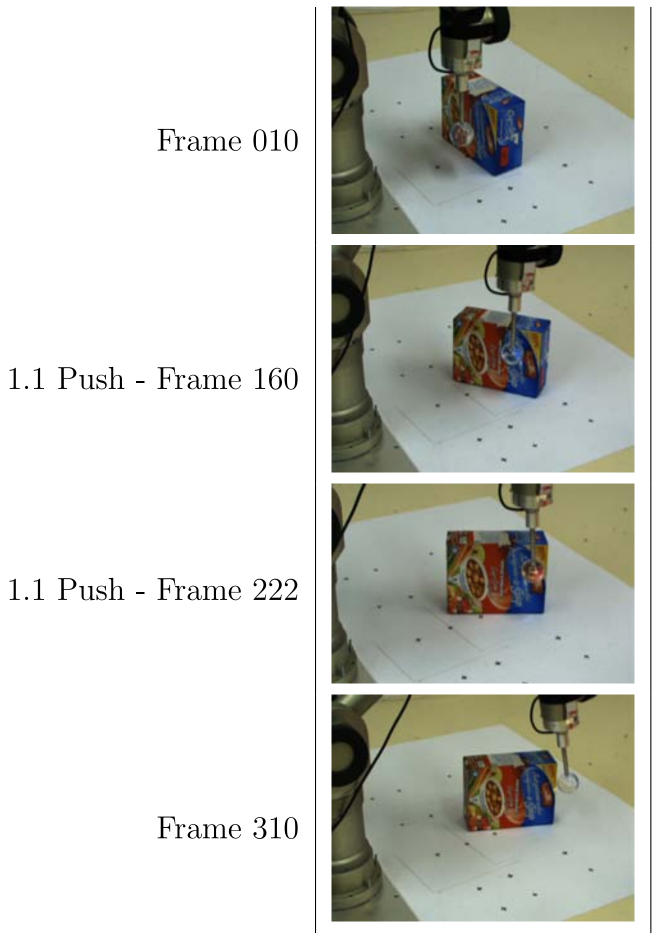


Figure 7.2: Video 1 **Simple Push**: As with all these videos, the box starts off in profile to the camera (frame 010, not a key frame). The robot finger pushes the right part of the box (frame 160, video key frame set 1.1), so that the box is face on to the camera (frame 222, video key frame set 1.1). The robot finger finally advances past the box (frame 310).

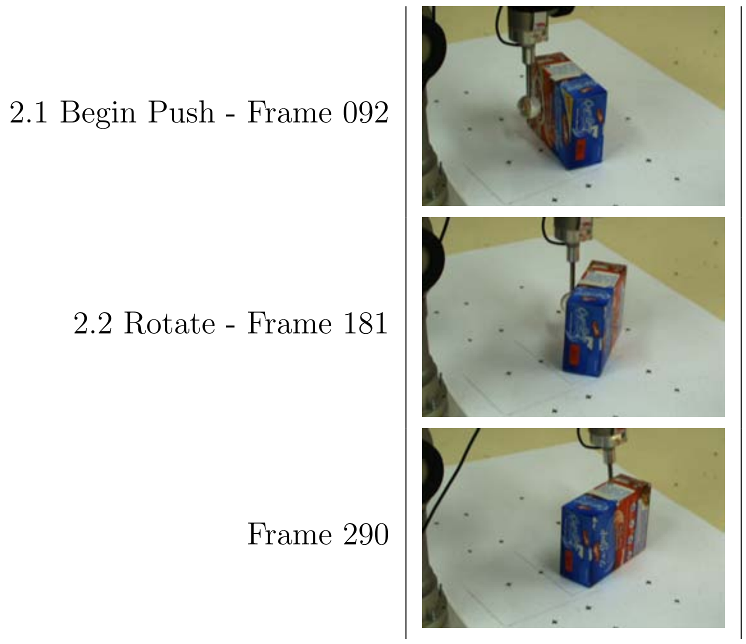
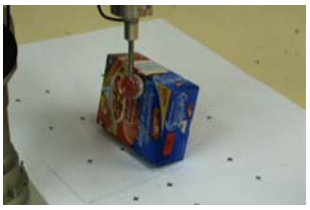
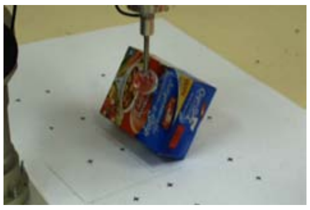


Figure 7.3: Video 2 **Push and rotate**: In this video the box starts off again in profile to the camera (frame 092, video key frame set 2.1), and is pushed from the left so that it rotates past a pose with its end to the camera (frame 181, video key frame set 2.2), coming to rest again in profile (frame 290, video key frame set 2.3). Here, video key frame set 2.2 is the challenging one as it presents a small image area for matching.

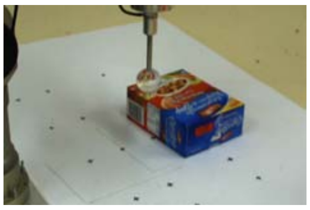
3.1 Start tip - Frame 100



3.1 Start tip - Frame 134



3.2 After tip - Frame 141



3.3 Resting - Frame 317

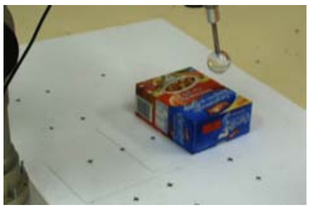


Figure 7.4: Video 3 **Tipping**: First, the object is pushed so that it is supported by the finger in a partial tipping position (frames 100 and 134, video key frame set 3.1). In the space of a few frames the object becomes unstable and leaves one energy configuration and enters another low energy configuration (frame 141, video key frame set 3.2). If a tracker fails at this point it may yet recover later on so a key frame set is examined about 6 seconds after this initial fall (frame 317, video key frame set 3.3).

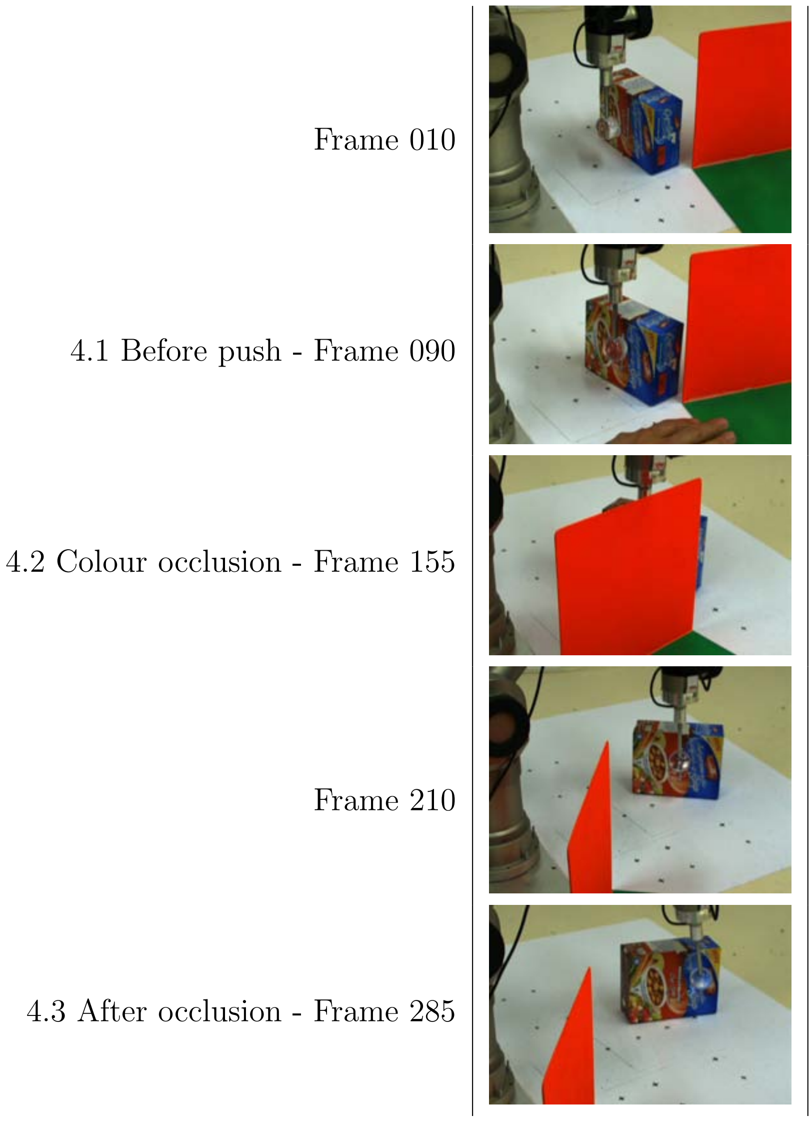

Frame 010
4.1 Before push - Frame 090
4.2 Colour occlusion - Frame 155
Frame 210
4.3 After occlusion - Frame 285

Figure 7.5: Video 4 **Colour distractor occlusion**: Again the object starts off in profile to the camera (frame 010), and is again pushed by the robot finger, with the point of contact slightly to the right of the centre of it (frame 090, video key frame set 4.1). In that frame the reader can see a human hand preparing to drag the coloured occluder in front of the object. The object is presently almost fully occluded (frame 155, video key frame set 4.2). The object appears out from behind the occlusion (frame 210) and another key frame set is gathered about 4 seconds after the occlusion (frame 285, key frame set 4.3).

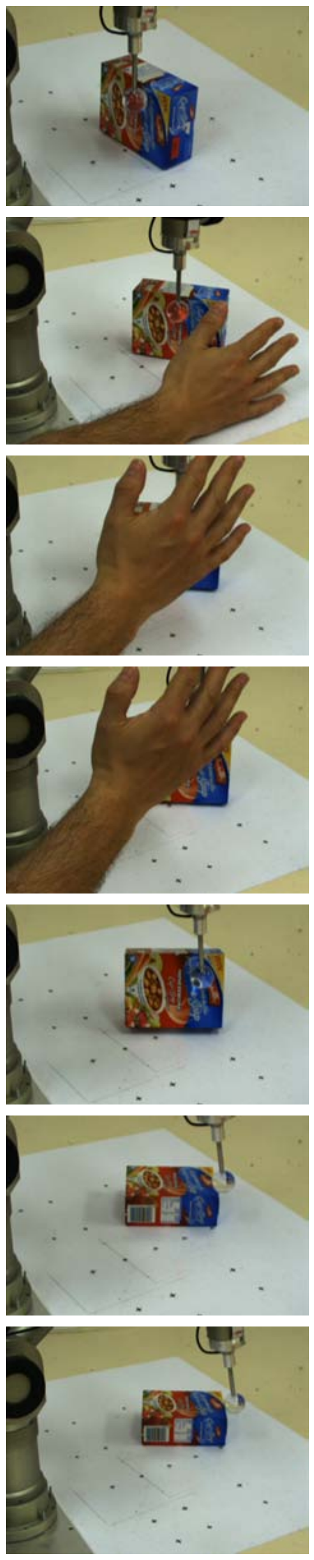
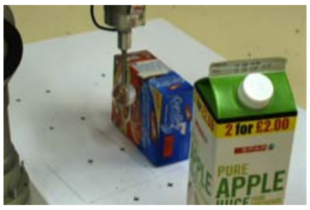
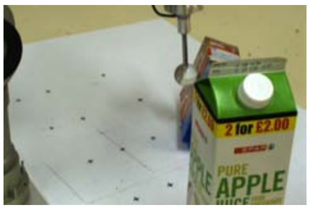

5.1 Initial push - Frame 090
Frame 160
5.2 Hand occlusion - Frame 190
Frame 200
5.3 After occlusion - Frame 230
5.4 Tipped - Frame 275
5.5 Resting Frame 328

Figure 7.6: Video 5 **Occlusion, recovery and tipping**: Initial track is checked (frame 090, key frame set 5.1), then a hand swept across the path of the object (frame 160) and waved in front of the object almost completely occluding it (frame 190, key frame set 5.2 and frame 200). The hand is then removed (frame 230, key frame set 5.3) and then the object is tipped over by the advancing robot finger (frame 275, key frame set 5.4). Two seconds later, the object remains tipped (frame 328, key frame set 5.5).

6.1 Begin push - Frame 110



6.2 Bottle occlusion - Frame 238



6.3 Occluded tip - Frame 306

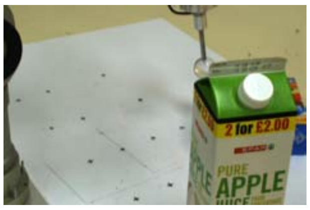


Figure 7.7: Video 6 **Tipping while occluded**: The object is pushed past a second object in the foreground of the scene (frame 110, video key frame set 6.1), and is pushed behind that second object so that it is occluded while it is tipped (frame 238, video key frame set 6.2) and is finally tipped over whilst mostly behind the occluding object (frame 306, key frame 6.3).

7.2 Quantitative criteria

Although only a handful of videos are examined here, and though each video generally contributes a handful of bits of information about each condition applied to it (e.g. track failure/success, time of onset of failure/recovery, degree of failure), the fact that the algorithm used is in theory nondeterministic (using sampling) means that there is the need to examine the behaviour of the algorithm across a number of trials. In this section, 40 trials are conducted for each condition on each video.

There are a number of different ways that the degree of success of an algorithm can be examined quantitatively. They are discussed in Chapter 3 Section 3.7. In these experiments, ground truth is not known so comparisons are done in image space with respect to hand-labelled poses[1]. The root mean error in vertex location in image space is obtained and statistics calculated over 40 trials.

Having established the target videos and time segments, and the method for measuring success, the next section discusses the set of algorithms used for comparison.

7.3 Algorithm setup

The Cartesian product of all the algorithmic variations discussed in the chapter 6 is a very large set. However, in these experiments as much of these combinations as is feasible is examined. Therefore there are a large number of conditions applied to the video set. These conditions are tabulated in Tables 7.1, 7.2, 7.3 and 7.4.

Table 7.1 shows all of the conditions that use the original basic dynamics model p_O coded according to condition[2]. All these conditions start with “O”. For those with retention schemes[3], this initial character is followed by “/R” for rank retention and “/M” for

[1]Comparing the poses directly would be misleading as hand-labelled poses can be far from the ground truth, particularly considering uncertainty in the depth dimension.

[2]See Section 6.5.1 for details of the original comparison dynamics model with constant displacement and dispersion noise.

[3]See Section 6.5.2 for details of retention schemes.

mixture model retention. The numerical model selection parameter (mixture coefficient or rank proportion) is given in the table. Finally the number of recursions at each time-step is given as "-1" or "-2" for one or two recursions respectively. For instance, O/R-2 is the dynamics model p_O with rank retention ($p_R[p_\delta, p_O, 0.25]$) and two recursions. Similarly O-1 has no particle retention scheme (p_O) and one recursion. The "O" can be thought of as "(O)riginal dynamics model".

Table 7.2 details all of the simulation with pose dispersion noise conditions using the dynamics model p_D[1]. These are coded in a similar way except that these conditions start with a "D". The "D" can be thought of as "simulation with (D)ispersion noise". Also, where the simulator includes the robot end effector[2], the condition is suffixed with "(K)". The mnemonic for this is that the finger position is "(K)nown". Because each condition can be tried with and without known finger location, there are twice the number of conditions in this set as in the set of conditions using the original basic dynamics model.

Table 7.3 shows all of the simulation with force noise conditions, using the dynamics model p_F[3], all prepended with the letter "F". Because only one recursion at each time-step is relevant with these conditions (since the noise is only introduced with the passing of time), there are half the number of conditions as there are in the simulation with dispersion noise case.

Finally, Table 7.4 illustrates a set of conditions that use a mixture model of the original, simulation with dispersion noise, and simulation with force noise dynamics models (and may still have rank or mixture retention schemes)[4]. These all start with "M" and are followed by the list of letters that refer to the component dynamics models. So for instance, "MDF(K)/R-1" is a mixture of the simulation with dispersion noise dynamics model and the simulation with force noise model with rank retention ($p_R[p_M[p_{D(K)}, 0.5, p_{F(K)}, 0.5], p_\delta, 0.25]$) and only one recursion. Weightings for the model components as well as the retention parameter are given in the tables. Since these mix-

[1]See Section 6.5.3 for details of the simulation with dispersion noise dynamics.
[2]See Section 6.5.5 for details of the use of dynamics models with known finger location.
[3]See Section 6.5.4 for details of the simulation with force noise dynamics.
[4]See Section 6.5.6 for details of the mixture model dynamics.

Table 7.1: 3D box pushing condition codes

Control conditions - no simulation.

Code	Model selection parameter	Dynamics models	No. recursions
O/R-1	$r^j = 0.25$	Straight copy	1
	Remainder	Pose dispersion (no simulation)	
O/R-2	$r^j = 0.25$	Straight copy	2
	Remainder	Pose dispersion (no simulation)	
O/M-1	$\pi = 0.10$	Straight copy	1
	$\pi = 0.90$	Pose dispersion (no simulation)	
O/M-2	$\pi = 0.10$	Straight copy	2
	$\pi = 0.90$	Pose dispersion (no simulation)	
O-1	All	Pose dispersion (no simulation)	1
O-2	All	Pose dispersion (no simulation)	2

ture conditions include the force noise dynamics as a component, it makes little sense to include conditions for multiple filter recursion at each time-step.

The various parameters of the algorithms used in this experiment, and how they were determined, are given in Table 7.5[1]. A fixed integration time-step size is used in the simulation, for repeatability.

Having described the experimental framework in some detail, the exposition moves on to the results.

[1]Colour intensity values are taken to be normalised to between 0 and 1.

Table 7.2: 3D box pushing condition codes

Simulation with pose dispersion noise conditions.

Code	Selection	Dynamics models	Recursions
D/R-1	$r^j = 0.25$	Straight copy	1
	Remainder	Simulation, pose dispersion	
D(K)/R-1	$r^j = 0.25$	Straight copy	1
	remainder	Simulation, pose dispersion (known robot location)	
D/R-2	$r^j = 0.25$	Straight copy	2
	Remainder	Simulation, pose dispersion	
D(K)/R-2	$r^j = 0.25$	Straight copy	2
	Remainder	Simulation, pose dispersion (known robot location)	
D/M-1	$\pi = 0.10$	Straight copy	1
	$\pi = 0.90$	Simulation, pose dispersion	
D(K)/M-1	$\pi = 0.10$	Straight copy	1
	$\pi = 0.90$	Simulation, pose dispersion (known robot location)	
D/M-2	$\pi = 0.10$	Straight copy	2
	$\pi = 0.90$	Simulation, pose dispersion	
D(K)/M-2	$\pi = 0.10$	Straight copy	2
	$\pi = 0.90$	Simulation, pose dispersion (known robot location)	
D-1	All	Simulation, pose dispersion	1
D(K)-1	All	Simulation, pose dispersion (known robot location)	1
D-2	All	Simulation with, dispersion	2
D(K)-2	All	Simulation, pose dispersion (known robot location)	2

Table 7.3: 3D box pushing condition codes

Simulation with force noise conditions.

Code	Selection	Dynamics models	Recursions
F/R-1	$r^j = 0.25$	Straight copy	1
	Remainder	Simulation, force noise	
F(K)/R-1	$r^j = 0.25$	Straight copy	1
	Remainder	Simulation, force noise (known robot location)	
F/M-1	$\pi = 0.10$	Straight copy	1
	$\pi = 0.90$	Simulation, force noise	
F(K)/M-1	$\pi = 0.10$	Straight copy	1
	$\pi = 0.90$	Simulation, force noise (known robot location)	
F-1	All	Simulation, force noise	1
F(K)-1	All	Simulation, force noise (known robot location)	1

Table 7.4: 3D box pushing condition codes

Mixture of models conditions.

Code	**Selection**	**Dynamics models**	**Recursions**
MOD/M-1	$\pi = 0.10$	Straight copy	1
	$\pi = 0.45$	Pose dispersion (no simulation)	
	$\pi = 0.45$	Simulation, pose dispersion	
MOD(K)/M-1	$\pi = 0.10$	Straight copy	1
	$\pi = 0.45$	Pose dispersion (no simulation)	
	$\pi = 0.45$	Simulation, pose disp. (known robot location)	
MODF/R-1	$r^j = 0.25$	Straight copy	1
	$\pi = 0.34$	Pose dispersion (no simulation)	
	$\pi = 0.33$	Simulation, pose dispersion	
	$\pi = 0.33$	Simulation, force noise	
MODF(K)/R-1	$r^j = 0.25$	Straight copy	1
	$\pi = 0.34$	Pose dispersion (no simulation)	
	$\pi = 0.33$	Simulation, pose disp. (known robot location)	
	$\pi = 0.33$	Simulation, force noise (known robot location)	
MDF/R-1	$r^j = 0.25$	Straight copy	1
	$\pi = 0.50$	Simulation, pose dispersion	
	$\pi = 0.50$	Simulation, force noise	
MDF(K)/R-1	$r = 0.25$	Straight copy	1
	$\pi = 0.50$	Simulation, pose disp. (known robot location)	
	$\pi = 0.50$	Simulation, force noise (known robot location)	

General tracker parameters		
Parameter	How obtained	Value
Num. posterior particles $\lvert J\rvert$	Original tracker	100

Observation model parameters		
Parameter	How obtained	Value
Convergence parameter C_{conv}	Original tracker tuned	7
Matching tolerance m_{thresh}	Original tracker tuned	0.2 (pixels)
Edge matching tolerance g_{thresh}	Original tracker tuned	0.03
Edge dilation iterations	Original tracker tuned	3
Edge dilation distance scaling	Original tracker tuned	0.75
Gaussian smoother size	Original tracker tuned	5×5

Dynamics model parameters		
Parameter	How obtained	Value
Displacement noise Σ_θ	Original tracker tuned	$\begin{bmatrix} \Sigma_{tra} & 0 \\ 0 & \Sigma_{rot} \end{bmatrix}$ $\Sigma_{tra} = .04(m)I_3$ $\Sigma_{rot} = 40I_3$
Extra z-axis noise Σ_{perp}	Original tracker tuned	$0.06(m)$
Force noise multiplier Σ_F	Cursory Tuning	0.003

Physics simulator parameters		
Parameter	How obtained	Value
Object density	Arbitrary	$1(kgm^{-3})$
Inertial matrix	Default	I_3 (Identity)
Coefficient of restitution	Cursory Tuning	0.1
Coefficient of static friction	Cursory Tuning	0.4
Coefficient dynamic friction	Cursory Tuning	0.2

Physical system		
Parameter	How obtained	Value
Finger radius	Engineered	1.5cm
Box dimensions	As manufactured	$10cm \times 5cm \times 12cm$

Table 7.5: Tracker and physics simulator parameters. The desired number of particles $|J|$ determines the re-sampling process. The convergence parameter C_{conv} determines the peakiness of the likelihood distribution. Matching tolerance m_{thresh} determines how close gradients must be (in chord length on the unit circle) to be counted as a matching pixel. Edge matching tolerance g_{thresh} determines how large the gradient magnitude must be to be considered an edge. Edge dilation distance scaling is how much effect on neighbouring pixels the edge information has during the edge dilation process. See Appendix F for specific details of the texture matching approach used.

Table 7.6: Key frame sets for numerical comparisons

Video	Case	Frame Nos	Times
1: Simple	Video 1.1	222	7.40s
2: End-on rotate	Video 2.1	91	3.03s
		92	3.07s
	Video 2.2	181	6.03s
		183	6.10s
3: Tipping	Video 3.1	100	3.67s
		118	3.93s
		119	3.97s
		134	4.47s
	Video 3.2	141	4.70s
		146	4.87s
		150	5.00s
	Video 3.3	308	10.27s
		317	10.57s
Video	Case	Frame Nos	Times
4: Colour occlusion	Video 4.1	89	2.97s
		90	3.00s
	Video 4.2	153	5.10s
		154	5.13s
		155	5.17s
		156	5.20s
		162	5.40s
	Video 4.3	279	9.30s
		285	9.50s
		286	9.53s
		287	9.57s
		288	9.60s
Video	Case	Frame Nos	Times
5: Hand occlusion, tipping	Video 5.1	86	2.87s
		87	2.90s
		90	3.00s
		107	3.57s
		109	3.63s
	Video 5.2	182	6.07s
		190	6.33s
	Video 5.3	221	7.37s
		222	7.40s
		230	7.67s
		231	7.70s
		239	7.97s
	Video 5.4	276	9.20s
		278	9.27s
		298	9.93s
	Video 5.5	321	10.70s
		328	10.93s
Video	Case	Frame Nos	Times
6: Occlusion while tipping	Video 6.1	101	3.37s
		110	3.67s
		122	4.07s
		123	4.10s
		124	4.13s
	Video 6.2	203	6.77s
		204	6.80s
		213	7.10s
		227	7.57s
		237	7.90s
		238	7.93s
	Video 6.3	263	8.77s
		264	8.80s
		274	9.13s
		302	10.07s
		303	10.10s
		304	10.13s
		306	10.23s

7.4 Full comparison

In this first section a full multi-way comparison is done between all of the approaches considered here. The main approaches considered are:

- The original no-simulation with dispersion noise model (O).
- The simulation with dispersion noise model (D).
- The simulation with force noise model (F).
- A mixture condition combining the first two (MOD).
- Combining all the models (MODF).
- Combining just the novel models (MDF).

Within the above approaches, different retention schemes are also analysed:

- Rank-retention (../R).
- Mixture-retention (../M).
- No retention (..).

Where possible, conditions involving different numbers of recursions of the filter at each time-step are added:

- One recursion (..-1)
- Two recursions (..-2).

In the simulation conditions, performance is also compared with performance obtained by adding the known robot finger is added to the simulation:

- No finger (..).
- Known finger (..(K)).

All algorithm box-whisker plots per condition. First, box-whisker plots for each condition are provided, ranging across all possible retention systems and recursion counts. These charts are found in Figures 7.8 and 7.9[1].

From these plots can be read the performance of any of the algorithms tested on any of the scenarios discussed here. It will be observed that the distributions of results here often have multiple modes. This is because there is likely to be a distractor or alternative behaviour, which causes relatively consistent penalties.

Partial ordering over algorithms. Next, in order to summarise the relative ability of each of these approaches on the scenarios used here, an automatically generated partial order over the algorithms is presented (Figure 7.10). For each pair of algorithms on each scenario, if the mean difference in error over 40 trials is over a threshold (20 pixels) then one of the algorithms is potentially better than the other[2].

For each pair of algorithms on each scenario for which the above threshold is reached, a Mann Whitney rank-sum test (H. B. Mann, 1947) is run by ranking the individual errors of the trials from each of the two algorithms run on the scenario. If the Mann Whitney test is significant ($p = 0.01$), domination of one algorithm by the other is accepted for this scenario. The set of all such dominations is rendered as a graph over algorithms, where each edge represents such a pairwise relationship. Clearly such a diagram contains a lot of cycles, so each cycle in the graph is removed by deleting all edges in the cycle. The resulting graph (Figure 7.10) only retains edges between two algorithms if one algorithm dominates the other in all scenarios where there is a significant difference.

As can be seen from these summary graphs, the simulation with force-noise conditions (F) are the clear winners on the scenarios in this set of experiments. There is no clear

[1]The box-whisker plots in Figures 7.8 and 7.9 show the numerical results across all scenarios and algorithms. Each chart represents the behaviour of all algorithms on one scenario (recall that each scenario is a small slice of time in one of the test videos). Box-whisker plots illustrate the median, maximum, minimum, upper and lower quartile error, giving a good impression of the spread of performance of an algorithm.

[2]The reason why a threshold on the mean error difference is required is that the ground truth is only approximate and so the actual errors are within an interval containing the measured error. In order for an error difference to be always attributable to something other than error in the ground truth estimate the error differences must be thresholded.

winner between the no-simulation (O) and simulation with dispersion noise (D) conditions, though (D) does win in the tipping case. Some of the mixture conditions (M..) win on the scenarios here, but are dominated by the force-noise conditions (F). Indeed, the mixture condition without force-noise as a component (MOD) does not fare noticeably better than the no-simulation and simulation with dispersion noise conditions.

In the next sections the three main dynamics models (O, D and F) will be compared more closely, the effect of known finger location analysed, and finally the results of an experiment analysing the behaviour of these simulation-based dynamics models presented.

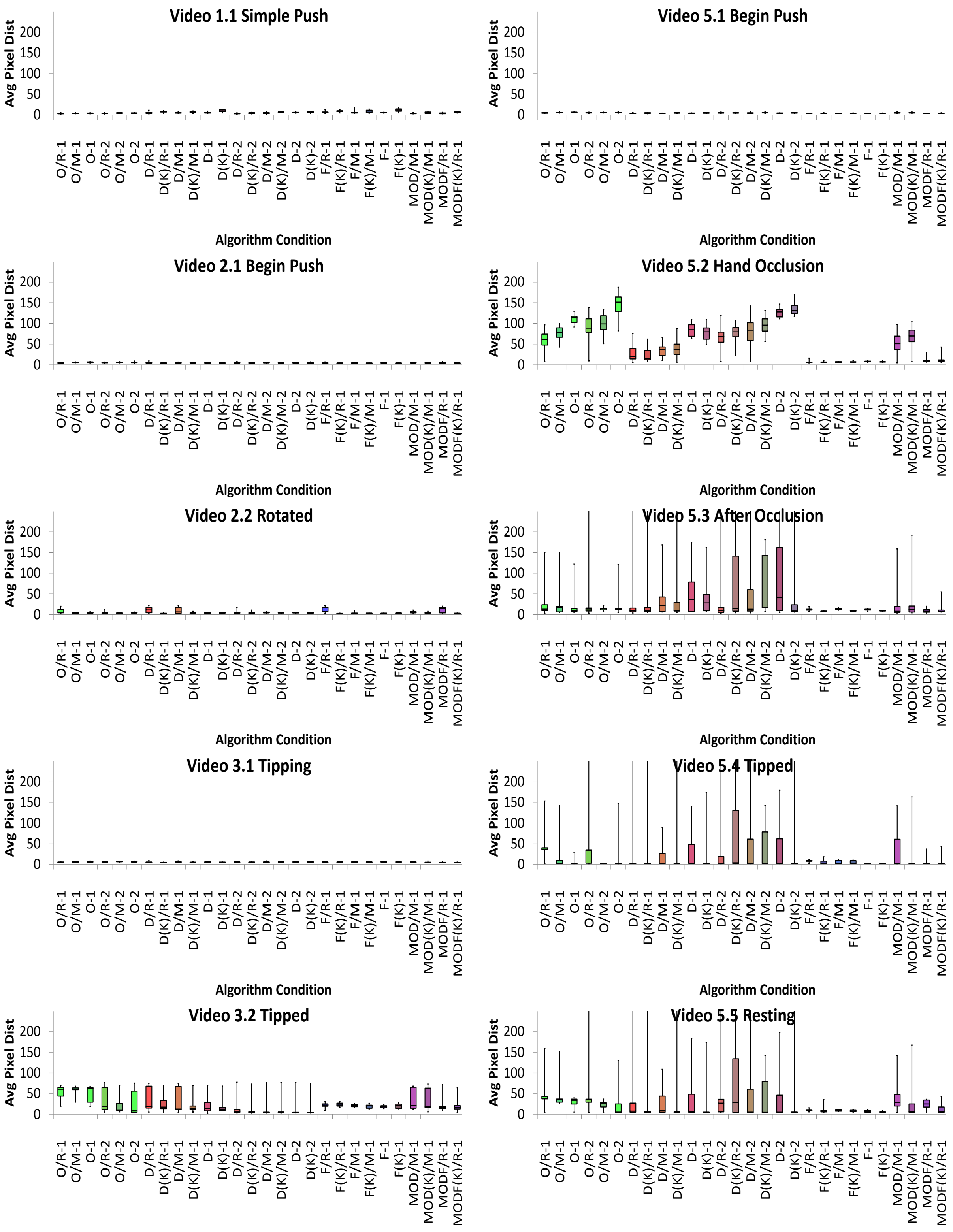


Figure 7.8: Error Analysis by Scenario. Box-whisker plots illustrate median, maximum, minimum, lower and upper quartiles of the error distribution. Each chart represents the performance of all algorithms over a single scenario, and each box-whisker represents the performance of one algorithm on that scenario.

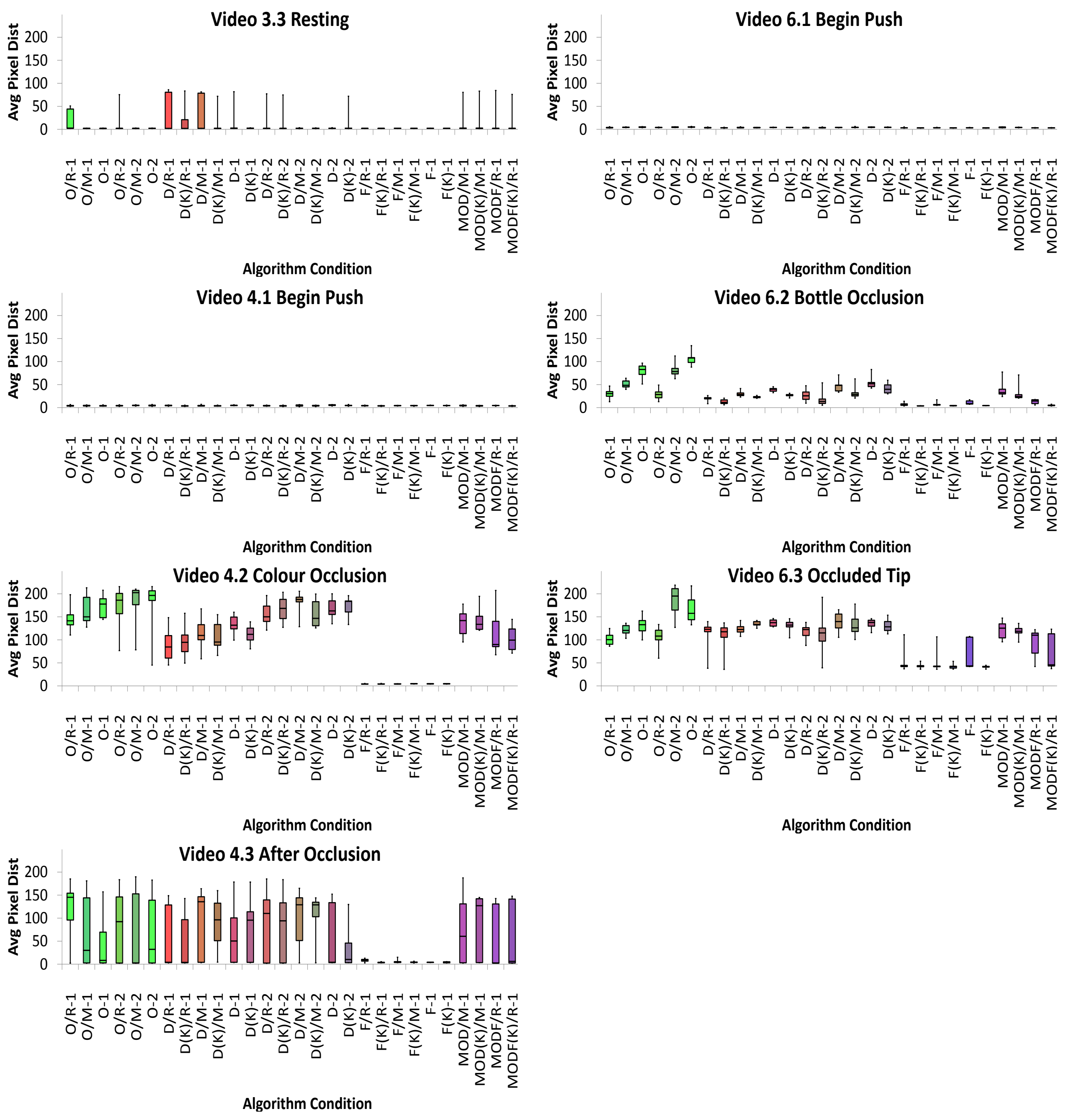


Figure 7.9: Error Analysis by Scenario. Box-whisker plots illustrate median, maximum, minimum, lower and upper quartiles of the error distribution. Each chart represents the performance of all algorithms over a single scenario, and each box-whisker represents the performance of one algorithm on that scenario.

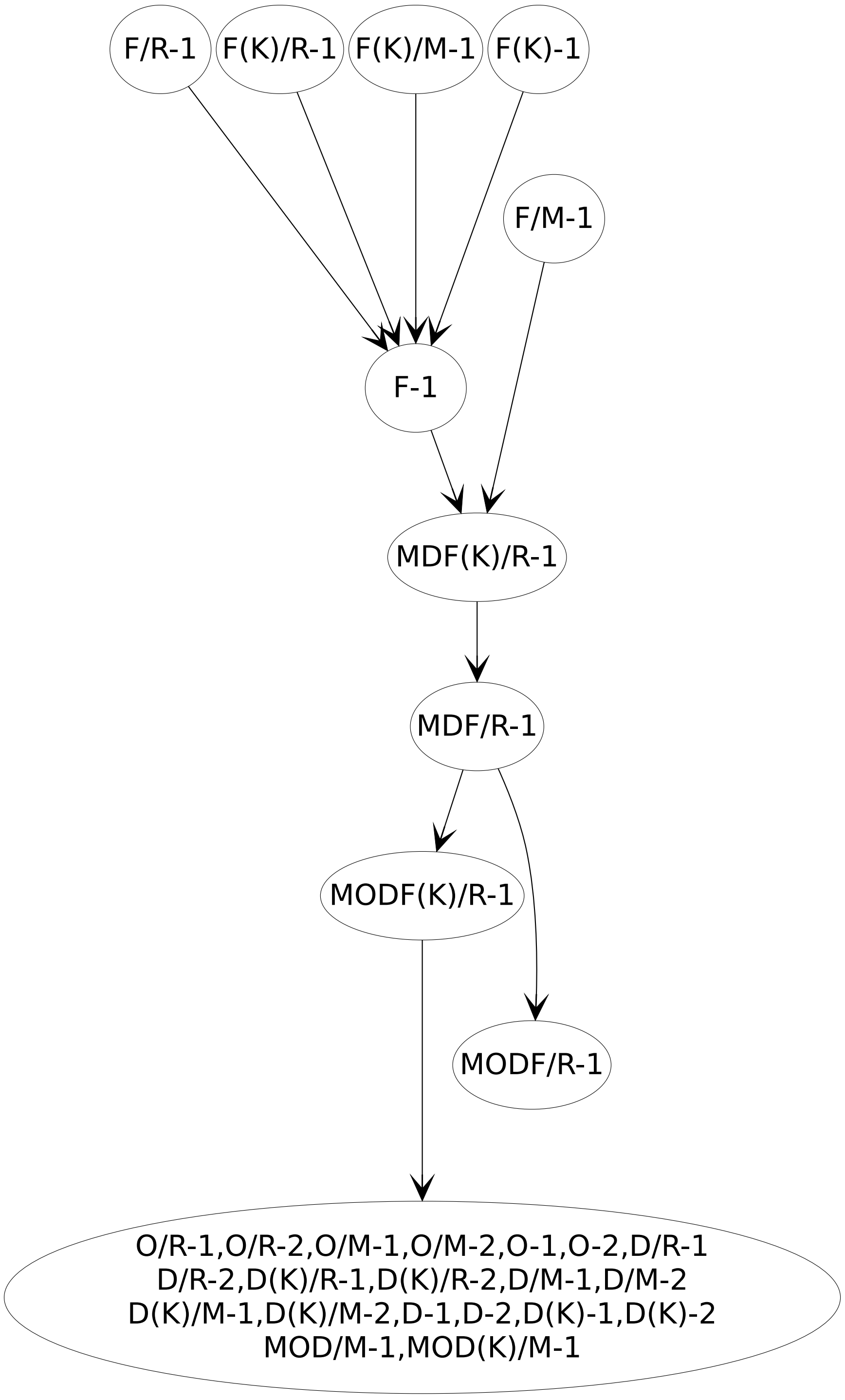

F/R-1
F(K)/R-1
F(K)/M-1
F(K)-1
F/M-1
F-1
MDF(K)/R-1
MDF/R-1
MODF(K)/R-1
MODF/R-1
O/R-1,O/R-2,O/M-1,O/M-2,O-1,O-2,D/R-1
D/R-2,D(K)/R-1,D(K)/R-2,D/M-1,D/M-2
D(K)/M-1,D(K)/M-2,D-1,D-2,D(K)-1,D(K)-2
MOD/M-1,MOD(K)/M-1

Figure 7.10: Partial ordering over all cases - each edge represents a mean difference in error of greater than 20 and a significant Mann-Whitney test in the same direction for a pair of conditions on one or more scenarios. Where cycles exist, all edges in the cycle are removed. Nodes and edges are merged where they are connected by the same edges or connect the same nodes.

7.5 Dynamics models comparison

This section focuses only on comparing the three primary dynamics models (no simulation - O, simulation with displacement noise - D, and simulation with force noise - F). Images are presented (Figures 7.11 to 7.16) illustrating performance. Results here are restricted to conditions using the rank retention scheme[1].

Then the mean performance of the three main dynamics models (O, D and F) is compared numerically across all key frame sets, for each of the rank retention scheme, mixture retention scheme and for no retention. These comparisons can be found in the line charts in Figures 7.17 7.18 and 7.19. As can be seen there, on many scenarios there is no clear winner, but on the harder scenarios, simulation with force-noise is the usual winner.

The results show that while the performance of simulation with dispersion noise is often equivocal with respect to the original no-simulation model, with the exception of the tipping scenario where it is a slight improvement, there is a clear improvement from the simulation with force noise model across all conditions.

[1]The rank retention scheme was employed in these experiments as this scheme seems to produce slightly better behaviour for the original (no simulation) dynamics and because it is was the scheme in use in the tracker before these experiments. This provides a marginally more difficult test for the new methods.

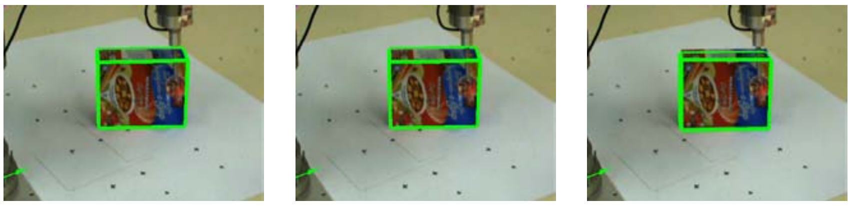


Figure 7.11: **Video 1**, phase 1.1 from top to bottom. **Left**: Gaussian dispersion, likelihood-rank retention, 2 recursions (condition O/R-2). **Middle**: Simulation with Gaussian dispersion, likelihood-rank retention, 2 recursions (condition D/R-2). **Right**: Simulation with force noise, likelihood-rank retention, 1 recursion (condition F/R-1). In the easy video 1 (box-whisker plots Figure 7.8) there is little difference between all conditions.

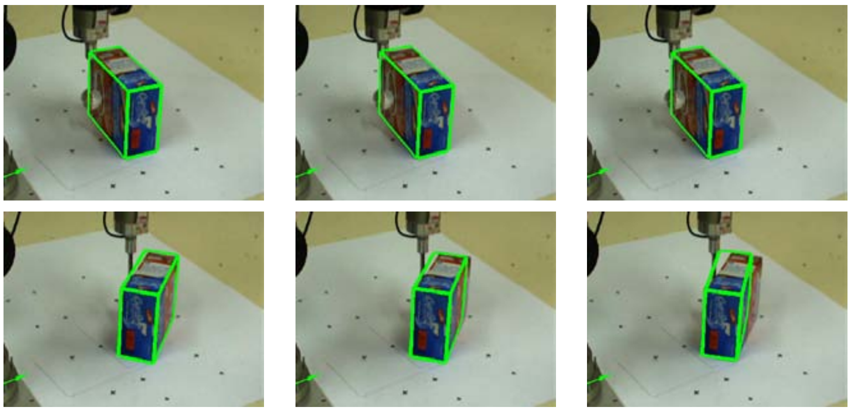


Figure 7.12: **Video 2**, phases 2.1 and 2.2 from top to bottom. **Left**: Gaussian dispersion, likelihood-rank retention, 2 recursions (condition O/R-2). **Middle**: Simulation with Gaussian dispersion, likelihood-rank retention, 2 recursions (condition D/R-2). **Right**: Simulation with force noise, likelihood-rank retention, 1 recursion (condition F/R-1). In video 2 (box-whisker plots Figure 7.8), where the object is pushed end-on to the camera, there is little difference in error rate - except that there is some lag in the force noise condition. This is attributable to the fact that the force noise condition presented in these images does not do multiple recursions, and consequently spends less time searching for presumably narrow observation likelihood optima. This conclusion is supported by the observation that with the single recursion non-simulation and simulation with dispersion noise conditions, similar lags occur - see the box-whisker plot 2.2 in Figure 7.8 for a summary of all conditions over this scenario. As can be seen in the line chart in Figure 7.19, not using a retention system at all solves the problem of lag - presumably because retention favours previously found observation likelihood optima which can have a higher observation likelihood than newly found correct optima, since newly found optima often need more refinement to obtain a better likelihood (particularly true with a peaky likelihood).

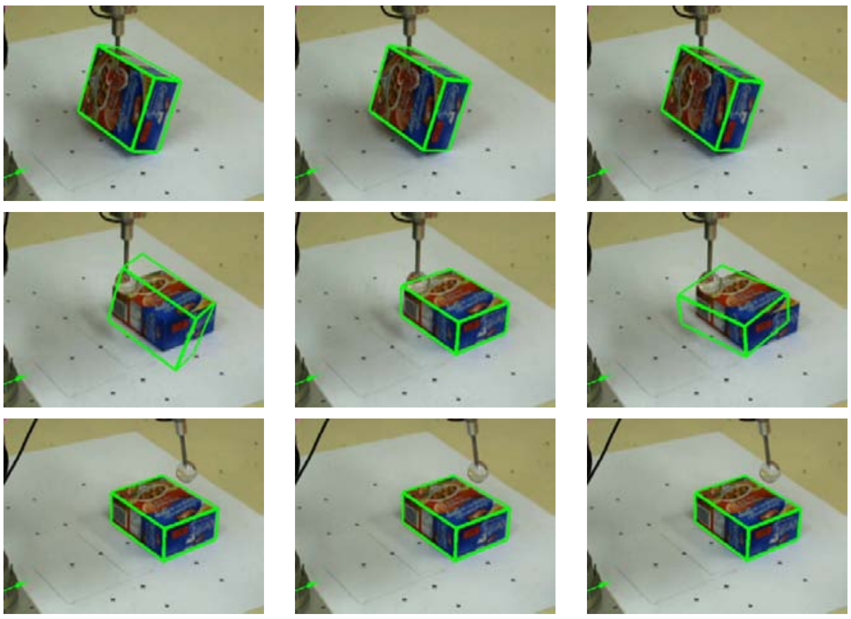


Figure 7.13: **Video 3**, phases 3.1, 3.2 and 3.3 from top to bottom. **Left**: Gaussian dispersion, likelihood-rank retention, 2 recursions (condition O/R-2). **Middle**: Simulation with Gaussian dispersion, likelihood-rank retention, 2 recursions (condition D/R-2). **Right**: Simulation with force noise, likelihood-rank retention, 1 recursion (condition F/R-1). Box-whisker plots are Figures 7.8 and 7.9. The existence of a simulation-based model of any kind enables the tracker to better track the object through the tip, where the object escapes from the search window of the basic tracker. An important aspect of this success is that the algorithm also tracks the beginning of the tip - even though the difference in pose from the original pose is small, for the simulation conditions, it puts the posterior distribution much closer to the next basin of attraction, and so in dynamics terms much closer to the tipped object. The improved dynamics model here guides sampling so that the new object location is more easily found. In the case of simulation with dispersion dynamics with one recursion, if the initial tip is not tracked, the algorithm finds it difficult to recover - this is because when the algorithm gets trapped in a bad pose, it finds a low-energy pose from which transition is difficult. This effect can be seen most clearly in the line chart in Figures 7.17 to 7.19.

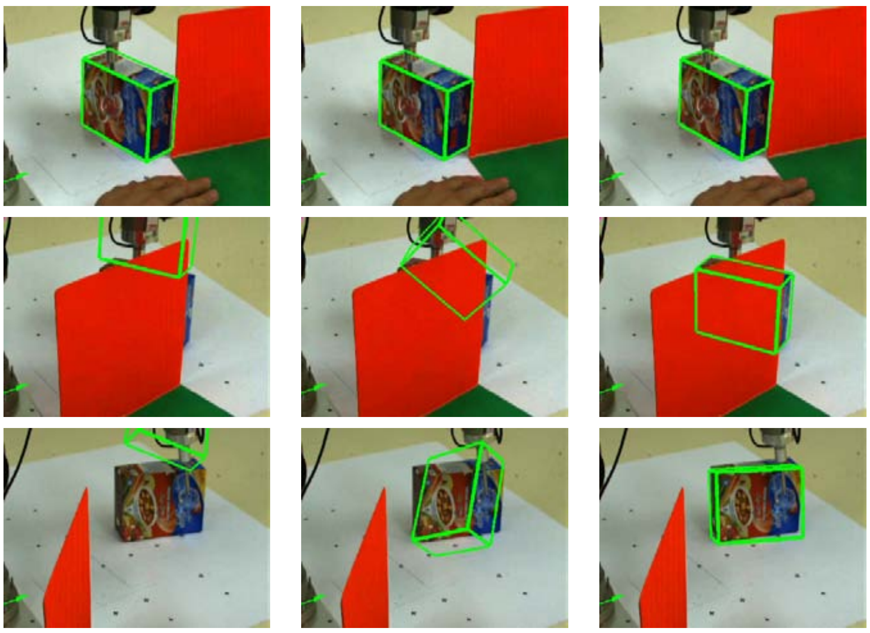


Figure 7.14: **Video 4**, phases 4.1, 4.2 and 4.3 from top to bottom. **Left**: Gaussian dispersion, likelihood-rank retention, 2 recursions (condition O/R-2). **Middle**: Simulation with Gaussian dispersion, likelihood-rank retention, 2 recursions (condition D/R-2). **Right**: Simulation with force noise, likelihood-rank retention, 1 recursion (condition F/R-1). In video 4 (box-whisker plots in Figure 7.9), the benefit of the simulation with force noise condition can be seen. When the target object is occluded by a plain coloured occluder and the only edges for the tracker to lock onto are on the background or the edge of the occluder, then the other methods fail by locking onto those distracting edges. However, the movements required to obtain such a lock are improbable with a force noise dynamics, so the stationary occluded object is tracked. It can also be seen that the simulation with dispersion noise condition offers some improvement by concentrating search closer to the ground plane, where the object is, thereby tending not to fail by as large a margin. Once the occluder is removed, if track was lost during occlusion, it is often not recovered.

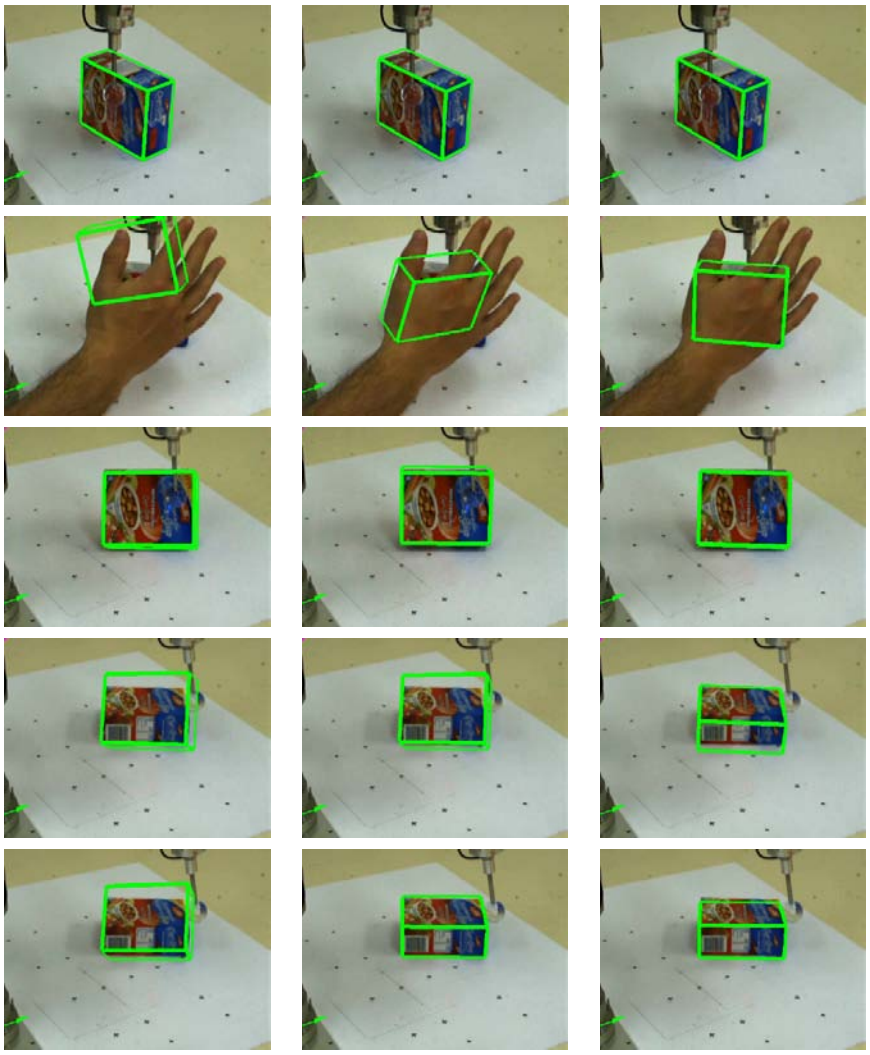


Figure 7.15: **Video 5**, phases 5.1, 5.2, 5.3, 5.4 and 5.5 from top to bottom. **Left**: Gaussian dispersion, likelihood-rank retention, 2 recursions (condition O/R-2). **Middle**: Simulation with Gaussian dispersion, likelihood-rank retention, 2 recursions (condition D/R-2). **Right**: Simulation with force noise, likelihood-rank retention, 2 recursions (condition F/R-1). In video 5 (box-wisker plots in Figure 7.8, the scenario with the object under occlusion (this time by a waving hand) is parallel with that in video 4: the simulation with force noise condition tracks the object accurately, while other conditions fail to varying degrees. Again, the simulation with dispersion noise does slightly better than the non-simulation conditions on this scenario. Most conditions are able to recover after the occluder in this case. This time the simulation with dispersion noise condition statistically is less able to recover (though under the basic 2-recursion rank-retention condition fares as well as other conditions). A failure to recover from the occlusion tends to persist through the subsequent tipping of the object also. The force-noise condition tracks throughout the whole video, and the simulation conditions track better through the tip if they have retained track to that point in the video.

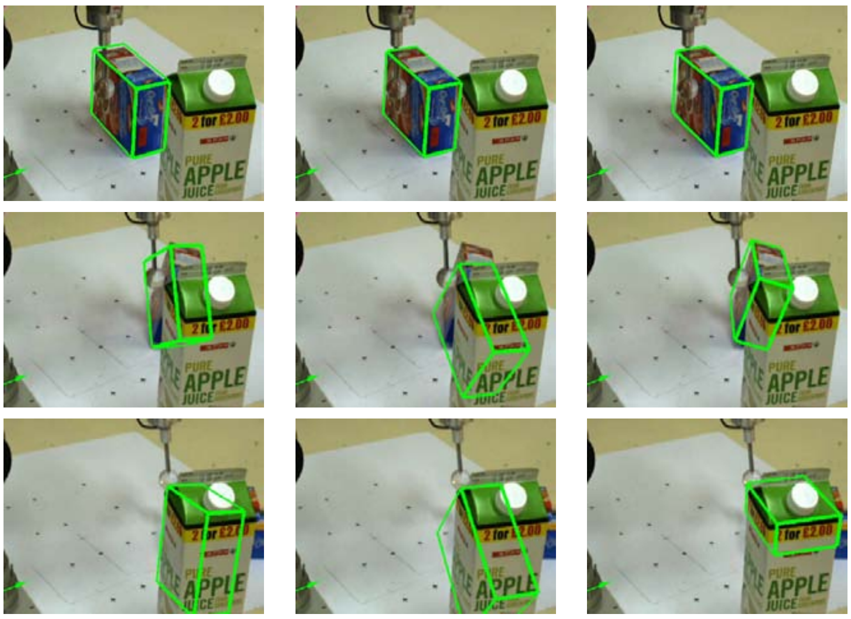


Figure 7.16: **Video 6**, phases 6.1, 6.2 and 6.3 from top to bottom. **Left**: Gaussian dispersion, likelihood-rank retention, 2 recursions (condition O/R-2). **Middle**: Simulation with Gaussian dispersion, likelihood-rank retention, 2 recursions (condition D/R-2). **Right**: Simulation with force noise, likelihood-rank retention, 1 recursion (condition F/R-1). The final video, video 6 (box-whisker plots of algorithm error distribution in Figure 7.9) is the most difficult case for any tracker due to simultaneous occlusion and fast object movement. Only the simulation with force-noise condition is able to succeed here, by tracking only plausible object motion. The largely successful track of the tipped box succeeds by locking onto contours of the occluding object.

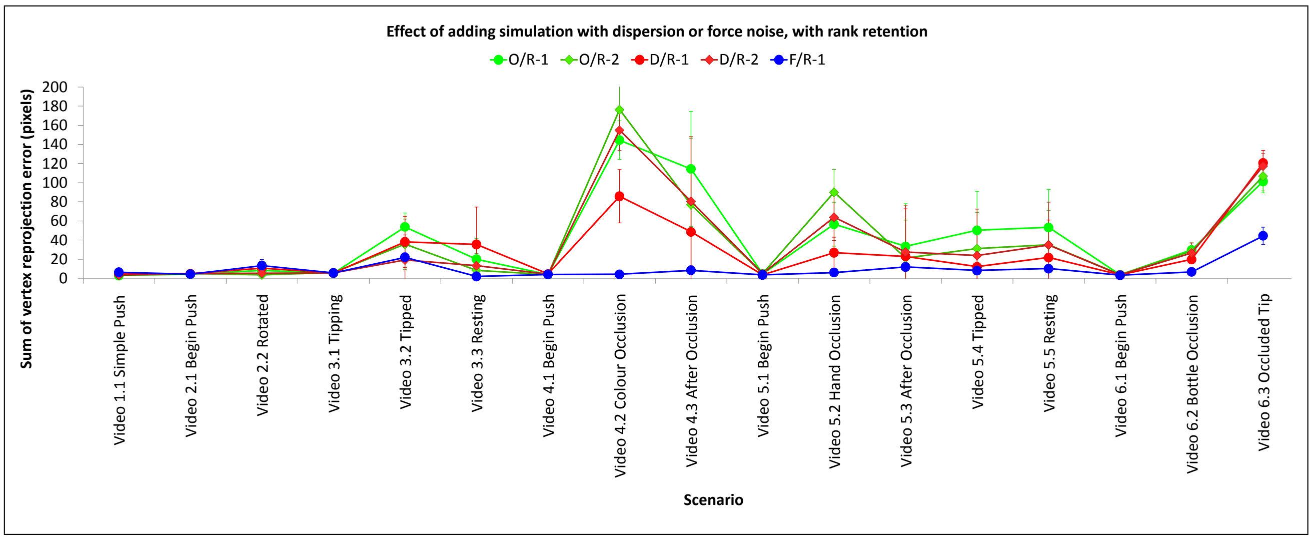


Figure 7.17: Mean error and standard deviation across all scenarios, comparing primary conditions with rank retention: **O/R-1**: no simulation, rank retention, 1 recursion. **O/R-2**: no simulation, rank retention, 2 recursions. **D/R-1**: simulation with dispersion noise, rank retention, 1 recursion. **D/R-2**: simulation with dispersion noise, rank retention, 2 recursions. **F/R-1**: simulation with force noise, rank retention, 1 recursion.

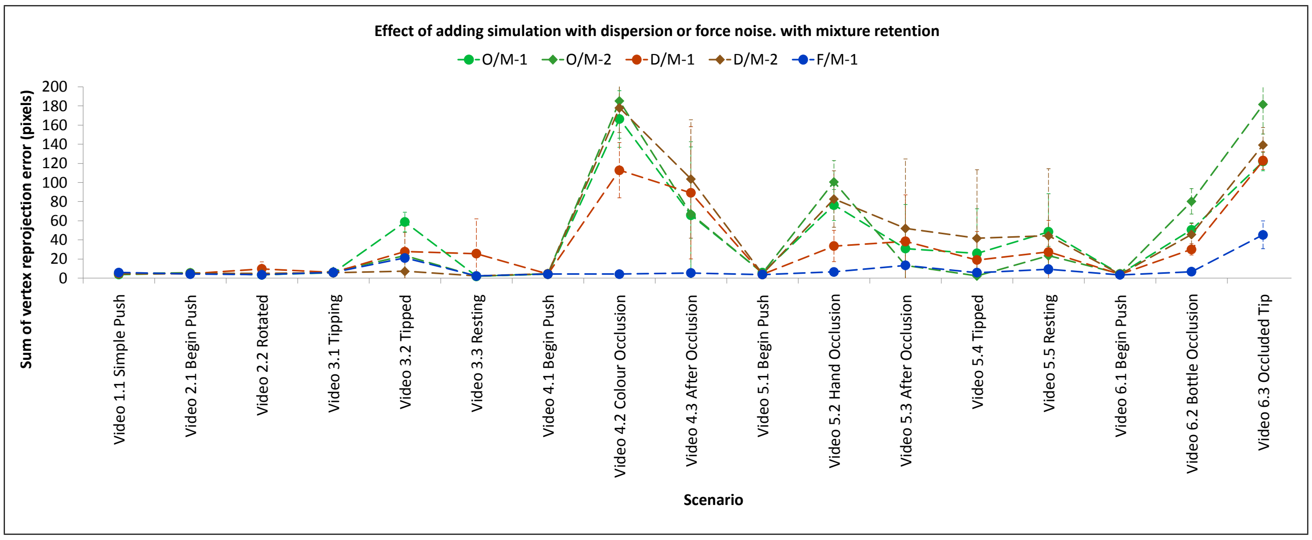


Figure 7.18: Mean error and standard deviation across all scenarios, comparing primary conditions with mixture retention: **O/M-1**: no simulation, mixture retention, 1 recursion. **O/M-2**: no simulation, mixture retention, 2 recursions. **D/M-1**: simulation with dispersion noise, mixture retention, 1 recursion. **D/M-2**: simulation with dispersion noise, mixture retention, 2 recursions. **F/M-1**: simulation with force noise, mixture retention, 1 recursion.

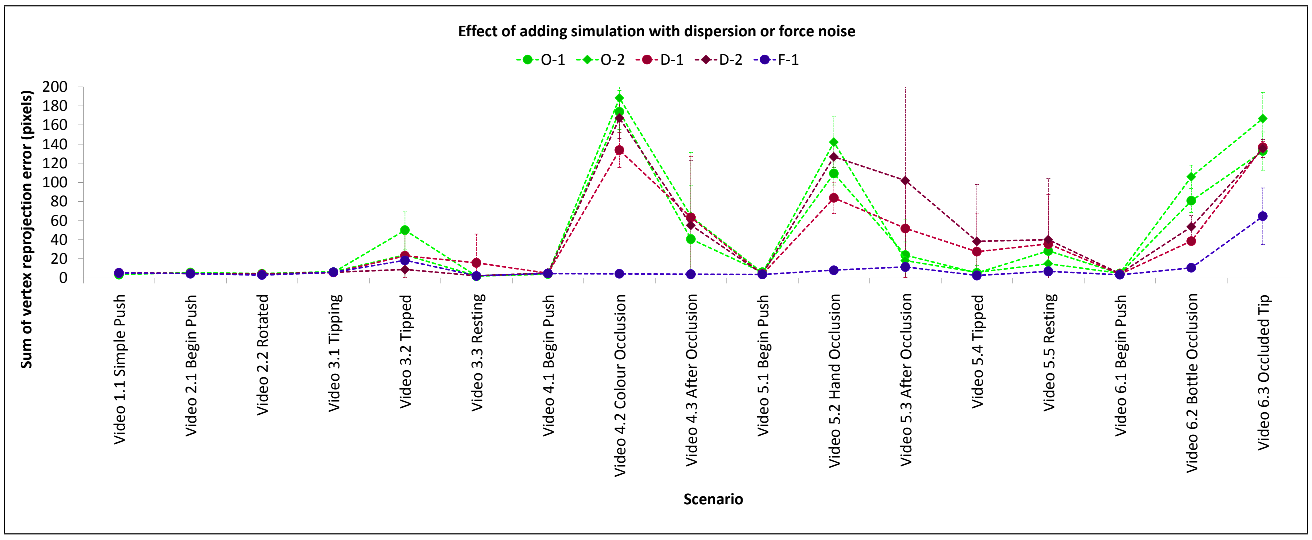


Figure 7.19: Mean error and standard deviation across all scenarios, comparing primary conditions with no retention: **O-1**: no simulation, no retention, 1 recursion. **O-2**: no simulation, no retention, 2 recursions. **D-1**: simulation with dispersion noise, no retention, 1 recursion. **D-2**: simulation with dispersion noise, no retention, 2 recursions. **F/M-1**: simulation with force noise, no retention, 1 recursion.

7.6 Effect of known robot finger location

In this subsection, the results are analysed with respect to the effect of including information about the known finger location.

For a numerical comparison of the mean performance of the simulation with dispersion noise dynamics model between unknown and known finger location, over all scenarios, see Figure 7.20 for the rank retention conditions, 7.21 for the mixture retention conditions and 7.22 for the no retention conditions. Analogously, Figures 7.23, 7.21 and 7.22 deal comparing the performance of the simulation with force noise conditions, with and without known finger location. Then, a set of images are presented illustrating the effect of adding known finger location to the simulation cases - Figures 7.26 to 7.31.

For simulation with dispersion noise, only in the tipping scenario (3.2, 3.3) is any improvement obtained by including finger location information. In all other cases the effect is very small compared to the choice of retention method. Improvement in the tipping is most likely obtained because the new information helps drive search locally, but the dispersion method of adding noise does not retain realistic dynamics.

Known finger location is helpful more often for the force noise condition. However, the impact of including known finger location does not compare to the impact already accrued by modelling the target object as a physical object subject to force noise.

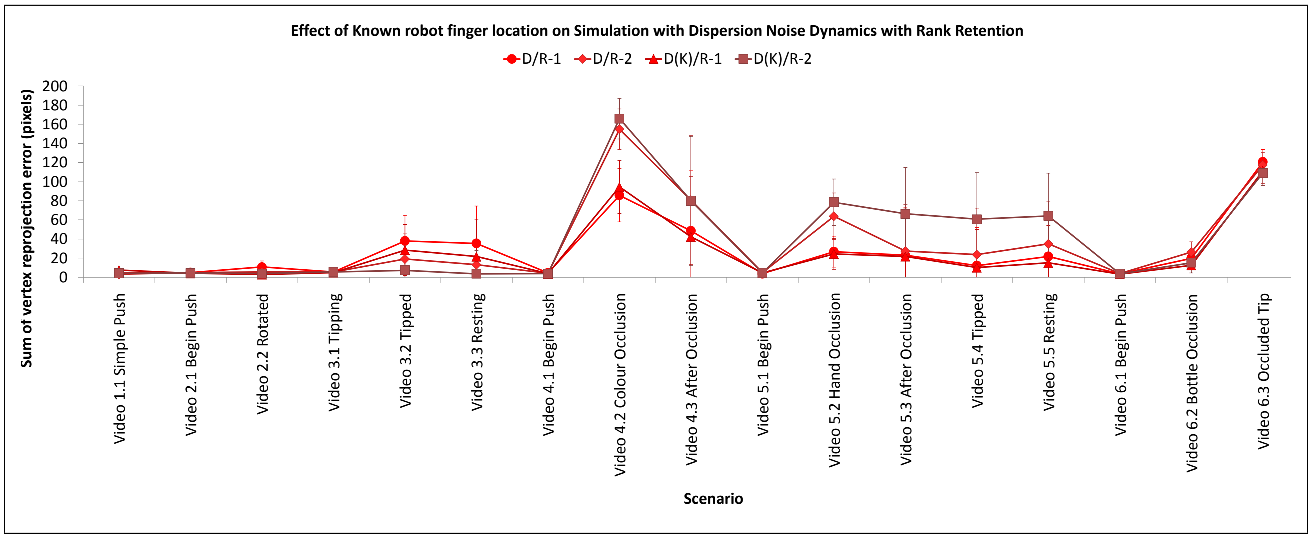


Figure 7.20: Mean error and standard deviation across all scenarios, comparing simulation with dispersion noise conditions with rank retention, with and without knowledge of finger location: **D/R-1**: simulation with dispersion noise, rank retention, 1 recursion. **D/R-2**: simulation with dispersion noise, rank retention, 2 recursions. **D(K)/R-1**: simulation with known finger location, dispersion noise, rank retention, 1 recursion. **D(K)/R-2**: simulation with known finger location, dispersion noise, rank retention, 2 recursions.

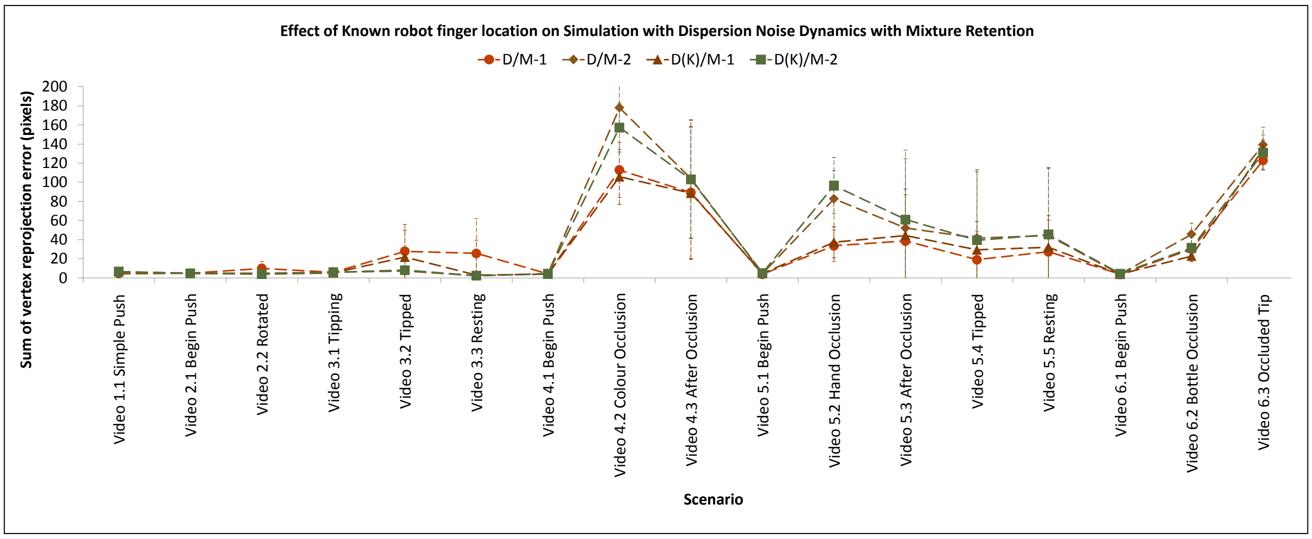


Figure 7.21: Mean error and standard deviation across all scenarios, comparing simulation with dispersion noise conditions with mixture retention, with and without knowledge of finger location: **D/M-1**: simulation with dispersion noise, mixture retention, 1 recursion. **D/M-2**: simulation with dispersion noise, mixture retention, 2 recursions. **D(K)/M-1**: simulation with known finger location, dispersion noise, mixture retention, 1 recursion. **D(K)/M-2**: simulation with known finger location, dispersion noise, mixture retention, 2 recursions.

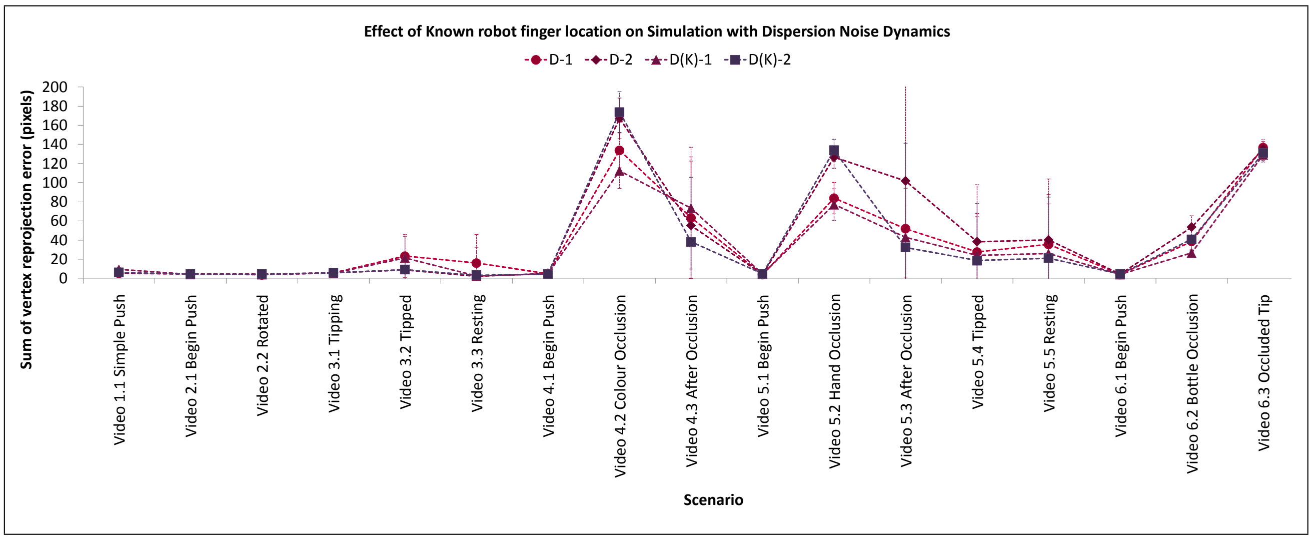


Figure 7.22: Mean error and standard deviation across all scenarios, comparing simulation with dispersion noise conditions with no retention, with and without knowledge of finger location: **D-1**: simulation with dispersion noise, no retention, 1 recursion. **D-2**: simulation with dispersion noise, no retention, 2 recursions. **D(K)-1**: simulation with known finger location, dispersion noise, no retention, 1 recursion. **D(K)-2**: simulation with known finger location, dispersion noise, no retention, 2 recursions.

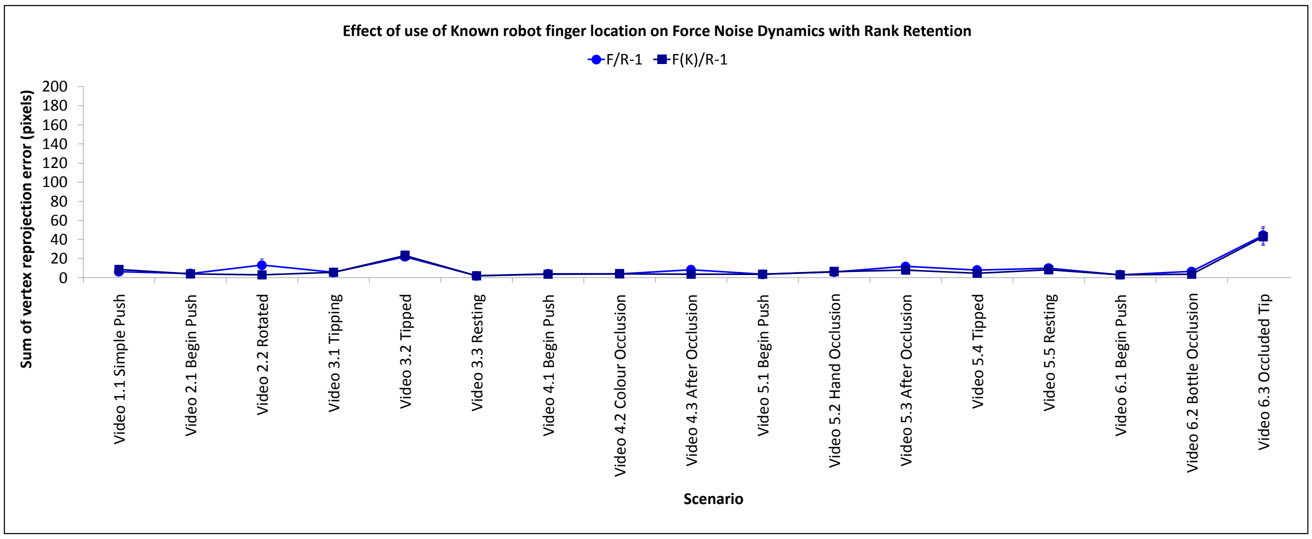


Figure 7.23: Mean error and standard deviation across all scenarios, comparing simulation with force noise conditions with rank retention, with and without knowledge of finger location: **F/R-1**: simulation with force noise, rank retention, 1 recursion. **F(K)/R-1**: simulation with force finger location, dispersion noise, rank retention, 1 recursion.

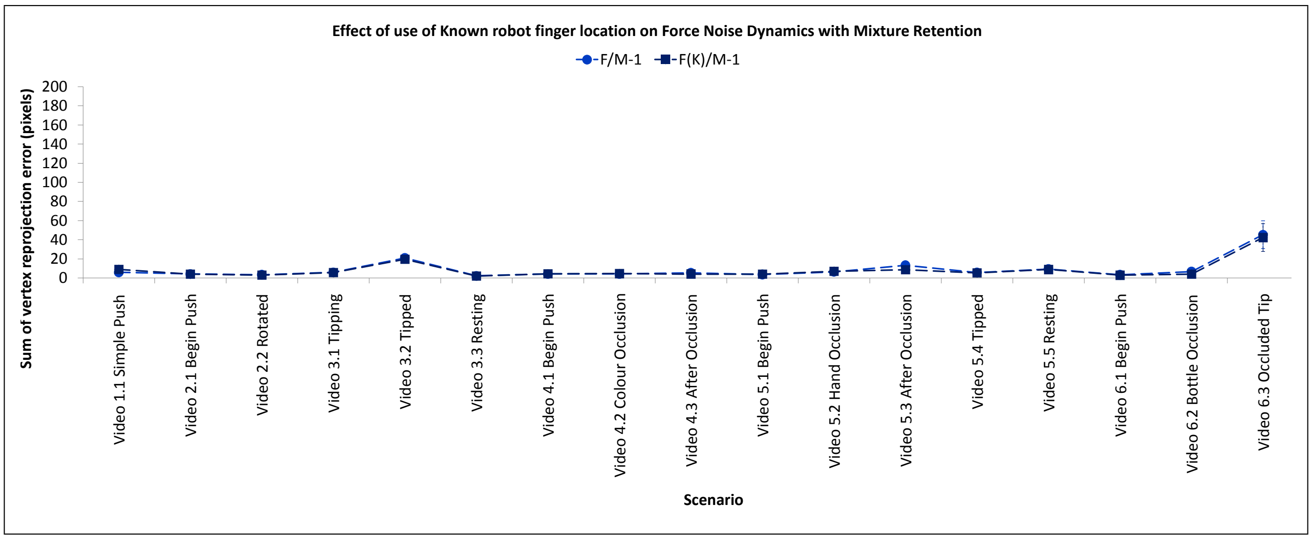


Figure 7.24: Mean error and standard deviation across all scenarios, comparing simulation with force noise conditions with mixture retention, with and without knowledge of finger location: **F/M-1**: simulation with force noise, mixture retention, 1 recursion. **F(K)/M-1**: simulation with force finger location, dispersion noise, mixture retention, 1 recursion.

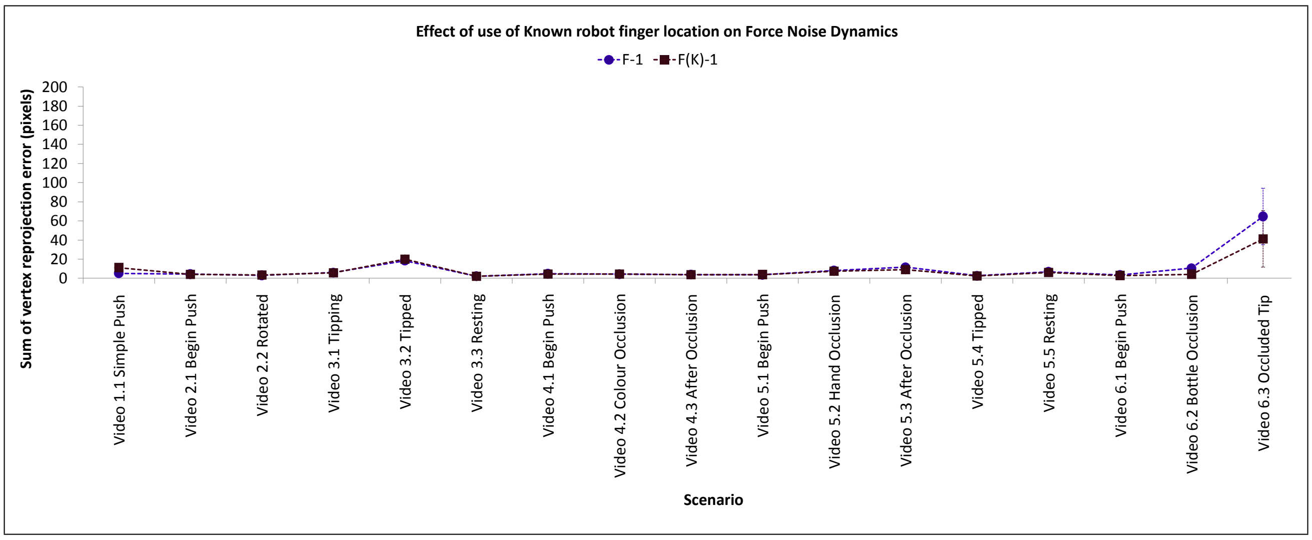
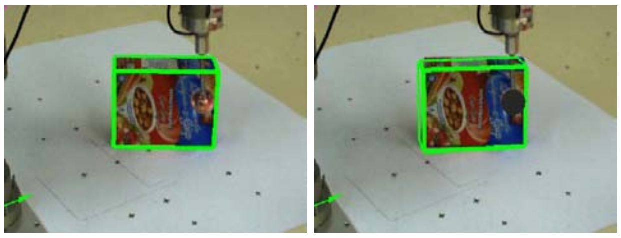
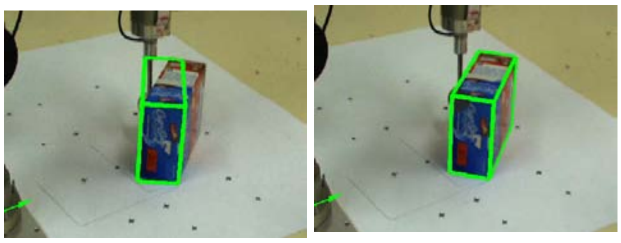


Figure 7.25: Mean error and standard deviation across all scenarios, comparing simulation with force noise conditions with no retention, with and without knowledge of finger location: **F-1**: simulation with force noise, no retention, 1 recursion. **F(K)-1**: simulation with force finger location, dispersion noise, no retention, 1 recursion.

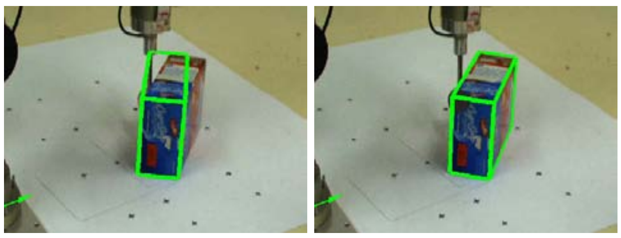


Frame 222 (scenario 1.1): Simulation dynamics with force noise, rank retention, known finger location - F/R-1 and F(K)/R-1

Figure 7.26: **Scenario 1.1 video slices.** The slight negative effect on this scenario of adding known finger location to the force-noise dynamics can be seen here. The finger's known location is rendered as a grey sphere on the image.

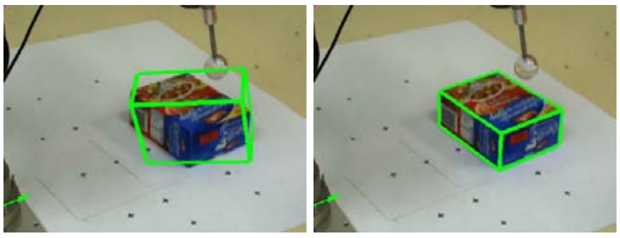


Frame 181 (scenario 2.2): Simulation dynamics with displacement noise, rank retention, one recursion at each time-step with and without known finger location - D/R-1 (left) and D/R-1(K) (right)

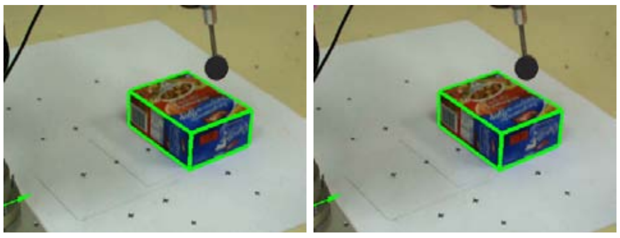


Frame 181 (scenario 2.2): Simulation dynamics with force noise , rank retention, one recursion at each time-step, with and without known finger location - F/R-1 (left) and F(K)/R-1 (right)

Figure 7.27: **Scenario 2.2 video slices.** The potential positive effect of including known finger location with force noise. Known finger conditions are shown on the right. The finger location also helps with the lag experienced by the force noise with rank retention condition in this video (see also Figure 7.8); this lag is only experienced under the rank retention scheme.

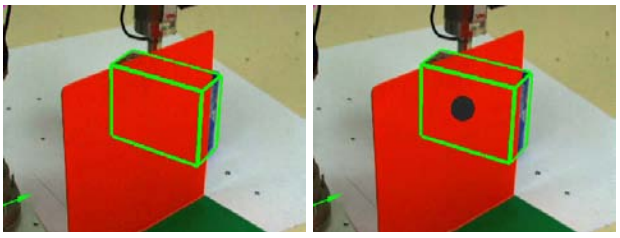


Frame 308 (scenario 3.3): Simulation dynamics with displacement noise, rank retention, one recursion at each time-step, without known finger location - D/R-1 (left and right)

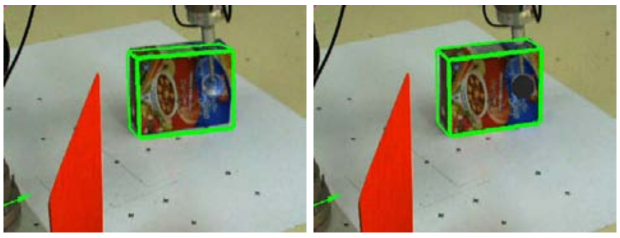


Frame 308 (scenario 3.3): Simulation dynamics with displacement noise, rank retention, one recursion at each time-step, with known finger location - D(K)/R-1 (left and right)

Figure 7.28: **Scenario 3.3 video slices.** Recovery from tip. Finger location noise improves the tracking in this scenario for all retention schemes (See also Chart 3.2 in Figure 7.8), particularly for simulation with displacement noise conditions, by reducing the probability of some unrealistic poses in which the finger intersects with the target object. Note that there are multiple outcomes for each condition because the tracker is non-deterministic (hence the statistics that were collected on its behaviour).

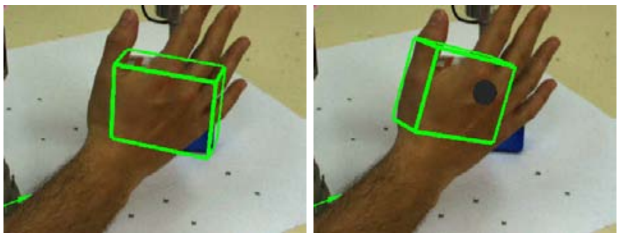


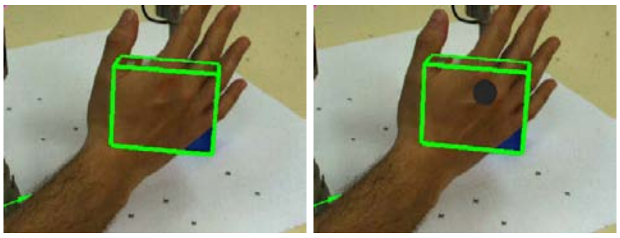
Frame 155 (scenario 4.2): Simulation dynamics with force noise , rank retention, one recursion at each time-step, with and without known finger location - F/R-1 (left) and F(K)/R-1 (right)

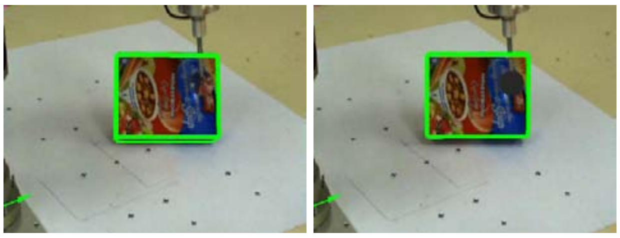


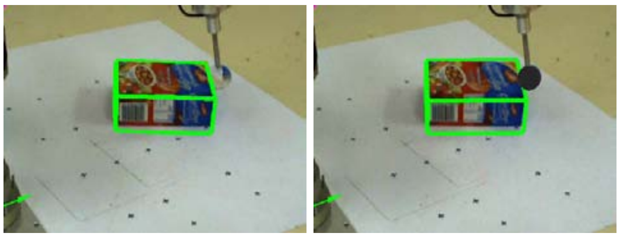
Frame 285 (scenario 4.3): Simulation dynamics with force noise , rank retention, one recursion at each time-step, with and without known finger location - F/R-1 (left) and F(K)/R-1 (right)

Figure 7.29: **Scenario 4.2 and 4.3 video slices.** The effect of known finger location in this video is equivocal (corresponding charts are in Figure 7.9).


Frame 190 (scenario 5.2): Simulation dynamics with displacement noise, rank retention, one recursion at each time-step, with unknown and known finger location - D/R-1 (left) and D(K)/R-1 (right)


Frame 190 (scenario 5.2): Simulation dynamics with force noise, rank retention, one recursion at each time-step, with and without known finger location - F/R-1 (left) and F(K)/R-1 (right)


Frame 239 (scenario 5.3): Simulation dynamics with force noise, rank retention, one recursion at each time-step, with and without known finger location - F/R-1 (left) and F(K)/R-1 (right)


Frame 278 (scenario 5.4): Simulation dynamics with force noise, rank retention, one recursion at each time-step, with and without known finger location - F/R-1 (left) and F(K)/R-1 (right)

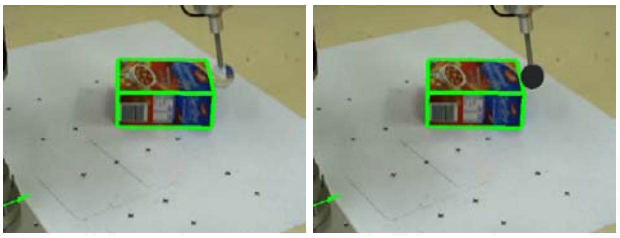
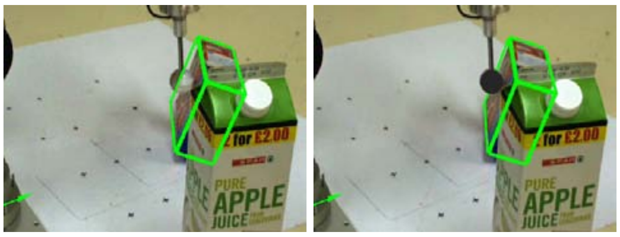
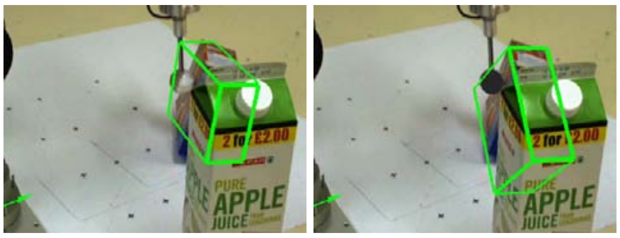
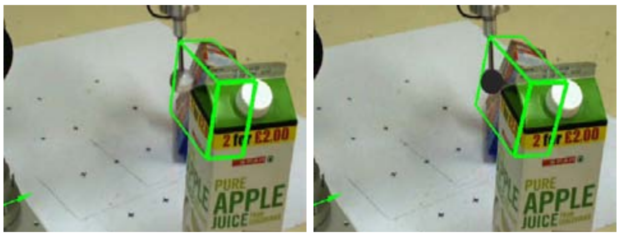
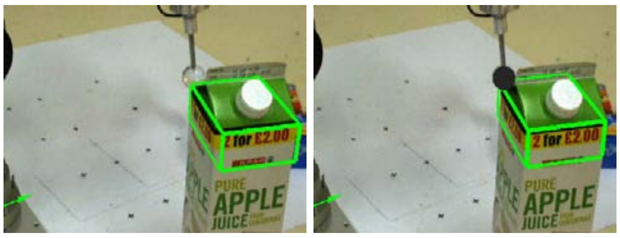

Frame 321 (scenario 5.5): Simulation dynamics with force noise, rank retention, one recursion at each time-step, with and without known finger location - F/R-1 (left) and F(K)/R-1 (right)

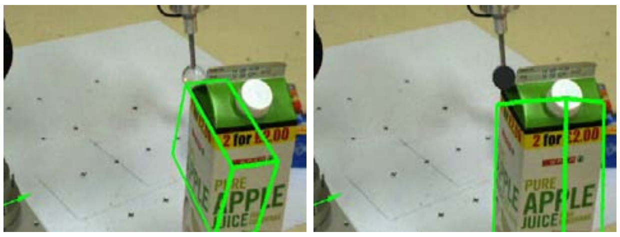
Figure 7.30: **Scenario 5.2, 5.3, 5.4 and 5.5 video slices.** The effect of having a finger in the force noise condition here is marginal. The effect of having a finger in simulation with displacement noise depends on the retention style used, as can be seen in Figure 7.8 Chart 5.5. There is no improvement during occlusion, but after the occlusion, known finger location improves slightly the simulation with force noise conditions; the improvement is on the order of a minor refinement however, presumably due to the inclusion of extra information constraining the exact location of the target object.
.


Frame 237 (scenario 6.2): Simulation dynamics with force noise, rank retention, one recursion at each time-step, with and without known finger location - F/R-1 (left) and F(K)/R-1 (right)


Frame 237 (scenario 6.2): Simulation dynamics with displacement noise, rank retention, two recursions at each time-step, with unknown and known finger location - D/R-2 (left) and D(K)/R-2 (right)


Frame 237 (scenario 6.2): Simulation dynamics with displacement noise, rank retention, one recursion at each time-step, with unknown and known finger location - D/R-1 (left) and D(K)/R-1 (right)


Frame 264 (scenario 6.3): Simulation dynamics with force noise , rank retention, one recursion at each time-step, with and without known finger location - F/R-1 (left) and F(K)/R-1 (right)


Frame 264 (scenario 6.3): Simulation dynamics with displacement noise, rank retention, two recursions at each time-step, with unknown and known finger location - D/R-2 (left) and D(K)/R-2 (right)

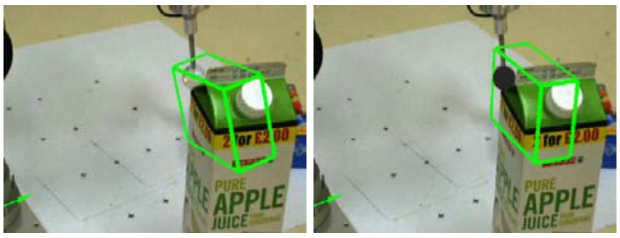

Frame 264 (scenario 6.3): Simulation dynamics with displacement noise, rank retention, one recursion at each time-step, with unknown and known finger location - D/R-1 (left) and D(K)/R-1 (right)

Figure 7.31: **Scenario 6.2 and 6.3 slices.** In these scenarios the retention scheme provides a small benefit to the simulation conditions (error plots in Figure 7.9) but this benefit can be replicated with known finger location. The reason for this non-additive combination is that in this scenario they produce the same intermediate benefit, the guiding of sampling towards the correct edges as the object becomes more obscured. This increased accuracy can be translated into an increase in accuracy after the occluded tip.

7.7 Mixture models

As previously discussed, it is possible to deploy mixtures of the primary dynamics models. Because the number of possible combinations is exceedingly large, only three alternative mixture models were tried - MOD/M-1, a mixture of no simulation and simulation with dispersion noise, a mixture retention scheme and 1 recursion at each time-step, MODF/R-1, a mixture of no simulation and simulation with dispersion noise, as well as simulation with force noise, a rank retention scheme and 1 recursion at each time-step, and MDF/M-1, a mixture of simulation with dispersion noise and simulation with force noise, a rank retention scheme and 1 recursion.

The use of mixtures could conceivably do better or worse than all of the existing models. In practice, the performance is, in general, a kind of average of the performance of the component models. Apart from the occasional interaction, this can be seen in Figures 7.32 and 7.33.

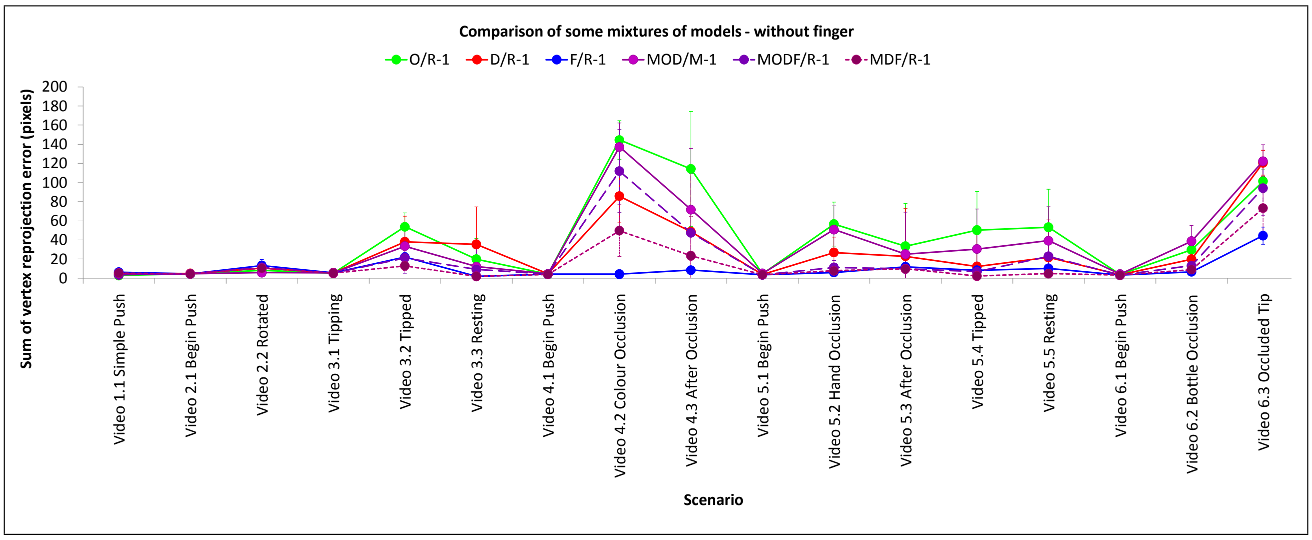


Figure 7.32: Mean error and standard deviation across all scenarios, comparing dynamics models based on single dynamics with those based on mixtures. **O/R-1**: dispersion-only dynamics, rank-retention. **D/R-1**: simulation with dispersion noise, rank retention. **F/R-1**: simulation with force noise, rank retention. **MOD/M-1**: a mixture of dispersion-only and simulation with dispersion dynamics, with mixture retention. **MDF/R-1**: a mixture of simulation with dispersion dynamics and simulation with force noise, rank retention. **MODF/R-1**: a mixture of all the main dynamics models, dispersion-only, simulation with dispersion dynamics and simulation with force noise, rank retention. All conditions use 1 recursion at each time-step.

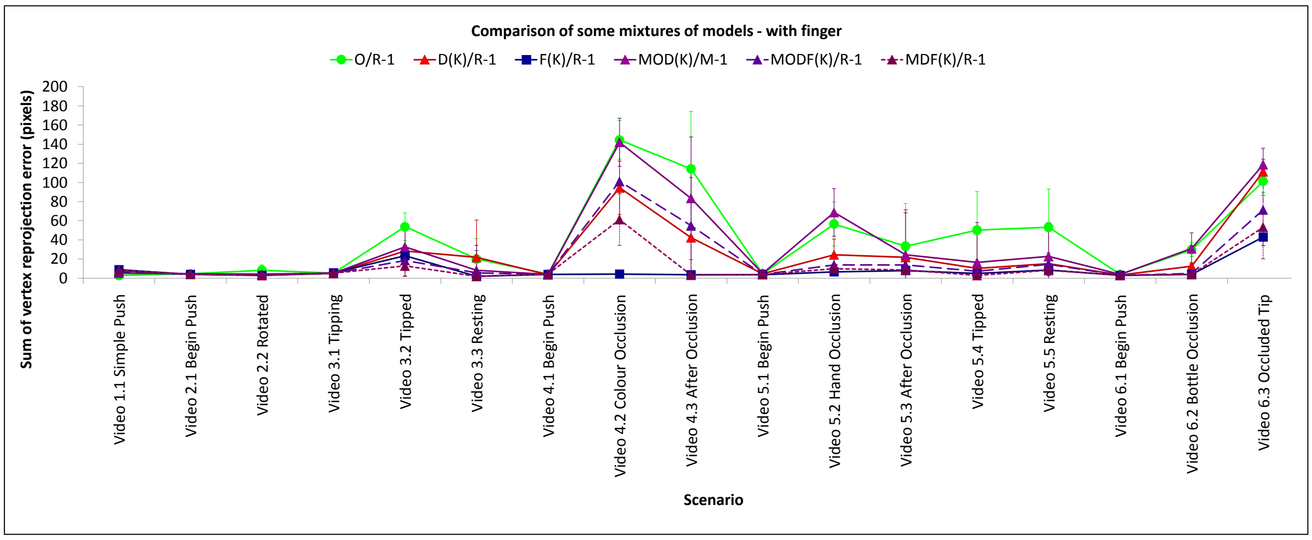


Figure 7.33: Mean error and standard deviation across all scenarios, comparing dynamics models based on single dynamics with those based on mixtures, with known finger location employed where applicable. **O/R-1**: dispersion-only dynamics, rank-retention. **D(K)/R-1**: simulation with dispersion noise, rank retention. **F(K)/R-1**: simulation with force noise, rank retention. **MOD(K)/M-1**: a mixture of dispersion-only and simulation with dispersion dynamics, with mixture retention. **MDF(K)/R-1**: a mixture of simulation with dispersion dynamics and simulation with force noise, rank retention. **MODF(K)/R-1**: a mixture of all the main dynamics models, dispersion-only, simulation with dispersion dynamics and simulation with force noise, rank retention. All conditions use 1 recursion at each time-step, and all simulation-based dynamics incorporate the known robot finger location.

7.8 Manufactured videos

Though the simulator-based stochastic dynamics models are likely to guide samples to plausible poses, they might not encompass movements that are not modelled by the simulator[1]. The proposition that these methods would retain robustness under model failure is put to the test directly by generating videos in which the assumptions of the simulator explicitly violated.

In the first such video (the first row of Figure 7.34), the whole image is gradually vertically rotated *up* in the field of view during the first half of the video and then down in the second half of the video. Since the camera is calibrated with respect to the ground plane before the film is taken, this has the effect that the object seems to float off the ground plane and then settle at the end.

In the second such video, the whole image is gradually vertically rotated *down* in the field of view in the first part of the video and back in the second half of the video. With respect to the calibrated camera and the model in the simulator, this would give the effect that the target object is sinking into the ground-plane and back. See the second row of results in Figure 7.34.

Instead of simulating the object floating or sinking, the physics-augmented trackers prefer hypotheses where the object stays close to the ground plane, thereby estimating the object receding or advancing along the ground plane in order to accommodate the object's vertical movement in the image plane. Future work is needed to remedy this fragility[2]. Proposals for increasing the robustness of the system are given below.

[1]In the case of simulation with added Gaussian noise, any movement is in theory possible due to the effect of the Gaussian noise (though some may be exceedingly improbable), and in the case of simulation with force noise, the only effect that is theoretically impossible is interpenetration of modelled objects, though unrealistic movements may be possible by exploiting the approximate nature of discrete physical simulation.

[2]Tests of tracker behaviour that account for real-world situations are necessary if the system is ever to be evolved to a state where it is robust in the kind of situation that is likely to be experienced during use and also where the dynamics models break down.

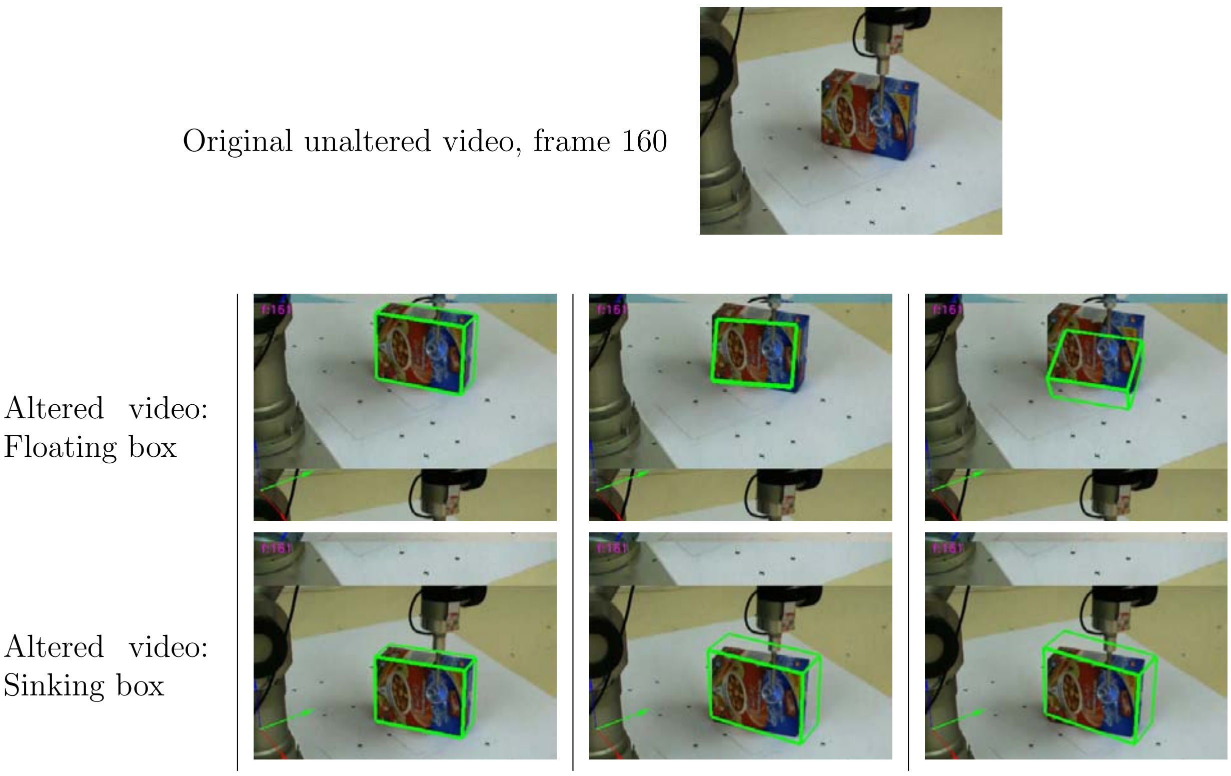

Original unaltered video, frame 160
Altered video: Floating box
Altered video: Sinking box

Figure 7.34: An illustration of the behaviour of the three basic dynamics models (left: without simulation with pose dispersion noise (**O**), middle: with simulation with pose dispersion noise (**O**), and right: simulation with force noise (**F**)) introduced in this thesis on two manufactured videos (one where the image rolls up during the video at top, effectively making the object float in the tracker's calibrated frame of reference, and one where the image rolls down at the bottom, conversely making the object in a sense appear to sink).

7.9 Sensitivity to parameters

The new, force noise based, tracker performs better than the non-physics tracker on the scenarios analysed here. However, robustness is important in this work, if the new approach is to be used practically. To this end, the sensitivity of the simulation-based approach to the extra simulation parameters should be analysed. This is done in this section by varying the level of dynamic and static friction, and the coefficient of restitution, while keeping other parameters fixed at the levels used for the above experiments.

Also, there are a large number of reasons why this improvement may be seen apart from the use of the realistic physics model, and it pays to eliminate the most obvious ones. One such possibility is that because the new approach incorporates noise in different units and so the noise levels of the old and new approaches are set separately[1], the effective level of noise in the new approach is what is causing the improvement. This is tested here by varying the noise level across both old (no physics) and new (force noise based physics) approaches.

7.9.1 Sensitivity to physics simulation parameters

The charts in Figure 7.35 show the effect of varying the coefficient of restitution of the condition with force noise and particle retention (F/R-1). The charts compare this with the performance of the corresponding original tracker without physics (O/R-1). Similarly, Figure 7.36 shows the effect of varying the coefficient of static friction. Figure 7.37 shows the effect of varying the coefficient of dynamic friction. In all experiments, other parameters are maintained at the levels used in previous experiments.

It can be seen that, in all cases, changing the coefficient of restitution has very little effect on the performance difference. Changing the coefficients of friction can have a

[1]The translation part of the dispersion noise is in units of metres, drawn from a a Gaussian distribution, the rotation part is in degrees. The noise for the force noise condition is in units of Newton seconds from a uniform distribution over an ellipse, incident at points drawn from a probability distribution over the shape surface.

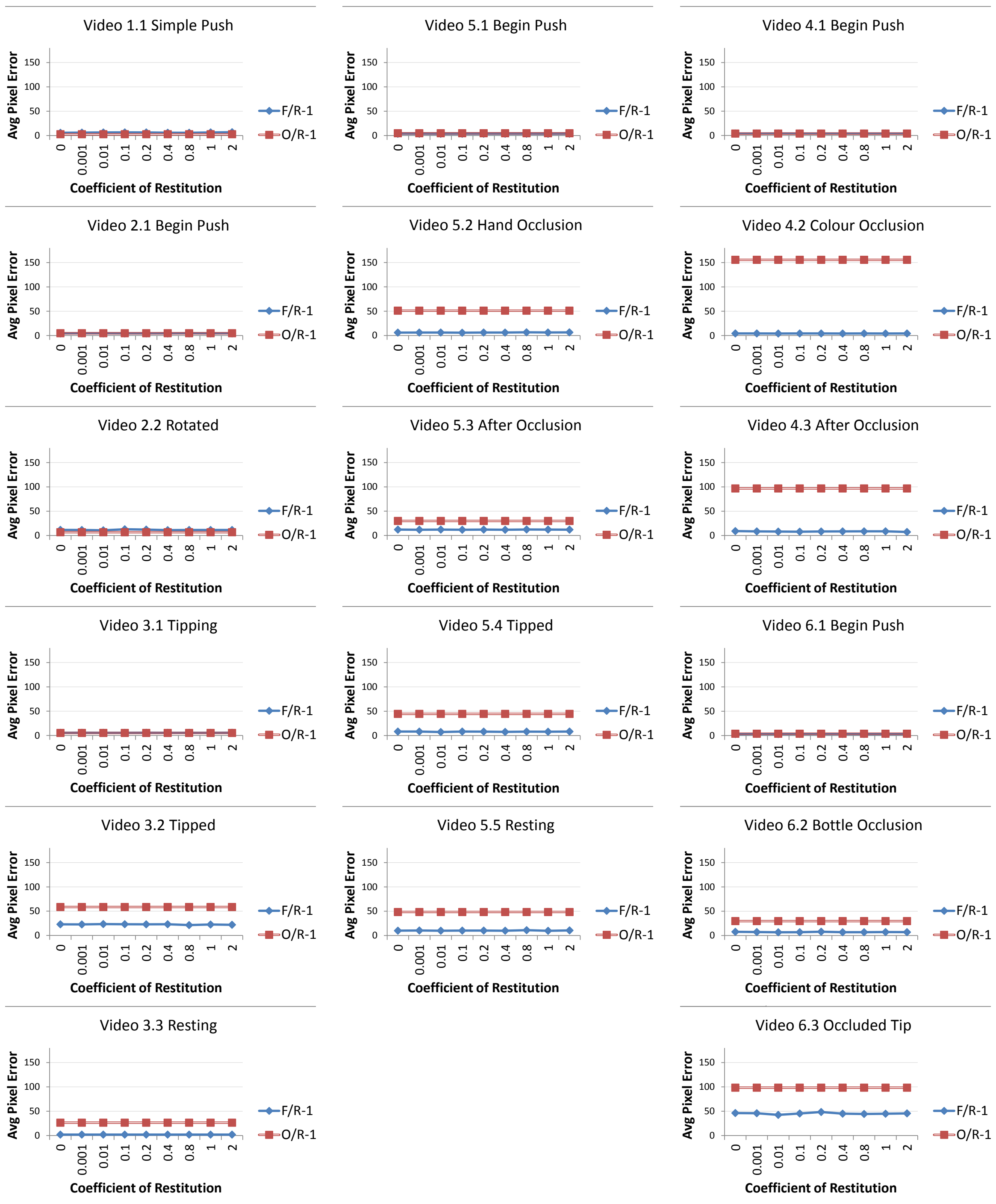


Figure 7.35: The effect of altering the *coefficient of restitution* on the physics simulation based tracker with force noise, and particle retention (F/R-1). Results for the non-physics based tracker with retention are also presented for comparison (O/R-1). Numbers are averaged over 40 trials. The default value of the coefficient in other experiments is 0.001.

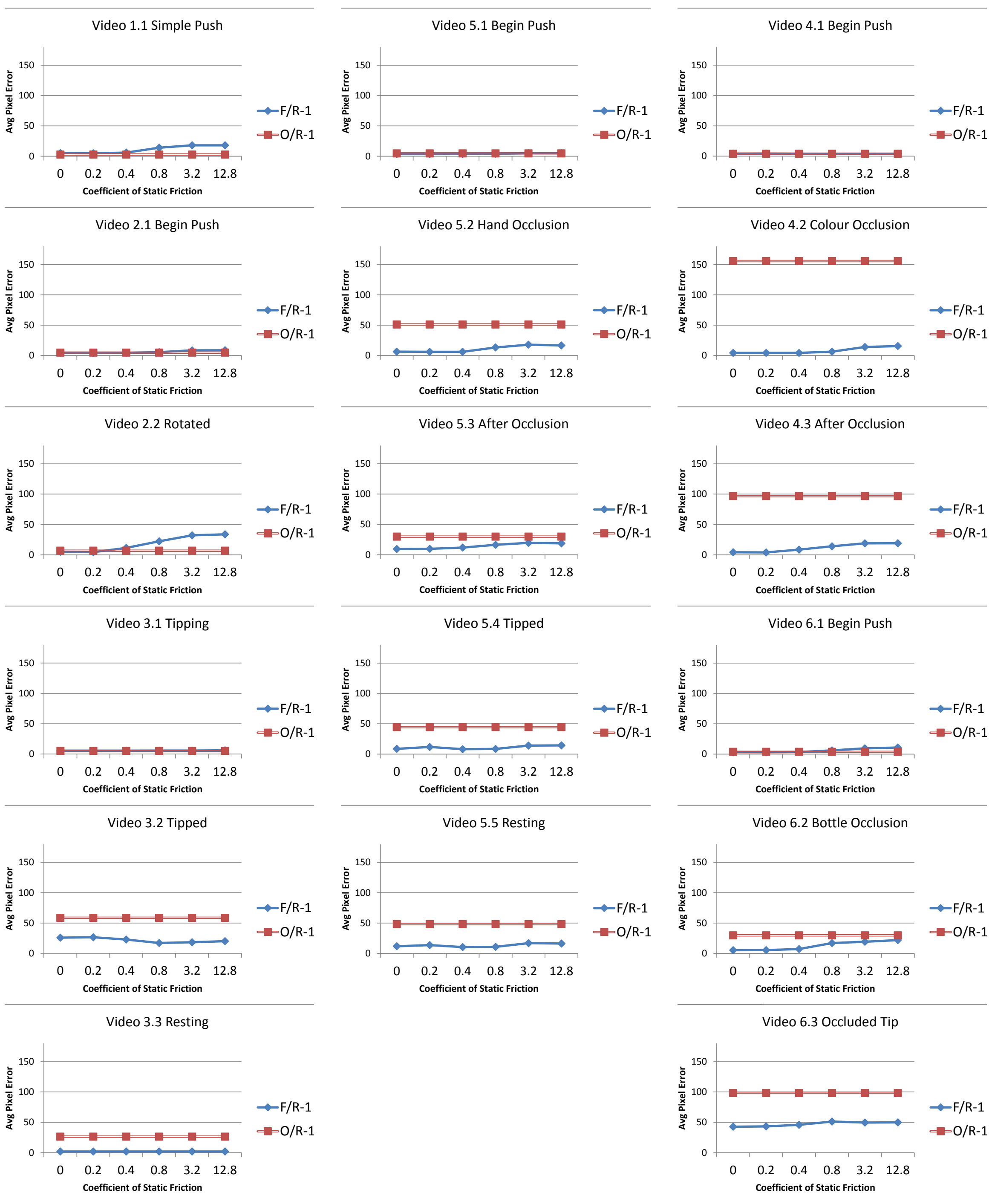


Figure 7.36: The effect of altering the *coefficient of static friction* on the physics simulation based tracker with force noise, and particle retention (F/R-1). Results for the non-physics based tracker with retention are also presented for comparison (O/R-1). Numbers are averaged over 40 trials. The default value of this coefficient in other experiments is 0.4.

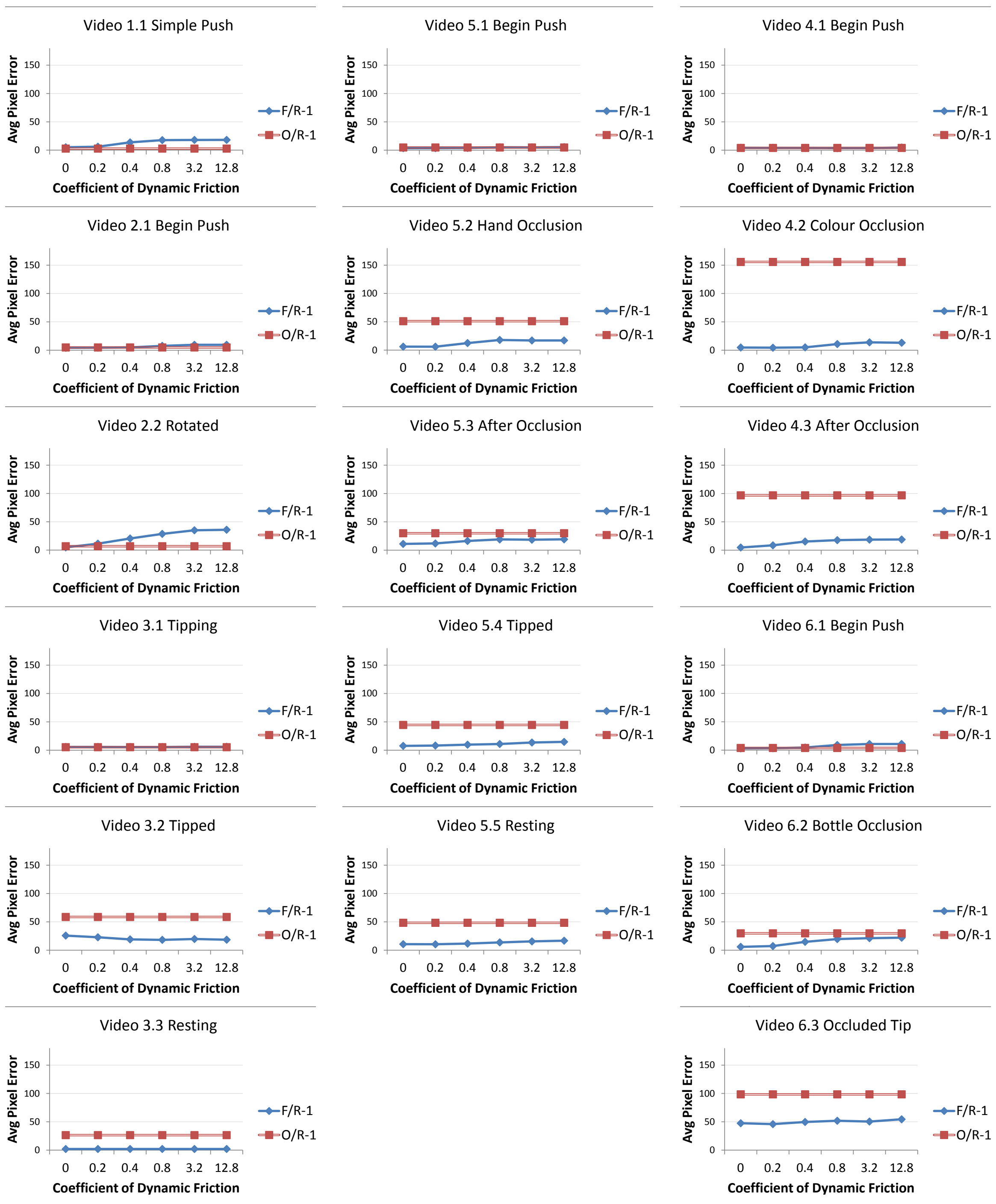


Figure 7.37: The effect of altering the *coefficient of dynamic friction* on the physics simulation based tracker with force noise, and particle retention (F/R-1). Results for the non-physics based tracker with retention are also presented for comparison (O/R-1). Numbers are averaged over 40 trials. The default value of this coefficient in other experiments is 0.2.

negative effect at a very high level of friction, but even then, improvement is generally seen over the comparison method. Dynamic friction has a bigger effect generally.

As a result, at least in the scenarios examined here, the method seems to be robust to simulation parameters. This robustness is attributed to the general noise model.

7.9.2 Sensitivity to noise

The charts in Figure 7.38 show the effect of varying noise parameters both on the condition with force noise and particle retention (F/R-1) and on the corresponding physics-less tracker (O/R-1).

It can be seen that the same improvement attributed to the use of a physics simulator in the context of occlusion can be obtained by turning the noise down a lot on the non-physics tracker. However, in order to be successful under fast movement (such as the tipping cases) a lot of noise is required. In contrast, there is a level of noise that is effective for both occlusion and tipping under the force noise condition.

It remains to be seen how well this noise level generalises to cases not examined in these experiments.

Another alternative hypothesis that needs to be explored further is the hypothesis that the benefit of the force noise is located only in the shape of the noise distribution or the way it differentially emphasises rotation and translation of the object[1]. It is possible to investigate the use of the simulator without gravity and without the ground plane to test this hypothesis, or by reconstructing the distribution of translations and rotations in the force noise condition and applying these directly as perturbations.

[1]While the simulation with displacement noise condition adds noise as a random translation and a random rotation (drawn from a Gaussian over the vector parameters of the unit quaternion), the translation and rotation of the object in the force noise condition is a consequence of a force acting at some point on the object surface.

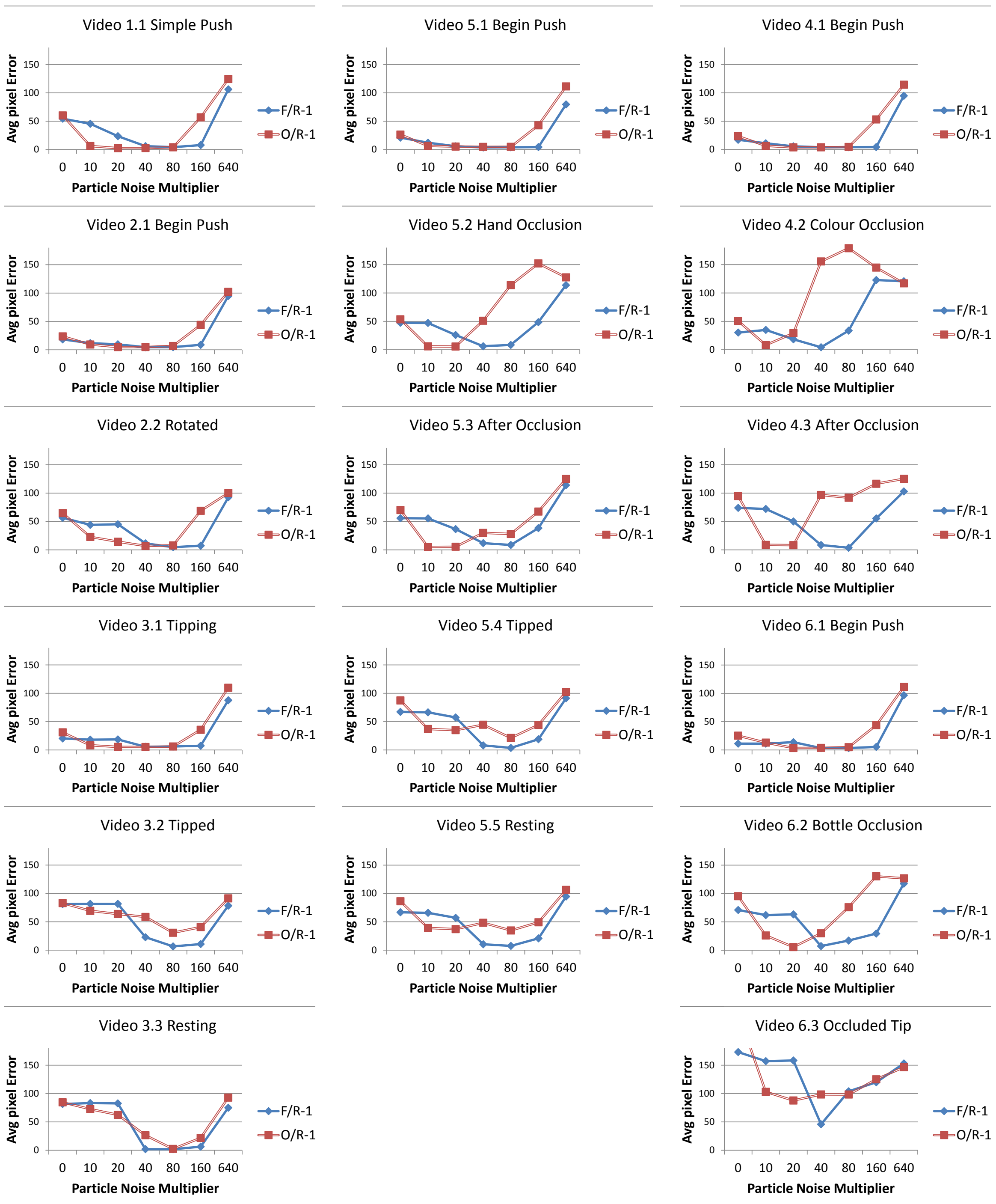


Figure 7.38: The effect of altering the *particle noise multiplier* on the physics simulation based tracker with force noise and particle retention (F/R-1), and on the non-physics based tracker with retention (O/R-1). Numbers are averaged over 40 trials. In the case of the non-physics tracker (O/R-1), the particle noise multiplier multiplies into the the rotation by a factor of 1 degree and into the translation by a factor of 1cm. The default value of the multiplier in other experiments is 40.

7.10 Discussion

"In February, the World Bank estimated that 44 million people have been pushed into poverty by food price spikes since June, 2010."

`It's time to get tough with the food casino,`

`Andrew Hepburn, The Globe and Mail, 4 August 2011`

7.10.1 Results summary

Simulation based dynamics. Coupling a simulator with visual tracking can lead to substantial improvements in the visual tracking, particularly in the presence of hard visual problems like occlusion and fast movement.

Incorporation of noise into the simulator. How the probabilistic dynamics model incorporates the simulation is important, with a force noise based model yielding the most improvement. Indeed, track is maintained with occlusion, distractors, fast movement, and fast movement under occlusion.

Incorporating non-visual sources of object information. Including information about the finger location marginally but significantly increases performance; but there are interactions with other conditions.

Mixtures of dynamics models. Mixtures of different dynamics models do not create the hoped super-additive effect on performance, rather tending to average the performance of component models.

Additive process noise. In Chapter 5 it was seen that adding additive process noise to the trajectory during refinement did little to improve the performance of the physics-based algorithms over simply putting noise in the initial conditions. Analogously, additive process noise in this particle filter (the simulation with dispersion noise condition) did little to improve performance over the non-physics approach. Although this is suggestive of a possible incompatibility of additive process noise with sophisticated physics-based dynamics, further work is necessary to test this. In particular, in Chapter 5, process noise was not incorporated into the RANSAC procedure; its lack of effect during refinement is not so important considering that refinement is less important than RANSAC.

7.10.2 Further work

Into the real world. The tracker is tested here in near-real-time (operating at about 10 frames a second) but it is expected that it would be capable of application in real-time with small tweaks. The simulation with force noise method has been shown to do significantly better on the videos tested on here but in order to move this line of investigation into the real-world, a wider sample of test videos is needed[1]. The manufactured videos above are a start, but what is needed are videos of typical human interaction in tabletop game situations, as well as even more unconstrained interaction, such as mobile manipulation, with and without the additional help of visible calibration patterns for unknown scene objects.

Restricting search vs guiding search. The interpretation advanced here is that the main consequence of using force noise is that the resulting sampler is only able to sample from physical behaviours that are physically plausible according to the physical simulator. This is in contrast to the simulation with dispersion noise condition, where the simulator only acts to guide search.

Each approach has its benefits; the restriction approach, while powerful here, would be expected to break down when the world model does not match the world (as it frequently does not in practice). In order to make the method mature enough to be used in applications (and hence receive the attention necessary to fully exploit it), it needs to be robust to model failure.

[1]While the experiments here are presented in the confirmatory fashion, the reader will be aware that the study here presented is more exploratory in nature. The videos tested here portray a subset in the kinds of situation that will be encountered in practice, even in the context of tabletop robotics.

7.10.3 Possible improvements

There are a number of different ways to improve this approach, and they will be discussed in the remainder of this chapter.

Better simulation. There is a computational ceiling at present as to the quality of the simulation, as well as a shortfall in present abilities to infer physical models from data. Some physical systems (such as human intentionality) require completely different technologies to predict.

Moreover, it is not clear that the techniques used here require good simulation for the good results obtained[1]. As long as the probabilistic model creates a tighter envelope encompassing plausible simulations, the method should be successful[2].

Better modelling of noise. Some kinds of object movement (for instance, when carried), require a different dynamics noise profile. Such force noise profiles may be learnt[3].

The current approach to sampling forces on the object is restricted to forces incident on the object surface. On one hand, where multiple forces add up to produce one net force, that net force may not be acting directly on the object surface. On the other hand, for each force acting at one point in the object frame, a different force acting at a different point on the object frame is typically equivalent. The question that needs to be determined is if the distribution of forces sampled from in the system described here is similar to the distribution of forces that would be applied to the object if, say, the object were picked up and moved by a human intervener[4].

Therefore, further work involves analysing the probability distribution[5] of forces on

[1]The PhysX engine, being a physics engine with mostly graphics and games applications, has a model of friction different to, and probably simpler than, the standard Coloumb model, does not simulate precession in unbalanced masses, and more. Moreover, the parameters used in the simulations in the experiments here (coefficients of restitution, for instance), were only estimated, not measured.

[2]Since games engines are tuned to the human eye, which is tuned to functionally relevant dynamics, it may be that what is plausible according to such simplified physics engines is sufficient for this task.

[3]Indeed, the structure of the noise may be more important than the model itself.

[4]The experiments in Section 7.8 indeed suggest that the force distribution is not up to the task of modelling the kind of stabilising forces that are required to lift the object slowly and put it down again.

[5]A simpler possible extension that is also applicable to the non-physics tracker involves modelling noise

an object or objects during normal human interaction with the object and testing the tracker with such a probability distribution[1].

Learning. Beyond learning the noise profile, the probabilistic dynamics model itself may be learnt (Khansari-Zadeh & Billard, 2010; Kim, Gribovskaya, & Billard, 2010; Kober & Peters, 2009; Kopicki, Stolkin, Zurek, Mörwald, & Wyatt, 2010; Nguyen-Tuong & Peters, 2010). However, such models, at least at present, are usually not as general as hand-created simulators, and just as fragile when generalised, though they are excellent for refining dynamics models within well-defined situations (Álvarez, Peters, et al., 2011; Nguyen-Tuong & Peters, 2010). The problem of using them robustly remains. Indeed, use of a model is used is an important motivator in the problem of learning the model.

In the long-term, if this approach is to be used more broadly, ways of acquiring models and learning or inferring the dynamics of unknown objects is essential.

Better handling of multiple hypotheses. In order to better accommodate different possible object motions (for instance, model conforming, and non-conforming), rather than collapsing the set of current hypotheses around a mode, maintaining distinct hypotheses may allow the tracker to entertain different dynamics models in a more disciplined way[2].

Trajectory-level estimation. The main problem with multiple hypotheses is the tendency to more computational complexity. In order to deal with this problem, rather than maintaining different hypotheses from time-step to time-step, hypotheses can be generated by sampling from plausible trajectories rather than states[3].

as a uniform rather than Gaussian distribution, which can help to deal with fast movements (Pupilli & Calway, 2005). Such an improvement can be complementary or supplementary to the physics based dynamics approach.

[1]Although the current tracker could be used to train the force distribution, for better results a more reliable tracker might be used (such as wide baseline stereo). Similarly, in order to train a non-parametric learner that was used to predict object movement in the limited context of a push, Mörwald, Kopicki, et al. (2011) used the same object tracker with an earlier, untrained, dynamics model. However, in the case proposed here, what would be learnt is a probabilistic model over forces rather than pose changes. On the other hand, representing actions (and by analogy forces) as distributions over (future) state itself does allow some nice parallels with conventional inference procedures (Todorov, 2009).

[2]The most related approach to multiple hypothesis maintenance is that of Vermaak et al. (2003) where particles are clustered into a mixture model and re-sampled in each mixture independently.

[3]The method tested in this chapter uses the dynamics distribution for propagation and sampling

So, a better way of combining multiple dynamics models is sought, that exploits both strict and permissive dynamics models. Indeed, a set of ideas have already been given here for doing this; more are possible.

Switching dynamics. The use of a slow-switching dynamics is proposed[1], such that each particle is augmented with an additional variable that captures its current dynamics; the variable probabilistically transitions with a low probability since dynamics don't switch very frequently. This approach would capture the notion that multiple dynamics models can exist, but not in quick succession as can happen with the current mixture model dynamics[2]. The problem of model selection is apparent in this case since less restricted models will garner more observation support in the short-term and quickly will have no particles; however, inclusion of information about the current level of likelihood support compared to a decaying maximum likelihood value should help to reduce the effect of distractors.

This approach is a straightforward extension and the most likely to succeed in making the system described in this thesis robust to model failure, and so applicable to general tracking application, and so should be the next one attempted.

Multiple object tracking. Using the techniques advanced here, multiple object tracking can be done if each object is treated in isolation. If the objects are to be tracked jointly, however, this means maintaining physical constraints between the objects, and so state estimates must be combined for simulation. Additionally, the states of multiple objects must be maintained together, creating an explosion in the complexity of the

and the observation distribution for re-weighting and re-sampling. A trajectory-based approach would need to remove this dependence by allowing trajectories to be generated by whatever means and tested against both dynamics and observation models; the computational complexity introduced by using full trajectories can be somewhat offset by a more bottom-up approach to their generation as is done, for example, in RANSAC. Sampling of plausible trajectories was done with ball motion in Chapter 5 and can provide a starting point for this work. An example of related work is RANSAC-PF that samples state directly from consecutive image pairs rather than from previous state (Lu, Dai, & Hager, 2006).

[1]Similar "hybrid" models have been employed in fault detection, event detection and action recognition (Dearden & Clancy, 2002; Hongeng & Wyatt, 2008; Williams, Quinn, & McIntosh, 2006), though not known to have been applied to the case where more permissive versus more restricted models with the same dimensionality are selected between.

[2]To see why the mixture model dynamics can lead to a particle switching too-fast between different dynamics models, consider that each particle is passed randomly through a potentially different dynamics at each time-step.

filtering task, which needs to be managed. Parallelisation onto graphics hardware, joint-separability (Lanz, 2005), broad-phase interaction tests, and maintenance of sparse inter-particle relationships (Tweed & Calway, 2002) can help to reduce this complexity.

Efficiency improvements. With some assumptions it would be possible to adapt this approach to a Kalman filter or one of its nonlinear extensions such as sigma-point sampling (Julier & Uhlmann, 1997) or mixture models (Alspach & Sorenson, 1972; Sorenson, 1970) or some combination of these with particle filters (Djuric & Kotecha, 2003; Doucet, de Freitas, Murphy, & Russell, 2000; Kandepu et al., 2008; Kotecha & Djurić, 2003; van der Merwe & Wan, 2003). The idea in using such parametric representations would be to increase efficiency to the point where many objects might be tracked. The mixture models would be the minimal required extension to the basic Kalman filter as the nonlinearity in the simulator can be shown to create highly bifurcated distributions. It is an open question as to how successful Gaussian-based dynamics may be given that sophisticated physics-based dynamics models also have a tendency to strongly distort Gaussian distributions, to the extent of creating zero or infinitesimal probability regions.

Finally, an ongoing advantage of this method is that it can be parallelised very easily. The observation model is currently implemented on a GPU, and the API being used in the physics simulation is also parallelisable on PPUs and GPUs. Hardware is likely to become more parallel in the future, so the decomposable structure of the particle filter approach can benefit even further.

Observation model improvement. Apart from the improvements to the dynamics models on which this work is focused, it may be thought that improvements in computer vision techniques may force the methods described here into the boutique part of the market. While this is unlikely to be true in the broad sense of synthesising physical reasoning and vision, there is one obvious set of improvements to the observation model component of this tracker that need to be considered.

At present the tracker does not reason about the presence of occluders or the quality of visual evidence available. If the tracker were able to reason about occluders, it might be able to alter the likelihood model in such a way that distractors had little effect.

On the other hand, for that approach to succeed overall, there needs to be a good source of information about object motion when it is occluded. For instance, in the difficult video in Figure 7.7 above, when a box is tipped behind a box. So an improvement to the observation model is likely to provide an independent improvement to the tracker, but in the real world much of what goes on is unobservable or partially observable, so any improvement is expected to behave even better together with a good simulator based dynamics model.

CHAPTER 8

DISCUSSION

"BP shares bounce back after Thursday's falls."

Daily Mail, 11 June 2010

8.1 Conclusions

Getting artificial cognitive systems to understand human-scale physical systems requires the understanding of many different kinds of problems since there are many different kinds of physical cognition. In this thesis, one kind, the ability to simulate physical systems, is recruited to assist in another, the ability to estimate the behaviour of physical systems from visual information.

This recruitment serves the practical purpose of augmenting the toolbox of approaches to visual tracking and motion estimation. It also serves the more amorphous purpose of looking afresh at simulation in the context of visual estimation and visa versa.

To investigate the requirements and consequences of this recruitment, experiments were carried out using a physics simulator based dynamics to improve motion estimation, in simulation and from colour, as well as model-based object tracking from colour texture edges. With the goal of this work in mind, from these experiments and accompanying analysis, the following high-level observations can be made:

$\longrightarrow$ A deterministic simulator can be used as a part of a solution to a problem involving estimation over time by adding theoretical noise to the inputs and outputs of the simulation. This theoretical noise accounts for failures of the dynamics embodied by the simulator to model real-world behaviour, and also to account for problems in the assumptions underlying the observation model as well as partial observability.

$\longrightarrow$ The resulting partially observable stochastic process model serves as an adequate formalism for conducting estimation, even using a deterministic simulator. For future work, using that stochastic process model to define the simulator itself will provide more opportunities for generality.

$\longrightarrow$ The use of a physical simulator based dynamics model in motion and recursive estimation is shown in this thesis to be able to compensate for missing or misleading visual observations - including fast movement, blur, glare, occlusion, distractors, and combinations of these.

$\longrightarrow$ For motion estimation, trajectories to be estimated can be parametrised by boundary conditions, a subset of states, or according to all states in the trajectory. The choice of parametrisation depends on the theoretical process noise, which can be somewhat alleviated by theoretical observation noise.

$\longrightarrow$ Practically, traditional non-linear optimisation of a MAP-derived least-squares cost function based on dynamics and observation costs is a straightforward way of merging data sources but is not by itself sufficient within this formalism, and needs to be augmented by sampling methods involving heuristic and robust estimation.

$\longrightarrow$ Reversibility of simulation comes in useful in initialising trajectories from a few observations. Typical simulations are not fully reversible however, collapsing multiple pre-states into single post states. The only general solution to this problem is stochastic simulation.

$\longrightarrow$ For more complex scenarios, for instance involving multiple interacting objects, an approach similar to MLESAC and MAPSAC is proposed based on sampling trajectories as well as simulator parameters, applying them to a generic cost function composed of robust observation terms as well as dynamics terms, and using a building-block approach to building trajectories from sub-trajectories.

$\longrightarrow$ In object tracking, a simulator with added noise fits well into a conventional particle filter model since the common approach is to sample forward in time from a dynamics model and re-weight from observations. Physics simulators are well-optimised so do not add too much overhead to the particle filter.

$\longrightarrow$ Sampling dispersion noise added to the output of the physics simulator does not improve performance, but sampling an input of the simulator in the form of noisy forces applied to the object provides a significant improvement.

$\longrightarrow$ Force noise is thought to do well due to how it constrains the shape of the proposal density and due to its maintenance of object behaviour largely within realistic bounds, particularly with respect to its interaction with the ground plane.

$\longrightarrow$ The addition of information about the robot finger movement in the robot push scenario tends to help only in the force noise condition, and then only marginally. This is thought to be because the force noise condition by itself enforces correctness of the inferred state. The simulation with dispersion noise condition usually cannot make use of the finger location information because it does not enforce a theoretically correct dynamics.

$\longrightarrow$ The most pressing concern in improving this object tracking method is to make it more robust to bad model failure; the current mixture model approach failed to show any merit, so the proposed first step is to introduce a switching model governing the choice of dynamics model.

This thesis is distinguished from earlier and contemporaneous work based on the principle of physics-in-vision in that it seeks improvement from the addition of a passive simulator without a trained controller, it provides robust solutions, it models noise in the inputs to the non-linear simulation, it concentrates on rigid body motion, and of course it focuses on new and untried scenarios and feature combinations. This thesis also bases its approach on a widely applicable stochastic model.

The rest of this chapter attempts to re-place the issues addressed by this thesis in a wider context.

8.2 Model correctness

This thesis fits into the category of model-based vision. A model is employed of the object to be tracked and basic geometric models of image formation. Furthermore, an explicit model exists of the dynamics of the physical system under observation[1]. Both estimation and simulation imply the existence of a dynamical model[2]. In inverse dynamics too (as used in control, and so forth), goals, states and actions at least are modelled. The first important point to make is that although models underlying simulation, estimation and planning relate to the same set of physical systems, they are constituted in very different ways because of their very different cognitive roles.

In a purely reactive system it might be argued that there is no model; though models may be developed to explain its behaviour - if such a system becomes stateful then the state might be argued to naturally model world and body state. For evolved systems there are arguments that models should emerge naturally (Clark & Grush, 1999; Grush, 2004). Not all approaches to physical cognition need strong models (for example, reactive control), but as a way of combining and coordinating different kinds of physical abilities, the use of models makes engineering sense.

One of the overriding concerns of the work described in this thesis has been model

[1]It is worth defining what is meant by model for clarity. A model is something that stands in for something else and so allows that original thing to be reasoned about by reasoning with the stand-in.

[2]In the case of estimation, at the very least, the values to be estimated theoretically reflect some real-world variable; the estimation procedure can depend on a model of how that variable interacts with other variables ultimately to produce observations. In simulation, the knowledge embodied in the simulator models how physical systems change over time, and a particular simulation can be made to model a real or theoretical physical scene.

fit. As mentioned in the literature review Chapter 3, Jakobi (1997) proposes a "radical envelope of noise" hypothesis as a way of making controllers evolved in simulation more applicable when applied in the real-world; otherwise there is a "reality gap"[1]. The idea is that the evolved controller should be evolved to depend as much as possible, and as robustly as possible, only on a base set of relevant behaviours modelled by the simulation. Behaviours of the simulator outside the base set that do not match the real-world are dealt with by adding noise both to non-base set and base set behaviours, to ensure robustness.

A broader principle is that introducing a theoretical noise model to a simulation (even if not sampling noise directly into it) can overcome the simulator's faults, a principle that has been born out in this thesis[2]. As the shape of the noise distribution gets closer to the actual motion of objects (due in this case to adding force noise to the inputs of the dynamics distribution), the efficiency of sampling or search improves.

The reality-gap problem of a pregiven simulator might be solved by machine learning (Kopicki, Stolkin, et al., 2010) though rigid body physics simulation is still usually more generally applicable than state of the art learning techniques. Moreover, learning can be combined with a pre-existing dynamics model (Álvarez, Peters, et al., 2011; Nguyen-Tuong & Peters, 2010). Machine learning has proved to be useful in well-defined problems and can be used to improve model fit. However, despite the best learning applied, there is always going to be unmodelled behaviour.

The reality gap problem has its analog in the area of learning in the problems of over-fitting, transfer and generalisation. As such, complementing the learning approach, work on robustifying dynamics models in appropriate ways is necessary so that the extent to which they are applied matches their applicability[3]. Productive further research involves

[1]The reality gap problem remains the main problem in the field of co-evolution of controllers and simulators, with many alternative solutions proposed (Bongard, Zykov, & Lipson, 2006; Zagal, Delpiano, et al., 2009; Zagal & Ruiz-del-Solar, 2007; Ziemke et al., 2005).

[2]In Chapter 5, the simulator was modelled with noise on boundary conditions, or on the location and velocity of the ball along a trajectory. The noisy model of observations compensated for model failure but ultimately the process noise is required. In the case of Chapters 6 and 7, noise was added to the simulator in order to both deal with a potential reality gap of the simulation, as well as the unknown shape of the observation likelihood induced by the chosen observation cost.

[3]In the visual estimation problem the slack of model fit is taken up by visual information and in control

finding efficient ways of dealing with a noisy or incorrect simulation, and this search is expected also to shed some light on the exact shape of this reality gap, and consequently on the nature of the problems that must be solved by autonomous systems acting in the physical world.

8.3 Human cognition

This thesis focuses on the practical problem of improving computer vision and the integration of computer vision with physical simulation. There is no model of human cognition in it. However, since the problem being solved here is related to problems solved by people, a short discussion of the relationship is warranted.

The literature review discussed work on humans that suggests that humans do mental simulation when solving some problems, particularly mechanical ones (Hegarty, 2004)[1]. One set of conclusions in that literature is that reasoning about physical systems should be analog in nature since the amount of time spent in inferring the outcome of physical systems often is proportional to the amount of time that the system itself would take to run. Indeed, while closed-forms exist for some kinds of simulation, Euler-like methods are typically used in practical computational simulations and the amount of processing required by these general methods is proportional to the amount of time that needs to be simulated. This could mean that, rather than being analog, what is going on in such cases is that a general purpose incremental kind of physical simulation is being run internally. Just as closed forms can be found or long-step simulations learned, so can humans train themselves to use tricks[2] to do these kinds of inferences in constant time. On the other hand, some studies do show that the reasoning process in more complicated simulations often involves a propagation of effects spatially around the system - for example, running

it is taken up by feedback. Approaches to robustness need to be able to rely on model information when it is available and feedback when it is available, and to combine them intelligently.

[1]See Chapter 4 for a summary of the literature on mental simulation, particularly Section 4.1.6.

[2]Such tricks of cognition used to short-cut physical simulation would typically be linguistic, but may also involve geometrical inference.

one gear first and then examining the consequences on another (Schwartz & J. Black, 1996); in practical simulations this approach requires some assumptions; there should not be many bidirectional constraints in the system for instance. As such, this finding has an implication for how simulations for physical cognition might productively be written, optimised for reasoning about small numbers of constraints.

In admitting these links, it must be assumed that the apparent similarity is not merely a coincidence, and an important question to ask is whether the similarity is due to the nature of the problem (simulation/mental simulation) or to a similarity in approaches to solving the problem that might be solved in other ways, or indeed due to some other factor. The former option is compelling because it allows one to reason that the link in behaviour between the human case and digital simulation is related to the nature of the problem of simulation itself rather than any implementation of a solution.

Finally a disclaimer: as mentioned numerous times in this thesis, there are many kinds of physical cognition, of which simulation is only one type. Humans have and do many kinds of physical cognition, and there is little evidence that capabilities for these different kinds of physical cognition can be separated in humans. Indeed some proposals require multiple kinds of physical cognition (such as forward and inverse in the work of Wolpert and Kawato (1998)) to coexist. Therefore, a perspective considering simulation, indeed, simulation and visual estimation, may help reveal some principles of human cognition, but at the same time may be be misleading when considered in isolation. In particular, starting with the assumption of a homomorphic relationship between simulator and world may be a simplification and the structure of a simulator would best be considered in the context of the particular problems it is being used for.

Finally, it needs to be mentioned that the use of physical cognition is ubiquitous in human cognition, even outside of the physical realm. Casual investigation of human language suggests that even the most abstract of concepts are couched linguistically in physical terms (e.g. the preceding sentence). This use of physical concepts to describe abstract ones has been observed time and again and many examples have been collected -

mind as space, progress as locomotion, numbers as motion, and so forth (Barnden, 2001; Lakoff & Johnson, 1984; Núñez, 2005). It is possible, by learning about physical cognition, to shed light on how that physical cognition might be re-used in abstract cognition[1].

Proof of concepts for this exist for specific simulated scenarios (Narayanan, 1999). If work on physical cognition in robots is to shed light on these issues, however, it will be by taking a system that does real physical reasoning and using it to do reasoning in other domains with minimal adaptation or with a disciplined scheme for adaptation[2].

Regarding the scenarios studied in this thesis as possible source domains for reasoning - in the case of bouncing balls, shares can bounce, in the case of pushes, it is possible to push to far, to topple a dictatorship, in the case of falling objects, share prices can fall, it is possible to for ideas to come into contact, and so forth. In all of these problems, there is an underlying state-based stochastic process and observation model that itself generalises physical events; this is certainly applicable to other domains, but it remains unclear whether a general mechanism might exist for applying it. In the end, it is most probable that the re-use of physical cognition will be possible only when systems for physical cognition are comprehensive enough that such abstract extensions are natural requirements in extending the cognitive system.

8.4 Prospects

A wide range of prospects for further work have been proposed with respect to each of the sets of experimental scenarios in Chapters 5, 6 and 7. This suggests that the research set out in this thesis can be built upon. As mentioned before, the most important thrust in any

[1]Some caveats are in order of course: it is controversial that humans do re-use physical cognition in a profound rather than shallow way: alternative explanations revolve around, for example, the argument that generalised internal languages might be necessary to explain the generative capability of human cognition (Hauser, Chomsky, & Fitch, 2002; Sloman, 2007). Furthermore, even if evolution produced a system that re-uses physical abilities to do abstract reasoning, this says nothing about the requirements of abstract reasoning with respect to all cognitive systems; it is quite possible that such a cognitive architecture is a legacy of phylogeny.

[2]One of the major difficulties in this approach would be, as with other approaches to analogy and metaphor in computation, recognising when the re-use is applicable.

such research is extending the software to degrade gracefully when faced with unmodelled situations. Specific proposals are given in Chapters 5, 6 and 7. The possible use of machine learning techniques to address various issues faced in this work is also high on the agenda, since this work exists in a context where machine learning is being applied successfully to similar problems in control and motion planning and the additional concerns of robustness that come out of the estimation problem should be valuable contributions to that field. New kinds of process noise models, tied closer to the details of simulation (for example, reversible stochastic simulation), are important to provide a greater flexibility in how the estimation routines come by solutions, while creating an envelope of noise more closely encompassing the behaviour of real processes.

Most important is a synthesis with other approaches to physical cognition such as motion planning, control, imitation and grasping. Both simulation and estimation don't by themselves require the generation of action or interaction with the world and while inclusion of such problems would coincide with new kinds of complexity, synergy would be expected; beyond this, the consideration of full systems acting autonomously is crucial in understanding the impact of approaches to subtasks, like simulation and estimation.

APPENDIX A

THE VISUAL APPARATUS

In order to be able to estimate variables in the world, it is useful to have a model of how these variables relate to the observations from which they are estimated[1]. To this end, any exposition of visual estimation usually begins with the visual apparatus. Although this thesis is about the application of physics ideas to computer vision, the physics in question is the physics of the scene under observation, not of the formation of images, so this exposition will not dwell deeply on the physical details of image formation.

At the broadest level, material in the world reflects light, which can be received by an organism or device and used to gauge the properties of these things, particularly spatial properties. The study of traditional "photogrammetry" is the study of reconstructing the spatial configuration of objects from their visual appearance, usually by identifying points from images or by measuring angles with a theodolite or sextant or other such apparatus (Karara, 1989). Computer vision is more broad in remit in that it goes beyond just the spatial configuration of objects and to their automated identification, real-time tracking, and so forth; all of the desiderata of visual perception by autonomous systems (Hartley & Zisserman, 2000; Jain et al., 1995).

The most widely used artificial apparatus for receiving and analysing light is the digital camera with lens, which, like the human eye, focuses light through an aperture onto a

[1]Even more specifically, a model of how observations are formed from world variables provides a view on the problem that aids in its understanding, at least as a starting point for designers of the estimation system.

area containing an array of light-sensitive elements. For the purposes of most day-to-day computer vision, the reflection, transmission and incidence of light can be assumed to be instantaneous [1].

In order to capture this arrangement as simply as possible, the lens itself is not directly modelled; rather, a pin-hole camera model is used and corrections introduced for distortions of the lens[2]. A pin-hole camera is a camera that captures light through a small lenseless aperture - the smaller the hole the more the camera matches the pin-hole camera model and the more accurate estimates of the direction of incident light; but of course the smaller the hole the less light is admitted through the aperture (in the limit, none, so the model is a true idealisation)[3].

The pin-hole camera model is also known as a projective camera model. By defining the focal length, this model can describe how light reflected off objects is projected as a ray from a point on the object to its corresponding point of incidence on the sensory array. After this projection, using the aspect ratio of the light-receiving elements in the sensory array, these points can be converted into pixel coordinates. At this point, the model can also account for skew in the sensory array. Finally, the correction for lens distortion can be applied.

Mathematically, the model is a mapping from points in object space, in homogeneous coordinates to pixel coordinates. The mapping can be broken into four parts. Typically, the first part is a transformation from object coordinate space into a world coordinate space (representing the rigid pose of the object with respect to some pre-defined world frame of reference); this allows points on all objects to be mapped into the same frame

[1]Though the digital processing of light often leads to delays that must be modelled and compensated for.

[2]Rather than modelling the lens, however, the experience of practitioners has shown that modelling the camera as a pin-hole camera and correcting for (unknown but usually able to be estimated) distortions introduced by a lens is simpler analytically (particularly since such distortions are often small and can, at many stages of analysis, be ignored). Lenses without these distortions are called metric lenses but are rarely used in the field of computer vision which typically makes use of more readily available components.

[3]In the camera with lens, the lens allows for the direction of light to be remembered after it passes through the aperture just as is the case with a pin-hole camera (since, as long as it falls on a light-receptive surface at the focal length, the location on the surface at which it falls will closely correspond to the direction of its origin). Moreover, the lens-filled aperture admits more light than a pin-hole.

of reference. This is followed by a transform into a coordinate system oriented with the camera (representing the rigid pose of the camera). After this transformation (which can be written as a composition of rotations and translations), the final transformations, as discussed in the previous paragraph, involves projection of the point in the camera's own coordinate system into pixels and the correction for distortion.

This process is summarised as:

- Transformation from object coordinates to world coordinates. $\boldsymbol{Y}^{Obj} \xrightarrow{\mathcal{T}_{O/W}} \boldsymbol{Y}^{Wld}$
- Transformation from world coordinates to camera coordinates. $\boldsymbol{Y}^{Wld} \xrightarrow{\mathcal{T}_{W/C}} \boldsymbol{Y}^{Cam}$
- Projection of camera coordinates to pixel coordinates. $\boldsymbol{Y}^{Cam} \xrightarrow{\mathcal{T}_{C/i}} \boldsymbol{Y}^{img}$
- Correction of lens distortion $\boldsymbol{Y}^{img} \xrightarrow{\mathcal{T}_{i/c}} \boldsymbol{Y}^{cor}$

The variables of interest are as follows. $\boldsymbol{Y}^{Obj}$ is a vector containing the coordinates of a point in the object's frame of reference. $\boldsymbol{Y}^{Wld}$ is a vector containing the coordinates of the same point in the "world" frame of reference, that is with respect to some fixed frame in the world. $\boldsymbol{Y}^{Cam}$ is a vector containing the coordinates of the same point in the frame of reference of the camera. $\boldsymbol{Y}^{img}$ describes the same point in the image plane in pixel units. $\boldsymbol{Y}^{cor}$ is the same point corrected for lens distortion.

$\boldsymbol{Y}^{Obj}$, $\boldsymbol{Y}^{Wld}$ and $\boldsymbol{Y}^{Cam}$ are generally represented as points in projective space. Projective space can be thought of as the same as the more familiar Euclidean space except that a point an infinite distance from the origin does exist in a projective space, and is the same point no matter what the direction infinity is approached in. Projective space is more amenable to certain kinds of analysis than the more well-known Euclidean space, since more operations on it are well defined[1]. If a point in projective space is to be numerically described, homogeneous coordinates are required, and contain one more coordinate than would normally be necessary to describe a point in Euclidean space, though this extra

[1]In particular, every point in projective space corresponds to a point in Euclidean space, with the exception of the point at infinity, which is defined in projective space but not Euclidean.

coordinate does not introduce any extra degrees of freedom[1]. The vector could be written $\boldsymbol{Y}^{Obj} = \begin{bmatrix} Y_1^{Obj} \\ Y_2^{Obj} \\ Y_3^{Obj} \\ Y_4^{Obj} \end{bmatrix}$[2].

The mappings $\mathcal{T}_{O/W}$, $\mathcal{T}_{W/C}$, $\mathcal{T}_{C/i}$ and $\mathcal{T}_{i/c}$ are the main mappings of interest for converting between these frames of reference. They will be described now. The mappings $\mathcal{T}_{O/W}$ and $\mathcal{T}_{W/C}$ are both rigid body transforms - that is, compositions of rotations and translations. They can be represented by matrices whose properties under composition are the same as the composition of rigid body transforms:

$$\mathcal{T}_{O/W} = \begin{bmatrix} R_{O/W} & \boldsymbol{T}_{O/W} \\ 0 & 1 \end{bmatrix} \tag{A.1}$$

$$\mathcal{T}_{W/C} = \begin{bmatrix} R_{W/C} & \boldsymbol{T}_{W/C} \\ 0 & 1 \end{bmatrix} \tag{A.2}$$

Here, from the perspective of the original coordinate frame, $R_{O/W}$ and $\boldsymbol{T}_{O/W}$ rotate then translate a point, having the effect of changing its coordinate frame (alternatively, the frame itself can be considered to be moved through the inverse of this rigid transform) from an object centred frame to a frame centred on some arbitrary "world" frame. $R_{O/W}$ is a 3×3 orthonormal matrix representing a rotation, and $\boldsymbol{T}_{O/W}$ a 3 dimensional vector representing a translation.

Similarly, $R_{W/C}$ and $\boldsymbol{T}_{W/C}$ transform a point into the frame of reference of the camera, defined by the optical axis and imaging plane.

If the camera is known to be fixed, then the world to camera transformation $\mathcal{T}_{W/C}$ is often either ignored or set to the identity, since the camera coordinate frame can serve the same function that the world coordinate frame would otherwise have served. This is the

[1]For more details of these models see Hartley and Zisserman (2000), Karara (1989), Lepetit and Fua (2005), Triggs et al. (2000), Trucco and Verri (1998).

[2]To transform a point in homogeneous coordinates into Euclidean coordinates, as long as the point is not the one at infinity (i.e. the fourth coordinate of the homogeneous coordinates is not zero), a simple calculation only is needed: $\boldsymbol{Y}^{Euc} = \begin{bmatrix} \frac{Y_1}{Y_4} \\ \frac{Y_2}{Y_4} \\ \frac{Y_3}{Y_4} \end{bmatrix}$.

case with the work done in Chapter 5 where the camera is fixed. Similarly, in Chapter 7, the camera does not move so the transformation $\mathcal{T}_{W/C}$ is acquired in advance of tracking through the use of a calibration pattern.

After these rigid body transforms, the following matrix encompasses the projection of the resulting points onto the image. This projection encompasses the change of units to pixels, the projection according to the focal length of the camera, and any effect due to the aspect ratio or skew of the photosensitive elements on the image plane.

$$\mathcal{T}_{C/i} = \begin{bmatrix} a_1 f & s & y_1^C & 0 \\ 0 & a_2 f & y_2^C & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{A.3}$$

Here, f is the focal length, a_1 and a_2 together determine the change of unit and the aspect ratio, s any skew in the imaging elements, and $[y_1^C, y_2^C]^T$ the location of the optical centre with respect to the origin in pixel coordinates. Because depth information is effectively forgotten by the projection, the right hand column of the matrix is all zero. The mapping takes a vector in homogeneous coordinates representing a 3D point (4 numbers) and produces a vector in homegeneous coordinates representing a 2D point (3 numbers).

A simplified model, called the orthogonal camera model, does not take into account the depth of a point when determining its image coordinates; this simplification is sometimes of use, though in general not correct because a point that recedes in depth also moves across the image plane. Such a projection would be written[1]:

[1]In Chapter 7 the projective camera matrix in Equation A.3 is calibrated in advance for the camera used since the parameters of this transformation are usually properties of the lens and camera system used. In the work in Chapter 5, these parameters were mostly estimated in an ad-hoc way by eye; moreover, because the objects tracked were expected to move parallel to the image plane (and hence with a fixed depth), an orthogonal camera model was assumed as with Equation A.4.

$$\mathcal{T}_{C/i} = \begin{bmatrix} a_1 f & s & 0 & y_1^C \\ 0 & a_2 f & 0 & y_2^C \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{A.4}$$

Note that in the orthogonal projection matrix, the third column is zero, because the depth of a point (information recorded in the 3rd component of the homogeneous coordinate vector) is not used when determining the image location of the projected correspondence to that point[1].

Finally, after rigid transformation and projection, the following mapping (the correction for lens distortion) can be applied, with $\boldsymbol{Y}^{img} = \begin{bmatrix} y_1^{img} \\ y_2^{img} \\ 1 \end{bmatrix}$:

$$\mathcal{T}_{i/c}(\boldsymbol{Y}^{img}) = D\left(\sqrt{(y_1^{img} - y_1^C)^2 + (y_2^{img} - y_2^C)^2}\right) \begin{bmatrix} y_1^{img} \\ y_2^{img} \end{bmatrix} \tag{A.5}$$

The function D here takes the radius as an argument and produces a scaling. Typically its inverse is modelled as a polynomial function of the radius; a truncated Taylor series:

$$r = 1 + d_1 \cdot D(r) + d_2 \cdot D(r)^2 + d_3 \cdot D(r)^3 + d_4 \cdot D(r)^4 \tag{A.6}$$

This mapping is expressed as its inverse because it is the inverse mapping that is generally applied - as a correction mapping from light incident on the visual array to where it would have been incident on the visual array, were the lens producing a perfect projective mapping.

The same story applies to all of the above mappings; in model-based vision the inverse mapping is of interest when it comes to the estimation problem; that is the task of finding the corrected pixel location from the acquired image, the depth of the point from the

[1]In Chapter 5, s was assumed zero, a_1 assumed to be 1 and a_2 assumed to be -1, and because the camera model is so simple, y_1^C and y_2^C serve only to shift the origin of the world coordinate frame to the origin of the image coordinate frame.

camera (e.g. the location from the camera frame), the rotation and translation of the camera with respect to the world frame, and the rotation and translation of the object with respect to the world frame[1]. Clearly, the forward model (from information about the configuration and content of the world to the image produced) is in most cases easier to achieve than the inverse. In some sense, the forward model is something like simulation in that it simulates the projection of light onto the visual array, the creation of an image from the world. However, simulation using a model tends to be easier than estimation using the same model, at least because estimation must involve efficient ways of picking between many possible hypotheses[2].

[1]And then there is the problem of finding the structure of objects in the world, not approached in this thesis, though discussed briefly in this background chapter.

[2]Furthermore, because the most easily designed and understood models of the world are ones that directly reflect its mechanics as humans can describe them, such models tend to be simulators.

APPENDIX B

OTHER FEATURES

In Chapter 3 Section 3.2, a discussion of edge and colour features is made. This section discusses features not discussed there; while these features are not directly used by this thesis, it is important to consider how they might be employed in conjunction with physics simulator based knowledge as well as in conjunction with each other.

B.1 Feature points

Given that the discussion of the visual apparatus in Appendix A is based on the intuition of the transformation points, it would be useful to identify known points in an image so that they can be used to disambiguate the pose of a target object (or to characterise a target object). The information in a single pixel is not enough to do this, but it turns out that information in an image in the pixel neighbourhood of a 2D point can often be sufficient to identify which of a family of candidate object points is projecting to the region of the image point. Such points are called *feature points*, or sometimes *interest points* (Harris & Stephens, 1988; Lepetit, Pilet, & Fua, 2004; Lowe, 2004; Özuysal, Fua, & Vincent Lepetit, 2007; S. M. Smith & Brady, 1995)[1]. If sufficient uncorrupted information exists in its neighbourhood, a feature point can often be located to sub-pixel accuracy.

Once identified, feature points can be used for all the major computer vision tasks

[1]Occasionally feature points are called corner points because they can be thought of as an elaboration on edges (which are discussed in Section 3.2.1).

such as estimating the identity and pose of an object, calculating the movement of a camera, and calibrating the camera. If the identification of feature points is correct and the identified image location of them close to correct, many methods exist for calculating the desired parameters (Fischler & Bolles, 1981; Gordon & Lowe, 2004; Hartley, 1997; Hartley & Zisserman, 2000; Pupilli & Calway, 2005; Vacchetti et al., 2004b). Feature points are useful in that they can be very distinctive, and where they exist they can be used to determine the pose of an object to a very fine degree. The main areas of research for feature points is in making their detection and identification invariant to illumination, rotation, scale, and so forth. This research can be divided up into the task of first detecting good feature points and then the task of matching them well (which may involve some pre-processing, machine learning, etc).

When it comes to tracking moving objects using feature points, they are useful because they do not require local image search and so can deal with large object displacements. However, the problem of blur means that during motion the interest points may not be recognisable (without accounting for the blur).

B.2 Templates & appearance

As noted above in the section on feature points, the appearance of an image in the neighbourhood of a feature point can help locate and identify a point on an object in the image. This idea of image appearance can be generalised so that rather than being used to locate and identify a point, the appearance of an object can be used to locate and identify an object, or part of an object, or multiple objects (Irani & Anandan, 1999; Jurie & Dhome, 2002; Masson et al., 2004).

Template matching is simply the matching of an image area between two images. Obviously the image appearance of an object and of object parts changes markedly in the presence of rotation, illumination, scale, shear, background, partial occlusion, and so forth, which is typically overcome with corrections such as transformations on the template, compressing the template into a classifier, re-updating the template dynamically over time, breaking the template up into multiple templates, etc. When tracking objects in full 3D, the visible object surfaces corresponding to templates tracked are often called patches.

Once a template has been matched, it can be refined at the pixel level to estimate pose, for instance, or the template can be used as a feature point might be for producing point matches between images (Masson et al., 2004).

With feature points the process is typically one of first extracting and matching the feature points and then using these matches to reconstruct object shape, pose or motion. With templates and like methods, the problem is instead often posed as one of simultaneously optimising template match while reconstructing the object shape, pose or motion. The distinction between the 2-step process or the simultaneous treatment is sometimes called the distinction between indirect and direct methods. While there may be a continuum of methods between these two extremes, in general, methods fall close to one or the other category (Irani & Anandan, 1999; Torr & Zisserman, 1999).

Appearance-based vision is similar to template-based approaches in that image regions are analysed based on their appearance. However, appearance-based vision is about, first, whole object matching rather than image patch matching, and, second, finding good ways of compiling exemplars of objects into visual routines that can detect, segment, and find the pose of objects. Appearance-based vision is particularly often applied to the task of recognising objects, segmenting them from an image, and tracking in the two dimensions of the image (Jepson, Fleet, & El-Maraghi, 2001; Leibe, Leonardis, & Schiele, 2008) but have been used in determining pose also (Mittrapiyanuruk, DeSouza, & Kak, 2004)[1].

[1]Appearance-based vision often works hand-in-hand with machine learning techniques; it is a case of learning the appearance of an object by clustering, building a classifier, etc., and applying that learnt data structure.

B.3 Optical flow

Optical flow is the movement of small elements in the visual field across the retina/image. Since optical flow is directly related to motion, it is clearly a very important technique in motion estimation problems.

There are numerous ways of attempting to calculate optical flow but the basic assumption underlying all methods is that as time progresses elements move across the image in a straight line (an assumption which generally holds for small movements). If this assumption holds, the optical flow is constrained linearly by the image gradients as well as image rate of change at each point.

So optical flow involves approximating the within-image image intensity gradient and calculating the rate of change of image pixels over time. As such, optical flow is a feature that applies over multiple frames. After the calculation of flow, an optimisation problem is generally solved (typically with additional constraints to make the problem well-formed) to find the movement of objects in the scene that best matches the optical flow (Aggarwal & Nandhakumar, 1988).

Unlike many of the features introduced here (but similar to other direct methods discussed above), optical flow does not fit easily into the two-part feature extraction then solution procedure. Optimisation of the calculation of the optical flow is typically done at the same time as calculation of object motion, since the kind of motion that is observed or expected has a strong relationship on the kinds of calculation needed to extract the flow, and the image gradient and pixel rate of change don't by themselves determine the flow (Aggarwal & Nandhakumar, 1988). This can have the additional benefit that optical flow can work to aid in the directed extraction of other features such as edges, since it is naturally early in the processing pipeline but can be used to predict higher level features such as the location of object contours (Decarlo & Metaxas, 2000).

Optical flow has the advantage over feature-point approaches to motion estimation in that the latter suffer badly from blur - though optical flow is not exempt from this problem either. Although one of the important features used in the estimation of motion, this thesis does not make use of optical flow, so it will not be considered further here.

B.4 Region and shapes

Another set of features are derived from the basic idea that objects produce distinctive regions in an image that can be identified and segmented from their background[1]. In very simple backlit images, silhouettes are available, where simple thresholding may be sufficient (Bhat et al., 2002), though real images are usually much more complex. In such cases, the local image intensity or texture can be used as cues, and then with that information, regions found where the intensities do not change profoundly, or where auto-convolution or correlation produce a good match score (the "affinity").

Techniques for building such regions generally start with such affinity values between pixels and build regions from these affinities. In single images, region growing techniques are the basic technique, similar to the algorithms used in image manipulation programs that require seed positions. These have the advantage of being directly geared to connected regions (Wren, Azarbayejani, et al., 1997). Global methods include methods that look at the affinity matrix built from the affinities, and graph-cut techniques that consider the affinities as a graph (Felzenszwalb & Huttenlocher, 2004; Weiss, 1999), and these approaches can be made to prefer connected regions too by appropriate design of the affinity matrix/graph.

Finally, segmentation using these region techniques is a good way to initialise optical flow techniques, since they generally specify the regions within which single-object constraints can be applied to the optical flow calculation. Segmentation using frame-differencing (analysing the differences between frames in terms of intensity changes or local changes, typically with a background model) is powerful when motion occurs, and provides good cues as to the motion of the object (Baumberg & Hogg, 1994b; Jain, 1981).

[1]Note that the problem of *segmentation* of an object in an image (determination of which pixels belong to the object) can make use of almost all the features discussed in this section, not just regions and shape.

Segmenting of objects can also be done using colour models (Wren, Azarbayejani, et al., 1997), an approach which is described in more detail in the Section 3.2.2. There is a large overlap between region-based and colour based models. Features that use information about both colour distribution as well as region (and possibly motion) are called "blob" based methods (Bradski, 1998; Comaniciu et al., 2003; Magee, 2001; Rasmussen & Hager, 2001; Wren, Azarbayejani, et al., 1997).

B.5 Stereo

Stereo vision involves exploiting a two-camera setup with the cameras generally at fixed poses with respect to each other. It has the advantage that the depth of cues is generally easier to discriminate. Stereo is generally applicable alongside all of the features mentioned here; for the most part it involves integrating features coming from both cameras. Finding interest points in the two views is the easiest method and allows their depth to be calculated; optical flow can be matched between two views, again revealing depth information; lines can be matched, again revealing depth information; region matching can give a depth information to varying quality and templates can, like optical flow, provide dense matching information (meaning that every pixel in an image has the potential to be matched to every pixel in the other).

Stereo somewhat simplifies the structure from motion problem in that depth is already calculated after the stereo processing stage (Aggarwal & Nandhakumar, 1988); its ability to discriminate depth makes it highly useful on problems where scene structure is not known in advance. However, when scene objects are known, monocular depth discrimination is still very bad, so stereo information can help there as well.

Stereo algorithms are often subsumed under general multiple view algorithms and a very general approach to multiple view geometry is given in (Hartley & Zisserman, 2000). Such approaches as used in structure from motion, for example, are also applicable to stereo, but stereo has several special characteristics such as the ability to specify the known stereo rig configuration very finely, and the ability to use good stereo algorithms to pre-process visual information into true 3D (depth) images, to simplify subsequent calculations. RGB-D cameras that use lighting structured in time or space can also provide depth information in addition to luminance; they have the disadvantage of needing to send a signal into the scene to probe it (even if the signal is generally invisible to humans), but have the advantage of working even when stereo cues are poor or difficult to extract, since they provide their own cues.

APPENDIX C

STRUCTURE AND MOTION

Although the problem addressed in this thesis is motion estimation of known objects, many of the techniques of photogrammetry and computer vision were developed with estimation of scene characteristics as well as camera or object movement (Hu & Ahuja, 1992a; Hu & Ahuja, 1992b; Iu & Wohn, 1990; Weng et al., 1987; Young, Chellappa, & Wu, 1991). In photogrammetry, estimation of the location of points is often tackled at the same time as camera transformations between images. More generally, structure and motion can involve obtaining object points or surfaces while at the same time obtaining camera and object motion.

Since a single view often does not contain enough information to determine both pose and scene characteristics, multiple views of a scene or object are often relied upon to disambiguate this kind of information; hence structure and motion. So, if five points are seen by two calibrated projective cameras then the resulting co-linearity equations are generally enough to estimate their locations with respect to both cameras (and hence also the camera transformation between the two images), again assuming no limiting conditions (Aggarwal & Nandhakumar, 1988; Hartley & Zisserman, 2000; Karara, 1989). If eight points are seen by two un-calibrated projective camera views then the resulting system of equations can be solved for most calibration parameters, pose and object point locations (Hartley, 1997; Hartley & Zisserman, 2000). These calculations proceed by finding the Essential Matrix or Fundamental Matrix, the relation defined such that $\boldsymbol{z}_t M \boldsymbol{z}_s = 0$, where $\boldsymbol{z}_s$ and $\boldsymbol{z}_t$ are image locations at different time points and M the Fundamental Matrix OR $\boldsymbol{z}_s$ and $\boldsymbol{z}_t$ image locations corrected according to the known camera calibration information and M the Essential Matrix. The literature on this subject is large, and as well as point-based methods, optical flow based methods are also a popular approach to solving the structure from motion problem (Aggarwal & Nandhakumar, 1988).

If the size of no single object in the scene is known, then there will always remain an unknown scale parameter, since large objects far away should generally produce the same image as small objects up close. This problem can be overcome with more information - from a computer vision perspective, using distance haze, for example, and from a photogrammetric perspective, any known object in the scene can be used to disambiguate all of the others as well as known relative camera locations.

A related problem is the problem of Simultaneous Localisation and Mapping, which is not always only a vision problem (using any feature such as laser range-finding), but involves creating and maintaining a map (equivalent to the structure described here) and updating a pose recursively (Davison, 2003; Davison et al., 2007; Pupilli & Calway, 2005).

If the task is to obtain only motion (defined as a sequence of camera or object poses) given that the structure of the scene is known, then multiple frames provide ways of evaluating hypothesis poses, and more ways of generating hypotheses. Motion estimation, particularly long sequence motion estimation, on the other hand, can make use of the expected kind of movement of objects to constrain or bias estimates of object motion, beyond the rigid body constraints. This is the approach taken in this thesis, and existing such approaches are discussed in Section 3.5.

APPENDIX D

LEVENBERG-MARQUADT FOR TRAJECTORIES

In Chapter 5, a non-linear optimisation algorithm is used to find solutions to the trajectory optimisation problem, with the cost function a least-squares error over observations, or over observations and dynamics costs. The algorithm used in that optimisation is Levenberg-Marquadt. The algorithm is given here in the notation used in that chapter. The problem here is posed as optimisation with respect to the initial parameters x_{t_0}, but is applicable with small changes to optimisation across a parameter space consisting of the whole trajectory and a dynamics too.

The Levenberg-Marquadt algorithm attempts to minimise the sum of squared errors in the observations by consecutively solving sets of linear equations based on a simplified approximation of the shape of the local cost surface. The equations are solved for a step through the parameter space. Because the algorithm uses the local shape of the cost surface it tends to converge to local optima.

So, at each iteration of the algorithm a parameter update $\delta \boldsymbol{x}_{t_0}^{j}$ is sought that, when added to the the previous parameter estimate $\boldsymbol{x}_{t_0}^{j-1}$, such that as $j \to \infty$ a local optimum is reached. The Levenberg-Marquadt algorithm, like the Gauss-Newton algorithm, uses the structure of the cost function as a sum of squares to approximate its curvature without needing to take the second derivative.

The usual assumption is made, that the local cost function can be approximated by a polynomial:

$$\begin{aligned} &Cost(\boldsymbol{x}_{t_0}^{j-1} + \delta\boldsymbol{x}_{t_0}^{j}) \\ &\quad = Cost(\boldsymbol{x}_{t_0}^{j-1}) + \left(\delta\boldsymbol{x}_{t_0}^{j-1}\right)^T \frac{d}{d\boldsymbol{x}_{t_0}} Cost(\boldsymbol{x}_{t_0}^{j-1}) + \frac{1}{2}\left(\delta\boldsymbol{x}_{t_0}^{j-1}\right)^T \frac{d^2}{d\boldsymbol{x}_{t_0}^2} Cost(\boldsymbol{x}_{t_0}^{j-1}) \left(\delta\boldsymbol{x}_{t_0}^{j-1}\right) \end{aligned} \tag{D.1}$$

An optimum will be found when the cost function gradient is zero, and so if the local approximation is assumed correct, the first derivative of the Taylor expansion of the cost in Equation D.1 can be set to zero and the following equation obtained which can be solved to step directly to the solution $\delta\boldsymbol{x}_{t_0}^{j}$ in the local Taylor approximation (this is the Newton step):

$$\left(\delta\boldsymbol{x}_{t_0}^{j}\right)^T \frac{d^2}{d\boldsymbol{x}_{t_0}^2} Cost(\boldsymbol{x}_{t_0}^{j-1}) = -\frac{d}{d\boldsymbol{x}_{t_0}} Cost(\boldsymbol{x}_{t_0}^{j-1}) \tag{D.2}$$

Since the present problem is a least-squares problem, it is possible to break the cost function into a sum of errors. For brevity, the dynamics function for each time point has been written as $\boldsymbol{f}_{dyn}^{i}$ so that $\boldsymbol{f}_{dyn}^{i}(\boldsymbol{x}_{t_i}) = \boldsymbol{f}_{dyn}(\boldsymbol{x}_{t_0}, \boldsymbol{\theta}, \boldsymbol{u}_{t_{0:j}}, \Delta t_{0:j})$. The gradient of the cost function can be calculated in terms of the Jacobian of the observations with respect to the parameter vector[1]:

$$\frac{d}{d\boldsymbol{x}_{t_0}} Cost(\boldsymbol{x}_{t_0}^{j-1}) = \left(\frac{d}{d\boldsymbol{x}_{t_0}} \boldsymbol{f}_{obs}(\boldsymbol{f}_{dyn}^{i}(\boldsymbol{x}_{t_0}^{j-1}))\right)^T S_{obs} \left(\boldsymbol{z}_{t_i} - \boldsymbol{f}_{obs}(\boldsymbol{f}_{dyn}^{i}(\boldsymbol{x}_{t_0}^{j-1}))\right) \tag{D.3}$$

The second derivative of the cost function with respect to the parameter vectors is approximated by ignoring any terms that include the second derivatives of the observations with respect to the parameter vectors. In the following approximation, these terms are packed away into H_{terms}:

$$\frac{d^2}{d\boldsymbol{x}_{t_0}^2} Cost(\boldsymbol{x}_{t_0}^{j-1}) = \left(\frac{d}{d\boldsymbol{x}_{t_0}} \boldsymbol{f}_{obs}(\boldsymbol{f}_{dyn}^{i}(\boldsymbol{x}_{t_0}^{j-1}))\right)^T S_{obs} \left(\frac{d}{d\boldsymbol{x}_{t_0}} \boldsymbol{f}_{obs}(\boldsymbol{f}_{dyn}^{i}(\boldsymbol{x}_{t_0}^{j-1}))\right) + H_{terms} \tag{D.4}$$

[1]When the cost function includes sum of squares dynamics terms, a similar decomposition applies.

The idea here is that if the function $\boldsymbol{f}_{obs}(\boldsymbol{f}^i_{dyn}())$ is linear, this approximation is exact. Therefore, for smooth cost functions there is always a neighbourhood in which the approximation works to a given accuracy. Therefore, there is a neighbourhood in which an algorithm based on this assumption should converge quickly.

This approximation can be applied to the Newton step in Equation D.2 to get the following equation for a step to the solution - this is called a Gauss Newton step, and the corresponding approximation the Gauss-Newton approximation:

$$\begin{aligned}(\delta\boldsymbol{x}^j_{t_0})^T \left(\frac{d}{d\boldsymbol{x}_{t_0}}\boldsymbol{f}_{obs}(\boldsymbol{f}^i_{dyn}(\boldsymbol{x}^{j-1}_{t_0}))\right)^T S_{obs} \left(\frac{d}{d\boldsymbol{x}_{t_0}}\boldsymbol{f}_{obs}(\boldsymbol{f}^i_{dyn}(\boldsymbol{x}^{j-1}_{t_0}))\right) \\ = -\left(\frac{d}{d\boldsymbol{x}_{t_0}}\boldsymbol{f}_{obs}(\boldsymbol{f}^i_{dyn}(\boldsymbol{x}^{j-1}_{t_0}))\right)^T S_{obs}\left(\boldsymbol{z}_{t_i} - \boldsymbol{f}_{obs}(\boldsymbol{f}^i_{dyn}(\boldsymbol{x}^{j-1}_{t_0}))\right)\end{aligned} \tag{D.5}$$

Since the consequence of the linear approximation of the observation function breaking down can be non-convergent oscillations, damping methods are useful. The addition that the Levenberg-Marquadt approach makes to this formulation is by damping the step size by altering the approximation of the second derivative term by adding in a damping factor that inflates the second derivative, thereby increasing the curvature of the cost function and bringing the solution closer to the current guess:

$$\frac{d^2}{d\boldsymbol{x}^2_{t_0}}Cost(\boldsymbol{x}^{j-1}_{t_0}) \tag{D.6}$$

$$\approx \left(\frac{d}{d\boldsymbol{x}_{t_0}}\boldsymbol{f}_{obs}(\boldsymbol{f}^i_{dyn}(\boldsymbol{x}^{j-1}_{t_0}))\right)^T S_{obs} \left(\frac{d}{d\boldsymbol{x}_{t_0}}\boldsymbol{f}_{obs}(\boldsymbol{f}^i_{dyn}(\boldsymbol{x}^{j-1}_{t_0}))\right) \tag{D.7}$$

$$+ \lambda_j Diag\left(\left(\frac{d}{d\boldsymbol{x}_{t_0}}\boldsymbol{f}_{obs}(\boldsymbol{f}^i_{dyn}(\boldsymbol{x}^{j-1}_{t_0}))\right)^T S_{obs} \left(\frac{d}{d\boldsymbol{x}_{t_0}}\boldsymbol{f}_{obs}(\boldsymbol{f}^i_{dyn}(\boldsymbol{x}^{j-1}_{t_0}))\right)\right) \tag{D.8}$$

The damping parameter λ_j is decreased until the step obtains an improvement and increased where possible to obtain faster convergence.

For more details of the Newton, Gauss-Newton and Levenberg-Marquadt approaches applied in this section, see Fletcher (2000), Nocedal and S. J. Wright (1999), Triggs et al. (2000).

APPENDIX E

A SIMPLE WELL-MOTIVATED TECHNIQUE FOR MEASURING POSE ERROR

As noted in the body of this thesis, it is not always clear how to reconcile error in rotational and translational error, though an error measure is necessary to calculate a residual in the fit of a pose or a trajectory. This appendix discusses a method of reconciling error in pose on the basis of the assumption that pose error is simply the sum of the translational error of each point in an object.

Since there are an infinite number of points on an object, obtaining this error is a case of integrating the error across the points in the object, something that can be done analytically. The easiest way of doing this integration is by integrating according to mass which produces a result containing known terms - mass and inertial matrix.

If the object coordinate centre is chosen to coincide with the rotation centre which is assumed to be the centre of mass of the object, then rotation and mass error become additive. Then, the integral produces a simple equation for the error in terms of rotation and translational error, as well as the object volume and the tensor of rotational inertia representing the object distribution in space. This approach is also a good approximation in the absence of knowledge of the mass distribution of the object; indeed, assuming unit density it is possible to use the same equations by integrating across volume rather than mass.

Since the error under consideration is sum of squared error, then in the current approach the error in each object point $\boldsymbol{Y}$ is the integral of squared error across the body. Let $\boldsymbol{Y}^{tra}$ be the point transformed according to the error translation $\boldsymbol{T}$ and rotation $R_{\theta,\boldsymbol{\Omega}}$ where θ is the rotation angle and $\boldsymbol{\Omega}$ the axis of rotation. Then the error $E_{\boldsymbol{Y}}$ in the point $\boldsymbol{Y}$ is:

$$\begin{aligned} E_{\boldsymbol{Y}} &= (\boldsymbol{Y}^{tra} - \boldsymbol{Y})^T(\boldsymbol{Y}^{tra} - \boldsymbol{Y}) \\ &= (R_{\theta,\boldsymbol{\Omega}}(\boldsymbol{Y}) + \boldsymbol{T} - \boldsymbol{Y})^T(R_{\theta,\boldsymbol{\Omega}}(\boldsymbol{Y}) + \boldsymbol{T} - \boldsymbol{Y}) \end{aligned} \tag{E.1}$$

Let $\mathcal{V}$ be the set of points in the mass (or volume) of the body. Then the quantity of interest is the average error $E_{\boldsymbol{T},R_{\theta,\Omega}}$ across the whole body, which is the integral of the error over all of the points in the body, weighted by the object density ρ at that point:

$$E_{\boldsymbol{T},R_{\theta,\Omega}} = \iiint^{\mathcal{V}} [\rho(\boldsymbol{Y})E_{\boldsymbol{Y}}]\, d\mathcal{V} \tag{E.2}$$

$$\begin{aligned} &= \iiint^{\mathcal{V}} \left[\rho(\boldsymbol{Y})(R_{\boldsymbol{\theta},\Omega}(\boldsymbol{Y}) + \boldsymbol{T} - \boldsymbol{Y})^T(R_{\boldsymbol{\theta},\Omega}(\boldsymbol{Y}) + \boldsymbol{T} - \boldsymbol{Y})\right] d\mathcal{V} \\ &= \iiint^{\mathcal{V}} [\rho(\boldsymbol{Y})R_{\boldsymbol{\theta},\Omega}(\boldsymbol{Y})^T R_{\boldsymbol{\theta},\Omega}(Y) + 2\rho(\boldsymbol{Y})R_{\boldsymbol{\theta},\Omega}(\boldsymbol{Y})^T\boldsymbol{T} \\ &\quad - 2\rho(\boldsymbol{Y})R_{\boldsymbol{\theta},\Omega}(\boldsymbol{Y})^T\boldsymbol{Y} - 2\rho(\boldsymbol{Y})\boldsymbol{T}^T\boldsymbol{Y} + \rho(\boldsymbol{Y})\boldsymbol{T}^T\boldsymbol{T} + \rho(\boldsymbol{Y})\boldsymbol{Y}^T\boldsymbol{Y}]d\mathcal{V} \end{aligned}$$

The quantity $\boldsymbol{Y}^T\boldsymbol{Y}$ is the norm of a point, which does not change under rotation, so:

$$R_{\theta,\boldsymbol{\Omega}}(\boldsymbol{Y})^T R_{\theta,\boldsymbol{\Omega}}(\boldsymbol{Y}) = \boldsymbol{Y}^T\boldsymbol{Y} \tag{E.3}$$

Using the assumption that the object has rotation centre centre that coincides with the coordinate system origin, one of the terms can be zeroed (this equation is recognisable as the centre of mass):

$$\iiint^{\mathcal{V}} \left[\rho(\boldsymbol{Y})\boldsymbol{T}^T\boldsymbol{Y}\right] d\mathcal{V} = \boldsymbol{T}^T \iiint^{\mathcal{V}} [\rho(\boldsymbol{Y})\boldsymbol{Y}]\, d\mathcal{V} \tag{E.4}$$

$$= 0 \tag{E.5}$$

Recognising that the rotation is a linear transform of the point it operates on, that is, it can be expressed as a matrix:

$$\iiint^{\mathcal{V}} \left[\rho(\boldsymbol{Y})R_{\theta,\boldsymbol{\Omega}}(\boldsymbol{Y})^T\boldsymbol{T}\right] d\mathcal{V} = \iiint^{\mathcal{V}} \left[\rho(\boldsymbol{Y})\left(R_{\theta,\boldsymbol{\Omega}}\boldsymbol{Y}\right)^T\boldsymbol{T}\right] d\mathcal{V} \tag{E.6}$$

$$= \boldsymbol{T}^T R_{\theta,\boldsymbol{\Omega}}^T \iiint^{\mathcal{V}} \left[\rho(\boldsymbol{Y})\boldsymbol{Y}\right] d\mathcal{V} \tag{E.7}$$

$$= 0 \tag{E.8}$$

Eliminating these terms from the error integral, the following terms remain:

$$\begin{aligned} E_{\boldsymbol{T},R_{\theta,\boldsymbol{\Omega}}} &= \iiint^{\mathcal{V}} [-2\rho(\boldsymbol{Y})R_{\theta,\boldsymbol{\Omega}}(\boldsymbol{Y})^T\boldsymbol{Y} + \rho(\boldsymbol{Y})\boldsymbol{T}^T\boldsymbol{T} + \rho(\boldsymbol{Y})\boldsymbol{Y}^T\boldsymbol{Y}]d\mathcal{V} \\ &= \iiint^{\mathcal{V}} [\rho(\boldsymbol{Y})\boldsymbol{T}^T\boldsymbol{T}]d\mathcal{V} + 2\iiint^{\mathcal{V}} [\rho(\boldsymbol{Y})\boldsymbol{Y}^T\boldsymbol{Y} - \rho(\boldsymbol{Y})R_{\theta,\boldsymbol{\Omega}}(\boldsymbol{Y})^T\boldsymbol{Y}]d\mathcal{V} \end{aligned} \tag{E.9}$$

These terms have now been separated into one term based purely on rotational error and a term based purely on translational error. Indeed, the first term is clearly the mass ($\iiint^{\mathcal{V}}[\rho(\boldsymbol{Y})]d\mathcal{V}$) multiplied by the squared norm of the translational error $\boldsymbol{T}^T\boldsymbol{T}$.

The second term can be analysed so that it is a simple expression of the rotational inertia according to the frame induced by the rotation axis $\boldsymbol{\Omega}$ multiplied by a trignometric function of the rotation angle θ. First, since the component of vector to each point parallel to the rotation axis $\boldsymbol{\Omega}$ does not contribute to the error, this component can be eliminated. Let $\boldsymbol{Y}_{\boldsymbol{\Omega}}^{\perp} = \boldsymbol{Y}\cdot\boldsymbol{\Omega}$ and $\boldsymbol{Y}_{\boldsymbol{\Omega}}^{\parallel} = \boldsymbol{Y} - \boldsymbol{Y}_{\boldsymbol{\Omega}}^{\perp}$ so that $\boldsymbol{Y} = \boldsymbol{Y}_{\boldsymbol{\Omega}}^{\perp} + \boldsymbol{Y}_{\boldsymbol{\Omega}}^{\parallel}$ and $\boldsymbol{Y}_{\boldsymbol{\Omega}}^{\perp}\cdot\boldsymbol{Y}_{\boldsymbol{\Omega}}^{\parallel} = 0$.

$$\boldsymbol{Y}^T\boldsymbol{Y} = \left|\boldsymbol{Y}_{\boldsymbol{\Omega}}^{\perp}\right|^2 + \left|\boldsymbol{Y}_{\boldsymbol{\Omega}}^{\parallel}\right|^2 \tag{E.10}$$

$$\begin{aligned} R_{\theta,\boldsymbol{\Omega}}(\boldsymbol{Y})^T\boldsymbol{Y} &= R_{\theta,\boldsymbol{\Omega}}(\boldsymbol{Y}_{\boldsymbol{\Omega}}^{\parallel})^T\boldsymbol{Y}_{\boldsymbol{\Omega}}^{\parallel} - R_{\theta,\boldsymbol{\Omega}}(\boldsymbol{Y}_{\boldsymbol{\Omega}}^{\perp})^T\boldsymbol{Y}_{\boldsymbol{\Omega}}^{\perp} \\ &= \left|\boldsymbol{Y}_{\boldsymbol{\Omega}}^{\perp}\right|^2 - \cos\theta\left|Y_{\boldsymbol{\Omega}}^{\perp}\right|^2 \end{aligned} \tag{E.11}$$

So the second term in Equation E.9 becomes:

$$2\iiint^{\mathcal{V}} [\rho(\boldsymbol{Y})\boldsymbol{Y}^T\boldsymbol{Y} - \rho(\boldsymbol{Y})R_{\theta,\boldsymbol{\Omega}}(\boldsymbol{Y})^T\boldsymbol{Y}]d\mathcal{V} \tag{E.12}$$

$$= 2\iiint^{\mathcal{V}} \left[\rho(\boldsymbol{Y})\left[\left|\mathbf{Y}_{\boldsymbol{\Omega}}^{\perp}\right|^2 - \cos\theta\left|\mathbf{Y}_{\boldsymbol{\Omega}}^{\perp}\right|^2\right]\right] d\mathcal{V} \tag{E.13}$$

$$= 2(1-\cos(\theta))\iiint^{\mathcal{V}} \left[\rho(\boldsymbol{Y})\left[\left|\mathbf{Y}_{\boldsymbol{\Omega}}^{\perp}\right|^2\right]\right] d\mathcal{V} \tag{E.14}$$

$$= 2(1-\cos(\theta))\mathcal{I}_{\boldsymbol{\Omega}} \tag{E.15}$$

The integral here is identifiable as the rotational inertia $\mathcal{I}_{\boldsymbol{\Omega}}$ of the object around the axis θ. This inertial value is derivable from the inertial matrix $\mathcal{I}$ according to the identity $\mathcal{I}_{\boldsymbol{\Omega}} = \boldsymbol{\Omega}^T\mathcal{I}\boldsymbol{\Omega}$. Combining this information, the integrated error measure is:

$$E_{\boldsymbol{T},R_{\theta,\boldsymbol{\Omega}}} = M\left|\boldsymbol{T}\right|^2 + 2(1-\cos(\theta))\boldsymbol{\Omega}^T\mathcal{I}\boldsymbol{\Omega} \tag{E.16}$$

This error is very easy to calculate, is simple, general, and correct given the assumption that it is the sum of point errors that is desired.

APPENDIX F

TEXTURE RE-PROJECTION OBSERVATION MODEL

In chapter 6, a method is described for incorporating a simulation-based dynamics in a particle filter. In the interests of space, in that chapter, a discussion of how the particle weights are obtained from comparing pose hypotheses and images is only summarised. Here, it is given in more detail.

Recall that the aim is to take a hypothesis pose $\boldsymbol{x}_{t_i}$ and the current image $I_{\boldsymbol{x}_{t_i}}$ and calculate a match score that can be used as a likelihood value $p(\boldsymbol{x}_{t_i}|\boldsymbol{x}_{t_i})$.

Texture edge projection. The observation likelihood calculation routine works, at base, by projecting the texture edges of the textured object model into the camera image plane given the hypothesis pose and comparing the projected texture edges with those extracted from the image. The more edge pixels that match, the higher the likelihood.

The top right of Figure 6.1 shows the result of running an edge detector on a candidate image, and the bottom right of that figure shows the result of re-projecting the texture of a candidate pose of that object into the image plane. These two textures can be compared.

There are some complexities of this procedure, introduced by Mörwald, Zillich, et al. (2009), that need to be discussed.

Firstly, the projection of edges is done using dedicated graphics hardware via OpenGL (3D Graphics Library) and the nVidia implementation of GLSL (the GL Shading Language) (Kessenich, Baldwin, & Rost, 2006; nVidia, 2006). This speeds up the calculation of the likelihood of any given hypothesis.

For each candidate pose, what is projected are pre-computed edge maps of the object texture model rather than the texture itself. This is because the computational expense of calculating an edge image for each candidate hypothesis is deemed prohibitive. In other words, the object is decorated with a texture that consists of an edge map rather than the original texture, and it is this “edge texture” that is projected.

This method of projection leads to another problem which is that the projection of edge maps leads to thinning of projected edges. The solution is to build the texture edge map by projecting a key pose that is an average of the best hypothesis poses, calculating the edges, and re-projecting the resultant map onto the object surface. That way, edge thinning is minimised when hypothesis poses are projected onto the image, since all relevant poses are assumed to be close to this average pose.

This leads to a further problem, which is that the resulting formulation has a natural proclivity to unimodality of the pose distribution. Since hypothesis poses away from the key averaged pose will suffer from edge thinning, the tracker has a tendency to favour poses close to the average.

This might be partially accounted for by making the particle filter adaptive (altering parameters of the tracker, such as the number of particles, online), but such an approach would no longer benefit from the ability to track multiple hypotheses, relying rather on re-finding alternative hypotheses after track is lost[1]. The fact that the tracker is a particle filter with extra recursions at each time-step also means that it has the ability to do extra search to regain track, but again this may not be as necessary with true multi-modality. The main problem with maintaining multi-modality is that its basic form presuppose a lot more particles. The computational expense could again be dealt with using a set of special techniques. **Edge image matching.** The way that edge likelihood when a fully textured object is known is as follows. First, a a way of comparing two edge images to

[1]The filter described here does make noise rates dependent on particle weight, but with little effect on the experiments here.

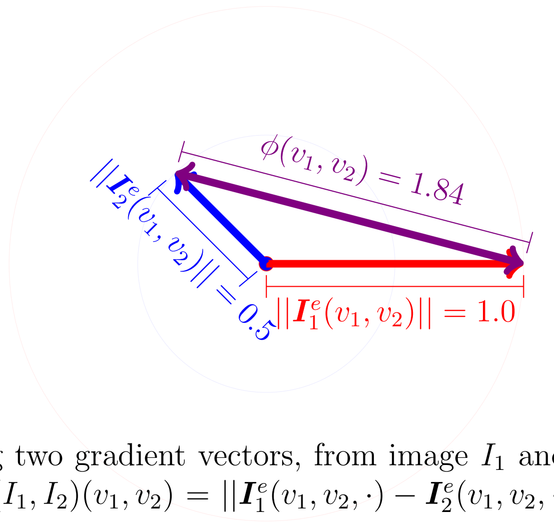

$\phi(v_1, v_2) = 1.84$
$||\boldsymbol{I}_2^e(v_1, v_2)|| = 0.5$
$||\boldsymbol{I}_1^e(v_1, v_2)|| = 1.0$

Figure F.1: Differencing two gradient vectors, from image I_1 and image I_2 at pixel u to obtain a match score $\phi(I_1, I_2)(v_1, v_2) = ||\boldsymbol{I}_1^e(v_1, v_2, \cdot) - \boldsymbol{I}_2^e(v_1, v_2, \cdot)||$. The resulting math score is a scalar.

get a match image is defined:

$$\Phi(I_1^e, I_2^e) = I_3^e \text{ s.t. } : \tag{F.1}$$

$$I_3^e(v_1, v_2) = ||\boldsymbol{I}_1^e(v_1, v_2, \cdot) - \boldsymbol{I}_2^e(v_1, v_2, \cdot)|| \tag{F.2}$$

Here, two (oriented) edge images $\boldsymbol{I}_1^e$ and $\boldsymbol{I}_2^e$ are transformed into a match image $\Phi(I_1^e, I_2^e)$ such that each pixel of that match image (where pixels have two pixel indexes v_1 and v_2) is the norm of the difference of the normalised gradient vectors at that pixel[1]. The notation $\boldsymbol{I}(a, b, \cdot)$ refers to components from all channels $(\cdot)$ of the pixel at pixel index (a, b) of the image (thus $\boldsymbol{I}(a, b, \cdot)$ is a vector). These channels are colour channels in the original image I and the directional gradient magnitudes in the edge image I^e. The match image, having only one channel, is indexed by only two indices. See Figure F.1 for a graphical illustration of this calculation.

This match image can then be used to compare the edge image acquired from the camera and an edge image based on a hypothesised pose, in order to obtain a match score. First, however, the process for obtaining a candidate edge image is illustrated.

Edge texture re-projection. Before each particle is compared to the image, the edge

[1]This is equivalent to the length of the chord on the unit circle defining gradient direction - note that gradient magnitude is not generally taken into account, except that the gradient vector length is either zero or one

map of the object must be calculated. Recalculating the edge map for each hypothesised pose adds too much computational expense for a real-time tracker, so what is projected to the image is an edge map. However, a single edge map does not suffice for all potential poses as the edges are distorted on projection into the image plane; therefore it is necessary to use an edge map that has been re-projected onto the object at a pose close to the one being considered. This process is summarised in the upper part of the process flowchart in Figure F.2.

First, $\boldsymbol{x}^{\mu}_{t_i}$, a mean pose of the best hypothesis particles is calculated[1]. This is used, along with the camera calibration parameters, to project the object colour texture S into the image plane.

In this work, the function $I(\boldsymbol{x}, S)$ takes a pose and a texture and creates an image of its projection to the image plane (taking into account the pre-calibrated parameters of the camera). This process is done in specialised graphics hardware and involves using the pose to calculate coordinates in the raster texture image for each projected pixel and interpolating the texture local to those coordinates. The process of projection of a texture is not addressed here, and is enabled by the use of the OpenGL graphics library (Segal & Akeley, 2006; Shreiner, Woo, Neider, & Tom Davis, 2005). The black-box calculation of the projected texture is written as:

$$I_{\mu_{t_i}} = I(\boldsymbol{x}^{\mu}_{t_i}, S) \tag{F.3}$$

Now will be described how the re-projected texture is obtained from the mean particle $\boldsymbol{x}^{\mu}_{t_i}$.

Edge detection. Once the mean pose $\boldsymbol{x}^{\mu}_{ti}$ has been used to project the texture S into the image plane, obtaining a projected image $I_{\mu_{t_i}}$, a Sobel edge detector is then passed over this image (having first smoothed it with a Gaussian). The output of the Sobel edge detector is thresholded and normalised so that all edges over a certain threshold are

[1] "Best" particles is the set of particles with the highest likelihood calculated at the previous time-step

given the same magnitude (but the direction of the gradient vector varies around the unit circle). The result of this also undergoes a small number n_γ of recursive edge dilation operations $Spr()$[1], with the magnitude of dilated edge pixels being a fraction (γ) of the magnitude of the gradient edges of neighbouring pixels. Multiple colour channels are taken care of by calculating the gradient for each channel and using the gradient with the highest magnitude.

To summarise this, the production of the edge image I^e from the original image I is written:

$$I^e_{\mu_{t_i}} = E(I_{\mu_{t_i}}) \tag{F.4}$$

Where:

$$\begin{aligned}
E(I) =& I^e \text{ s.t. } \forall v_1, v_2 : \\
& l_{max} = \underset{l}{argmax} \frac{\partial I(v_1, v_2, l)}{\partial [v_1, v_2]} \\
& \boldsymbol{I}^g(v_1, v_2, \cdot) = \frac{\partial I(v_1, v_2, l_{max})}{\partial [v_1, v_2]}, \\
& \boldsymbol{I}^\gamma(v_1, v_2, \cdot) = \begin{cases} 0 \text{ if } ||\boldsymbol{I}^g(v_1, v_2, \cdot)|| < g_{thresh} \\ \frac{\boldsymbol{I}^g(v_1, v_2, \cdot)}{||\boldsymbol{I}^g([v_1, v_2, \cdot])||} \text{ otherwise} \end{cases} \\
& I^e = Spr(I^\gamma, n_\gamma, \gamma)
\end{aligned} \tag{F.5}$$

Here, I^g is the image of the result of the Sobel calculation of the most edge-responsive colour channel l_max. This is subsequently thresholded to produce $\boldsymbol{I}^\gamma$, which undergoes edge dilation to produce I^e.

The original image is a multi-channel colour image and the resulting edge image is also multi-channel (channel indexed by l), each channel corresponding to one component of the gradient 2-vector at each pixel. The edge response is chosen as the maximum edge

[1]The dilation reduces the peakiness of the likelihood function

response of each of the colour channels, and thresholded by the number g_{thresh}[1].

The dilation algorithm is illustrated in Figure F.3. It is not a typical dilation algorithm in that the dilation is done in image gradient space and there is a discount factor on the gradient magnitude. However, the basic algorithm of repeated neighbourhood search is typical. It is hypothesised that the use of this dilation creates a better-behaved image likelihood function, by broadening the peaks in the likelihood function.

This edge image is then re-projected back onto the object surface, creating an edge texture[2]:

$$S^e_{\mu_{t_i}} = I^{-1}(\boldsymbol{x}^{\mu}_{t_i}, I^e_{\mu_{t_i}}) \tag{F.6}$$

This edge texture is taken, for the remainder of the calculations in this recursion of the particle filter, to be the edge texture for all particles in the particle set. Thus, in order to calculate the edge map of a projected particle the mean edge image is again projected from the particle's pose:

$$I^e_{\boldsymbol{x}_{t_i}[j]} = I(\boldsymbol{x}_{t_i}[j], S^e_{\mu_{t_i}}) \tag{F.7}$$

For each time step, a parallel process of edge detection is carried out for the observed image, $I_{\boldsymbol{z}_{t_i}}$, where $E()$ is the function previously defined above:

$$I^e_{z_{t_i}} = E(I_{z_{t_i}}) \tag{F.8}$$

For each particle, the projected edge image of the particle can be compared to the edge image of the observed camera frame to obtain a match image:

$$\Phi_{t_i}[j] = \Phi(I^e_{\boldsymbol{x}_{t_i}[j]}, I^e_{t_i}) \tag{F.9}$$

See Figure F.2 for a diagram of how the calculation of $\Phi_{t_i}[j]$ is carried out.

This match image is used as the basis of the particle weight calculation, the approxi-

[1]Colour intensity values are taken to be normalised to between 0 and 1

[2]In practice this new texture is maintained in the 2D image coordinate frame and indexed per pixel as needed according to the saved projection $\boldsymbol{I}(\boldsymbol{x}^{\mu}_{t_i}, \cdot)$, so that I^{-1} need not be calculated explicitly.

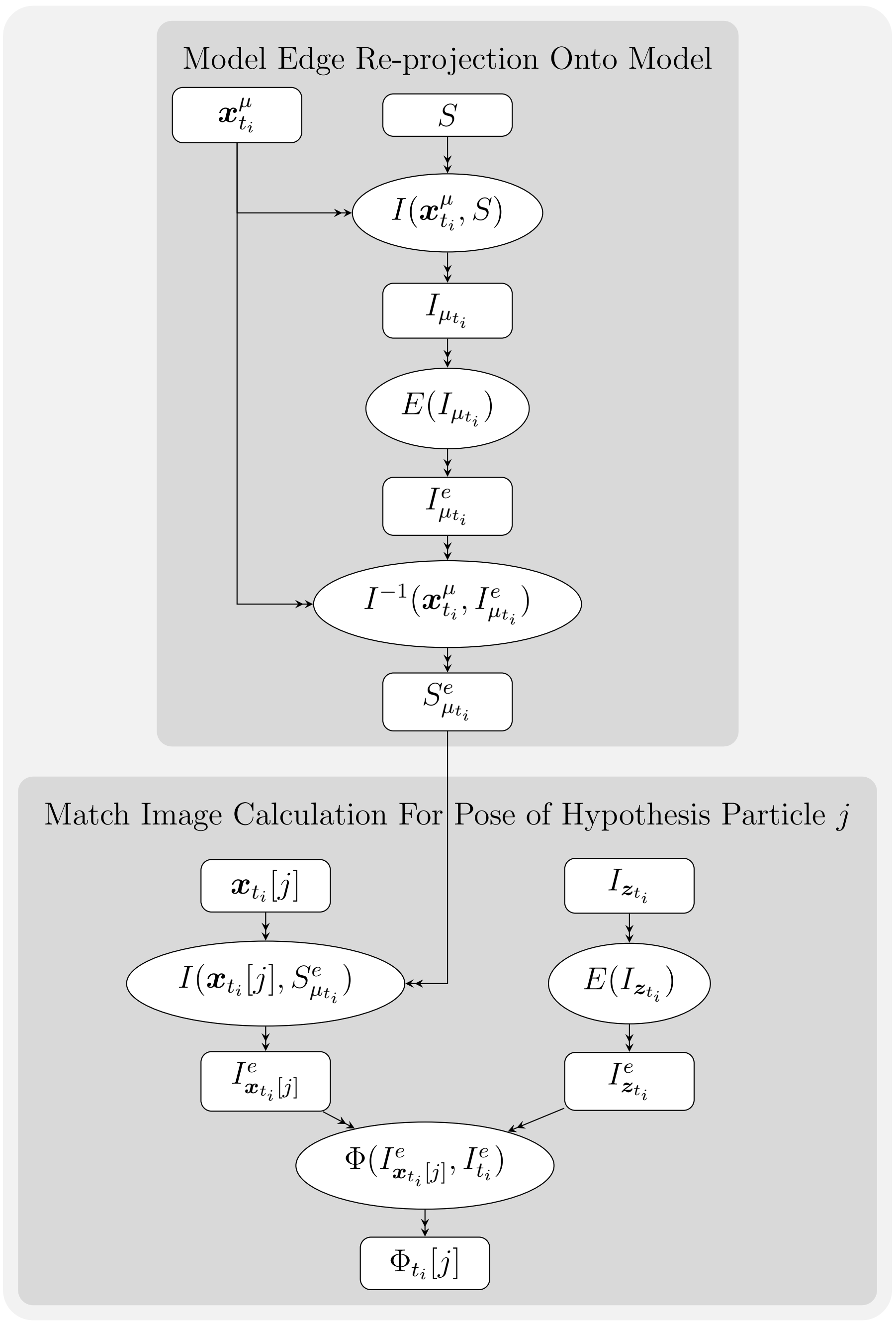

Model Edge Re-projection Onto Model
$\boldsymbol{x}^{\mu}_{t_i}$
S
$I(\boldsymbol{x}^{\mu}_{t_i}, S)$
$I_{\mu_{t_i}}$
$E(I_{\mu_{t_i}})$
$I^e_{\mu_{t_i}}$
$I^{-1}(\boldsymbol{x}^{\mu}_{t_i}, I^e_{\mu_{t_i}})$
$S^e_{\mu_{t_i}}$
Match Image Calculation For Pose of Hypothesis Particle j
$\boldsymbol{x}_{t_i}[j]$
$I_{\boldsymbol{z}_{t_i}}$
$I(\boldsymbol{x}_{t_i}[j], S^e_{\mu_{t_i}})$
$E(I_{\boldsymbol{z}_{t_i}})$
$I^e_{\boldsymbol{x}_{t_i}[j]}$
$I^e_{\boldsymbol{z}_{t_i}}$
$\Phi(I^e_{\boldsymbol{x}_{t_i}[j]}, I^e_{t_i})$
$\Phi_{t_i}[j]$

Figure F.2: Creation of the match image $\Phi_{t_i}[j]$ for the pose of a given particle j at time t_i, by, first, re-projecting the texture S based on a mean particle $\boldsymbol{x}^{\mu}_{t_i}$ (which is done once per recursion of the particle filter), and then the comparing the projected texture $I^e_{\boldsymbol{x}_{t_i}[j]}$ from particle j pose $\boldsymbol{x}_{t_i}[j]$ with the edge image $I^e_{\boldsymbol{z}_{t_i}}$ of observation $\boldsymbol{z}_{t_i}$. This is then used directly to calculate the likelihood of the particle. Note that the images I_* are multi-channel images indexed by pixel address, containing different colour channels or different channels corresponding to the components of the gradient vector. $I(\boldsymbol{x}, S_*)$, $I^{-1}(\boldsymbol{x}, I_*)$ are the projection and re-projection between textures and images via pose. $E()$ is the edge detection / image gradient calculation function.

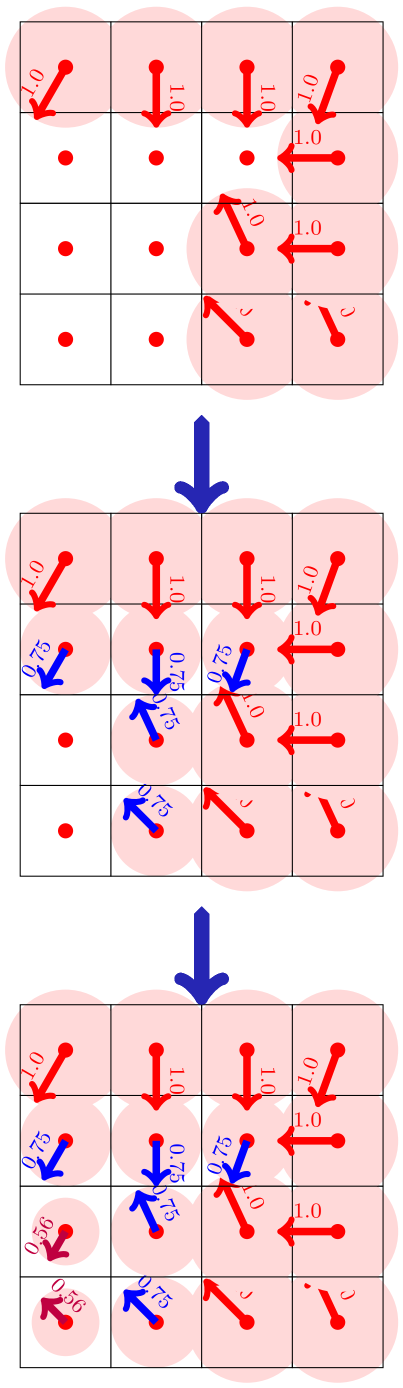


Figure F.3: A graphical representation of the slightly generalised edge dilation algorithm used in this work and that of Mörwald, Zillich, et al. (2009). The top grid is a representation of a simple binary gradient image, with gradient magnitude thresholded to 0 or 1. The bottom two grids represent successive runs of the edge dilation algorithm. The discount factor in this illustration is $\gamma = 0.75$, and the 8-neighbourhood of each pixel is searched for each dilation. Note that much of the time the choice of exactly which orientation is used is arbitrary when many neighbourhood pixels have the same magnitude. In practice this presently depends on pixel search order.

mation of the likelihood of the particle given the observed image. The basic idea is that the number of pixels where the match value is below a certain threshold are counted and that count is used as the weight for the particle.

Calculating particle weights. The weight calculation is:

$$w^{-}_{t_i}[j] = \frac{(n_{t_i}[j] + n^{max}_{t_i})}{2.n^{max}_{t_i}} \frac{m_{t_i}[j]}{n_{t_i}[j]} \tag{F.10}$$

$$w_{t_i}[j] = \frac{1}{W}(w^{-}_{t_i}[j])^{C_{conv}} \tag{F.11}$$

$$W = \sum_j w^{-}_{t_i}[j] \tag{F.12}$$

The calculation expresses the number of matches of the particle $m_{t_i}[j]$ (as per the current threshold m_{thresh}) as a proportion of the number of potential matches of the particle $n_{t_i}[j]$. This is adjusted with respect to the highest number of potential matches across all particles $n^{max}_{t_i}$ so that poses with faces parallel to the imaging plane are not overly preferred. For more details see Mörwald, Zillich, et al. (2009). The weights are made more peaky by using a predetermined "convergence" parameter, C_{conv}. W is chosen so that $\sum_j w_{t_i}[j] = 1$.

Writing the calculation of pixel match counts, potential match counts and maximum potential match counts across across all particles:

$$m_{t_i}[j] = |\{[v_1, v_2] \text{ s.t. } \Phi_{t_i}[j](v_1, v_2) < m_{thresh}\}| \tag{F.13}$$

$$n_{t_i}[j] = \left|\left\{[v_1, v_2] \text{ s.t. } ||\boldsymbol{I}^{e}_{\boldsymbol{x}_{t_i}[j]}(v_1, v_2)|| > 0\right\}\right| \tag{F.14}$$

$$n^{max}_{t_i} = \max_j n_{t_i}[j] \tag{F.15}$$

On the basis of these weights, the observation likelihood, with observation $\boldsymbol{z}_{t_i}$ fixed, for each particle j is approximated as:

$$p_{obs}(\boldsymbol{z}_{t_i}|\boldsymbol{x}_{t_i}[j]) \approx w_{t_i}[j] \tag{F.16}$$

APPENDIX G

EFFECT OF MULTIPLE RECURSIONS AT EACH TIME-STEP ON A SIMPLIFIED ANNEALED PARTICLE FILTER WITH PHYSICS-BASED DYNAMICS

In Chapter 6, a simplified annealed particle filter is described that is applicable to dynamics models that introduce only process noise during state-evolution. This algorithm was described by Mörwald, Zillich, et al. (2009) and is an alteration to the particle filter to re-run the particle filter multiple times on an image at one time-step to encourage convergence.

Chapter 7 describe some experiments with that particle filter, focusing on the integration of new kinds of dynamics models based on physical simulation with different ways of adding noise. See that chapter for background on the terminology used in this appendix.

The present appendix extends those results by presenting video slices illustrating the costs and benefits that the extra recursions present, particularly in the context of strong dynamics models. These slices are shown in Figures G.1 to G.8.

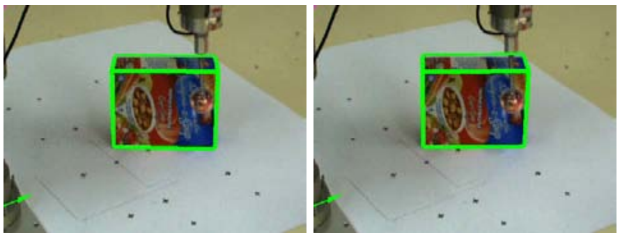
Multiple recursions at a time-step generally lead to improved track for visible objects due to the increased amount of searching, but are more able to be distracted by false likelihood optima in the presence of distractors. Further, they can interfere with the beneficial effects of a good dynamics model.

Including multiple recursions of the particle filter at each time-step can enable a more extensive search of the likelihood space, where possible. This argument is relevant to the core thesis in that there are motivations for comparing the novel conditions to the 2-recursion non-simulation condition, because if multiple recursions are truly better on the cases considered here then it constitutes the best alternative algorithm for comparison. On the other hand, since the best novel condition (simulation with force noise) is only feasible with 1 recursion, it also makes sense to only compare the 1 recursion conditions.

Moreover, this appendix can constitute an experimental analysis of a heuristic technique that was only recently introduced. It has been predicted that with narrow-peaked likelihood distributions, the extra search should be advantageous, as with the annealed particle filter (Deutscher et al., 2000), but that multiple recursions at one time-step should lead to problems when the likelihood contains only distracting information (such as during occlusion).

G.1 Results

Video 1 (Figure G.1), does not show any significant difference between any of the conditions.

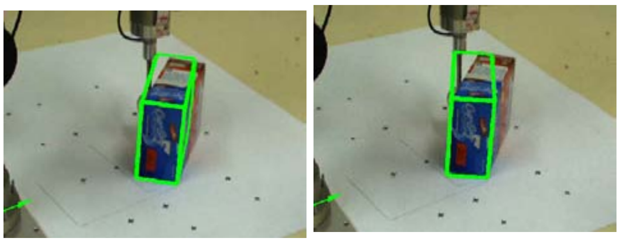


Frame 222 (scenario 1.1): No simulation, rank retention, 1 recursion (left) and 2 recursions (right) - O/R-1 and O/R-2

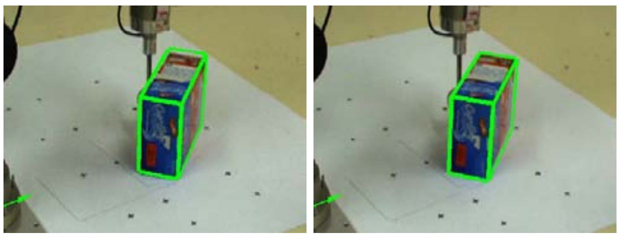
Figure G.1: **Scenario 1.1 video slices.** Number of recursions at each time-step makes little difference on easy tasks.

Average performance on video 2 is improved by increasing the number of recursions at each time-step, since the biggest problem in this video is tracking an object as it becomes end-on to the camera (Figure G.2), which makes for a smaller target likelihood valley in a scenario where it is postulated that the observation model makes likelihood optima already somewhat difficult to find, and may interact with the trade-off between likelihood based on visible edges as a proportion of total edges in the scene and as a proportion of total expected visible edges. Conversely, multiple recursions contributes to the finding of the narrow likelihood valleys.

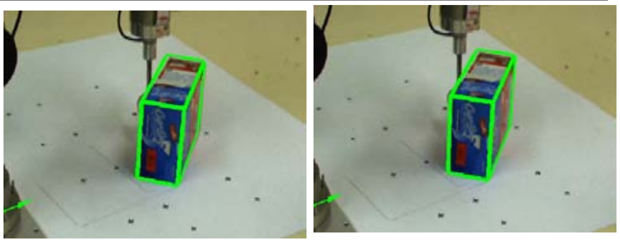
Also note that on video 2, performance is improved if a retention scheme is not used; most likely because retention focuses the search around previously found observation likelihood optima.



Frame 181 (scenario 2.2): Without (left) and with (right) simulation, dispersion noise, with rank-retention, one recursion of the particle filter between each frame - O/R-1 (left) and D/R-1 (right)



Frame 181 (scenario 2.2): Without (left) and with (right) simulation, dispersion noise, no retention, one recursion - O-1 and D-1

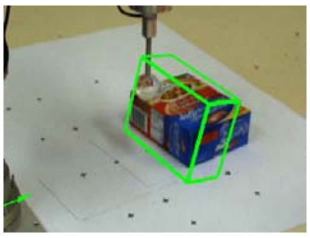


Frame 181 (scenario 2.2): Without (left) and with (right) simulation, dispersion noise, with rank-retention, two recursions of the particle filter between each frame - O/R-2 (left) and D/R-2 (right)

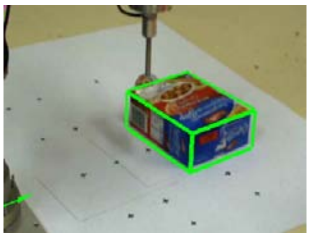
Figure G.2: **Scenario 2.2 video slices.** Here the positive effect of using multiple recursions at each time-step can be seen, as well as the effect of the retention scheme.

Average performance on the tipping video 3 (example images in Figure G.3 and G.4) is also increased. Since this involves fast object motion, more search is more likely to find the object that has moved away from the primary search window. If the object is not found soon after it tips, the tracker can become trapped in a different local optima.

As can be seen in videos 4 and 5, during occlusion (example images in Figure G.5 and G.6) multiple recursions can take the object lock further away from the target by

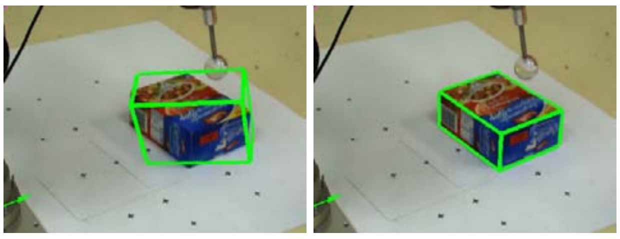


Frame 150 (scenario 3.2): Simulation with dispersion noise, rank retention and only one recursion of the particle filter - D/R-1

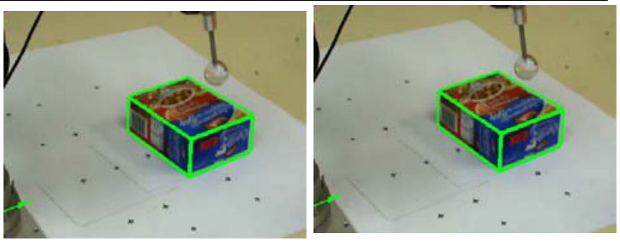


Frame 150 (scenario 3.2): Simulation with dispersion noise, rank retention and two recursions of the particle filter - D/R-2

Figure G.3: **Scenario 3.2 video slices.** Here can be seen the positive effect of using multiple recursions at each time-step to increase search during fast object movement.



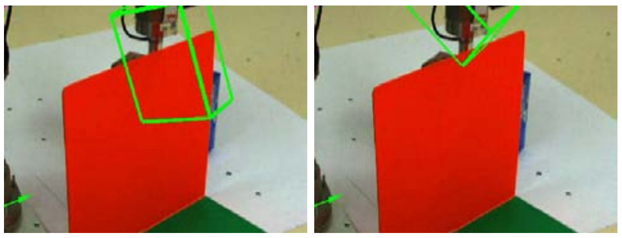
Frame 308 (scenario 3.3): Simulation with dispersion noise, rank retention and 1 recursion - D/R-1



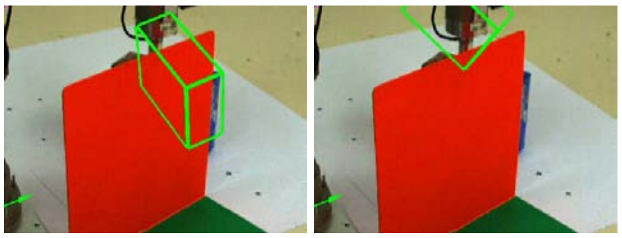
Frame 308 (scenario 3.3): Simulation with dispersion noise, rank retention and 2 recursions - D/R-2

Figure G.4: **Scenario 3.3 video slices.** Recovery from track is improved by increasing the number of recursions at each time-step. This effect is robust across simulation and non-simulation dispersion noise conditions.

giving it more opportunities to lock onto distractors.



Frame 155 (scenario 4.2): No simulation with dispersion noise, rank retention and 1 recursion (left) and 2 recursions (right) - O/R-1 and O/R-2

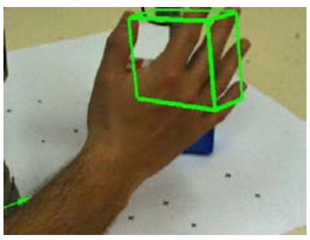
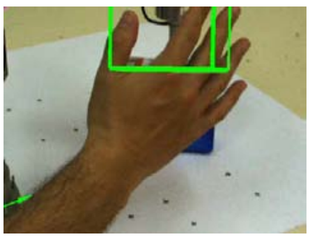


Frame 155 (scenario 4.2): Simulation with dispersion noise, rank retention and 1 recursion (left) and 2 recursions (right) - D/R-1 and D/R-2 and

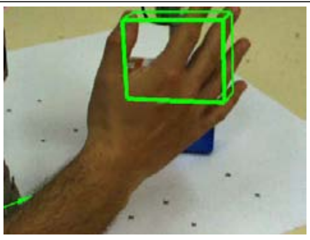
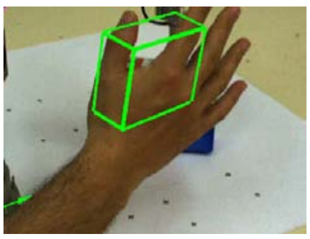
Figure G.5: **Scenario 4.2 video slices.** The tendency of multiple recursions to be captured by distractors is illustrated, as well as the moderating effect of a simulation model when added to dispersion noise.

However, as borne out in the remainder of video 5 (example images in Figure G.7), multiple recursions at each time-step can aid in recovering track due to the improved search for newly revealed edges. Additionally, there is a negative interaction in track recovery between the use of multiple recursions and the simulation with dispersion noise condition - possibly because the vertical movement in hypothesised position necessary to return track to the target object is more likely to be associated with a rotation away from the true object pose when simulation is involved.

There is no clear effect of multiple recursions in video 6 (example images in Figure G.8). Though, as can be seen in the section on the introduction of finger location in Section 6.5.5, in Figure 7.31 there is an interaction between multiple recursions and the modelling of the finger, with the multiple recursions causing distractors to have a larger effect, while existence of the finger pushes the already incorrect pose further from the correct one.

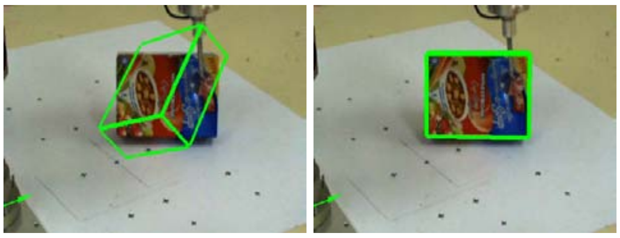



Frame 190 (scenario 5.2): No simulation with dispersion noise, rank retention and 1 recursion (left) and 2 recursions (right) - O/R-1 and O/R-2

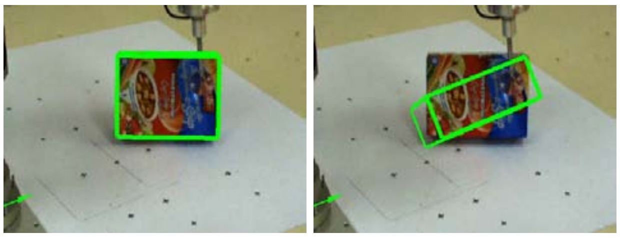



Frame 190 (scenario 5.2): Simulation with dispersion noise, rank retention and 1 recursion (left) and 2 recursions (right) - D/R-1 and D/R-2

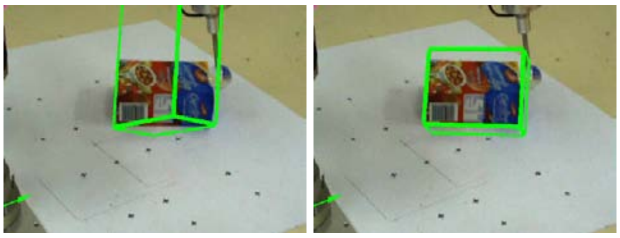
Figure G.6: **Scenario 5.2 video slices.** The tendency of multiple recursions to be captured by distractors is illustrated, as well as the moderating effect of a simulation model when added to dispersion noise.



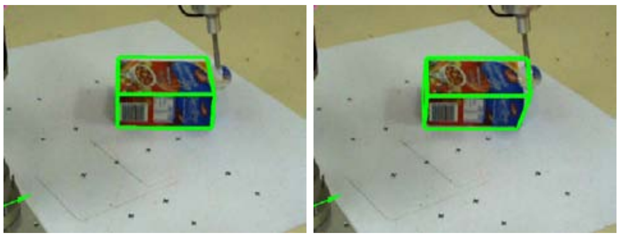
Frame 239 (scenario 5.3): No simulation with dispersion noise, rank retention and 1 recursion (left) and 2 recursions (right) - O/R-1 and O/R-2



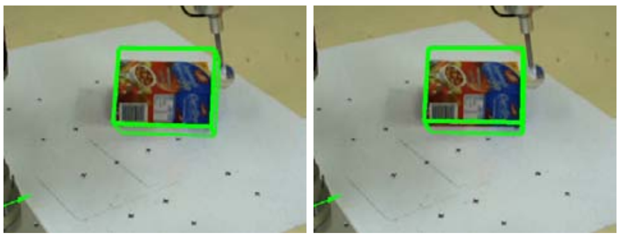
Frame 239 (scenario 5.3): Simulation with dispersion noise, rank retention and 1 recursion (left) and 2 recursions (right) - D/R-1 and D/R-2



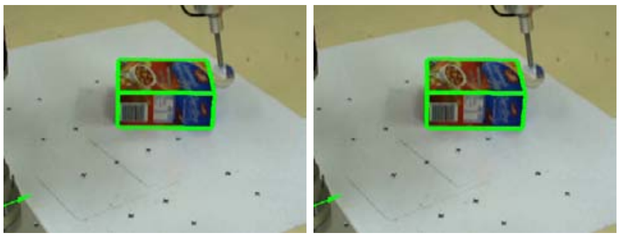
Frame 278 (scenario 5.4): No simulation with dispersion noise, rank retention and 1 recursion (left) and 2 recursions (right) - O/R-1 and O/R-2



Frame 278 (scenario 5.4): Simulation with dispersion noise, rank retention and 1 recursion (left) and 2 recursions (right) - D/R-1 and D/R-2



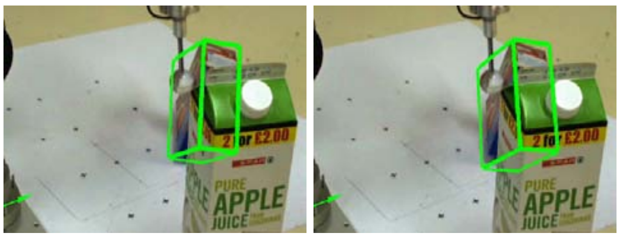
Frame 321 (scenario 5.5): No simulation with dispersion noise, rank retention and 1 recursion (left) and 2 recursions (right) - O/R-1 and O/R-2



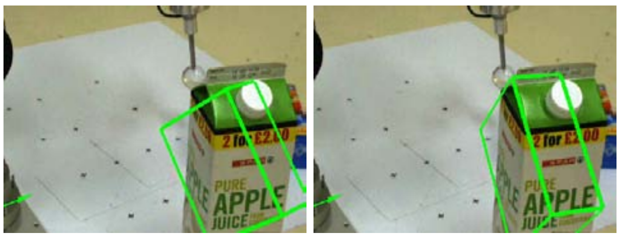
Frame 321 (scenario 5.5): Simulation with dispersion noise, rank retention and 1 recursion (left) and 2 recursions (right) - D/R-1 and D/R-2

Figure G.7: **Scenario 5.3, 5.4 and 5.5 video slices.** Multiple recursions at each time-step can increase the probability of a successful recovery of track.

.



Frame 237 (scenario 6.2): No simulation with dispersion noise, rank retention and 1 recursion (left) and 2 recursions (right) - O/R-1 and O/R-2



Frame 264 (scenario 6.3): No simulation with dispersion noise, rank retention and 1 recursion (left) and 2 recursions (right) - O/R-1 and O/R-2

Figure G.8: **Scenario 6.2 and 6.3 video slices.** There is no clear effect of multiple recursions at each time-step in this difficult case, though there are some interactions - see for example Figure 7.31.

For numerical results, see the box-whisker charts in Figures 7.8 and 7.9.

G.2 Discussion

The effect of the use of particle retention is equivocal in the results presented here, interacting in complex ways with the other factors explored.

For the basic tracker (with the original pose dispersion dynamics model), multiple recursions at each time-step leads to an improvement in those situations where there is no occlusion. This is attributable to the extra search performed by generating more particles, so that a high-likelihood area is more likely to be found. On the other hand, this search leads to a deterioration of performance under occlusion, since high-likelihood areas are likely to be distractors where there is occlusion. To clarify, since the actual object is not visible under occlusion, the highest matching pose in the image is unlikely to be from the object or in the same image location or pose as the object. On the other hand, this extra search does assist in re-finding track after occlusion.

In general, the use of multiple recursions at each time-step has not been as well motivated by this study as has the use of a simulator based dynamics model. Previous studies have shown that similar methods can bear fruit, however (Deutscher et al., 2000; Pupilli & Calway, 2005). The challenge is to obtain those benefits while at the same time retaining the benefits obtained by using force noise with simulation.

APPENDIX

BIBLIOGRAPHY

Aggarwal, J., & Nandhakumar, N. (1988). On the computation of motion from sequences of images - a review. *Proceedings of the IEEE*, *76*(8), 917–935. doi:`10.1109/5.5965`

Alspach, D. L., & Sorenson, H. W. (1972). Nonlinear Bayesian estimation using Gaussian sum approximations. *IEEE Transactions on Automatic Control*, *17*(4), 439–448.

Alur, R., Henzinger, T. A., Lafferriere, G., & Pappas, G. J. (2000). Discrete abstractions of hybrid systems. *Proceedings of the IEEE*, *88*(7), 971–984.

Álvarez, M. A., Luengo, D., & Lawrence, N. D. (2009). Latent force models. In D. van Dyk & M. Welling (Eds.), *Proceedings of the twelfth international conference on artificial intelligence and statistics* (pp. 9–16). AISTATS. Clearwater Beach, Florida, USA: JMLR Workshop & Conference Proceedings. Retrieved from `http://jmlr.csail.mit.edu/proceedings/papers/v5/alvarez09a/alvarez09a.pdf`

Álvarez, M. A., Peters, J., Schölkopf, B., & Lawrence, N. D. (2011). Switched latent force models for movement segmentation. In J. Lafferty, C. K. Williams, J. Shawe-Taylor, R. Zemel & A. Culotta (Eds.), *Advances in neural information processing systems* (Vol. 23, pp. 1–9). 24th Annual Conference on Neural Information Processing Systems 2010. Red Hook, NY, USA: Curran.

Armstrong, M., & Zisserman, A. (1995). Robust object tracking. In *Proceedings of the asian conference on computer vision* (Vol. 1, pp. 58–61). Lecture Notes in Computer Science (LNCS) 1035. ACCV. Singapore: Springer.

Baillargeon, R., & Hanko-Summers, S. (1990). Is the top object adequately supported by the bottom object? young infants' understanding of support relations. *Cognitive Development*, *5*(1), 29–53.

Balan, A. O., Sigal, L., & Black, M. J. (2005). A quantitative evaluation of video-based 3D person tracking. In *Proceedings of the 14th international conference on computer communications and networks* (pp. 349–356). ICCCN. San Diego, CA, USA: IEEE Computer Society. Retrieved 18 August 2010, from `http://portal.acm.org/citation.cfm?id=1259814`

Baraff, D. (1993). Non-penetrating rigid body simulation. In *State of the art reports* (pp. 1–23). Eurographics. Barcelona, Spain. Retrieved from `http://www.cs.cmu.edu/~baraff/papers/eg93.pdf`

Barnden, J. (2001). The utility of reversed transfers in metaphor. In *Proceedings of the twenty-third annual meeting of the cognitive science society* (pp. 57–62). COGSCI. Edinburgh, Scotland. Retrieved from `http://www.cs.bham.ac.uk/~jab/Publications/metaphor-alone.list.html`

Baumberg, A. M., & Hogg, D. C. (1994a). An efficient method for contour tracking using active shape models. In *Proceedings of the workshop on motion of nonrigid and articulated objects.* (pp. 194–199). Austin, TX, USA: IEEE Computer Society. Retrieved from `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.16.1396`

Baumberg, A. M., & Hogg, D. C. (1994b). Learning flexible models from image sequences. In *Proceedings of the european conference on computer vision* (Vol. 1, pp. 299–308). ECCV. Stockholm, Sweden. Retrieved from `http://citeseer.ist.psu.edu/viewdoc/summary?doi=10.1.1.16.1712`

Bender, J., & Schmitt, A. (2006). Constraint-based collision and contact handling using impulses. In *Proceedings of the 19th international conference on computer animation and social agents* (pp. 3–11). CASA. Uni Dufour, Geneva, Switzerland.

Bennett, B., Magee, D. R., Cohn, A. G., & Hogg, D. C. (2004). Using spatio-temporal continuity constraints to enhance visual tracking of moving objects. In *Proceedings* *of the European conference on artificial intelligence* (pp. 922–926). ECAI. Valencia, Spain. Retrieved from `http://www.comp.leeds.ac.uk/drm/ecai04-1.pdf`

Berclaz, J., Fleuret, F., & Fua, P. (2009). Multiple object tracking using flow linear programming. In *12th IEEE international workshop on performance evaluation of tracking and surveillance* (pp. 1–8). Winter-PETS. Snowbird, Utah, USA.

Bhat, K. S., Seitz, S. M., Popovic, J., & Khosla, P. K. (2002). Computing the physical parameters of rigid-body motion from video. In *Proceedings of the European conference on computer vision* (pp. 551–565). Lecture Notes in Computer Science (LNCS) 2350. ECCV. Copenhagen, Denmark.

Biederman, I., Mezzanotte, R. J., & Rabinowitz, J. C. (1982). Scene perception: detecting and judging objects undergoing relational violations. *Cognitive Psychology*, *14*(2), 143–177.

Birnbaum, L., Brand, M., & Cooper, P. (1993). Looking for trouble: using causal semantics to direct focus of attention. In *Proceedings of the international conference on computer vision* (pp. 49–56). ICCV. Berlin, Germany. Retrieved from `http://citeseer.ist.psu.edu/viewdoc/summary?doi=10.1.1.47.8320`

Black, D. (2011). Under the PhysX hood - any publication pointers? - NVIDIA developer forums. Retrieved 28 February 2011, from `http://developer.nvidia.com/forums/index.php?showtopic=5758`

Blackhall, L., & Rotkowitz, M. (2008). Recursive sparse estimation using a Gaussian sum filter. In *Proceedings of the 17th world congress of the international federation of automatic control*. IFAC. Seoul, Republic of Korea. Retrieved from `http://www.ee.unimelb.edu.au/people/mcrotk/publications/ifac08_br_proc.pdf`

Blackman, S. (2004). Multiple hypothesis tracking for multiple target tracking. *IEEE Aerospace and Electronic Systems Magazine*, *19*(1), 5–18. doi:`10.1109/MAES.2004.1263228`

Blake, A., Curwen, R., & Zisserman, A. (1993). A framework for spatiotemporal control in the tracking of visual contours. *International Journal of Computer Vision*, *11*(2), 127–145.

Bongard, J., Zykov, V., & Lipson, H. (2006). Resilient machines through continuous self-modeling. *Science*, *314*(5802), 1118–1121. doi:10.1126/science.1133687

Bradski, G. R. (1998). Computer vision face tracking for use in a perceptual user interface. *Intel Technology Journal*, *Q2*. doi:10.1.1.14.7673

Brand, M. (1997). Physics-based visual understanding. *Computer Vision and Image Understanding*, *65*(2), 192–205. doi:06/cviu.1996.0572

Brand, M., Cooper, P., & Birnbaum, L. (1995, June). Seeing physics, or: physics is for prediction. In *Proceedings of the workshop on physics-based modeling in computer vision, 1995* (pp. 144–150). PBMCV. Cambridge, MA, USA: IEEE. doi:10.1109/PBMCV.1995.514679

Brankart, J.-M. (2006). *Optimal nonlinear filtering and smoothing assuming truncated Gaussian probability distributions in a reduced dimension space of adaptive size.* Technical Report. MEOM-LEGI, Grenoble. Retrieved from http://www-meom.hmg.inpg.fr/Web/pages-perso/brankart/Publis_pdf/tgf.pdf

Broida, T. J., & Chellappa, R. (1991). Estimating the kinematics and structure of a rigid object from a sequence of monocular images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *13*(6), 497–513. Retrieved from http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=87338&userType=inst

Brown, D. (2008). Video modeling: combining dynamic model simulations with traditional video analysis. In *American association of physics teachers summer meeting*. Poster PST3-15AAPT. AAPT. Edmonton, Alberta, Canada.

Bullet Physics Library. (2011). Retrieved 24 January 2011, from http://bulletphysics.org/mediawiki-1.5.8/index.php/Bullet_User_Manual_and_API_documentation

Candes, E. J., & Tao, T. (2005). Decoding by linear programming. *IEEE Transactions on Information Theory*, *51*(12), 4203–4215. doi:`10.1109/TIT.2005.858979`

Canny, J. (1987). A computational approach to edge detection. In M. A. Fischler & O. Firschein (Eds.), *Readings in computer vision: issues, problems, principles, and paradigms* (Vol. 184, 87–116). San Francisco, CA, USA: Morgan Kaufmann.

Cao, H., Ohnishi, N., Takeuchi, Y., Matsumoto, T., & Kudo, H. (2007). Gauss-Newton particle filter. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, *E90-A*(6), 1235–1239.

Catto, E. (2005). Iterative dynamics with temporal coherence. In *Game developers conference*. GDC. San Francisco, CA, USA. Retrieved from `http://www.bulletphysics.com/ftp/pub/test/physics/papers/IterativeDynamics.pdf`

Cellier, F. E., & Roddier, N. (1991). Qualitative state spaces: a formalization of the naive physics approach to knowledge-based reasoning. In *Proceedings of the conference on AI, simulation and planning in high autonomy systems: integrating qualitative and quantitative system knowledge* (40–49). Cocoa Beach, FA, USA.

Chang, C., Ansari, R., & Khokhar, A. (2005). Multiple object tracking with kernel particle filter. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 566–573). CVPR. San Diego, CA, USA. Retrieved from `http://ieeexplore.ieee.org/iel5/9901/31472/01467318.pdf?arnumber=1467318`

Chan, M., Metaxas, D. N., & Dickinson, S. (1994). Physics-based tracking of 3D objects in 2D image sequences. In *Proceedings of the international conference on pattern recognition* (pp. 432–436). ICPR. Seattle, WA, USA.

Chenney, S., & Forsyth, D. (2000). Sampling plausible solutions to multi-body constraint problems. In *Proceedings of the ACM international conference on computer graphics and interactive techniques* (pp. 219–228). SIGGRAPH. New Orleans, LA, USA. Retrieved from `http://portal.acm.org/citation.cfm?id=344882`

Chesterton, G. (1987). *Saint Francis of Assisi*. New York, NY, USA: Image.

Chum, O., & Matas, J. (2002). Randomized RANSAC with T d,d test. *Image and Vision Computing*, *22*, 448–457. Retrieved 21 May 2010, from `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.6.5136`

Clark, A. (2011). Whatever next? predictive brains, situated agents, and the future of cognitive science. unpublished draft. Unpublished draft.

Clark, A., & Grush, R. (1999). Towards a cognitive robotics. *Adaptive Behavior*, *7*(1), 5–16.

Comaniciu, D., Ramesh, V., & Meer, P. (2003). Kernel-based object tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *25*(5), 564–577. Retrieved from `http://www.caip.rutgers.edu/~comanici/Papers/KernelTracking.pdf`

CRISIS Physics Library. (2011). Retrieved 10 July 2011, from `http://crisis.sourceforge.net/`

Davis, E. (2008). Physical reasoning. *Foundations of Artificial Intelligence*, *3*, 597–620.

Davison, A. J. (2003). Real-time simultaneous localisation and mapping with a single camera. In *Proceedings of the international conference on computer vision* (pp. 1403–1411). ICCV. Nice, France.

Davison, A. J., Reid, I. D., Molton, N. D., & Stasse, O. (2007). MonoSLAM: real-time single camera SLAM. *IEEE Transations on Pattern Analysis and Machine Intelligence*, *29*(6), 1052–1067.

Dearden, R., & Clancy, D. (2002). Particle filters for real-time diagnosis of planetary rovers. In *Proceedings of the thirteenth international workshop on principles of diagnosis* (pp. 1–6). DX. Semmering, Austria.

Decarlo, D., & Metaxas, D. N. (2000). Optical flow constraints on deformable models with applications to face tracking. *International Journal of Computer Vision*, *38*(2), 99–127.

DeGroot, M. (1986). *Probability and statistics* (2nd). Wokingham, England: Addison-Wesley.

De Kleer, J., & Brown, J. S. (1984). A qualitative physics based on confluences. *Artificial Intelligence*. Artificial Intelligence, *24*(1-3), 7–83. Retrieved 24 August 2010, from `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.91.3833`

Deutscher, J., Blake, A., & Reid, I. (2000). Articulated body motion capture by annealed particle filtering. In *Proceedings of the IEEE computer society conference on computer vision and pattern recognition* (Vol. 2, pp. 21–26). CVPR. Retrieved 22 January 2011, from `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.94.4250`

Djuric, P. M., & Kotecha, J. H. (2003). Gaussian sum particle filtering. *IEEE Transactions on Signal Processing*, *51*(10), 2602–2612.

Doucet, A., de Freitas, N., Murphy, K., & Russell, S. (2000). Rao-Blackwellised particle filtering for dynamic Bayesian networks. In *Proceedings of the conference on uncertainty in artificial intelligence* (pp. 176–183). UAI. Stanford, CA, USA. Retrieved 21 January 2011, from `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.36.4120`

Drummond, T., & Cipolla, R. (2002). Real-time visual tracking of complex structures. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *24*(7), 932–946.

Duff, D. J. (2011a). Angular momentum precesses, why? - NVIDIA developer forums. Retrieved 10 July 2011, from `http://forums.developer.nvidia.com/index.php?showtopic=3939&st=0&p=17950`

Duff, D. J. (2011b). Simple experiment testing static friction failed - NVIDIA developer forums. Retrieved 10 July 2011, from `http://forums.developer.nvidia.com/index.php?showtopic=5536&st=0&p=17923`

Duff, D. J., Mörwald, T., Stolkin, R., & Wyatt, J. (2011). Physical simulation for monocular 3D model based tracking. In *Proceedings of the IEEE international conference on robotics and automation* (p. In Print). ICRA. Shanghai, China: IEEE. Retrieved from `http://www.cs.bham.ac.uk/~nah/bibtex/papers/damien_jade_duff_physical_2011.pdf`

Duff, D. J., Wyatt, J., & Stolkin, R. (2010, May). Motion estimation using physical simulation. In *Proceedings of the IEEE international conference on robotics and automation* (pp. 1511–1517). ICRA. Anchorage, AL, USA: IEEE. Retrieved from `http://dx.doi.org/10.1109/ROBOT.2010.5509590`

Eberly, D. H. (2010, April). *Game physics, second edition* (2nd ed.). San Francisco, CA, USA: Morgan Kaufmann.

Ennesser, F., & Medioni, G. (1995). Finding Waldo, or focus of attention using local color information. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *17*(8), 805–809. Retrieved from `http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=400571&isnumber=9031`

Erdem, Ç. E., Tekalp, A. M., & Sankur, B. (2001). Metrics for performance evaluation of video object segmentation andtracking without ground-truth. In *International conference on image processing* (pp. 69–72). ICIP. Thessaloniki, Greece.

Erdem, Ç. E., Sankur, B., & Tekalp, A. M. (2004). Performance measures for video object segmentation and tracking. *IEEE Transactions on Image Processing*, *13*, 937–951.

Ettlin, A., Buchler, P., & Bleuler, H. (2005). A simulation environment for robot motion planning. In *Proceedings of the fifth international workshop on robot motion and control* (pp. 277–282). RoMoCo. Dymaczewo, Poland. doi:`10.1109/ROMOCO.2005.201436`

Evensen, G. (2003). The ensemble Kalman filter: theoretical and practical implementation. *Ocean Dynamics*, *53*, 343–367.

Felzenszwalb, P. F., & Huttenlocher, D. P. (2004). Efficient graph-based image segmentation. *International Journal of Computer Vision*, *59*(2). Retrieved from `http://people.cs.uchicago.edu/~pff/segment/`

Fischler, M. A., & Bolles, R. C. (1981). Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, *24*, 381–395.

Flea 2 Specifications. (2009, February). http://www.ptgrey.com/products/flea2/flea2.pdf. Point Grey. Retrieved 20 December 2010, from http://www.ptgrey.com/products/flea2/flea2.pdf

Fletcher, R. (2000, May). *Practical methods of optimization* (2nd ed.). Cornwall, England: Wiley.

Fontanelli, D., Ricciato, L., & Soatto, S. (2007). A fast RANSAC–based registration algorithm for accurate localization in unknown environments using LIDAR measurements. In *International conference on automation science and engineering* (pp. 597–602). CASE. Scottsdale, AZ, USA: IEEE. Retrieved from http://www.centropiaggio.unipi.it/robpublications/pub/papers/Localization-CASE07.pdf

Fu, J., Srinivasa, S., Pollard, N., & Nabbe, B. (2007). Planar batting under shape, pose, and impact uncertainty. In *IEEE international conference on robotics and automation*. ICRA. Rome, Italy: IEEE. Retrieved from http://www.ri.cmu.edu/pubs/pub_5685.html

Geeter, J. D., Brussel, H. v., Schutter, J. D., & Decreton, M. (1997). A smoothly constrained Kalman filter. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *19*(10), 1171–1177. Retrieved from http://doi.ieeecomputersociety.org/10.1109/34.625129

Gentner, D. (2002). Mental models, psychology of. In N.J. Smelser & P.B. Bates (Eds.), *International encyclopedia of the social and behavioral sciences* (pp. 9683–9687). Amsterdam, Netherlands: Elsevier Science. Retrieved from http://groups.psych.northwestern.edu/gentner/papers/Gentner02d.pdf

Gibbons, J. D., & Chakraborti, S. (2010). *Nonparametric statistical inference* (4th Rev). New York, NY, USA: CRC Press.

Gillespie, D. T. (1992). *Markov processes: an introduction for physical scientists*. San Diego, CA, USA: Academic Press.

Gillespie, R. B., Patoğlu, V., Hussein, I. I., & Westervelt, E. R. (2005). Online symbolic constraint embedding for simulation of hybrid dynamical systems. *Multibody System Dynamics*, *14*(3-4), 387–417. doi:10.1007/s11044-005-0269-0

F. Glover, & G. A. Kochenberger (Eds.). (2003). *Handbook of metaheuristics*. International series in operations research & management science. Boston, MA, USA: Kluwer Academic Publishers.

Gordon, I., & Lowe, D. G. (2004). Scene modelling, recognition and tracking with invariant image features. In *International symposium on mixed and augmented reality* (pp. 110–119). ISMAR. Arlington, VA,USA.

Gottschalk, S., Lin, M. C., & Manocha, D. (1996). OBBTree: a hierarchical structure for rapid interference detection. In *Proceedings of the ACM international conference on computer graphics and interactive techniques* (171–180). ACM ID: 237244. SIGGRAPH. New Orleans, LA, USA: ACM. doi:10.1145/237170.237244

Grush, R. (2004). The emulation theory of representation: motor control, imagery, and perception. *Behavioral and Brain Sciences*, *27*, 377–442.

Grzeszczuk, R., Terzopoulos, D., & Hinton, G. (1998). NeuroAnimator: fast neural network emulation and control of physics-based models. In *Proceedings of the ACM international conference on computer graphics and interactive techniques* (pp. 9–20). SIGGRAPH. Orlando, Florida: ACM. Retrieved from www.cs.toronto.edu/~hinton/absps/siggraph98.pdf

Hairer, E., & Söderlind, G. (2005). Explicit, time reversible, adaptive step size control. *SIAM Journal on Scientific Computing*, *26*(6), 1838. doi:10.1137/040606995

Harris, C., & Stennett, C. (1990). RAPID - a video rate object tracker. In *Proceedings of the British machine vision conference* (73–77). BMVC. Oxford, England.

Harris, C., & Stephens, M. (1988). A combined corner and edge detector. In *Proceedings of the Alvey vision conference* (Vol. 15, p. 50). Manchester, England.

Hartley, R. I. (1997). In defense of the eight-point algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *19*(6), 580–593. Retrieved 24 August 2010, from `http://portal.acm.org/citation.cfm?id=262634`

Hartley, R. I. (1998). Minimizing algebraic error. In *Sixth IEEE international conference on computer vision* (Vol. 356, pp. 1175–1192). ICCV. Bombay, India: IEEE. doi:`10.1109/ICCV.1998.710760`

Hartley, R. I., & Kahl, F. (2009, January). Global optimization through rotation space search. *International Journal of Computer Vision*, *82*(1), 64–79. doi:`10.1007/s11263-008-0186-9`

Hartley, R. I., & Zisserman, A. (2000). *Multiple view geometry in computer vision*. Cambridge, England: Cambridge University Press.

Hau, D. T., Coiera, E., & Muggleton, S. (1997). Learning qualitative models of dynamic systems. *Machine Learning*, *26*, 177–211. Retrieved from `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.28.5266`

Hauser, M. D., Chomsky, N., & Fitch, W. (2002). The faculty of language: what is it, who has it, and how did it evolve? *Science*, *298*(5598), 15–69.

Havok Physics. (2011). Retrieved 24 January 2011, from `http://www.havok.com/index.php?page=havok-physics`

Hayes, P. (1979). The naive physics manifesto. In D. Michie (Ed.), *Expert systems in the microelectronic age* (pp. 242–270). Edinburgh, Scotland: Edinburgh University Press.

Heap, T., & Hogg, D. C. (1998). Wormholes in shape space: tracking through discontinuous changes in shape. In *Proceedings of the IEEE international conference on computer vision* (pp. 344–349). ICCV. Bombay, India. Retrieved from `http://citeseer.ist.psu.edu/viewdoc/summary?doi=10.1.1.37.2884`

Hegarty, M. (2004). Mechanical reasoning by mental simulation. *Trends in Cognitive Sciences*, *8*(6), 280–285.

Hettmansperger, T. P., & McKean, J. W. (2010). *Robust nonparametric statistical methods* (2nd Ed.). London, England: CRC Press.

Hogg, D. C. (1984). *Interpreting images of a known moving object.* PhD Thesis, School of Cognitive and Computing Sciences, University of Sussex, Brighton, UK.

Hongeng, S., & Wyatt, J. (2008). Learning causality and intentional actions. In *Proceedings of the 2006 international conference on towards affordance-based robot control* (27–46). ACM ID: 1787360. Dagstuhl Castle, Germany: Springer-Verlag. Retrieved from `http://portal.acm.org/citation.cfm?id=1787357.1787360`

Huber, P. J. (1981). *Robust statistics.* New York, NY, USA: John Wiley and Sons.

Hu, X., & Ahuja, N. (1992a). Estimating motion of constant acceleration from image sequences. In *Proceedings 11th IAPR international conference on pattern recognition, conference a: computer vision and applications* (pp. 655–659). ICPR-CVA. The Hague, Netherlands.

Hu, X., & Ahuja, N. (1992b). Long image sequence motion analysis using polynomial motion models. In *IAPR workshop on machine vision applications* (pp. 109–114). MVA. Tokyo, Japan.

Ibáñez, L. (2001). Tutorial on quaternions. Retrieved 20 January 2011, from `http://www.itk.org/CourseWare/Training/QuaternionsI.pdf;http://www.itk.org/CourseWare/Training/QuaternionsII.pdf`

Irani, M., & Anandan, P. (1999). All about direct methods. In *Vision algorithms: theory and practice* (pp. 267–277). Lecture Notes in Computer Science (LNCS) 1883. International Workshop on Vision Algorithms. Corfu, Greece: Springer.

Isard, M., & Blake, A. (1998a). CONDENSATION – conditional density propagation for visual tracking. *International Journal of Computer Vision*, *29*(1), 5–28.

Isard, M., & Blake, A. (1998b). ICONDENSATION: unifying low-level and high-level tracking in a stochastic framework. In *Proceedings of the 5th european conference on computer vision* (pp. 893–908). ECCV. Freiburg, Germany.

Ito, K., & Xiong, K. (2000). Gaussian filters for nonlinear filtering problems. *IEEE Transactions on Automatic Control*, *45*(5), 910–927. doi:`10.1109/9.855552`

Iu, S.-L., & Wohn, K. (1990). Estimation of general rigid body motion from a long sequence of images. In *International conference on pattern recognition* (pp. 217–219). ICPR. Atlantic City, NJ, USA. Retrieved from `http://repository.upenn.edu/cgi/viewcontent.cgi?article=1561&context=cis_reports`

Jain, R. (1981, August). Dynamic scene analysis using pixel-based processes. *Computer*, *14*(8), 12–18. doi:`10.1109/C-M.1981.220557`

Jain, R., Kasturi, R., & Schunck, B. G. (1995). *Machine vision* (1st ed.). New York, NY, USA: McGraw-Hill.

Jakobi, N. (1997). Evolutionary robotics and the radical envelope-of-noise hypothesis. *Adaptive Behavior*, *6*(2), 325–368. doi:`10.1177/105971239700600205`

Jepson, A. D., Fleet, D. J., & El-Maraghi, T. F. (2001). Robust online appearance models for visual tracking. *IEEE Transations on Pattern Analysis and Machine Intelligence*, *25*(10), 1296–1311. Retrieved 24 August 2010, from `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.6.4265`

Jiménez, P. (2008). Brief survey on model-based manipulation planning of rigid objects [Institut de robotica i informatica industrial (CSIC - UPC)]. Retrieved from `http://www.iri.upc.edu/people/jimenez/manipulation_rigid.pdf`

Jordan, M. I., & Rumelhart, D. E. (1992). Forward models: supervised learning with a distal teacher. *Cognitive Science*, *16*, 307–354. Retrieved from `http://brick3.ist.psu.edu/viewdoc/summary?doi=10.1.1.106.873`

Julier, S. J., & Uhlmann, J. K. (1997). A new extension of the Kalman filter to nonlinear systems. In *SPIE international symposium on aerospace/defense sensing, simulation and control*. Orlando, Florida. Retrieved from `http://tinyurl.com/4dqojq`

Jurie, F., & Dhome, M. (2002). Real time robust template matching. In *Proceedings of the British machine vision conference* (pp. 123–131). BMVC. Cardiff, Wales.

Kakadiaris, I. A., Metaxas, D. N., & Bajcsy, R. (1994). Active part-decomposition, shape and motion estimation of articulated objects: a physics-based approach. In *Proceedings of the IEEE computer society conference on computer vision and pattern recognition* (pp. 980–984). CVPR. Seattle, WA, USA. doi:`10.1109/CVPR.1994.323938`

Kakadiaris, I. A., Metaxas, D. N., & Bajcsy, R. (1997, February). Inferring 2d object structure from the deformation of apparent contours. *Computer Vision and Image Understanding*, *65*(2), 129–147. doi:`06/cviu.1996.0580`

Kalman, R. E. (1960). A new approach to linear filtering and prediction problems. *Transactions of the ASME - Journal of Basic Engineering*, *82*, 35–45.

Kandepu, R., Foss, B., & Imsland, L. (2008). Applying the unscented Kalman filter for nonlinear state estimation. *Journal of Process Control*, *18*(7-8), 753–768. doi:`10.1016/j.jprocont.2007.11.004`

Karara, H. M. (1989, May). *Non-topographic photogrammetry* (2nd Ed.). Bethesda, MD, USA: ASPRS Publications.

Kessenich, J., Baldwin, D., & Rost, R. (2006, September). The OpenGL shading language specification: version 1.20 rev 8. 3DLabs. Retrieved from `http://www.opengl.org/registry/doc/GLSLangSpec.Full.1.20.8.pdf`

Khansari-Zadeh, S., & Billard, A. (2010). Bm: an iterative algorithm to learn stable non-linear dynamical systems with gaussian mixture models. In *Robotics and automation (icra), 2010 ieee international conference on* (pp. 2381–2388). Robotics and Automation (ICRA), 2010 IEEE International Conference on. doi:`10.1109/ROBOT.2010.5510001`

Kim, S., Gribovskaya, E., & Billard, A. (2010). Learning motion dynamics to catch a moving object. In *10th ieee-ras international conference on humanoid robots*.

Klank, U., Pangercic, D., Rusu, R. B., & Beetz, M. (2009). Real-time CAD model matching for mobile manipulation and grasping. In *The 9th IEEE-RAS international conference on humanoid robots* (pp. 290–296). Humanoids. Paris, France. Retrieved from `http://files.rbrusu.com/publications/Klank09Humanoids.pdf`

Klein, G., & Murray, D. W. (2006). Full 3D edge tracking with a particle filter. In *Proceedings of the 17th British machine vision conference* (pp. 1119–1128). BMVC. Edinburgh, Scotland. Retrieved from `http://www.robots.ox.ac.uk/~dwm/Publications/klein_murray_bmvc2006/klein_murray_bmvc2006.pdf`

Kober, J., & Peters, J. (2009). Learning motor primitives for robotics. In *IEEE international conference on robotics and automation* (pp. 2112–2118). ICRA. Kobe, Japan. doi:`10.1109/ROBOT.2009.5152577`

Koller, D., Daniilidis, K., & Nagel, H.-H. (1993). Model-based object tracking in monocular image sequences of road traffic scenes. *International Journal of Computer Vision*, *10*(3), 257–281.

Kopicki, M., Stolkin, R., Zurek, S., Mörwald, T., & Wyatt, J. (2010). Predicting workpiece motions under pushing manipulations using the principle of minimum energy. In *Procedings of the robotic science and systems workshop on representation for object grasping and manipulation in single and dual arm tasks*. RSS. Zaragoza, Spain. Retrieved from `http://www.cas.kth.se/~danik/RSS/RSSWorkshop2010.pdf`

Kopicki, M., Wyatt, J., & Stolkin, R. (2009). Prediction learning in robotic pushing manipulation. In *Proceedings of the international conference on advanced robotics* (pp. 1–6). ICAR. Munich, Germany.

Kotecha, J. H., & Djurić, P. M. (2003). Gaussian particle filtering. *IEEE Transations on Signal Processing*, *51*, 2592–2601. Retrieved 21 January 2011, from `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.129.6900`

Kragic, D., Miller, A. T., & Allen, P. K. (2001). Real-time tracking meets online grasp planning. In *Proceedings of the IEEE international conference on robotics and automation* (pp. 2460–2465). ICRA. Seoul, Republic of Korea.

Krainin, M., Henry, P., Ren, X., & Fox, D. (2010, May). Manipulator and object tracking for in hand model acquisition. ICRA Mobile Manipulation Workshop. Anchorage, AL, USA. Retrieved from `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.176.4735`

Kuipers, B. J. (2001). Qualitative simulation. In R. A. Meyers (Ed.), *Encyclopedia of physical science and technology* (3rd Ed, pp. 287–300). San Diego, CA, USA: Academic Press.

Lakoff, G., & Johnson, M. (1984). *Metaphors we live by*. Chicago, IL, USA: University Of Chicago Press.

Lanz, O. (2005). Hybrid joint-separable multibody tracking. In *Computer vision and pattern recognition* (pp. 413–420). CVPR. San Diego, CA, USA. Retrieved from `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.132.9541`

Lanz, O. (2006). Approximate Bayesian multibody tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *28*(9), 1436–1449. doi:`10.1109/TPAMI.2006.177`

LaValle, S. M. (2011a, June). Motion planning part I. *IEEE Robotics & Automation Magazine*, *18*(2), 108–118. doi:`10.1109/MRA.2011.941635`

LaValle, S. M. (2011b, June). Motion planning part II. *IEEE Robotics & Automation Magazine*, *18*(2), 108–118. doi:`10.1109/MRA.2011.941635`

Leibe, B., Leonardis, A., & Schiele, B. (2008). Robust object detection with interleaved categorization and segmentation. *International Journal of Computer Vision*, *77*(1-3), 259–289. Retrieved from `http://www.springerlink.com/content/j1269614728g1767/`

Lengyel, E. (2003). *Mathematics for 3D game programming and computer graphics* (2nd Ed.). Boston, MA, USA: Charles River Media.

Lepetit, V., & Fua, P. (2005). Monocular model-based 3D tracking of rigid objects: a survey. *Foundations and Trends in Computer Graphics and Vision*, *1*, 1–89.

Lepetit, V., Pilet, J., & Fua, P. (2004). Point matching as a classification problem for fast and robust object pose estimation. In *Proceedings of the ieee international conference on computer vision and pattern recognition* (Vol. 2, pp. 244–250). CVPR. Washington, DC, USA: IEEE. doi:`10.1109/CVPR.2004.1315170`

Lin, M. C., & Gottschalk, S. (1998). Collision detection between geometric models: a survey. In *IMA conference on mathematics of surfaces* (pp. 37–56). Birmingham, England. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.45.3926

Liu, H. (2008, December). A fuzzy qualitative framework for connecting robot qualitative and quantitative representations. *IEEE Transactions on Fuzzy Systems*, *16*(6), 1522–1530. doi:10.1109/TFUZZ.2008.2005004

Loscalzo, S., & Wright, R. (2010). Automatic methods for continuous state space abstraction. In *AAAI conference on artificial intelligence*. AAAI. Atlanta, GA, USA.

Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, *60*(2), 91–110.

Lu, L., Dai, X., & Hager, G. (2006). Efficient particle filtering using ransac with application to 3d face tracking. *Image and Vision Computing*, *24*, 581–592.

MacCormick, J., & Blake, A. (2000). A probabilistic exclusion principle for tracking multiple objects. *International Journal of Computer Vision*, *39*(1), 57–71.

Magee, D. R. (2001). Tracking multiple vehicles using foreground, background and motion models. *Image and Vision Computing*, *22*, 143–155. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.16.1390

Mann, H. B. (1947). On a test of whether one of two random variables is stochastically larger than the other. *The Annals of Mathematical Statistics*, *18*(1), 50–60. doi:10.1214/aoms/1177730491

Mann, R., & Jepson, A. D. (1998). Towards the computational perception of action. In *Proceedings of the IEEE computer society conference on computer vision and pattern recognition* (794–799). CVPR. Santa Barbara, CA, USA.

Marchand, E., Bouthemy, P., Chaumette, F., & Moreau, V. (1999). Robust real-time visual tracking using a 2D-3D model-based approach. In *Proceedings of the seventh IEEE international conference on computer vision* (pp. 262–268). ICCV. Kerkyra, Greece. doi:10.1109/ICCV.1999.791229

Maronna, R. A., Martin, R. D., & Yohai, V. J. (2006). *Robust statistics: theory and methods*. Chichester, England: John Wiley & Sons.

Masson, L., Dhome, M., & Jurie, F. (2004). Robust real time tracking of 3D objects. In *International conference on pattern recognition* (pp. 252–255). ICPR. Cambridge, England.

Masson, L., Dhome, M., & Jurie, F. (2005). Tracking 3D object using flexible models. In *Proceedings of the British machine vision conference* (pp. 1–10). BMVC. Oxford, England.

McShaffry, M. (2009). *Game coding complete, 3rd* (3rd ed.). Boston, MA, USA: Charles River Media.

Metaxas, D. N. (1997). *Physics-based deformable models: applications to computer vision, graphics, and medical imaging*. Berlin, Germany: Springer.

Metaxas, D. N., & Terzopoulos, D. (1993). Shape and nonrigid motion estimation through physics-based synthesis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *15*(6), 580–591. doi:10.1109/34.216727

Metaxas, D. N., & Kakadiaris, I. A. (2002). Elastically adaptive deformable models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *24*(10), 1310–1321. doi: 10.1109/TPAMI.2002.1039203

Michalewicz, Z., & Fogel, D. B. (2004). *How to solve it: modern heuristics* (2nd Ed.). Berlin, Germany: Springer.

Mihaylova, L., Brasnett, P., Canagarajah, N., & Bull, D. (2007). Object tracking by particle filtering techniques in video sequences, In *Advances and challenges in multisensor data and information* (8). NATO Security Through Science Series. Netherlands: IOS Press. Retrieved 24 August 2010, from http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.60.8510

Mirtich, B., & Canny, J. (1995). Impulse-based simulation of rigid bodies. In *Proceedings of the 1995 symposium on interactive 3D graphics* (pp. 181–188). SI3D95. Monterey, CA, USA.

Mittrapiyanuruk, P., DeSouza, G. N., & Kak, A. C. (2004). Calculating the 3D-pose of rigid-objects using active appearance models. In *Proceedings of the IEEE international conference in robotics and automation*. ICRA. New Orleans, LA, USA.

Montemerlo, M., Thrun, S., Koller, D., & Wegbreit, B. (2002). FastSLAM: a factored solution to the simultaneous localization and mapping problem. In *In proceedings of the AAAI international conference on artificial intelligence* (pp. 593–598). AAAI-02. Edmonton, Alberta, Canada. Retrieved 24 August 2010, from `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.16.2153`

Moravánszky, A., & Terdiman, P. (2004). Fast contact reduction for dynamics simulation, In *Game programming gems* (Vol. 4). Boston, MA, USA: Charles River Media. Retrieved 11 July 2011, from `http://www.bulletphysics.com/ftp/pub/test/index.php?dir=physics/papers/&file=Fast_Contact_Reduction_for_Dynamics_Simulation012_draft9.doc`

Mörwald, T., Kopicki, M., Stolkin, R., Wyatt, J., Zurek, S., Zillich, M., & Vincze, M. (2011). Predicting the unobservable: visual 3d tracking with a probabilistic motion model. In *Proceedings of the IEEE international conference on robotics and automation* (p. In Print). ICRA. Shanghai, China. Retrieved from `http://cogx.eu/data/cogx/publications/moerwald11predicting.pdf`

Mörwald, T., Prankl, J., Richtsfeld, A., Zillich, M., & Vincze, M. (2010). Blort - the blocks world robotic vision toolbox. In *Best practice in 3d perception and modeling for mobile manipulation*. ICRA. Anchorage, AL, USA: IEEE.

Mörwald, T., Zillich, M., & Vincze, M. (2009, October). Edge tracking of textured objects with a recursive particle filter. In *International conference on computer graphics and vision* (pp. 96–103). GraphiCon. Moscow, Russia. Retrieved from `http://cogx.eu/data/cogx/publications/moerwald2009edge.pdf`

Murray, R. M., Li, Z., & Sastry, S. S. (1994). *A mathematical introduction to robotic manipulation*. London, England: CRC Press.

Myatt, D., Torr, P. H. S., Nasuto, S. J., Bishop, J. M., & Craddock, R. (2002). NAPSAC: high noise, high dimensional robust estimation-it's in the bag. In *Proceedings of the British machine vision conference* (pp. 458–467). BMVC. Cardiff, Wales.

Naeem, A., Mills, S., & Pridmore, T. (2006). Structured combination of particle filter and kernel mean-shift tracking. In *International conference on image and vision computing new zealand* (Vol. 21). IVCNZ. Great Barrier Island, New Zealand. Retrieved from `http://www.cs.nott.ac.uk/~tpp/papers/naeemrevised.pdf`

Narayanan, S. (1999). Moving right along: a computational model of metaphoric reasoning about events. In *Proceedings of the AAAI national conference on artificial intelligence* (pp. 121–128). AAAI-99. Orlando, FL, USA.

Needham, C. J., & Boyle, R. D. (2003). Performance evaluation metrics and statistics for positional tracker evaluation. In *Proceedings of the computer vision systems third international conference* (Vol. 2626, pp. 278–289). ICVS. Graz, Austria. Retrieved from `http://citeseer.ist.psu.edu/viewdoc/summary?doi=10.1.1.103.2617`

Neuronics - Katana. (2011). Retrieved 4 February 2011, from `http://www.neuronics.com/cms_en/web/index.php?id=244&s=katana`

Newton Game Dynamics. (2011). Retrieved 10 July 2011, from `http://newtondynamics.com/forum/newton.php`

Ng, B., Pfeffer, A., & Dearden, R. (2005). Continuous time particle filtering. In *Proceedings of the 19th international joint conference on AI* (pp. 1360–1365). IJCAI. Edinburgh, Scotland. Retrieved from `http://www.eecs.harvard.edu/~bmng/papers/ctpf.pdf`

Nguyen-Tuong, D., & Peters, J. (2010). Using model knowledge for learning inverse dynamics. In *IEEE international conference on robotics and automation* (pp. 2677–2682). ICRA. Anchorage, AL, USA: IEEE. doi:`10.1109/ROBOT.2010.5509858`

Nickels, K., & Hutchinson, S. (1998, May). Weighting observations: the use of kinematic models in object tracking. In *Proceedings of the IEEE international conference on* *robotics and automation* (Vol. 2, pp. 1677–1682). ICRA. Leuven, Belgium: IEEE. doi:10.1109/ROBOT.1998.677401

Nielsen, P. (1988). Qualitative mechanics: envisioning the clock [Qualitative reasoning group, department of computer science, university of illinois at urbana-champaign]. Retrieved from http://www.qrg.northwestern.edu/papers/Files/qr-worksho ps/QP88/Nielsen_Qualitative_Mechanics_Envisioning_Clock.pdf

Nocedal, J., & Wright, S. J. (1999). *Numerical optimization*. New York, NY, USA: Springer.

Nummiaro, K., Koller-Meier, E., & Van Gool, L. (2002). Object tracking with an adaptive color-based particle filter. In *Proceedings of the 24th DAGM symposium on pattern recognition* (Vol. 2449, pp. 353–360). DAGM. Zurich, Switzerland. doi:10.1.1.11. 5314

Núñez, R. E. (2005). Creating mathematical infinities: metaphor, blending, and the beauty of transfinite cardinals. *Journal of Pragmatics*, *37*(10), 1717–1741. doi:doi:DOI:1 0.1016/j.pragma.2004.09.013

nVidia. (2006, November). Release notes for NVIDIA OpenGL shading language support. nVidia. Retrieved from http://developer.download.nvidia.com/opengl/glsl/ glsl_release_notes.pdf

Open Dynamics Engine Manual. (2011). Retrieved 24 January 2011, from http://opend e.sourceforge.net/wiki/index.php/Manual_%28All%29

Open source computer vision library: reference manual. (2001)(Intel Report No. A77028-004). Retrieved from http://software.intel.com/sites/oss/pdfs/OpenC Vreferencemanual.pdf

Ozden, K., Schindler, K., & Van Gool, L. (2010). Multibody structure-from-motion in practice. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, *32*(6), 1134–1141. doi:10.1109/TPAMI.2010.23

Özuysal, M., Fua, P., & Vincent Lepetit. (2007). Fast keypoint recognition in ten lines of code. In *IEEE conference on computer vision and pattern recognition* (pp. 1–8).

CVPR. Minneapolis, MN, USA. Retrieved from http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4270148

Özuysal, M., Lepetit, V., Fleuret, F., & Fua, P. (2006). Feature harvesting for tracking-by-detection. In *European conference on computer vision* (Vol. 3953, pp. 592–605). ECCV. Graz, Austria. Retrieved from http://cvlab.epfl.ch/publications/publications/2006/OzuysalLFF06.pdf

Perez, P., Hue, C., Vermaak, J., & Gangnet, M. (2002, June). Color-based probabilistic tracking. In *Proceedings of the European conference on computer vision* (pp. 661–675). Lecture Notes in Computer Science (LNCS) 2350. ECCV. Copenhagen, Denmark. Retrieved from http://www.irisa.fr/vista/Papers/2002/perez_hue_eccv02.pdf

Peters, J., Janzing, D., & Schölkopf, B. (2010). Identifying cause and effect on discrete data using additive noise models. In *Proceedings of the 13th international conference on artificial intelligence and statistics* (Vol. 9, 597–604). AISTATS. La Palma, Canary Islands.

PhysBAM. (2011). Retrieved 11 July 2011, from http://physbam.stanford.edu/

PhysX Features. (2010, May). Retrieved 7 May 2010, from http://developer.nvidia.com/object/physx_features.html

Pitt, M. K., & Shephard, N. (1999). Filtering via simulation: auxiliary particle filters. *Journal of the American Statistical Association*, *94*(446), 590–599. Retrieved from http://www.jstor.org/stable/2670179

Plungpongpun, K., & Naik, N. (2008). Multivariate analysis of variance using a Kotz type distribution. In *World conference on engineering* (pp. 1076–1081). Lecture Notes in Computer Science (LNCS) 2171. WCE. London, England. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.148.9654

Popovic, J. (2001). *Interactive design of rigid-body simulations for computer animation.* (PhD Dissertation, Carnegie Mellon University). PhD Thesis, School of Computer Science, Carnegie Mellon University.

Popovic, J., Seitz, S. M., Erdmann, M., Popovic, Z., & Witkin, A. (2000). Interactive manipulation of rigid body simulations. In *Proceedings of the ACM international conference on computer graphics and interactive techniques* (pp. 209–218). SIGGRAPH. New Orleans, LA, USA.

Pupilli, M., & Calway, A. (2005). Real-time camera tracking using a particle filter. In *Proceedings of the British machine vision conference* (pp. 519–528). BMVC. Oxford, England. Retrieved 24 August 2010, from `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.60.1137`

Pupilli, M., & Calway, A. (2006). Real-time camera tracking using known 3D models and a particle filter. In *Proceedings of the international conference on pattern recognition* (pp. 199–203). ICPR. Hong Kong. Retrieved from `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.152.8577`

Pylyshyn, Z. (2003). Return of the mental image: are there really pictures in the brain? *Trends in cognitive sciences*, *7*(3), 113–118.

Rasmussen, C., & Hager, G. D. (1998). Joint probabilistic techniques for tracking multipart objects. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 16–21). CVPR. Santa Barbara, CA, USA.

Rasmussen, C., & Hager, G. D. (2001). Probabilistic data association methods for tracking complex visual objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *23*(6), 560–576.

Reissell, L.-M., & Pai, D. K. (2001). Modeling stochastic dynamical systems for interactive simulation. *Computer Graphics Forum*, *20*(3), 339–348. doi:`10.1111/1467-8659.00526`

Richtsfeld, A., Mörwald, T., Zillich, M., & Vincze, M. (2010). Taking in shape: detection and tracking of basic 3D shapes in a robotics context. In *Computer vision winter workshop* (91–98). CVWW. Nove Hrady, Czech Republich. Retrieved from `http://cogx.eu/datogx/publications/richtsfeldCVWW2010.pdf`

Sachenbacher, M., & Struss, P. (2005). Task-dependent qualitative domain abstraction. *Artificial Intelligence*, *162*(1-2), 121–143.

Say, A. C. C., & Akın, H. L. (2003, October). Sound and complete qualitative simulation is impossible. *Artificial Intelligence*, *149*(2), 251–266. doi:16/S0004-3702(03)00077-8

Schwartz, D. L. (1999). Physical imagery: kinematic versus dynamic models. *Cognitive Psychology*, *38*(3), 433–464.

Schwartz, D. L., & Black, J. (1996). Analog imagery in mental model reasoning: depictive models. *Cognitive Psychology*, *30*(2), 154–219. PMID: 8660784. Retrieved 16 July 2011, from http://www.ncbi.nlm.nih.gov/pubmed/8660784

Segal, M., & Akeley, K. (2006, December). OpenGL 2.1 specification. Silicon Graphics. Retrieved from http://www.opengl.org/registry/doc/glspec21.20061201.pdf

Shabana, A. A. (2001, February). *Computational dynamics*. New York, NY, USA: Wiley-IEEE.

Shreiner, D., Woo, M., Neider, J., & Tom Davis. (2005). *OpenGL programming guide 5th edition*. Upper Saddle River, NJ, USA: Addison-Wesley.

Sim, R., Elinas, P., Griffin, M., & Little, J. J. (2005). Vision-based SLAM using the Rao-Blackwellised particle filter. In *International joint conference on artificial intelligence workshop on reasoning with uncertainty in robotics* (9–16). IJCAI-RUR. Edinburgh, Scotland.

Sim, R., Elinas, P., & Little, J. J. (2007). A study of the Rao-Blackwellised particle filter for efficient and accurate vision-based SLAM. *International Journal of Computer Vision*, *74*(3), 303–318.

Siskind, J. M. (2003). Reconstructing force-dynamic models from video sequences. *Artificial Intelligence*, *151*(1-2), 91–154.

Sloman, A. (2007). Architectural and representational requirements for seeing processes and affordances. In H. Dietmar (Ed.), *Computational modelling in behavioural neuroscience: closing the gap between neurophysiology and behaviour* (p. 303). London, England: Psychology Press.

Sminchisescu, C., & Triggs, B. (2003). Estimating articulated human motion with covariance scaled sampling. *International Journal of Robotics Research*, *22*(6), 371–393.

Sminchisescu, C., & Triggs, B. (2005). Building roadmaps of minima and transitions in visual models. *International Journal of Computer Vision*, *61*(1), 81–101.

Smith, B., & Casati, R. (1994). Naive physics: an essay in ontology. *Philosophical Psychology*, *7*(2), 225–244. Retrieved 15 July 2011, from `http://ontology.buffalo.edu/smith/articles/naivephysics.html`

Smith, S. M., & Brady, J. M. (1995). SUSAN - a new approach to low level image processing. *International Journal of Computer Vision*, *23*, 45–78. Retrieved from `http://citeseer.ist.psu.edu/viewdoc/summary?doi=10.1.1.24.2763`

Sonka, M., Hlavac, V., & Boyle, R. (1998). *Image processing, analysis, and machine vision*. Pacific Grove, CA, USA: PWS Publishing.

Sorenson, H. W. (1970). Least-squares estimation: from Gauss to Kalman. *IEEE Spectrum*, *7*, 63–68.

Sorenson, H. W., & Alspach, D. L. (1971). Recursive bayesian estimation using gaussian sums. *Automatica*, *7*, 465–479.

Stark, L., Hoover, A., Goldgof, D., & Bowyer, K. (1993, June). Function-based recognition from incomplete knowledge of shape. In *Proceedings of the IEEE workshop on qualitative vision* (pp. 11–22). New York City, NY , USA: IEEE. doi:`10.1109/WQV.1993.262954`

Stenger, B., Thayananthan, A., Torr, P. H. S., & Cipolla, R. (2006). Model-based hand tracking using a hierarchical bayesian filter. *IEEE Transations on Pattern Analysis and Machine Intelligence*, *28*(9), 1372–1384.

Stewart, D. E. (2000). Rigid-body dynamics with friction and impact. *SIAM Review*, *42*(1), 3. doi:`10.1137/S0036144599360110`

Stoianovici, D., & Hurmuzlu, Y. (1996). A critical study of the applicability of rigid-body collision theory. *ASME Journal of Applied Mechanics*, *63*, 307–316. Retrieved from `http://citeseer.ist.psu.edu/viewdoc/summary?doi=10.1.1.73.9464`

Stolkin, R., Greig, A., & Gilby, J. (2005). Video with ground-truth for validation of visual registration, tracking and navigation algorithms. In *Canadian conference on computer and robot vision* (pp. 210–217). CRV. Victoria, BC, Canada. doi:`10.1109/CRV.2005.86`

Struss, P. (2002). Automated abstraction of numerical simulation models-theory and practical experience. In *Sixteenth international workshop on qualitative reasoning*. Sitges, Catalonia, Spain.

Swain, M., & Ballard, D. (1990). Indexing via color histograms. In *Proceedings of the third international conference on computer vision* (pp. 390–393). ICCV. Osaka, Japan. Retrieved from `http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&isnumber=&arnumber=139558`

Tang, D., & Ngo, J. T. (1995). N-body spacetime constraints. *Journal of Visualization and Computer Animation*, *6*(3), 143–154.

Terdiman, P. (2001). Memory-optimized bounding-volume hierarchies [OPCODE: optimized collision detection]. Retrieved from `http://www.codercorner.com/Opcode.pdf`

Todorov, E. (2009). Parallels between sensory and motor information processing. In M. S. Gazzaniga (Ed.), *The cognitive neurosciences* (4th Ed.). Cambridge, MA, USA: MIT Press.

Todorov, E. (2011). A convex, smooth and invertible contact model for trajectory optimization. In *IEEE international conference in robotics and automation* (p. In Print). ICRA. Beijing, China.

Tordoff, B. J., & Murray, D. W. (2005). Guided-MLESAC: faster image transform estimation by using matching priors. *IEEE Transactions on Pattern Analysis and Machine*

Intelligence, *27*(10), 1523–1535. Retrieved 5 July 2010, from http://portal.acm.org/citation.cfm?id=1083982

Torr, P. H. S. (2002). Bayesian model estimation and selection for epipolar geometry and generic manifold fitting. *International Journal of Computer Vision*, *50*(1), 35–61. ACM ID: 598525. doi:10.1023/A:1020224303087

Torr, P. H. S., & Zisserman, A. (1999). Feature based methods for structure and motion estimation. In *Vision algorithms: theory and practice* (pp. 278–292). Lecture Notes in Computer Science (LNCS) 1883. International Workshop on Vision Algorithms. Corfu, Greece: Springer. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.37.1517

Torr, P. H. S., & Zisserman, A. (2000). MLESAC: a new robust estimator with application to estimating image geometry. *Computer Vision and Image Understanding*, *78*(1), 138–156. ACM ID: 344583. doi:http://dx.doi.org/10.1006/cviu.1999.0832

Toussaint, M. (2009). Probabilistic inference as a model of planned behavior. *Künstliche Intelligenz*, *3*(3), 23–29.

Triggs, B., McLauchlan, P. F., Hartley, R. I., & Fitzgibbon, A. W. (2000). Bundle adjustment: a modern synthesis. In *Vision algorithms: theory and practice* (pp. 153–177). Lecture Notes in Computer Science (LNCS) 1883. International Workshop on Vision Algorithms. Corfu, Greece: Springer.

Trucco, E., & Verri, A. (1998). *Introductory techniques for 3-D computer vision*. New Jersey, USA: Prentice Hall.

Tsai, R. (1987, August). A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-shelf TV cameras and lenses. *IEEE Journal on Robotics and Automation*, *3*(4), 323–344. doi:10.1109/JRA.1987.1087109

Tweed, D., & Calway, A. (2002). Tracking many objects using subordinated CONDENSATION. In *Proceedings of the British machine vision conference* (pp. 283–292). BMVC. Cardiff, Wales. Retrieved from http://www.bmva.ac.uk/bmvc/2002/papers/155/full_155.pdf

Ullman, S. (1983). Maximizing rigidity: the incremental recovery of 3-D structure from rigid and nonrigid motion. *Perception*, *13*, 255–274. Retrieved from http://citeseer.ist.psu.edu/viewdoc/summary?doi=10.1.1.18.3273

Vacchetti, L., Lepetit, V., & Fua, P. (2004a). Combining edge and texture information for real-time accurate 3D camera tracking. In *Proceedings of the third IEEE and ACM international symposium on mixed and augmented reality* (pp. 48–57). ISMAR. Arlington, VA, USA.

Vacchetti, L., Lepetit, V., & Fua, P. (2004b). Stable real-time 3D tracking using online and offline information. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *26*(10), 1385–1391.

van der Merwe, R., Doucet, A., Freitas, N. d., & Wan, E. (2001). The unscented particle filter. In *Advances in neural information processing systems* (pp. 584–590). NIPS 13. Denver, CO, USA.

van der Merwe, R., & Wan, E. (2003). Gaussian mixture sigma-point particle filters for sequential probabilistic inference in dynamic state-space models. In *Proceedings of the international conference on acoustics, speech, and signal processing* (Vol. 6, pp. 701–704). ICASSP. Hong Kong: IEEE.

Vermaak, J., Doucet, A., & Perez, P. (2003). Maintaining multi-modality through mixture tracking. In *Proceedings of the ninth IEEE international conference on computer vision* (pp. 1110–1116). ICCV. Nice, France: IEEE. Retrieved from www.irisa.fr/vista/Papers/2003_iccv_vermaak.pdf

Vlassis, N., Terwijn, B., & Krose, B. (2002). Auxiliary particle filter robot localization from high-dimensional sensor observations. In *Proceedings of the IEEE international conference on robotics and automation* (Vol. 1, pp. 7–12). ICRA. Washington, DC, USA: IEEE. Retrieved from http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1013331

Vondrak, M., Sigal, L., & Jenkins, O. C. (2008). Physical simulation for probabilistic motion tracking. In *Proceedings of the IEEE computer society conference on computer* *vision and pattern recognition* (pp. 1–8). CVPR. Los Alamitos, CA, USA: IEEE Computer Society. doi:http://doi.ieeecomputersociety.org/10.1109/CVPR.2008.4587580

Wang, Q., & Cellier, F. E. (1990). Time windows: an approach to automated abstraction of continuous-time models into discrete-event models. In *Proceedings of the conference on AI, simulation and planning in high autonomy systems* (204–211). Perth, Australia.

Weiss, Y. (1999). Segmentation using eigenvectors: a unifying view. In *Proceedings of the IEEE international conference on computer vision* (pp. 975–982). ICCV. Kerkyra, Greece. Retrieved from http://www.cs.huji.ac.il/~yweiss/iccv99.pdf

Weng, J., Huang, T. S., & Ahuja, N. (1987, May). 3-D motion estimation, understanding, and prediction from noisy image sequences. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *PAMI-9*(3), 370–389. doi:10.1109/TPAMI.1987.4767920

What is Newton Physics Engine? (2011). Retrieved 24 January 2011, from http://newtondynamics.com/forum/newton.php

Williams, C., Quinn, J., & McIntosh, N. (2006). Factorial switching Kalman filters for condition monitoring in neonatal intensive care. *Advances in Neural Information Processing Systems*, *18*(June), 1513–1520.

Witkin, A., & Baraff, D. (1997). Physically based modeling: SIGGRAPH 1997 course notes. Retrieved 24 August 2010, from http://www.cs.cmu.edu/~baraff/sigcourse/

Witkin, A., Baraff, D., & Kass, M. (1997). An introduction to physically based modelling. In *ACM international conference on computer graphics and interactive techniques*. SIGGRAPH. New Orleans, LA, USA. Retrieved from http://www.cs.cmu.edu/~baraff/pbm/pbm.html

Wolpert, D. M., & Kawato, M. (1998). Multiple paired forward and inverse models for motor control. *Neural Networks*, *11*(7-8), 1317–1329. ACM ID: 302597. doi:http://dx.doi.org/10.1016/S0893-6080(98)00066-5

Wren, C. R., Azarbayejani, A., Darrell, T., & Pentland, A. P. (1997). Pfinder: real-time tracking of the human body. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *19*(7), 780–785. doi:`10.1109/34.598236`

Wren, C. R., & Pentland, A. (1998). Dynamic models of human motion. In *Proceedings of the ieee international conference on automatic face and gesture recognition* (pp. 22–27). FG. Nara, Japan. doi:`10.1109/AFGR.1998.670920`

Yin, F., Makris, D., & Velastin, S. (2007, October). Performance evaluation of object tracking algorithms. In *10th IEEE international workshop on performance evaluation of tracking and surveillance*. PETS2007. Rio de Janeiro, Brazil. Retrieved from `http://dircweb.king.ac.uk/papers/Yin,%20Fei2007_841079/PerformanceEvaluationofObjectTrackingAlgorithms.pdf`

Young, G., & Chellappa, R. (1990). 3-D motion estimation using a sequence of noisy stereo images: models, estimation, and uniqueness results. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *12*(8), 735–759. Retrieved 28 April 2009, from `http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=00057666`

Young, G., Chellappa, R., & Wu, T. (1991). Monocular motion estimation using a long sequence of noisy images. In *Proceedings of the international conference on acoustics, speech, and signal processing* (pp. 2437–2440). ICASSP. Toronto, Ontario, Canada. Retrieved from `http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=150893`

Zabkar, J., Mozina, M., Bratko, I., & Demsar, J. (2011, June). Learning qualitative models from numerical data. *Artificial Intelligence*, *175*(9-10), 1604–1619. doi:`10.1016/j.artint.2011.02.004`

Zagal, J. C., Delpiano, J., & Ruiz-del-Solar, J. (2009). Self-modeling in humanoid soccer robots. *Robotics and Autonomous Systems*, *57*(8), 819–827. doi:`10.1016/j.robot.2009.03.010`

Zagal, J. C., & Ruiz-del-Solar, J. (2007). Combining simulation and reality in evolutionary robotics. *Journal of Intelligent and Robotic Systems*, *50*(1), 19–39. doi:10.1007/s10846-007-9149-6

Zhang, Z. (2000). A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *22*(11), 1330–1334. doi:10.1109/34.888718

Zhang, Z. (1998). *A flexible new technique for camera calibration* (Technical Report No. TR98-71). Microsoft Research. Retrieved from http://research.microsoft.com/en-us/um/people/zhang/Papers/TR98-71.pdfhttp://research.microsoft.com/en-us/um/people/zhang/Papers/TR98-71.pdf

Ziemke, T., Jirenhed, D.-A., & Hesslow, G. (2005). Internal simulation of perception: a minimal neuro-robotic model. *Neurocomputing*, *68*, 85–104. doi:10.1016/j.neucom.2004.12.005

☺