Eindhoven University of Technology

MASTER

Model-Based Visual Object Tracking with Collision Models

Jongeneel, M.J.

Award date:
2020

Link to publication

Disclaimer
This document contains a student thesis (bachelor's or master's), as authored by a student at Eindhoven University of Technology. Student theses are made available in the TU/e repository upon obtaining the required degree. The grade received is not published on the document as presented in the repository. The required complexity or quality of research of student theses may vary by program, and the required minimum study period may vary in duration.

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain

EINDHOVEN UNIVERSITY OF TECHNOLOGY

Department of Mechanical Engineering
Dynamics and Control section

Model-Based Visual Object Tracking with Collision Models

MSc. graduation project

M.J. Jongeneel (0909144)

DC 2020.024

Coaches:
dr. ir. A. Saccon — TU/e, Eindhoven
dr. A. Bernardino — IST, Lisbon, Portugal

Supervisor:
prof. dr. ir. N. van de Wouw — TU/e, Eindhoven

Committee:
prof. dr. ir. N. van de Wouw — TU/e, Eindhoven
dr. ir. A. Saccon — TU/e, Eindhoven
dr. A. Bernardino — IST, Lisbon, Portugal
dr. ir. G. Dubbelman — TU/e, Eindhoven

Eindhoven, March 17, 2020

Abstract

With a growing e-commerce market, warehousing companies could benefit from faster package handling to increase their throughput. Currently, handling these objects is mainly done by robotic systems, due to labor shortage. In these systems, high speed handling is limited, as robotic arms have to execute the complete trajectory of an object to move it around. To avoid controller complexity, picking and placing the objects happens at zero relative velocity of the robotic arm, limiting the throughput even more.

Tossing the objects is considered as a possible solution, resulting in faster object handling, and allowing robots to place objects outside their reach. However, by tossing the objects, they need to be followed in 3D space, to keep track of their location. Following the motion of an object is referred to as *object tracking*, and is essential for other robots to interact with the object.

Object tracking is an essential technique in computer vision and has many fields of application, such as robot navigation, sports, human-robot interaction, and object handling in logistics. In this study, the focus lies on the applications within logistics as described above. More specifically, in this thesis, an object tracking algorithm is presented, capable of tracking cuboids that, tossed from an initial position, can collide with surfaces. This involves tracking the 3D motion of the cuboid by using its 2D projection onto images of a single RGB camera. In this tracking process, the pose, i.e., the 3D position and orientation of the object, is estimated in each image of a video sequence.

The approach taken in this work, uses a newly proposed particle filter, named the *Geometric Unscented Particle Filter*. This filter is based on the unscented particle filter and makes use of prior knowledge on the shape and mass of the cuboid to describe its dynamical behavior, both during free flight as well as across the impacts. Existing methods use constant velocity models to describe the dynamical behavior, and thereby disregard the effects of gravity and impacts on the motion of the object. However, in this thesis, we provide a dynamical model accurate to the true dynamics of the cuboid, which limits the search area for the cuboid and significantly improves the overall tracking performance. The proposed method is compared to state-of-the-art methods in simulation, involving challenging motion sequences in which the cuboid makes multiple impacts before it comes to rest. The results show a major improvement, as the overall error in position and orientation is reduced up to 87% and 67%, respectively.

Preface

I first encountered the topic of object tracking during my internship, at Instituto Superior Técnico (IST) in Lisbon. There, I met Alexandre Bernardino, who introduced me to the world of computer vision. Although the topic was new to me, I quickly got inspired by the many enthusiastic researchers and students around me at IST. During my internship, I worked with simple collision models, with the goal to improve object tracking. I soon realized that there was much more potential in this project, as by accurately describing the dynamics of the object, the tracking performance would improve. This sparked the idea of continuing in this direction during my master thesis.

At the start of this project, I lacked the understanding of what it meant to introduce rigid-body dynamics to the world of object tracking. Not only did I have little knowledge on rigid-body dynamics itself, I also did not know that this was related to Lie group theory which, at that time, I had never heard of. With the help of Alessandro Saccon, I started to understand Lie groups, nonsmooth mechanics and what would be required to combine these topics with the filtering techniques that Alexandre taught me. The challenge that came, motivated me throughout the project and ultimately led to this thesis report.

Although I compiled this thesis on my own, I would not have been able to deliver this work if it wasn't for the exchange of ideas and discussions I had with the people around me. Therefore, I find it inappropriate to write this thesis in first person singular. Instead, we will continue in first person plural as we move through the argument. Before proceeding, I want to acknowledge the group of people that made this work come to light.

I want to thank Alexandre for his supervision, support and time. Both during my internship and graduation project he has been a source of inspiration. Without him, I would not have pursued my study in this direction. I owe great gratitude to Alessandro, who taught me all about rigid-body dynamics and Lie groups. Many hours we spent together to tackle the problems that turned up during this project. Even when time was limited, he always had a minute to help me. I want to thank Nathan van de Wouw for his support and supervision during this project, for reviewing this thesis, and providing helpful feedback.

I also want to thank my fellow students of the robotics lab, providing me with a nice work environment. Being able to share my struggles with them has made the process of this thesis much easier. I want to thank them for the inspiring discussions we had, but also for the fun activities and the friendships that have grown from that.

Finally, I want to thank my parents and sister. They have always motivated me to perform at my best. Their unconditional support and love is incredible, and for this, I am very grateful.

Maarten Jongeneel
Eindhoven, March 2020

Contents

Abstract i

Preface iii

Nomenclature ix

1 Introduction 1
- 1.1 Flexible systems for robotic part handling . . . 1
- 1.2 State-of-the-art in object tracking . . . 3
  - 1.2.1 Object detection . . . 3
  - 1.2.2 Object tracking . . . 9
- 1.3 Discussion . . . 10
  - 1.3.1 Object detection . . . 11
  - 1.3.2 Object tracking . . . 11
- 1.4 Problem definition . . . 12
  - 1.4.1 Object detection . . . 12
  - 1.4.2 Object tracking . . . 12
- 1.5 Structure of the report . . . 12

2 Mathematical Preliminaries 15
- 2.1 Object tracking using particle filters . . . 15
  - 2.1.1 Sequential Monte Carlo . . . 16
  - 2.1.2 Resampling . . . 17
  - 2.1.3 Choice of proposal distribution . . . 19
  - 2.1.4 Using an unscented Kalman filter to create a proposal distribution . . . 20
- 2.2 Lie groups: basic notation and probability distributions . . . 23
  - 2.2.1 Preliminaries on Lie groups . . . 23
  - 2.2.2 The Special Orthogonal group SO(3) . . . 25
  - 2.2.3 The Special Euclidean group . . . 25
  - 2.2.4 Gaussian distributions on Lie groups . . . 26
- 2.3 Group structure of the state . . . 28
- 2.4 Multibody notation for rigid-body dynamics . . . 28
  - 2.4.1 Notation of coordinate frames and points . . . 29
  - 2.4.2 Notation of rotational velocity . . . 30
  - 2.4.3 Notation of rigid-body velocity . . . 30
  - 2.4.4 Wrench notation . . . 31
- 2.5 Image formation . . . 32
  - 2.5.1 Perspective projection . . . 32
  - 2.5.2 Projection of the cuboid in the image plane . . . 33

2.6 Concluding remark . . . 34

3 The Geometric Unscented Particle Filter for Rigid Body Dynamics 35
3.1 Initialization of the particle filter . . . 35
3.2 Creating the sigma points . . . 36
3.3 Propagating the sigma points . . . 38
3.4 Obtaining a measurement . . . 39
3.4.1 Measurement from likelihood estimation . . . 39
3.4.2 Measurement from ground truth data . . . 40
3.5 Update of the state . . . 40
3.6 Computing the weights for each particle . . . 41
3.7 Final estimate . . . 41
3.8 Conclusion . . . 42

4 Applying the GUPF for Visual Tracking of a Cuboid 43
4.1 Geometric model of the cuboid . . . 43
4.2 Likelihood computation from color histograms . . . 44
4.3 Motion model with impacts . . . 47
4.3.1 Mechanical systems with unilateral constraints . . . 47
4.3.2 Signorini's contact law . . . 48
4.3.3 Newton's impact law . . . 50
4.3.4 Coulomb's friction law . . . 52
4.3.5 Numerical integration using time-stepping . . . 55
4.3.6 Fixed-point iteration for solving the nonlinear algebraic equations . . . 58
4.3.7 Conclusion on the motion model . . . 60
4.4 Conclusion . . . 60

5 Numerical Validation 61
5.1 Ground truth trajectories for simulation . . . 61
5.2 Measurement data . . . 63
5.3 Numerical evaluation . . . 65
5.3.1 Comparing the GUPF with the state-of-the-art . . . 66
5.3.2 Comparing the motion models: Results of the first scenario . . . 67
5.3.3 Comparing the motion models: Results of the second scenario . . . 70
5.4 Conclusion . . . 72

6 Conclusions and Recommendations 75
6.1 Conclusions . . . 75
6.1.1 Detection of the cuboid . . . 75
6.1.2 Tracking of the cuboid . . . 76
6.2 Recommendations . . . 77
6.2.1 Object detection . . . 77
6.2.2 Object tracking . . . 77
6.2.3 Validation by simulations and experiments . . . 78

Bibliography 79

A Behavior of the Likelihood Function **89**
A.1 Varying the x- and y-position of the cuboid 89
A.2 Varying the z-direction 91
A.3 Varying the orientation 92
A.4 Conclusion 93

B Group Operations for the Pose and States Lie Groups **95**
B.1 The group of the poses 95
B.2 The group of the states 96

Nomenclature

Groups, algebras, and sets

$\mathbb{R}$	The set of real numbers
$\mathbb{R}^+$	The set of non-negative real numbers
$\mathbb{R}^3_\times$	Vector representation of an element $\mathbf{x} \in \mathfrak{so}(3)$
SO(3)	The Special Orthogonal group in 3 dimensions
$\mathfrak{so}(3)$	The Lie algebra of SO(3)
SE(3)	The Special Euclidean group in 3 dimensions
$\mathfrak{se}(3)$	The Lie algebra of SE(3)
$\mathcal{A}$	The Lie group of the augmented states: $\mathcal{A} = \mathcal{S} \times \mathcal{S} \times \mathcal{P}$
$\mathcal{P}$	The Lie group of the poses: $\mathcal{P} = \mathrm{SO}(3) \times \mathbb{R}^3$
$\mathcal{S}$	The Lie group of the states: $\mathcal{S} = \mathcal{P} \times \mathfrak{p}$
$\mathfrak{a}$	Lie algebra of $\mathcal{A}$
$\mathfrak{p}$	Lie algebra of $\mathcal{P}$
$\mathfrak{s}$	Lie algebra of $\mathcal{S}$
$T_\mathbf{x}\mathcal{G}$	Tangent space of a Lie group $\mathcal{G}$ considered at $\mathbf{x} \in \mathcal{G}$
$\mathcal{I}$	The set of all contact points
$\mathcal{I}_c$	The set of all closed contact points
$\mathcal{C}_{Ti}$	The set of all admissible friction momenta of contact point i
$\mathcal{C}_N$	The set of all admissible normal momenta of all closed contact points
$\mathcal{H}$	The set of possible pixel values
$\mathcal{U}$	The set of all pixel points used for computation of a histogram

Roman symbols

A	World coordinate frame
$[A]$	Orientation frame associated to A
B	Body fixed frame, located at the center of mass
$\mathbb{B}_H$	Number of bins related to the *hue* of a pixel
$\mathbb{B}_S$	Number of bins related to the *saturation* of a pixel
$\mathbb{B}_I$	Number of bins related to the *intensity* of a pixel
$B[A]$	Frame with origin o_B and orientation $[A]$
$\mathbf{b}(\mathbf{d})$	Bin index at pixel location $\mathbf{d}$
C	Contact surface frame
D	Distance between histograms
$\mathbf{d}$	General location of a pixel in the image
$\mathbb{E}$	Expected value
$\boldsymbol{e}_N$	Diagonal matrix containing the coefficients of restitution in normal direction e_{Ni} of each closed contact point
$\boldsymbol{e}_{Ti}$	Coefficient of restitution of closed contact i in tangential direction

f	Focal length (as symbol) or motion model (as function)
${}_B\mathbf{f}$	Coordinates of the wrench $\mathbf{f}$ w.r.t. B
g_{Ni}	Gap-function of contact point i, denoting the distance in normal direction between the contact point and the contact surface
${}^A\mathbf{H}_B$	Homogeneous transformation from B to A
$\boldsymbol{\mathcal{H}}$	Pose parameters of the object, element of $\mathfrak{p}$
$\boldsymbol{H}$	Color histogram
$\mathbf{K}$	Camera intrinsic matrix
$\boldsymbol{K}$	Kalman gain
$L^{(i)}$	Likelihood assigned to particle i
${}_B\mathbb{M}_B$	6×6 inertia tensor expressed w.r.t. frame B
n	Dimension of the state
n_a	Dimension of the augmented state
n_c	Number of closed contacts
N	Number of particles
N_f	Number of visible faces
${}^A\mathbf{o}_B$	Coordinates of the origin $\mathbf{o}_B$ w.r.t. to A
$\mathbf{p}_i$	Contact point i
${}^A\mathbf{p}_i$	Coordinates of point $\mathbf{p}_i$ w.r.t. to A
$\mathbf{P}_N$	Momentum associated to the contact in normal direction
$\mathbf{P}_T$	Momentum associated to the contact in tangential direction
$\boldsymbol{P}^{\mathbf{x}}$	Covariance of the state, element of $\mathfrak{s} \otimes \mathfrak{s}$
$\boldsymbol{P}^{\mathbf{xy}}$	State-observation cross-covariance, element of $\mathfrak{s} \otimes \mathfrak{p}$
$\boldsymbol{P}^{\mathbf{yy}}$	Observation noise covariance, element of $\mathfrak{p} \otimes \mathfrak{p}$
$\boldsymbol{Q}$	Process noise covariance
$\mathbf{q}$	Projected point on the image plane
$\boldsymbol{R}$	Measurement noise covariance
${}^A\mathbf{R}_B$	Rotation matrix from B to A
r	Convergence parameter for proximal functions
S	Bhattacharyya similarity between two histograms
t	Global time
u_0	x-coordinate of the principal point in pixels
v_0	y-coordinate of the principal point in pixels
${}^B\mathbf{v}_{A,B}$	Twist expressing the velocity of B w.r.t. to A written in B
${}^B\mathbf{v}^{\wedge}_{A,B}$	4×4 matrix representation of ${}^B\mathbf{v}_{A,B}$
${}^B\mathbf{v}_{A,B}\bar{\times}^*$	6×6 matrix representation of the dual cross product
${}^B\boldsymbol{v}_{A,B}$	Linear velocity of B with respect to A, written in B
w	Weight of a particle
$\mathbf{W}_N$	Matrix containing the normal force directions of all closed contacts
$\mathbf{W}_T$	Matrix containing the tangential force directions of all closed contacts
W_j	Weight of the sigma point j
$\mathbf{w}^T_{Ni}$	Row vector containing the normal force directions of contact point i
$\mathbf{w}^T_{Ti}$	Matrix containing the friction force directions of contact point i
${}_B\mathbf{X}^A$	Wrench transformation from A to B
$\boldsymbol{\mathcal{X}}^{\mathbf{a}}$	Sigma point of the augmented state, element of $\mathfrak{a}$
$\boldsymbol{\mathcal{X}}^{\mathbf{x}}$	Sigma point of the state, element of $\mathfrak{s}$
$\boldsymbol{\mathcal{X}}^{\mathbf{m}}$	Sigma point of the process noise, element of $\mathfrak{s}$
$\boldsymbol{\mathcal{X}}^{\mathbf{n}}$	Sigma point of the observation noise, element of $\mathfrak{p}$
$\mathbf{x}$	State of the cuboid, written as element of $\mathcal{S}$
$\boldsymbol{x}$	General column vector

$\mathbf{Z}_t$	Measurement at time t, element of $\mathcal{P}$
$\mathcal{Z}_t$	Measurement at time t, element of $\mathfrak{p}$

Greek symbols

α	Focal length in terms of pixel coordinates
α_k	Scaling parameter of the unscented Kalman filter with $0 < \alpha < 1$
β	Normalizing constant
β_k	Scaling parameter of the unscented Kalman filter
$\boldsymbol{\gamma}_{Ti}$	Relative tangential velocity of contact point i
$\boldsymbol{\gamma}_N$	Contact velocities in normal direction of all closed contact points
Δt	Time interval of a time-step
δ	Dirac delta function
δ_k	Kronecker delta function
ϵ	Scaling parameter of the likelihood function
$\boldsymbol{\zeta}$	Scaling parameter of the unscented Kalman filter
$d\eta$	Atomic measure
θ	Angle to determine if a face of the cuboid is visible
κ	Weighting parameter for the distance function
$\boldsymbol{\lambda}_N$	Column of all normal forces
$\boldsymbol{\lambda}_T$	Column of all friction forces
$\boldsymbol{\Lambda}_N$	Column of all normal impulsive forces
$\boldsymbol{\Lambda}_T$	Column of all tangential impulsive forces
μ	Friction coefficient
$\boldsymbol{\rho}$	Innovation of the Kalman filter, element of $\mathfrak{p}$
$\boldsymbol{\Sigma}$	General covariance matrix
$\boldsymbol{\chi}$	General representation of the state in $\mathfrak{s}$
${}^B\boldsymbol{\omega}_{A,B}$	Angular velocity of B with respect to A, written in B

Special functions and operators

$\mathrm{Exp} : \mathfrak{a} \to \mathcal{A}$	Exponential map on $\mathcal{A}$ (same symbol for other Lie groups)
$\mathrm{Log} : \mathcal{A} \to \mathfrak{a}$	Logarithmic map on $\mathcal{A}$ (same symbol for other Lie groups)
$\mathrm{Sign}(x)$	The set-valued Sign function, $\mathrm{Sign}(x) = -1$ for $x < 0$, $\mathrm{Sign}(x) = 1$ for $x > 0$ and $\mathrm{Sign}(0) \in [-1, 1]$
$\mathcal{L} : \mathcal{P} \to \mathbb{R}$	Likelihood function. Computes the likelihood of a particle $\mathbf{x} \in \mathcal{S}$
$f : \mathcal{S} \to \mathcal{S}$	Nonlinear discrete time motion model
$h : \mathcal{S} \to \mathcal{P}$	Observation model
$\mathrm{prox}_{\mathcal{C}_N}$	Proximal function related to the set $\mathcal{C}_N$
$\mathrm{prox}_{\mathcal{C}_{Ti}}$	Proximal function related to the set $\mathcal{C}_{T,i}$
$\otimes$	Tensor product between two vector spaces
$\bar{\times}^*$	Dual cross-product on $\mathbb{R}^6$
$\lvert \cdot \rvert$	Absolute value
$\lVert \cdot \rVert$	Euclidean norm
$\mathrm{tr}(\cdot)$	The trace of a $n \times n$ matrix, sum of its diagonal elements
$(\cdot)^T$	Transpose
$\circ$	Lie group operator (exact definition depending on the Lie group structure)
$\boldsymbol{x} = \mathrm{col}\{x_i\}$	Column-vector $\boldsymbol{x} = \begin{bmatrix} x_1 & x_2 & \dots & x_n \end{bmatrix}^T$

Subscripts and superscripts

$(\cdot)^{\vee}$	'Vee' operator: matrix to vector representation of algebra element
$(\cdot)^{\wedge}$	'Hat' operator: vector to matrix representation of algebra element
$(\cdot)_N$	Related to the normal direction of the contact surface
$(\cdot)_T$	Related to the tangential direction of the contact surface
$(\cdot)_i$	Related to contact point i
$(\cdot)^{(i)}$	Related to particle i
$(\cdot)^-$	The left limit
$(\cdot)^+$	The right limit
$(\cdot)^{(m)}$	Related to the mean
$(\cdot)^{(c)}$	Related to the covariance
$\dot{(\cdot)}$	First time derivative
$\hat{(\cdot)}$	Estimated variable
$\bar{(\cdot)}$	Mean value
$\tilde{(\cdot)}$	Sampled variable
$(\cdot)_{t-1}$	Variable at time $t-1$
$(\cdot)_t$	Variable at time t
$(\cdot)_{t\|t-1}$	Variable at time t, given the information at the previous time step
$(\cdot)_{obj}$	Related to the object (the cuboid)
$(\cdot)_{cin}$	Related to the inside of the contouring edges of the cuboid
$(\cdot)_{cout}$	Related to the outside of the contouring edges of the cuboid
$(\cdot)_{ref}$	Related to the reference image
$(\cdot)_a$	Variable at time t_a, the beginning of a time step
$(\cdot)_m$	Variable at time t_m, the mid-point of a time step
$(\cdot)_e$	Variable at time t_e, the end-point of a time step

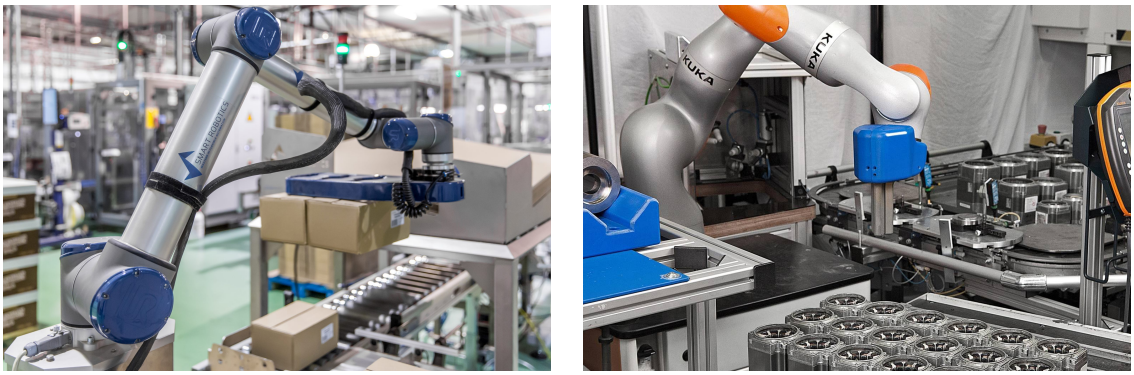
Chapter 1

Introduction

1.1 Flexible systems for robotic part handling

Warehousing companies are continuously looking for ways to run their operations more efficiently due to an ever growing e-commerce market and shortage of labor. As a result, they demand the development of robotic solutions for object handling, including pick-and-place solutions. These are common operations in both industrial setups, such as assembling or order picking as well as in domestic environments, and in many fields, this already is a mature process [1]. Examples can be found in sorting or packaging applications, where robots execute repetitive tasks. The objects with which they interact are located at known positions with respect to the robot, allowing for predefined trajectories of the robot's end-effector. This is not the case for robots operating in more complex environments, encountering varying scenarios, of which two examples are shown in Figure 1.1. In Figure 1.1a, a robot has to handle packages that come in different sizes, and in Figure 1.1b the robot has to pick up items from a cart placed by an operator. In those cases, a lack of robustness, efficiency, and flexibility limit the performance, while the market demands for more flexible systems that work with increased speed. This requires the development of systems with robust environment perception, trajectory planning, and flexible grasping strategies.

An example of such systems is the SIR (Smart Item Robotics) solution presented by Vanderlande [4], shown in Figure 1.2. In this system, various items can be picked by a robot, allowing for bin-to-bin and bin-to-belt tasks, where by the use of perceptive sensors, the



(a) UR10 robot [2] **(b)** KUKA LBR [3]

Figure 1.1: Examples of robots working in complex environments: A robot manipulating cuboids with varying sizes (a) and a robot picking objects from a cart (b).

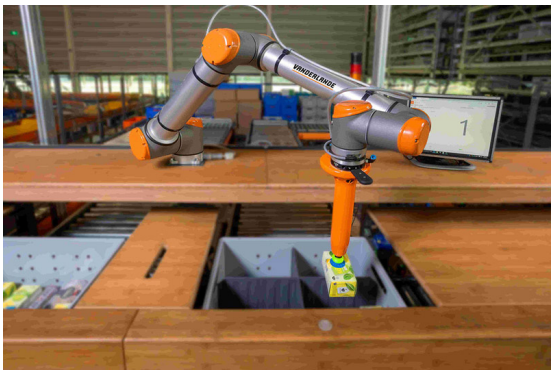


Figure 1.2: Smart Item Robotics presented by Vanderlande.

robot is able to locate and identify objects. Even though this provides a solution towards a more flexible work environment, operating speed is still limited as the robot has to execute the complete trajectory to move a part from one position to another.

A solution to speed up this process is tossing the objects onto the target location. Tossing objects is not only faster than manipulating the object, it also allows to place objects outside the robot's reach. Within logistic applications, this could reduce the number of robots needed to distribute items into various bins and increase throughput on lines [5]. However, one of the challenges that comes into play, is that these type of systems require continuous localization of the objects in the 3D world, to allow other robots to interact with the object.

For humans, this ability to localize objects in the 3D world is something very natural. Even when objects are moving with complex dynamics, such as balls bouncing off the cushions of a pool table, we are able to follow their movement. This is not the case for robotic systems, where following the movement of objects is often referred to as *object tracking*. In a generic setup, a video camera is used where the task of object tracking is to locate the object in each frame of the video. A priori information about the object, the environment, and the camera being used, is needed to create a mapping between the location of the object in the image and in the 3D world.

In Section 1.2, existing object tracking methods are discussed. It will become clear, that many of these methods focus on small changes in the object's *pose*, i.e. its position and orientation, from frame to frame. The complex dynamics involved with tossing objects, such as gravity, impacts, and friction, are neglected. Especially the presence of impacts and friction have a high influence on the object's trajectory as they introduce *nonsmooth* behavior, that is, they introduce velocity jumps. This makes it hard for existing tracking methods to follow objects subjected to these type of dynamics.

This study focuses on filling that gap in literature, by developing an object tracking algorithm able to follow objects which, after being tossed, can collide with surfaces. More specifically, the focus will be on tracking cuboids, which is a common shape for objects within the logistic applications described above. The remainder of this chapter focuses on state-of-the-art object tracking methods, discussed in Section 1.2, and their strengths and weaknesses are reviewed and classified in Section 1.3. Based on this classification, a research goal is formulated in Section 1.4.

(a) (b) (c)

Figure 1.3: Feature based tracking of [12]. Grey-scale image from the camera, with a tracking block placed around the target (a), cut-out of the target object (b), and set of feature pixels obtained from the target object (c). Images taken from [12].

1.2 State-of-the-art in object tracking

Object tracking, in which the goal is to follow the movement of objects in the 3D space, can be seen as a subclass within computer vision. It has various applications such as robot navigation, surveillance, and computer-human interaction and nowadays research in this field is quite mature [6]. A review paper which gives a good overview on the whole topic is given by Yilmaz et al [7].

There are two key steps in tracking objects: *detection* of the object and *tracking* the object. In object detection, the pose of the object is estimated using image data only. Object tracking can then be defined as maintaining an estimate of the object's state along time. Numerous approaches for object tracking and object detection have been proposed. In the following sections, some of these methods are reviewed.

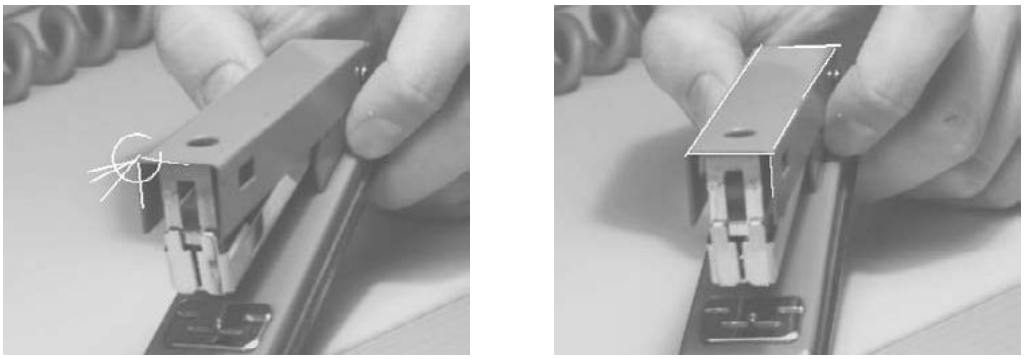
1.2.1 Object detection

In general, the problem of object detection can be divided in two categories: *feature-based methods* and *model-based methods* [8–10]. The former method is solely based on 2D information of the image and tracks object features, such as object contours, points, lines, or colored regions. The latter explicitly uses a geometric model of the target object, which provides 3D information of the object shape. In the following sub-sections, both methods will be discussed.

1.2.1.1 Feature-based methods

Feature-based methods focus on detection of the object by the use of object features, such as edges, points, and colors. Some of the advantages of feature-based tracking are, that it is computationally efficient and relatively easy to implement [11]. Therefore, this method is widely used and various papers have been published based on this method.

In [12], an edge-based method is used, where a histogram is created from the image of the object to be tracked. After initialization of the object in the image, a bounding box is placed around the object as shown by Figure 1.3a. A cut-out of this bounding box is shown in Figure 1.3b. By the use of edge-detection and thresholding, a binary image of the target object is



(a) **(b)**

Figure 1.4: Feature based tracking of [16]. Edges found on the image (a) and top surface lines found (b). Images taken from [16].

created as shown in Figure 1.3c. From this binary image, a 2D histogram is computed which acts as a reference histogram of the object. In this histogram, the pixel values (zero or one) of the reference image (Figure 1.3c) are stored. By comparing the pixels in a search window with the object-histogram, the object can be detected in consecutive video frames. Similar approaches are taken in [13–15], where the detection algorithm is based on feature-matching.

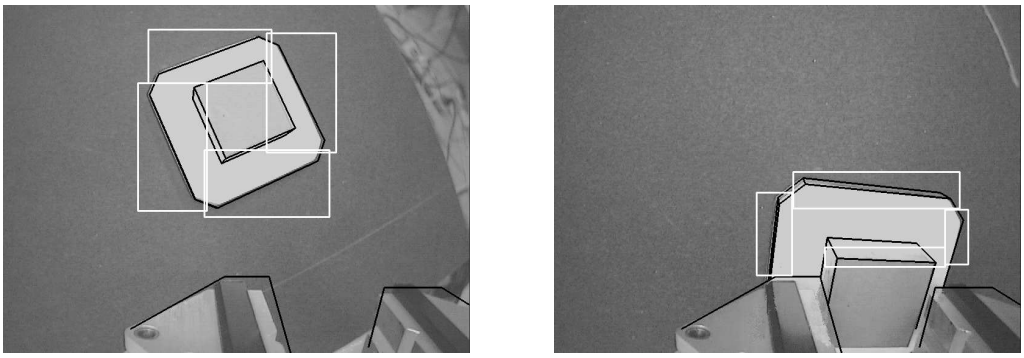
Another, slightly different method, is that of Vincze et al. in [16]. They use a wire-frame database containing the features of the object. This database contains lines, corners and circles. These features are then mapped to edges found in the image, to locate the object in the image plane. This is done by an iterative process of matching features, resulting in a final estimate of the object's pose. Figure 1.4 shows the steps of the algorithm after the fourth iteration (Fig.1.4a) and after the 19th iteration (Fig.1.4b).
Other feature-based tracking methods use background subtraction [17], color [18], or the combination of both [15]. To increase robustness, various features are often used simultaneously [16]. This could be, for example, the combination of edge-detection and optical flow [19], or edge-detection and color [20].

A downside to feature-based tracking is that it is often only applicable for 2D tracking. In other words, the object can be followed in the image plane but its 3D position remains unknown, as is also the case in the papers discussed above. In order to take the step to 3D tracking, a *model-based* approach is often taken. This will be described in the next section.

1.2.1.2 Model-based methods

In model-based methods, 3D geometric information of the target object is used to detect the object. Model-based algorithms are often more robust [8, 10, 16, 21], as they allow for a combination of geometric and dynamic models. The geometric model of the object can be used to cope for partial occlusions and pose estimation. The dynamic model can, at the same time, be used to predict the location of the object for each time step, and thereby limiting the search area and therefore computation time.

Several works have been published using a model-based approach. Wunch and Hirzinger [22], for example, use a model-based method to reconstruct the object pose in the image. After a 3D prediction of the pose is obtained, a local search is executed on the projection of that predicted object in the image plane. A hidden-line removal method, as proposed in [23],



(a) **(b)**

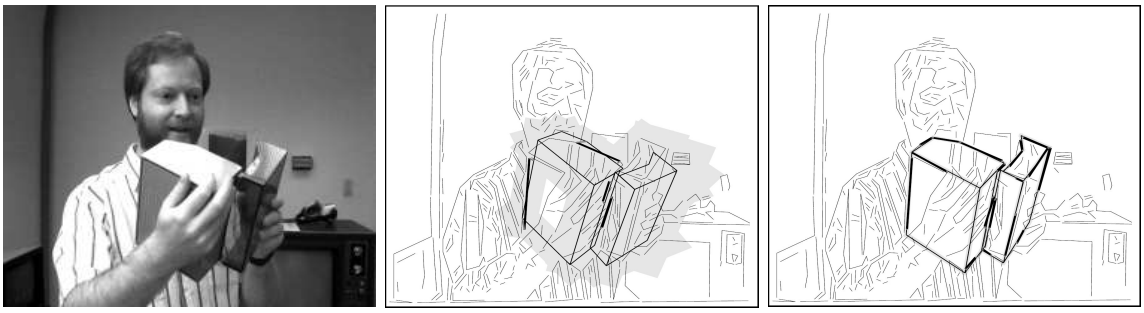
Figure 1.5: Sample frames of the sequence used when the object is fully shown (a) and partially occluded by the gripper (b). Images taken from [22].

is then used to account for occlusions, after which the Hough transform with a limited parameter space is used to detect the contours of the object. Figure 1.5 shows the results of their algorithm. In Figure 1.5a, the estimated pose of the object is shown with black lines. In the bottom part of the image, a gripper of a robot arm is shown, which is partially modeled as well. The white boxes show the regions of interest used for feature extraction. Figure 1.5b shows the handling of occlusions, as the object is partly occluded by the gripper.

In [24], Lowe uses an edge detector to extract line segments from the image. An iterative matching procedure is then used, together with a probability model, to minimize the match error and only keep those line segments which most likely fit the model. Figure 1.6a shows the input image of the algorithm and Figure 1.6b consequently shows the results after the first iteration. The heavy line segments show the first matches with the image, while the shaded area shows the uncertainty for the feature locations. The model is shown as well, superimposed from its initial position. Figure 1.6c shows the result of the tracking algorithm after the third and final iteration, in which the uncertainty is reduced significantly. However, due to the high computational cost of the iterative procedure, special purpose hardware was needed to achieve frame-rate processing.

Similar approaches to the methods discussed above are also presented in [25] and [26]. In the former, the pose of the object is estimated in the image plane from an initialization. A search for contouring edges is then executed in the image plane to detect the object. Between frames, small changes in the object pose in 3D are assumed. In the latter, a similar approach is taken where an initial estimate of the pose of the object is projected in the image. Next, the image is searched for evidence of nearby edges. This search is executed along the normals of each projected boundary line and small changes on the model are assumed that move each boundary from frame to frame.

A different approach is taken in [27] and [28]. In these works, the 3D pose of a cuboid is estimated by projecting hypothesized states (3D poses) of the object onto the image, and evaluating the projected features in the image plane. This evaluation is then done by the use of a likelihood function. In [27] and [28], for example, this likelihood function, based on color histograms, is used to evaluate the projected features in the image. For a given pose, contouring points of the object are projected on the image and a color histogram is created from their corresponding pixel points. Figure 1.7 shows the 3D model used in [28], where the blue and red points are both used to compute different color histograms. A downside to



(a) (b) (c)

Figure 1.6: Input image of the algorithm (a), the result of the algorithm after the first iteration (b), and after the third and final iteration (c). Images taken from [24].

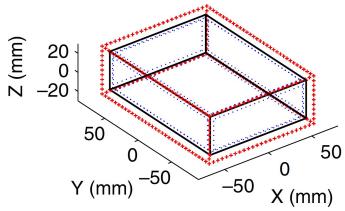


Figure 1.7: 3D model of the cuboid. Image taken from [28].

this method is that the hypothesized states need to stay close to the true state of the cuboid in order to detect the object in the image.

1.2.1.3 Alternative methods

Besides feature based or model based methods, alternative detection methods exist. One of these methods makes use of *depth cameras*, also called RGB-D cameras, which provides information about the distance of objects from the camera. In [29], for example, Min et al. use a Microsoft Kinect camera to detect humans. In their system, two cameras are used; one conventional RGB camera that provides a color image, and a second depth camera, providing a so-called *depth image*. Figure 1.8 shows the resulting images of the setup used in [29], where Figure 1.8a shows the RGB image and Figure 1.8b shows the corresponding depth image. In their studies, Min et al. use the depth information to extract the 3D position of

(a) (b)

Figure 1.8: Example of using a depth camera. RGB image (a) and its corresponding depth image (b). Images taken from [29].

a person. The depth information is, in their case, also used to distinguish the object from the background. In [30], Nakamura uses both depth information from a depth camera and color information from a RGB camera to create a "depth color image". Color features of the object are used to locate the object in the image, while the depth information is used to link this detection to its 3D pose. A similar approach is also taken in [31], [32], and [33]. Although these methods provide a good estimation of the object in 3D space, their performance is often limited due to a limited depth range of the camera and high sensitivity to daylight outdoors.

Other studies, for example [34] and [35], make use of neural networks to detect the object. In the latter study, Dwibedi et al. are able to find cuboids inside a 2D image and localize their vertices. By the use of perspective projection methods, the 3D position can then be estimated. Although these methods are quite robust, they require a lot of training data in order for the neural network to work effectively.

1.2.1.4 Challenges in object detection

Several challenges can affect the robustness of the detection algorithm. Below, we will discuss occlusions, illumination changes, and motion blur, which are three of the main challenges in object detection.

Handling of occlusions. One of the main difficulties in object tracking is handling occlusions of the object. There are two types of occlusions; self-occlusions and external occlusions. In the first case, certain parts of the object are occluded by the object itself such that they are not visible from the camera, whereas in the second case the object is occluded by other objects.

In [22], Wunsch and Hirzinger deal with self-occlusions by the use of an efficient hidden-line removal algorithm from [23]. This algorithm uses a 3D CAD model of the object together with the camera parameters to compute which parts of the object are visible from the camera for a given pose of the object. Only the potentially visible features of the objects are then searched for in the image. Similar approaches are also taken in [25, 36–38] and [39]. In the later study, however, an additional step is taken in which outlying edges are rejected when insufficient support is found for them. This allows to cope for external occlusions as well, as is shown in Figure 1.9.

The works discussed above are detection algorithms that use line segments as features to track the object. In [40], however, a detection mechanism is used based on color histograms, as described in [41]. A likelihood function is used, which is directly related to the appearance of the object in the image plane. Occlusions and reappearance events are detected by a threshold on the likelihood function. A detection process is run every time the likelihood drops below its threshold, hence when the object is occluded. This detection process detects the edges in the image to locate where the object will reappear. A controller, used from [42], is able to estimate the velocity of the object in world coordinates, based on on-line learning of the target dynamics. The combination of the two methods allows to know when, where, and with what velocity the object will reappear. In [43], this method is also praised for its robustness for occlusions. Due to occlusions, some features as lines may disappear which highly affect the appearance of the object. Color histograms however, only change little under occlusion, and allow to keep track of the object even when it is partially occluded.

(a) (b)

Figure 1.9: Handling external occlusions. Detecting a cuboid without occlusions (a) and with occlusions (b). Images taken from [39].

Illumination changes. Another challenge is to be invariant under illumination changes. Dealing with illumination changes can especially be difficult for feature based methods. Edge-based methods, for example, are able to locate edges in an image based on contrast. However, intensity changes in an image can have a major influence on the contrast levels and therefore highly affect the detection of edges. Several approaches have focused on combining features, such as grouping data features as edges and points, to cope for illumination changes [20], but this remains a sensitive method.

Another popular method to increase robustness is by the use of RANSAC [44]. In [16], RANSAC is used to find the most likely line from a set of input edgels [1]. In [39], the same method is used to detect outliers among edges, and in [45], the pose estimation of the object is robustified by the use of RANSAC.

Detection methods that make use of color cues have shown greater robustness to illumination changes. Examples of works that make use of color cues are [27, 28, 33, 46–48]. These methods make use of the HSV color space, in which the intensity (i.e. value) is decoupled from the color (i.e. hue and saturation). This decoupling allows to keep track of the color cue, even if the illumination of the object changes. In most of these papers, a reference color histogram of the object is created which is used as a model throughout the tracking process. In [49], Stern et al. use an adaptive color space method. As their goal is to separate the target object from the background as good as possible, their algorithm iteratively searches for the best color space model that fulfills this task. However, even though their adaptive approach showed improvement over existing methods using a single color space model, it turned out that their algorithm used the HSV color space 86% of the time, indicating that this color space model is significantly more effective than other color space models.

Motion blur. Motion blur can occur when an object has a relatively high velocity with respect to the video frame rate of the camera. As a result, the representation of the object in the image plane is blurred out over the image. One can imagine that motion-blur can be an obstacle for exact localization of the object. Especially tracking methods based on edge-detection tend to fail detecting the object when the object appears blurred in the image [50]. Where object features like edges or lines disappear when the object is blurred, color features remain. This is why tracking algorithms that make use of color histograms have proven to

[1] An edgel is a pixel in the image that is found to be the edge of something.

be very robust to motion blur [28, 51, 52].

1.2.2 Object tracking

In the previous section, object detection was discussed, which covers the topic of estimating the pose of the object in each frame of a video. In the latter, we will refer to this as the *observation*, indicating what the system observes at a given time instant. Tracking the object is then referred to as maintaining an estimate of the objects's state along time, based on these observations. Although it seems natural to directly use these observations from frame to frame to track the object, this will result in poor tracking performance. Due to several challenges, of which some were discussed in Section 1.2.1.4, the observations invariably contain noise, which highly affect the overall tracking performance. To this end, filtering techniques are used, and in this section, some of the most commonly used filtering techniques are discussed.

The goal of tracking is to estimate the state $\mathbf{x}_t$ at time t, given all the observations up to that point $\mathbf{y}_{1:t}$. This estimation is represented by the probability distribution $p(\mathbf{x}_t|\mathbf{y}_{1:t})$, also known as the *belief* of the system, which reflects the probability of that state. The *prediction* of the system is the distribution of the state $\mathbf{x}_t$ at time t, given all the observations but the current one. This distribution reflects the probability of the current state before incorporating the current observation. The prediction uses the belief of the previous time step and a *motion model* to derive the prior of the current state, that is, $p(\mathbf{x}_t|\mathbf{y}_{1:t-1})$. Computation of the *belief* from the *prediction* is then done by the *update* step, which employs the *likelihood* $p(\mathbf{y}_t|\mathbf{x}_t)$ of the current observation.

1.2.2.1 Filering techniques

One of the most commonly used filters is the *Bayes Filter*, which computes the belief, formally known as the *filtering density*, in a recursive way. When the variables are normally distributed and the state transition is linear, the Bayes filter becomes equal to the well-known *Kalman Filter*. The Kalman filter is a widely used filter within computer vision and in its pure form used in for example [39] and [25]. However, application of the Kalman filter is limited to systems with linear state transitions. To this end, the *Extended Kalman Filter* is often employed when the state transition shows non-linear behavior. This filter is, for example, used in [22, 53–56] and there proven to be an effective filtering technique for object tracking. Like the ordinary Kalman filter, the extended Kalman filter also assumes the state variables to be Gaussian. As a result, these filters will give poor estimations for non-Gaussian state variables. In such cases, the step is often taken towards the *Unscented Kalman Filter* (UKF) or the *Particle Filter* (PF). In the unscented Kalman filter, the state is approximated by a chosen set of deterministic points, and we discuss this filter in detail in Section 2.1.4. In the particle filter, further discussed in Section 2.1.1, the filtering density is approximated by a set of weighted states, also known as particles. Each particle is seen as a hypothesis of the state estimate. Therefore, the particle filter is known to be more robust, as it maintains several hypotheses over time. It is therefore widely used in literature, and some examples can be found in [41, 46, 57, 58].

1.2.2.2 Motion models

As stated in [11], a downside to the particle filter is that often a large number of particles is needed, especially when the motion of these particles is poorly defined. The motion of

these particles is defined in so-called *motion models*, which describe the state transition of each particle. In case of 3D tracking, this motion model describes the state transition in 3D, often done by the use of a discrete-time model described by

$$\begin{aligned} \mathbf{x}_{t+1} &= f(\mathbf{x}_t) + \mathbf{m}_{t+1}, \\ \mathbf{y}_{t+1} &= h(\mathbf{x}_{t+1}) + \mathbf{n}_{t+1}, \end{aligned} \tag{1.1}$$

where $f(\cdot)$ describes the 3D motion of the object and $h(\cdot)$ is the function that links the object pose to its appearance in the image. Furthermore, $\mathbf{m}$ and $\mathbf{n}$ are Gaussian noises with zero mean and predefined covariance matrices. The motion model $f(\cdot)$ is often modeled as a constant velocity model, such that the state vector $\mathbf{x}_{t+1}$ is obtained by

$$\mathbf{x}_{t+1} = \mathbf{A}\mathbf{x}_t + \mathbf{B}\mathbf{m}_{t+1}, \tag{1.2}$$

where the matrices $\mathbf{A}$ and $\mathbf{B}$ are given by

$$\mathbf{A} = \begin{bmatrix} \mathbf{I} & (\Delta t)\mathbf{I} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \quad \text{and} \quad \mathbf{B} = \begin{bmatrix} \frac{(\Delta t)^2}{2}\mathbf{I} & \mathbf{0} \\ \mathbf{0} & (\Delta t)\mathbf{I} \end{bmatrix}, \tag{1.3}$$

where $\mathbf{I}$ denotes the identity matrix and Δt the step size. The sizes of both $\mathbf{I}$ and $\mathbf{0}$ depend on the size of the state. In the works of [8, 53, 55, 59], for example, this state contains the 3D position, orientation, linear velocity, and angular velocity, thereby making the state 12-dimensional.

The constant velocity model is sufficient for free moving objects, but in many cases, the dynamics of the object are more restricted [55]. Hereto, the dynamic model should be imposed by the true motion of the object to improve the tracking performance. Although some studies show improvements over the constant velocity model, such as learning the dynamics on the fly [60], all of these models assume the motion is smooth with no abrupt changes [7]. The complex dynamics involved in our specific case of tossing objects, such as friction and impacts, that lead to *nonsmooth* behavior, are neglected in the literature.

1.2.2.3 Challenges in object tracking

As stated in Section 1.2.2.2, the tracking performance can be highly affected by poor estimation of the object's motion. In our specific case of tracking an object subjected to impact and friction, this gives rise to several challenges. Especially the abrupt changes in velocity that occur upon impact of the object, may lead to failure of the tracking algorithm. To this end, it is important to precisely model this nonsmooth behavior. Besides, a small error in estimation of the impact time can lead to drastically different estimation of the trajectory. The particle filter may resolve some of these problems by spreading the particles across the state space. Furthermore, a well known extended version of the particle filter, the Unscented Particle Filter (UPF) [61], may further increase robustness. This filter uses the observation of each frame to move particles closer to the true state, thereby increasing tracking performance.

1.3 Discussion

In this section, we will reflect upon the literature presented in Section 1.2 by providing a short summary for both object detection and object tracking, and motivate our approach.

1.3.1 Object detection

Some works presented in Section 1.2.1.3 make use of depth cameras. Although this type of cameras provide additional information to the user, it also requires a lot of extra computation as the depth images need to be synchronized with the RGB images. Furthermore, depth cameras only work indoor, close range, and on non-reflective surfaces. Here, the choice is made to use a single RGB-camera, due to its simplicity. The works presented in [27, 28] for example, have shown that a single RGB-camera can be sufficient for accurate tracking in 3D.

As discussed in Sections 1.2.1.1 and 1.2.1.2, a main division can be made between feature-based and model based tracking algorithms. The feature-based methods are shown to be effective and computationally efficient, but are mostly focused on localizing the object in the image plane. Model-based methods on the other hand, enable the localization of the objects in the 3D space. These type of algorithms come with extra information about the geometry of the target object, making the algorithm more robust for occlusions. To this end, in this project a model-based algorithm will be used.

Using a RGB-camera still allows for a wide use of observation models. Some of these models are more robust than others, as can be concluded from Section 1.2.1.4. An interesting conclusion from this section can be that, for all discussed challenges, the use of color cues as an observation model seems to be the most robust method. The works of [40] and [43] for example, show effective use of color histograms to cope for occlusions. Handling illumination changes can be done very effectively with the HSV color space, as was demonstrated in [27, 28, 33, 46–48]. Section 1.2.1.4 also shows that color histograms are the most effective method in dealing with motion blur. To this end, a color histogram model will be used for detection of the object.

1.3.2 Object tracking

As a tracking algorithm, we will adopt a method based on particle filtering. This method is widely used in literature and has proven to be an effective tracking technique [47, 58, 62–64]. More specifically, we will use the Unscented Particle Filter of [61], as this methods has proven to outperform the standard particle filter. The advantage of particle filters is that they are easy to implement, scalable, and are able to handle non-Gaussian state distributions. This last point is very important, as in case of impacts the state transition is non-linear. Furthermore, the particle filter may cope for the inaccurate estimation of the impact time, due to the use of multiple hypotheses of the state estimate.

The state transition from frame to frame, describing the motion of the object, can be done in several ways. The literature in Section 1.2.2.2 shows that most methods make use of constant-velocity models. These models are efficient when free moving objects are tracked, but if the motion of an object is more restricted, different type of models should be used [55]. Since the objective of this project is to track objects that are subjected to gravity, impacts, and friction, these phenomena should be included in the motion model. To this end, a geometric approach will be taken to model the dynamics. Examples in literature that use this geometric approach, can be found in [37, 65–68] for particle filters, or [69, 70] for the unscented Kalman filter. In these works, the state transition is described on SE(3), which is a matrix Lie group. However, to our knowledge, no literature exists in which the unscented Particle filter is used on the Lie group structure. To this end, this research aims to fill that

gap in literature by generalizing the UPF to Lie groups.

1.4 Problem definition

The objective of this thesis is to *develop an algorithm* that is able to track an object subjected to complex dynamics, involving gravity, friction, and impacts, by the use of a single RGB camera. The objective is divided into two main tasks; detection of the object and tracking the object. In the following sections, these tasks are further explained.

1.4.1 Object detection

Object detection focuses on *localizing the object* in each image of a video. The assumption is made that a single RGB camera will provide these images at a frequency of 60Hz. A mapping should be created that is able to link the localization of the object in the image to the *state* of the object, that is, its position, orientation, linear velocity and angular velocity in the 3D space. More specifically, the object to be tracked will be a cuboid, and the assumption is made that its shape, mass, and colors are known. Therefore, the research objective related to object detection is stated as:

Estimate the state, i.e., the pose and velocity, of the cuboid in 3D space, based on images from a RGB camera.

1.4.2 Object tracking

Object tracking focuses on *following the motion* of the cuboid and *predicting* the current state of the cuboid based on its previous state estimate. The assumption is made that an estimation of the initial state is given. This could be at the end-effector of a robotic arm, which tosses the cuboid. The cuboid will then be subjected to gravity and collide multiple times with a surface before it comes to rest. The assumption is made that the location of this surface is known, and the coefficients of friction and restitution are given. The goal is then to develop a *motion model* that describes the behavior of the cuboid both during free flight as well as across the impacts in order to accurately predict the state of the cuboid. To this end, the research objective related to object tracking is stated as:

Develop a tracking algorithm that, based on the estimate of the previous state, gives a prediction of the current state of the cuboid.

1.5 Structure of the report

This report has the following structure. The rigid-body dynamics and the background material on Lie groups is discussed in Chapter 2. Furthermore, this chapter covers the derivation of the standard unscented particle filter and provides background material on image formation. The generalization of the UPF to Lie groups leads to the Geometric Unscented Particle Filter, which is described in Chapter 3. The key elements of this filter, the geometric model, likelihood function, and the motion model, are then further detailed in Chapter 4, where we discuss the specific application of tracking cuboids. The resulting tracking algorithm is then evaluated in Chapter 5 by the use of simulations. Based on the results of these simulations, conclusions are drawn and recommendations for further research are given in Chapter 6. A graphical representation of the thesis outline can be seen in Figure 1.10.

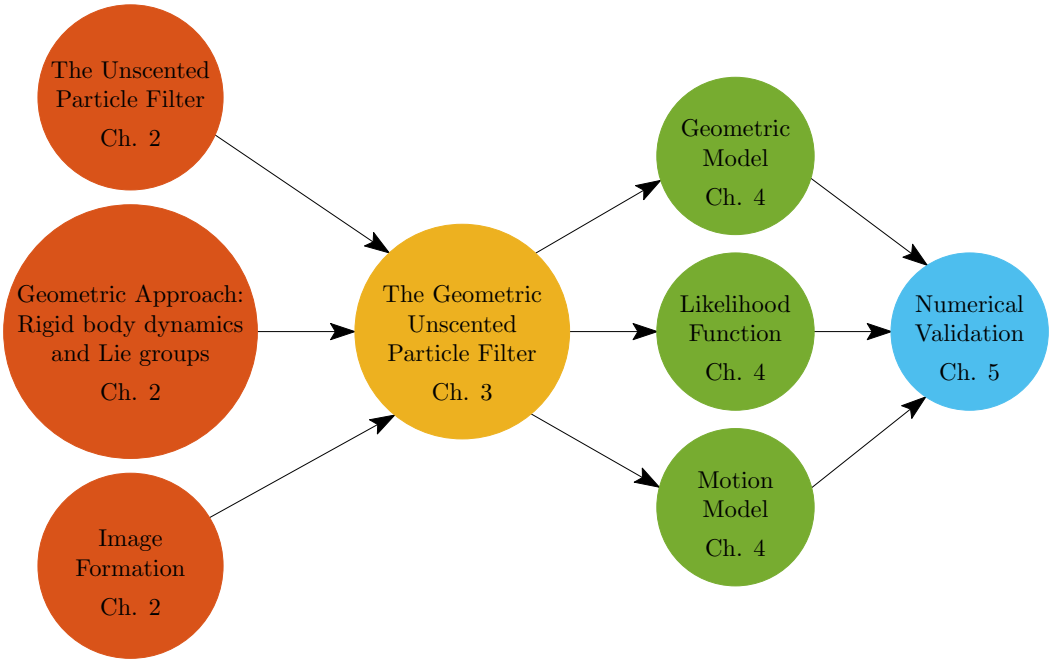


Figure 1.10: Graphical representation of the thesis contents.

Chapter 2

Mathematical Preliminaries

In this chapter, we provide an overview of the background material relevant for this research. More specifically, in Section 2.1, we give an overview of the UPF, which is the key element behind our proposed tracking algorithm. This UPF will in Chapter 3 be generalized to Lie groups, resulting in the GUPF, and the required background information on Lie groups is given in Section 2.2. Section 2.3 then describes the group structure of the state as used in the GUPF. Two main elements of this filter include a likelihood estimation based on image data and a description of the dynamical behavior of the object. Section 2.4 covers an overview of the notation used to describe this dynamical behavior and explains how we express the position, orientation, and velocity of the cuboid in 3D space. Since the tracking algorithm that we will use is partly based on image data, we will use Section 2.5 to explain image formation and show how the pose of the cuboid in 3D space is represented in a 2D image.

2.1 Object tracking using particle filters

From a statistical point of view, the problem of object tracking can be seen as estimating the *state* (position, orientation, linear velocity, and angular velocity) of the object recursively in time. At every time instant, a multivariate probability distribution reflects the probability of the object having a certain state. This distribution is known as the *filtering density*, denoted by $p(\mathbf{x}_t|\mathbf{y}_{1:t})$, and gives the probability of a state $\mathbf{x}_t$ at time t, given all the observations up to that point $\mathbf{y}_{1:t}$. The observation of the state is not a measurement obtained from image processing as one might think. Rather, we will consider the observation to be the image itself, and compute the *likelihood* of that image being created by a certain state. Let us now use the Bayesian rule to rewrite the filtering density as

$$p(\mathbf{x}_t|\mathbf{y}_{1:t}) = \frac{p(\mathbf{y}_t|\mathbf{x}_t, \mathbf{y}_{1:t-1})p(\mathbf{x}_t, \mathbf{y}_{1:t-1})}{p(\mathbf{y}_{1:t})}. \tag{2.1}$$

As common practice, the observations are assumed to be conditionally independent given the states. This means that the current observation $\mathbf{y}_t$ depends only on the current state $\mathbf{x}_t$, independently of previous observations $\mathbf{y}_{1:t-1}$. Hence, (2.1) can be rewritten as

$$p(\mathbf{x}_t|\mathbf{y}_{1:t}) = \frac{p(\mathbf{y}_t|\mathbf{x}_t)p(\mathbf{x}_t, \mathbf{y}_{1:t-1})}{p(\mathbf{y}_{1:t})}. \tag{2.2}$$

In addition, the state evolution over time is considered to be a Markov process with initial distribution $p(\mathbf{x}_0)$ and a transition distribution $p(\mathbf{x}_t|\mathbf{x}_{t-1})$, known as the *prior*. In a Markov process, the current state $\mathbf{x}_t$ is directly computed from the previous state $\mathbf{x}_{t-1}$ and thus independent of all other previous states $\mathbf{x}_{0:t-2}$. This allows to rewrite the term $p(\mathbf{x}_t, \mathbf{y}_{1:t-1})$

in (2.2) as

$$p(\mathbf{x}_t, \mathbf{y}_{1:t-1}) = \int_{\mathbf{x}_{t-1}} p(\mathbf{x}_t, \mathbf{x}_{t-1}, \mathbf{y}_{1:t-1}) d\mathbf{x}_{t-1}, \tag{2.3}$$

$$= \int_{\mathbf{x}_{t-1}} p(\mathbf{x}_t|\mathbf{x}_{t-1}) p(\mathbf{x}_{t-1}, \mathbf{y}_{1:t-1}) d\mathbf{x}_{t-1}, \tag{2.4}$$

where the integral is taken over all the possible values of $\mathbf{x}_{t-1}$, known as the Chapman-Kolmogorov equation [71, 72]. Substituting the result of (2.3) into (2.2) gives

$$p(\mathbf{x}_t|\mathbf{y}_{1:t}) = \frac{p(\mathbf{y}_t|\mathbf{x}_t) \int_{\mathbf{x}_{t-1}} p(\mathbf{x}_t|\mathbf{x}_{t-1}) p(\mathbf{x}_{t-1}, \mathbf{y}_{1:t-1}) d\mathbf{x}_{t-1}}{p(\mathbf{y}_{1:t})}. \tag{2.5}$$

Making use of the fact that

$$p(\mathbf{x}_{t-1}, \mathbf{y}_{1:t-1}) = p(\mathbf{x}_{t-1}|\mathbf{y}_{1:t-1}) p(\mathbf{y}_{1:t-1}) \tag{2.6}$$

and

$$p(\mathbf{y}_{1:t}) = p(\mathbf{y}_t|\mathbf{y}_{1:t-1}) p(\mathbf{y}_{1:t-1}), \tag{2.7}$$

(2.5) can be written as

$$p(\mathbf{x}_t|\mathbf{y}_{1:t}) = \frac{p(\mathbf{y}_t|\mathbf{x}_t) \int_{\mathbf{x}_{t-1}} p(\mathbf{x}_t|\mathbf{x}_{t-1}) p(\mathbf{x}_{t-1}|\mathbf{y}_{1:t-1}) p(\mathbf{y}_{1:t-1}) d\mathbf{x}_{t-1}}{p(\mathbf{y}_t|\mathbf{y}_{1:t-1}) p(\mathbf{y}_{1:t-1})}. \tag{2.8}$$

Note that in (2.8), the term $p(\mathbf{y}_{1:t-1})$ is independent of $\mathbf{x}_{t-1}$ and can therefore be taken out of the integral to cancel out the same term in the denominator. Hence, (2.8) can now be expressed as

$$p(\mathbf{x}_t|\mathbf{y}_{1:t}) = \frac{p(\mathbf{y}_t|\mathbf{x}_t) \int_{\mathbf{x}_{t-1}} p(\mathbf{x}_t|\mathbf{x}_{t-1}) p(\mathbf{x}_{t-1}|\mathbf{y}_{1:t-1}) d\mathbf{x}_{t-1}}{p(\mathbf{y}_t|\mathbf{y}_{1:t-1})} \tag{2.9}$$

$$\propto p(\mathbf{y}_t|\mathbf{x}_t) \int_{\mathbf{x}_{t-1}} p(\mathbf{x}_t|\mathbf{x}_{t-1}) p(\mathbf{x}_{t-1}|\mathbf{y}_{1:t-1}) d\mathbf{x}_{t-1}, \tag{2.10}$$

where in the last step we make use of the fact that

$$p(\mathbf{y}_t|\mathbf{y}_{1:t-1}) = \int_{\mathbf{x}_t} p(\mathbf{y}_t|\mathbf{x}_t) p(\mathbf{x}_t|\mathbf{y}_{1:t-1}) d\mathbf{x}_t \tag{2.11}$$

is a normalizing constant. The filtering density given by (2.10) can thus be computed recursively by the use of the likelihood $p(\mathbf{y}_t|\mathbf{x}_t)$, the transition prior $p(\mathbf{x}_t|\mathbf{x}_{t-1})$, and the estimate of the previous time step $p(\mathbf{x}_{t-1}|\mathbf{y}_{1:t-1})$.

2.1.1 Sequential Monte Carlo

In sequential Monte Carlo simulations (also known as *particle filters*), as set of particles $\{\mathbf{x}_t^{(i)}\}_{i=1}^N$ is used to approximate the filtering density (2.10) by the use of the empirical estimate

$$p(\mathbf{x}_t|\mathbf{y}_{1:t}) \approx \frac{1}{N} \sum_{i=1}^{N} \delta(\mathbf{x}_t - \mathbf{x}_t^{(i)}), \tag{2.12}$$

with δ the Dirac delta function and N a chosen number of particles. It is, however, not possible to sample from this distribution, as the exact distribution is unknown [61, 71–73]. However, if it is possible to evaluate the filtering density, one can sample particles from a distribution similar to the filtering density, a so called *proposal distribution* given as $q(\mathbf{x}_t^{(i)}|\mathbf{y}_{1:t})$, and assign weights to each particle which reflect the difference between the

proposal distribution and the filtering density [73]. This procedure is called *importance sampling*. The unnormalized weights $\tilde{w}_t^{(i)}$ are given as the ratio between the filtering density and the proposal distribution evaluated for each particle $\mathbf{x}_t^{(i)}$, sampled from $q(\mathbf{x}_t^{(i)}|\mathbf{y}_{1:t})$, such that

$$\tilde{w}_t^{(i)} \propto \frac{p(\mathbf{x}_t^{(i)}|\mathbf{y}_{1:t})}{q(\mathbf{x}_t^{(i)}|\mathbf{y}_{1:t})}, \tag{2.13}$$

after which these weights can be normalized according to

$$w_t^{(i)} = \frac{\tilde{w}_t^{(i)}}{\sum_{j=1}^{N} \tilde{w}_t^{(j)}}. \tag{2.14}$$

The filtering density can now be approximated by the set of weighted particles $\{\mathbf{x}_t^{(i)}, w_t^{(i)}\}_{i=1}^{N}$ such that

$$p(\mathbf{x}_t|\mathbf{y}_{1:t}) \approx \sum_{i=1}^{N} w_t^{(i)} \delta\left(\mathbf{x}_t - \mathbf{x}_t^{(i)}\right) \tag{2.15}$$

from which the optimum state estimate can be approximated by the use of the Monte Carlo approximation:

$$\mathbb{E}(\mathbf{x}_t|\mathbf{y}_{1:t}) \approx \sum_{i=1}^{N} w_i^{(i)} \mathbf{x}_t^{(i)} = \bar{\mathbf{x}}, \tag{2.16}$$

also known as a *weighted mean*. The unnormalized weights, given by (2.13), can be computed in a recursive way. First, the result of (2.10) is substituted into (2.13) to obtain

$$\tilde{w}_t^{(i)} \propto \frac{p(\mathbf{y}_t|\mathbf{x}_t^{(i)}) \int_{\mathbf{x}_{t-1}^{(i)}} p(\mathbf{x}_t^{(i)}|\mathbf{x}_{t-1}^{(i)}) p(\mathbf{x}_{t-1}|\mathbf{y}_{1:t-1}) d\mathbf{x}_{t-1}^{(i)}}{q(\mathbf{x}_t^{(i)}|\mathbf{y}_{1:t})}, \tag{2.17}$$

where after the result of (2.15) is substituted to finally obtain

$$\tilde{w}_t^{(i)} \propto w_{t-1}^{(i)} \frac{p(\mathbf{y}_t|\mathbf{x}_t^{(i)}) p(\mathbf{x}_t^{(i)}|\mathbf{x}_{t-1}^{(i)})}{q(\mathbf{x}_t^{(i)}|\mathbf{y}_{1:t})}, \tag{2.18}$$

known as the importance function [74].

2.1.2 Resampling

As illustrated by (2.18), the weight of each particle is computed recursively by the use of the likelihood $p(\mathbf{y}_t|\mathbf{x}_t)$, the transition prior $p(\mathbf{x}_t|\mathbf{x}_{t-1})$, and the proposal distribution $q(\mathbf{x}_t|\mathbf{y}_{1:t})$. As a result, the variance in the weights increases over time, and after some iterations one of the normalized weights tends to 1, while the other weights tend to zero [75, p. 368], [76, p. 247]. A particle drifting away from the area with high likelihood is thus effectively removed from the particle set. It is therefore of high importance to include a selection step in the algorithm, to only concentrate on particles with a high weight [74]. This selection step is known as *resampling*, which is done according to the weight of each particle: particles corresponding to a low weight get discarded from the system while particles with a high weight get resampled into multiple *children*. Multiple resampling techniques exist, of which *Multinomial Resampling*, *Residual Resampling*, *Stratified Resampling*, and *Systematic Resampling* are the four main resampling methods proposed in literature. Various articles show a comparison between these methods [77–80]. In this work, the choice is made for systematic resampling due to its simplicity, high resampling quality, and low computational cost. Formally, the

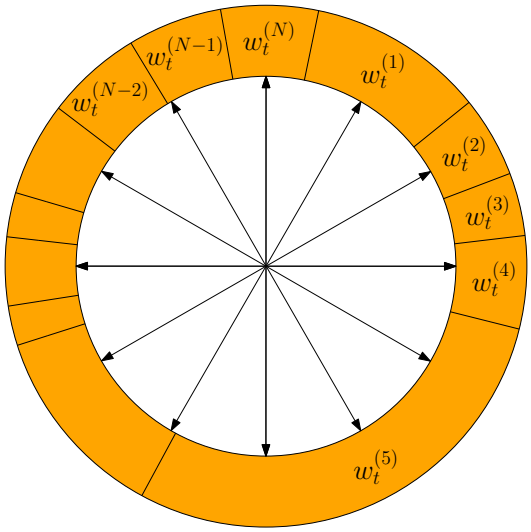


Figure 2.1: Illustration of systematic resampling.

goal of resampling is to convert the weighted distribution to the unweighted density. Hence, the distribution

$$p(\mathbf{x}_t|\mathbf{y}_{1:t}) \approx \sum_{i=1}^{N} w_t^{(i)} \delta\left(\mathbf{x}_t - \mathbf{x}_t^{(i)}\right) \tag{2.19}$$

is replaced by

$$\hat{p}(\mathbf{x}_t|\mathbf{y}_{1:t}) \approx \frac{1}{N}\sum_{k=1}^{N} \delta\left(\mathbf{x}_t - \mathbf{x}_t^{*(k)}\right) = \frac{n_i}{N}\sum_{i=1}^{N} \delta\left(\mathbf{x}_t - \mathbf{x}_t^{(i)}\right), \tag{2.20}$$

with n_i the number of children of particle $\mathbf{x}_t^{(i)}$ in the new set of particles $\{\mathbf{x}_t^{*(k)}\}_{k=1}^N$ [81]. The children n_i should be chosen such that the density $\hat{p}(\mathbf{x}_t|\mathbf{y}_{1:t})$ is close to $p(\mathbf{x}_t|\mathbf{y}_{1:t})$ [75, 77] in the sense that for any function $f(\mathbf{x})$

$$\mathbb{E}\left[\left(\int f(\mathbf{x}_t)p(\mathbf{x}_t|\mathbf{y}_{1:t}) - \int f(\mathbf{x}_t)\hat{p}(\mathbf{x}_t|\mathbf{y}_{1:t})\right)^2\right] \xrightarrow{N\to\infty} 0. \tag{2.21}$$

In systematic resampling, there are N ordered numbers generated according to

$$u_k = \frac{(k-1)+\tilde{u}}{N}, \text{ with } \tilde{u} \sim U[0,1), \tag{2.22}$$

which are used to select $\mathbf{x}_t^{*(k)}$ according to the multinomial distribution. That is,

$$\begin{aligned}\mathbf{x}_t^{*(k)} &= \mathbf{x}_t\left(F^{-1}(u_k)\right) \\ &= \mathbf{x}_t^{(i)} \text{ with } i \text{ such that } u_k \in \left[\sum_{s=1}^{i-1} w_t^{(s)}, \sum_{s=1}^{i} w_t^{(s)}\right),\end{aligned}$$

where F^{-1} denotes the generalized inverse of the cumulative probability distribution of the normalized weights [77, 79]. Figure 2.1 shows a wagon wheel representation of systematic resampling. The weights are represented on the arc and the arc length is proportional to their weight. In this wagon wheel, N spokes are chosen according to (2.22). Every spoke hitting a certain weight will copy its corresponding particle $\mathbf{x}_t^{(i)}$ to the new set. In the example of Figure 2.1, the particle $\mathbf{x}_t^{(5)}$ corresponding to $w_t^{(5)}$ will be resampled into 3 new particles, whereas the particle corresponding to $w_t^{(3)}$ will be discarded from the particle set.

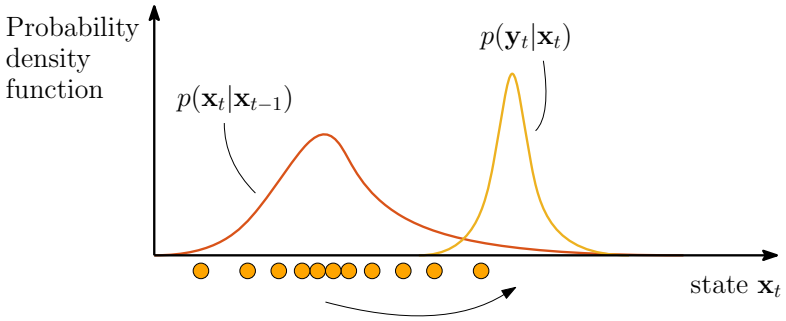


Figure 2.2: Choosing the prior as the proposal distribution. If we sample from the transition prior, the particles (here represented by the orange dots) can move away from the area with high likelihood, resulting in degeneracy of the particle set.

2.1.3 Choice of proposal distribution

As (2.13) illustrates, the choice of proposal distribution is of high importance for the approximation of the filtering density. Often, the choice is made to use the prior distribution as a proposal, as is for example done in [28, 47]. This yields the Bootstrap filter as discussed in [75, Chapter 1]. In that case, the weight of each particle is directly proportional to the likelihood of that particle, and (2.18) becomes

$$\tilde{w}_t^{(i)} \propto w_{t-1}^{(i)} p(\mathbf{y}_t|\mathbf{x}_t^{(i)}). \tag{2.23}$$

In this case, the particles from the previous state $\mathbf{x}_{t-1}$ are directly propagated to the next time step through the motion model, without the use of the available observation $\mathbf{y}_t$. Consequently, it may happen that the complete set of particles moves away from the area with high likelihood. This often occurs when the motion model does not match the true motion of the object. As a result, after a few iterations only few particles will have a significant weight after their likelihood is evaluated, as is illustrated in Figure 2.2. A solution would be to move the particles closer to the area with high likelihood before the weight of each particle is computed, hence by creating a proposal distribution which is closer to the true posterior of the state. In order to do so, the information of the current observation $\mathbf{y}_t$ should be used. So, let us assume we can obtain a rough measurement $\mathbf{z}_t$ of the state parameters, for example by using some image processing routines. Even if noisy, such measurement could improve the proposal distribution. Assuming the proposal distribution, the measurement, and the noise in the transition model are all Gaussian, we can use a form of Kalman filter (extended or unscented) to compute the mean and covariance of the proposal distribution.

In [61], Van der Merwe et al. show both the effects of using an extended Kalman filter or unscented Kalman filter to create a proposal distribution. They show that incorporating the latest observation to generate a proposal distribution, improves the tracking accuracy significantly. Moreover, their results show that the particle filter with an unscented Kalman filter as proposal, which they call the *Unscented Particle Filter*, outperforms the particle filter with extended Kalman filter as proposal significantly. Besides, the unscented Kalman filter allows for non-linear state transitions, whereas the extended Kalman filter does not. To this end, we prefer to use the unscented Kalman filter to create a proposal distribution, and in the next section we will show how this can be done.

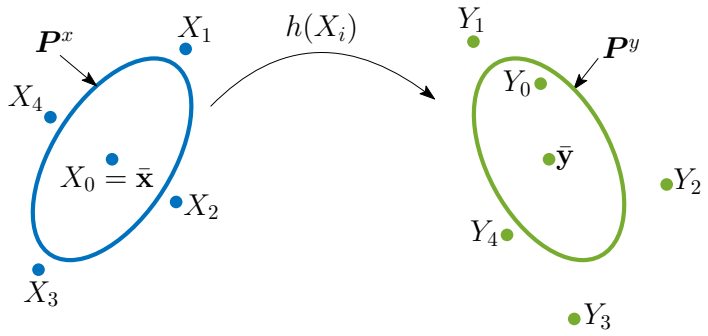


Figure 2.3: Schematic representation of the unscented transformation for a two-dimensional state, with $n_x = 2$. The sigma points X_i of the state $\mathbf{x}$ are propagated through the non-linear model h to obtain the points Y_i. These points are then used to compute the mean $\bar{\mathbf{y}}$ and covariance $\boldsymbol{P}^y$.

2.1.4 Using an unscented Kalman filter to create a proposal distribution

2.1.4.1 The unscented transformation

The key idea behind Unscented Kalman Filter (UKF) is the so-called *unscented transformation*. In this transformation, the state $\mathbf{x}$ is assumed to have mean $\bar{\mathbf{x}}$ and covariance $\boldsymbol{P}^x$. Given n_x as the size of the state, a set of $2n_x + 1$ weighted samples or *sigma points* that represent the true mean and covariance of the state $\mathbf{x}$ is generated as in [82]. Each sigma point is denoted by $S_i = \{X_i, W_i\}$ where X_i is given by

$$\begin{aligned} X_0 &= \bar{\mathbf{x}}, \\ X_i &= \bar{\mathbf{x}} + \left(\sqrt{(n_x+\lambda)\boldsymbol{P}^x}\right)_i \quad \text{for} \quad i = 1, \ldots, n_x, \\ X_i &= \bar{\mathbf{x}} - \left(\sqrt{(n_x+\lambda)\boldsymbol{P}^x}\right)_i \quad \text{for} \quad i = n_x+1, \ldots, 2n_x, \end{aligned} \tag{2.24}$$

where $\left(\sqrt{(n_x+\lambda)\boldsymbol{P}^x}\right)_i$ denotes the i-th column of the matrix square root of $(n_x+\lambda)\boldsymbol{P}^x$. Furthermore, the weights W_i are given by

$$\begin{aligned} W_0^{(m)} &= \lambda/(n_x+\lambda), \\ W_0^{(c)} &= \lambda/(n_x+\lambda) + (1 - {\alpha_k}^2 + \beta_k), \\ W_i^{(m)} &= 1/(2(n_x+\lambda)) \quad \text{for} \quad i = 1, \ldots, 2n_x, \\ W_i^{(c)} &= 1/(2(n_x+\lambda)) \quad \text{for} \quad i = 1, \ldots, 2n_x, \end{aligned} \tag{2.25}$$

where the superscripts (m) and (c) indicate, respectively, if the weight is contributing to the mean or the covariance. Using (2.24) and (2.25), the mean and covariance of the state $\mathbf{x}$ can be estimated by

$$\bar{\mathbf{x}} = \sum_{i=0}^{2n_x} W_i^{(m)} X_i \tag{2.26}$$

and

$$\boldsymbol{P}^x = \sum_{i=0}^{2n_x} W_i^{(c)} \left(X_i - \bar{\mathbf{x}}\right)\left(X_i - \bar{\mathbf{x}}\right)^T, \tag{2.27}$$

respectively. In (2.24) and (2.25), λ is chosen as $\lambda = {\alpha_k}^2(n_x + \zeta) - n_x$ as in [83]. The value ζ is a scaling parameter that scales the sigma points towards ($\zeta < 0$) or away ($\zeta > 0$) from the

mean. The parameter β_k is a weighting term, which for a Gaussian prior has an optimum value of 2. The parameter α_k is a scaling parameter for which holds that $0 < \alpha_k < 1$. It is used to scale the size of the distribution. Each sigma point is then propagated through the non-linear observation model h by

$$Y_i = h\left(X_i\right) \quad \text{for} \quad i = 0, \ldots, 2n_x, \tag{2.28}$$

such that the mean and covariance of $\mathbf{y}$ are given by

$$\bar{\mathbf{y}} = \sum_{i=0}^{2n_x} W_i^{(m)} Y_i \tag{2.29}$$

and

$$\boldsymbol{P}^y = \sum_{i=0}^{2n_x} W_i^{(c)} \left[Y_i - \bar{\mathbf{y}}\right] \left[Y_i - \bar{\mathbf{y}}\right]^T, \tag{2.30}$$

respectively. In Figure 2.3, a schematic representation is shown for the two-dimensional case, hence when $n_x = 2$.

2.1.4.2 The unscented Kalman filter

If we assume Gaussian noise is added to a motion model f and observation model h, we can describe the dynamics of the system as

$$\begin{aligned} \mathbf{x}_t &= f(\mathbf{x}_{t-1}) + \mathbf{m}_{t-1}, \\ \mathbf{y}_t &= h(\mathbf{x}_t) + \mathbf{n}_t, \end{aligned} \tag{2.31}$$

where $\mathbf{m}_t$ denotes the process noise and $\mathbf{n}_t$ the observation noise. In order to implement the unscented transformation into the Kalman filter, the state mean and covariance are augmented with the process noise $\mathbf{m}_t$ and measurement noise $\mathbf{n}_t$ to obtain the augmented state mean and covariance, written as

$$\bar{\mathbf{x}}_t^a = \begin{bmatrix} \bar{\mathbf{x}}_t \\ \mathbf{m}_t \\ \mathbf{n}_t \end{bmatrix} \quad \text{and} \quad \boldsymbol{P}_t^a = \begin{bmatrix} \boldsymbol{P}_t^x & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{P}_t^m & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \boldsymbol{P}_t^n \end{bmatrix}. \tag{2.32}$$

In (2.32), $\boldsymbol{P}_t^m$ and $\boldsymbol{P}_t^n$ are the process noise covariance and measurement noise covariance, respectively. Confusion is often made that, since the process noise and measurement noise are additive, the computational cost can be reduced by using the non-augmented unscented transformation. However, as stated in [84], using the augmented state yields better results, as the sigma points of the augmented state capture more statistical information than the non-augmented state. By substituting $\bar{\mathbf{x}}_t^a$ and $\boldsymbol{P}_t^a$ in (2.24), we obtain $2n_a + 1$ sigma points (n_a being the dimension of the augmented state), where each sigma point i is given by

$$X_i^a = \begin{bmatrix} \left(X_i^x\right)^T & \left(X_i^m\right)^T & \left(X_i^n\right)^T \end{bmatrix}^T, \tag{2.33}$$

with X_i^x, X_i^m, and X_i^n the sigma points of the state, the process noise, and the measurement noise respectively. Each sigma point is now subjected to the functions f and h as described by (2.31). This results in

$$\begin{aligned} X_{i,t|t-1}^x &= f\left(X_{i,t-1}^x\right) + X_{i,t-1}^m \quad \text{for} \quad i = 0, \ldots, 2n_a, \\ Y_{i,t|t-1}^x &= h\left(X_{i,t|t-1}^x\right) + X_{i,t-1}^n \quad \text{for} \quad i = 0, \ldots, 2n_a, \end{aligned} \tag{2.34}$$

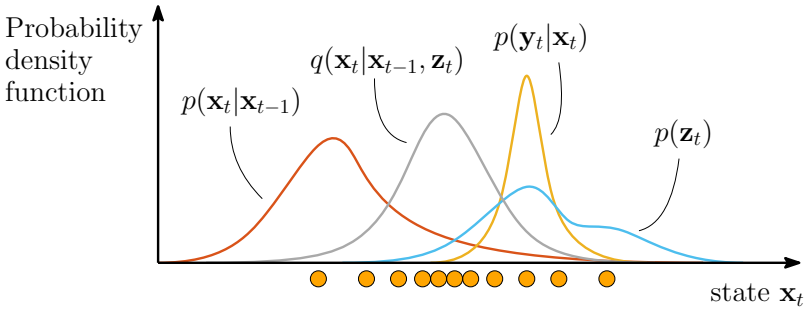


Figure 2.4: Using the UKF to create a proposal distribution. By incorporating a measurement $\mathbf{z}_t$ at time t, the transition prior $p(\mathbf{x}_t|\mathbf{x}_{t-1})$ can be updated by the use of an UKF to create a better proposal distribution $q(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{z}_t)$ which is closer to the area with high likelihood $p(\mathbf{y}_t|\mathbf{x}_t)$.

where the mean and covariance of the state are respectively given by

$$\bar{\mathbf{x}}_{t|t-1} = \sum_{i=0}^{2n_a} W_i^{(m)} X_{i,t|t-1}^x \tag{2.35}$$

and

$$\boldsymbol{P}_{t|t-1}^x = \sum_{i=0}^{2n_a} W_i^{(c)} \left[X_{i,t|t-1}^x - \bar{\mathbf{x}}_{t|t-1}\right] \left[X_{i,t|t-1}^x - \bar{\mathbf{x}}_{t|t-1}\right]^T. \tag{2.36}$$

The mean and covariance of the observation are then given by

$$\bar{\mathbf{y}}_{t|t-1} = \sum_{i=0}^{2n_a} W_i^{(m)} Y_{i,t|t-1} \tag{2.37}$$

and

$$\boldsymbol{P}_{t|t-1}^y = \sum_{i=0}^{2n_a} W_i^{(c)} \left[Y_{i,t|t-1} - \bar{\mathbf{y}}_{t|t-1}\right] \left[Y_{i,t|t-1} - \bar{\mathbf{y}}_{t|t-1}\right]^T. \tag{2.38}$$

The state-observation cross covariance is obtained by

$$\boldsymbol{P}_{t|t-1}^{xy} = \sum_{i=0}^{2n_a} W_i^{(c)} \left[X_{i,t|t-1} - \bar{\mathbf{x}}_{t|t-1}\right] \left[Y_{i,t|t-1} - \bar{\mathbf{y}}_{t|t-1}\right]^T, \tag{2.39}$$

from which the Kalman gain can be computed as

$$\boldsymbol{K}_t = \boldsymbol{P}_{t|t-1}^{xy} \left(\boldsymbol{P}_{t|t-1}^y\right)^{-1}. \tag{2.40}$$

Finally, the state mean and covariance are updated by the use of the Kalman gain which results in

$$\bar{\mathbf{x}}_t = \bar{\mathbf{x}}_{t|t-1} + \boldsymbol{K}_t \left(\mathbf{z}_t - \bar{\mathbf{y}}_{t|t-1}\right) \tag{2.41}$$

and

$$\boldsymbol{P}_t^x = \boldsymbol{P}_{t|t-1}^x - \boldsymbol{K}_t \boldsymbol{P}_{t|t-1}^y \mathbf{K}_t^T, \tag{2.42}$$

where $\mathbf{z}_t$ is the measurement obtained at time t. This measurement can be obtained in several ways. One method could be the detection of color blobs in the image, from which a 3D measurement of the object can be obtained. This approach is taken, for example, in [85]. Further details on how we can obtain a measurement of the pose from a 2D image will be

discussed in Chapter 3. The transition prior and proposal distribution are now given as normal distributions with a mean and covariance as computed by the UKF. Hence, they can be written as

$$p(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}\left(\bar{\mathbf{x}}_{t|t-1}, \boldsymbol{P}^x_{t|t-1}\right) \tag{2.43}$$

and

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{z}_t) = \mathcal{N}\left(\bar{\mathbf{x}}_t, \boldsymbol{P}^x_t\right). \tag{2.44}$$

Van de Merwe et al. proposed in [61] to use the above described UKF for each particle, resulting in the *Unscented Particle Filter* (UPF). The advantage of using the UKF to create the proposal distribution (2.44) is that a particle sampled from this distribution is now closer to the area with high likelihood, due to the update using the measurement. This is also schematically depicted in Figure 2.4.

To conclude, we want to stress the importance of (2.18), which now contains the *likelihood* given as $p(\mathbf{y}_t|\mathbf{x}_t^{(i)})$, the *transition prior* given as $p(\mathbf{x}_t^{(i)}|\mathbf{x}_{t-1}^{(i)})$, and the *proposal distribution* given by $q(\mathbf{x}_t^{(i)}|\mathbf{x}_{t-1}^{(i)}, \mathbf{z}_t)$. The likelihood function will be based on image data, and requires projection of the 3D position/orientation of the particle to the image plane, which we will discuss in Section 2.5. The transition prior and proposal distribution are obtained via the UKF, given by (2.43) and (2.44), and require a motion model, describing the state transition in time. As mentioned in 1.3.2, we will take a geometric approach to this motion model, which requires a generalization of the UPF to Lie groups. In the next section, we therefore provide the required background information from this field.

2.2 Lie groups: basic notation and probability distributions

In this section, first the basic notation and operations on Lie groups are discussed. Next, in Sections 2.2.2 and 2.2.3, two well-known Lie groups are discussed: the Special Orthogonal group and the Special Euclidean group. Finally, in Section 2.2.4, it will be shown how Gaussian distributions are defined for Lie groups.

2.2.1 Preliminaries on Lie groups

A *Lie group* is a group, which also is a smooth differentiable manifold, consisting of a set $\mathcal{G}$ and a an operator $\circ$, such that the following properties hold [86]:

- For any $g_1, g_2 \in \mathcal{G}$ it holds that if $g_1 \circ g_2 = g_3$, then $g_3 \in \mathcal{G}$.
- For any $g_1, g_2, g_3 \in \mathcal{G}$ it holds that $(g_1 \circ g_2) \circ g_3 = g_1 \circ (g_2 \circ g_3)$.
- There exist an element $e \in \mathcal{G}$, known as the *identity*, such that for any $g_1 \in \mathcal{G}$ it holds that $g_1 \circ e = g_1$ and $e \circ g_1 = g_1$.
- Every element $g_1 \in \mathcal{G}$ has an *inverse*, denoted by g_1^{-1} such that $g_1^{-1} \in \mathcal{G}$ and for which holds that $g_1^{-1} \circ g_1 = e$ and $g_1 \circ g_1^{-1} = e$.

In a Lie group, the group operator and inversion are smooth functions. Furthermore, a Lie group is known to be *abelian* if $g_1 \circ g_2 = g_2 \circ g_1$ for $g_1, g_2 \in \mathcal{G}$. It is however well known that both the Special Euclidean group SE(3) as well as the Special Orthogonal group SO(3) are nonabelian groups, such that we can distinguish a *left* and a *right* translation, which will become clear in Section 2.4. In the latter, we will refer to Lie groups by the use of

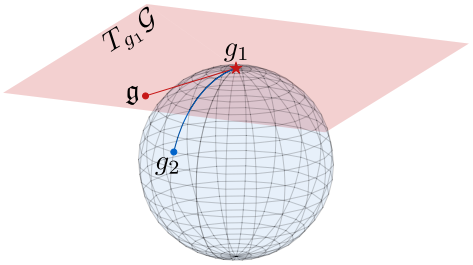


Figure 2.5: The exponential and logarithmic mappings on a Lie group transfer points between the group and the tangent space. Here, g_2 is mapped to $\mathfrak{g}$ in the tangent space at g_1 by the logarithmic mapping. The inverse operation is accomplished by the exponential mapping.

calligraphic capital letters. We will assume the operation is understood from the context, such that we will omit its symbol for the sake of brevity.

The exponential and logarithmic map of a Lie group. Every Lie group comes with a *Lie algebra*, which is the tangent space of the Lie group at the identity endowed with a binary operation. The exponential and logarithmic map allow to transfer elements between the Lie group and the Lie algebra. The exponential map locally maps an element of the tangent space to the group, whereas the logarithmic map of a Lie group transfers elements from the Lie group to its tangent space. In Lie group theory, the group product and inversion allow to transfer the exponential and logarithmic mappings over the entire group. Consider Figure 2.5, where g_1 and g_2 are both elements of the group $\mathcal{G}$. A tangent space is considered at g_1, indicated by $T_{g_1}\mathcal{G}$. A point $\mathfrak{g} \in T_{g_1}\mathcal{G}$ can now be mapped to the group, resulting in g_2, according to

$$g_2 = g_1 \operatorname{Exp}(\mathfrak{g}). \tag{2.45}$$

On the other hand, g_2 can be mapped to the tangent space $T_{g_1}\mathcal{G}$ resulting in $\mathfrak{g}$ according to

$$\mathfrak{g} = \operatorname{Log}(g_1^{-1} g_2). \tag{2.46}$$

Note that in (2.45) and (2.46), we use the expressions of Exp and Log to denote the exponential and logarithmic mappings on Lie groups, thereby making a distinction with the ordinary exponential and logarithmic functions, which we will denote by exp and log, respectively. For more information on Lie groups and Lie algebras, the reader is referred to [87].

Direct product of groups. Another property of Lie groups allows us to combine groups. Given the Lie group $\mathcal{H}$ with operation $\bullet$ and group $\mathcal{G}$ with operation $\circ$, it is given that the direct product of these groups, denoted as $\mathcal{H} \times \mathcal{G}$, is a Lie group as well. The formed Lie group consists of a set, which is a Cartesian product of the sets of $\mathcal{H}$ and $\mathcal{G}$, which results in the ordered pairs (h, g) where $h \in \mathcal{H}$ and $g \in \mathcal{G}$. The resulting group operation is defined component-wise such that

$$(h_1, g_1)(h_2, g_2) = (h_1 \bullet h_2, g_1 \circ g_2). \tag{2.47}$$

2.2.2 The Special Orthogonal group SO(3)

The 3-dimensional *Special Orthogonal* group is formed by the set of $\mathbb{R}^{3\times3}$ orthogonal matrices with determinant equal to 1 given by

$$\mathrm{SO}(3) \coloneqq \left\{\mathbf{R} \in \mathbb{R}^{3\times3} \;\middle|\; \mathbf{R}^T\mathbf{R} = \mathbf{I}_3, \det(\mathbf{R}) = 1\right\}, \tag{2.48}$$

which becomes a Lie group under the matrix product. The matrices $\mathbf{R}$ are referred to as *rotation matrices* and describe the relative rotation of one frame to another. The Lie algebra of SO(3), denoted as $\mathfrak{so}(3)$, is identified by the 3×3 skew-symmetric matrices of the form

$$\boldsymbol{\omega}^\wedge \coloneqq \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix} \in \mathfrak{so}(3), \tag{2.49}$$

where $\boldsymbol{\omega} = \begin{bmatrix}\omega_x & \omega_y & \omega_z\end{bmatrix}^T \in \mathbb{R}^3$ and $(\cdot)^\wedge$ is known as the *hat*-operator as in [88, 89]. The exponential mapping for SO(3) can efficiently be computed using Rodrigues' formula as in [90, 91] such that

$$\mathbf{R} = \mathbf{I}_3 + \frac{\sin\left(\|\boldsymbol{\omega}\|\right)}{\|\boldsymbol{\omega}\|}\boldsymbol{\omega}^\wedge + \frac{\left(1-\cos\left(\|\boldsymbol{\omega}\|\right)\right)}{\|\boldsymbol{\omega}\|^2}\left(\boldsymbol{\omega}^\wedge\right)^2, \tag{2.50}$$

where $\|\boldsymbol{\omega}\| = \sqrt{\omega_x^2+\omega_y^2+\omega_z^2}$. It is convenient to know that ω_x, ω_y, and ω_z form the direction of rotation and $\theta = \|\boldsymbol{\omega}\|$ gives the corresponding angle of rotation. The logarithmic map is then effectively computed by inverting (2.50), which yields

$$\theta = \begin{cases} \cos^{-1}\left(\frac{\mathrm{tr}(\mathbf{R})-1}{2}\right) & \text{if} \quad \mathbf{R} \neq \mathbf{I}_3, \\ 2\pi k & \text{if} \quad \mathbf{R} = \mathbf{I}_3, \end{cases} \tag{2.51}$$

$$\boldsymbol{\omega}^\wedge = \begin{cases} \frac{\theta}{2\sin(\theta)}\left(\mathbf{R}-\mathbf{R}^T\right) & \text{if} \quad \mathbf{R} \neq \mathbf{I}_3, \\ \mathbf{0} & \text{if} \quad \mathbf{R} = \mathbf{I}_3, \end{cases} \tag{2.52}$$

where k is an arbitrarily chosen integer. Note that the exponential map is a many-to-one map such that representations of $\mathbf{R}$ that rely on $\boldsymbol{\omega}$ are not uniquely covering SO(3) [92]. Furthermore, the inverse is given by the matrix inverse and the identity element is given by the 3×3 identity matrix $\mathbf{I}_3$.

2.2.3 The Special Euclidean group

The 3-dimensional *Special Euclidean* group is formed by the set given by

$$\mathrm{SE}(3) \coloneqq \left\{\mathbf{H} = \begin{bmatrix}\mathbf{R} & \mathbf{p} \\ \mathbf{0} & 1\end{bmatrix} \in \mathbb{R}^{4\times4} \;\middle|\; \mathbf{R} \in \mathrm{SO}(3), \mathbf{p} \in \mathbb{R}^3\right\}, \tag{2.53}$$

which becomes a Lie group under the matrix product. It is often referred to as the group of rigid-body transformations, and $\mathbf{H}$ is often referred to as a *homogeneous transformation matrix*. The Lie algebra of SE(3), denoted by $\mathfrak{se}(3)$, is identified by the 4×4 matrices of the form

$$\begin{bmatrix}\boldsymbol{\omega}^\wedge & \boldsymbol{v} \\ \mathbf{0} & 0\end{bmatrix} \in \mathfrak{se}(3), \tag{2.54}$$

with $\boldsymbol{\omega}^\wedge \in \mathfrak{so}(3)$ and $\boldsymbol{v} \in \mathbb{R}^3$. Given a vector $\mathbf{v} = \begin{bmatrix}\boldsymbol{v} & \boldsymbol{\omega}\end{bmatrix}^T \in \mathbb{R}^6$ with $\boldsymbol{v} \in \mathbb{R}^3$ and $\boldsymbol{\omega} \in \mathbb{R}^3$, we apply the hat-operator to write $\mathbf{v}$ as an element of $\mathfrak{se}(3)$ according to

$$\mathbf{v}^\wedge = \begin{bmatrix}\boldsymbol{v} \\ \boldsymbol{\omega}\end{bmatrix}^\wedge = \begin{bmatrix}\boldsymbol{\omega}^\wedge & \boldsymbol{v} \\ \mathbf{0} & 0\end{bmatrix} \in \mathfrak{se}(3). \tag{2.55}$$

Since $\boldsymbol{\omega}^\wedge$ is equal to an element in $\mathbb{R}^3$ with the cross product, we will in the remainder refer to this as $\boldsymbol{\omega} \in \mathbb{R}^3_\times$, where the subscript $(\cdot)_\times$ refers to the cross product on $\mathbb{R}^3$. Physically, if $\boldsymbol{v}$ and $\boldsymbol{\omega}$ are expressed in a body fixed frame, $\boldsymbol{v}$ corresponds to the linear velocity of the origin of that frame, while $\boldsymbol{\omega}$ corresponds to the angular velocity of the rigid-body. In kinematics, elements of $\mathfrak{se}(3)$ are referred to as *twists*. The exponential mapping, which maps an element of $\mathfrak{se}(3)$ to SE(3), is given by the matrix exponential and, as shown in [89], every rigid transformation can be written as the exponential of some twist. Furthermore, the inverse of an element $\mathbf{H} \in \mathrm{SE}(3)$ is given by

$$\mathbf{H}^{-1} = \begin{bmatrix} \mathbf{R}^T & -\mathbf{R}^T\mathbf{p} \\ \mathbf{0} & 1 \end{bmatrix}, \tag{2.56}$$

and the identity element is given by the 4×4 identity matrix $\mathbf{I}_4$.

2.2.4 Gaussian distributions on Lie groups

Recall that the n-dimensional multivariate Gaussian distribution is given by

$$p\left(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}\right) = \frac{1}{\sqrt{(2\pi)^n|\boldsymbol{\Sigma}|}} \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right), \tag{2.57}$$

with $\boldsymbol{\mu} \in \mathbb{R}^n$ the mean and $\boldsymbol{\Sigma} \in \mathbb{R}^{n\times n}$ a positive definite covariance matrix. The term inside the exponent is directly related to the Mahalanobis distance, known as

$$d(g_1, g_2) = \sqrt{(g_2 - g_1)^T\boldsymbol{\Sigma}^{-1}(g_2 - g_1)}, \tag{2.58}$$

which is a Euclidean distance. This Euclidean distance function is a *metric* and should therefore satisfy the following axioms:

- **Non-negativity:** $d(g_1, g_2) \geq 0$,
- **Identity of indiscernibles:** $d(g_1, g_2) = 0 \Leftrightarrow g_1 = g_2$,
- **Symmetry:** $d(g_1, g_2) = d(g_2, g_1)$,
- **Triangle inequality:** $d(g_1, g_3) \leq d(g_1, g_2) + d(g_2, g_3)$.

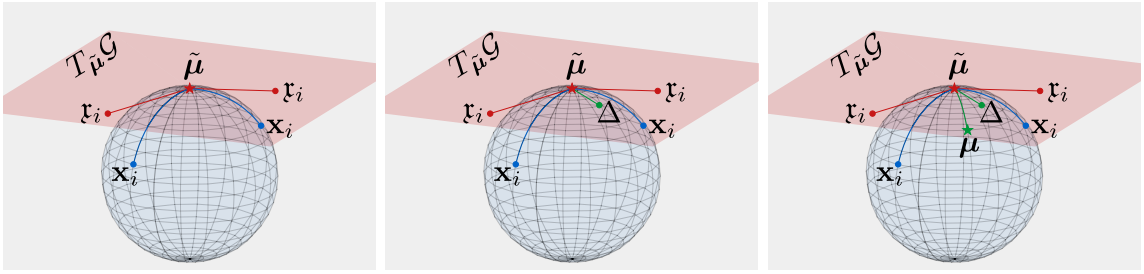
We now try to extend this idea of an unimodal distribution to the case where the random variables live in a Lie group, rather than in Euclidean space. This is a rather complicated concept, as the idea of *mean* and *covariance* are not always well defined. As pointed out by Chirikjian in [94], for example, any point on the uniform distribution on SO(3) could be called a mean. To this end, we follow the approach of [92, 95], in which the following substitution is used:

$$(g_2 - g_1)^T\boldsymbol{\Sigma}^{-1}(g_2 - g_1) \rightarrow \mathrm{Log}(g_1^{-1}g_2)^T\boldsymbol{\Sigma}^{-1}\mathrm{Log}(g_1^{-1}g_2), \tag{2.59}$$

where $\mathrm{Log}(g_1^{-1}g_2)$ represents the distance between two points on the Lie group, mapped to the tangent space as in (2.46). The substitution of (2.59) into (2.57) then leads to the maximum entropy distribution, as also given in [93,94], which for an n-dimensional Lie group $\mathcal{G}$ is given by

$$p\left(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}\right) = \frac{1}{\sqrt{(2\pi)^n|\boldsymbol{\Sigma}|}} \exp\left(-\frac{1}{2}\mathrm{Log}(\boldsymbol{\mu}^{-1}\mathbf{x})^T\boldsymbol{\Sigma}^{-1}\mathrm{Log}(\boldsymbol{\mu}^{-1}\mathbf{x})\right), \tag{2.60}$$

with mean $\boldsymbol{\mu} \in \mathcal{G}$ and covariance $\boldsymbol{\Sigma} \in \mathfrak{g}$. Note that (2.60) is valid under the assumption that the covariance is small, such that a set of random points $\mathbf{x}_i \in \mathcal{G}$ are closely centered around one point on the group. Furthermore, this formulation of the Gaussian depends on a distance



(a) (b) (c)

Figure 2.6: Schematic visualization of the iterative procedure of computing the mean $\boldsymbol{\mu}$ of a set of points $\mathbf{x}_i$ on a Lie group $\mathcal{G}$. The points $\mathbf{x}_i$ are projected onto the tangent space at $\tilde{\boldsymbol{\mu}}$ using the logarithmic mapping resulting in the points $\mathfrak{x}_i$ (a). The mean of the points $\mathfrak{x}_i$ is computed, resulting in the update Δ (b). The update is projected back to the group by the exponential mapping (c). The steps (a-c) are then repeated until $\|\Delta\|$ is converged to a pre-defined convergence threshold.

preserving mapping from the Lie group to the tangent space, which allows for computation of the Mahalanobis distance. As a result, the statistical computations of a *mean* and *covariance* are all performed in the tangent space. Since the point $\boldsymbol{\mu}$ is mapped to the point $\mathbf{0}$ in its own tangent space, the distribution of (2.60) can be thought of as a normal distribution with zero mean and covariance $\boldsymbol{\Sigma}$ in the tangent space of the point $\boldsymbol{\mu}$ on the Lie group.

What remains, is to understand how the mean $\boldsymbol{\mu}$ and consequently the covariance $\boldsymbol{\Sigma}$ of random variables on a Lie group are computed. In the next paragraph, we will show how the mean and covariance can be estimated.

Computing the mean and covariance on a Lie group. As stated in [92], estimating the mean and covariance of a distribution on a Lie group boils down to finding the parameters $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ that maximize the distribution of (2.60) for a given set of points $\mathbf{x}_i \in \mathcal{G}$. In order to do so, we will closely follow the procedures of [92, 95], where the mean in computed iteratively. This procedure is also shown in Figure 2.6. We start with an initial guess for $\tilde{\boldsymbol{\mu}} \in \mathcal{G}$, which for now we assume is close to the true mean $\boldsymbol{\mu}$. In Section 3.3, we will further elaborate on how we obtain this initial guess for a given set of points. First, all the points $\mathbf{x}_i \in \mathcal{G}$ are mapped to a tangent space at $\tilde{\boldsymbol{\mu}}$ according to

$$\mathfrak{x}_i = \mathrm{Log}(\tilde{\boldsymbol{\mu}}^{-1}\mathbf{x}_i) \quad \text{for} \quad i = 1, \ldots, N. \tag{2.61}$$

Next, the mean of the points $\mathfrak{x}_i$ is computed in the tangent space by

$$\Delta = \frac{1}{N}\sum_{i=1}^{N} \mathfrak{x}_i, \tag{2.62}$$

where Δ serves as an update of the mean such that

$$\boldsymbol{\mu} = \tilde{\boldsymbol{\mu}}\mathrm{Exp}(\Delta). \tag{2.63}$$

These steps are then repeated until $\|\Delta\|$ is converged to a pre-defined convergence threshold. When the mean $\boldsymbol{\mu}$ is obtained, the covariance can be computed according to

$$\boldsymbol{\Sigma} = \frac{1}{\sum_{i=1}^{N} w_i}\sum_{i=1}^{N} w_i \mathrm{Log}(\boldsymbol{\mu}^{-1}\mathbf{x}_i)\mathrm{Log}(\boldsymbol{\mu}^{-1}\mathbf{x}_i)^T, \tag{2.64}$$

where w_i may be some weighting factor, generally set to 1. Note that the term $\text{Log}(\boldsymbol{\mu}^{-1}\mathbf{x}_i)$ indicates the distance from the point $\mathbf{x}_i$ to the mean $\boldsymbol{\mu}$, defined in the tangent space of $\boldsymbol{\mu}$ by the logarithmic mapping. In the next section, we will stress the importance of this distance to be the result of a minimum distance mapping, which will lead to our choice for the group structure of the state space.

2.3 Group structure of the state

A possible way to describe the pose of an object is by the use of the homogeneous transformation matrix $\mathbf{H} \in \text{SE}(3)$. This approach is, for example, taken in [37, 65, 66], where the state of the object is considered to be the pose, parametrized by the homogeneous transformation matrix. The state transition is then modeled by the use of *screw motions*, which describe rigid transformations [96], and in Section 2.2.3 we mentioned that these can be written as the exponential of some twist. The advantage of screw motions is that they provide a coordinate independent description [92, 97], and are therefore often used in the literature.

Besides representing the pose as an element of SE(3), it is also possible to describe this pose in $\text{SO}(3) \times \mathbb{R}^3$. Both contain the same set, but under a different topology. This becomes clear in their group operation. Consider two poses, given by $(\mathbf{A}, \mathbf{b})$ and $(\mathbf{R}, \mathbf{v})$. Applying the group operation on $\text{SO}(3) \times \mathbb{R}^3$ yields

$$(\mathbf{A}, \mathbf{b})(\mathbf{R}, \mathbf{v}) = (\mathbf{AR}, \mathbf{b} + \mathbf{v}), \tag{2.65}$$

whereas for SE(3) this yields

$$(\mathbf{A}, \mathbf{b})(\mathbf{R}, \mathbf{v}) = (\mathbf{AR}, \mathbf{Av} + \mathbf{b}). \tag{2.66}$$

It is clear that for $\text{SO}(3) \times \mathbb{R}^3$ the rotational part and translation part are treated separately. As stated in [92, 95], this is a very important property. They state that the Lie group logarithmic map on SE(3) is not related to a metric, and therefore stress the importance of using $\text{SO}(3) \times \mathbb{R}^3$ instead of SE(3) as group structure to represent the pose. As a result, we will use $\text{SO}(3) \times \mathbb{R}^3$ in our statistical framework to represent the pose of the cuboid. In the remainder of this work, we will refer to this group as the *group of the poses*, defined as $\mathcal{P} \coloneqq \text{SO}(3) \times \mathbb{R}^3$ with the corresponding Lie algebra denoted by $\mathfrak{p}$. The state of the cuboid will be defined as a combination of the pose, the linear, and the angular velocity of the cuboid. Since the linear velocity $\boldsymbol{v}$ and angular velocity $\boldsymbol{\omega}$ can be represented as elements of $\mathbb{R}^3$ and $\mathbb{R}^3_\times$ respectively, we will define the group of the states as $\mathcal{S} \coloneqq \text{SO}(3) \times \mathbb{R}^3 \times \mathbb{R}^3 \times \mathbb{R}^3_\times$, with its corresponding Lie algebra denoted by $\mathfrak{s}$. We assume at this point the reader is familiar with the operations on Lie groups, such that the group operation, inverse, the identity element, and the exponential and logarithmic mappings of the groups $\mathcal{P}$ and $\mathcal{S}$, as well as their corresponding Lie algebras, are clear from the group structure, and we will not present them here. They can, however, be found in Appendix B.

2.4 Multibody notation for rigid-body dynamics

In this section, we present the notation of [88], as will be used throughout this report to describe rigid-body dynamics. First, in Section 2.4.1 it is shown how we formally write the position and orientation of a rigid-body. Next, in Sections 2.4.2 and 2.4.3, it is shown how the velocity of a rigid-body is expressed. Finally, Section 2.4.4 shows how external forces and torques applied to the rigid-body are expressed.

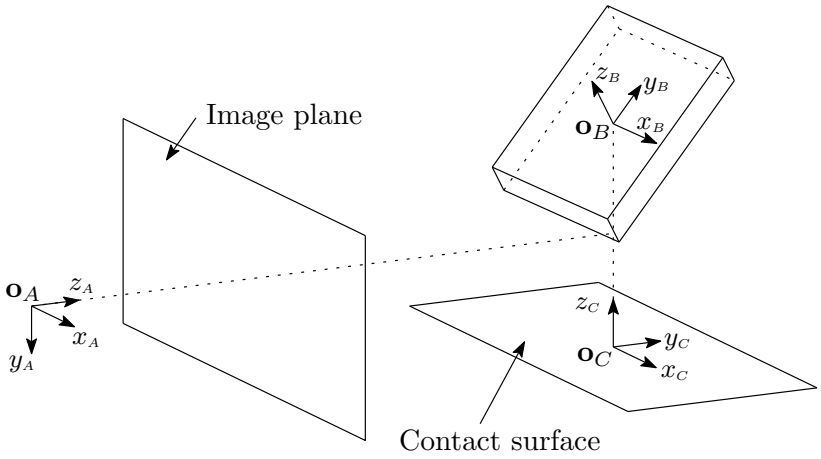


Figure 2.7: Illustration of a cuboid which is about to make contact with a surface. The coordinate frames of the world (A), the center of mass of the cuboid (B), and the reference frame of the contact surface (C) are shown.

2.4.1 Notation of coordinate frames and points

A coordinate frame is defined as the combination of a point, called the *origin*, and an *orientation frame* in the 3D space. A capital letter is used to denote each frame. As example, for a frame A, we will write its origin as $\mathbf{o}_A$ and its orientation frame as $[A]$ such that, formally, we write $A = (\mathbf{o}_A, [A])$. Coordinate frames can move through space in time with respect to other coordinate frames. They are typically assigned to rigid bodies to describe their position and orientation in space.

In this work, we will use frame A as an inertial frame, fixed to the world's surface and we express all motions with respect to this frame. For simplicity, we assume this frame coincides with the camera coordinate frame, such that all motions of rigid bodies in front of the camera can be expressed relative to the camera. We assign the frame B to the cuboid, located at its center of mass and we assign the frame C to the contact surface with which the cuboid can collide. These coordinate frames are schematically depicted in Figure 2.7.

For a point $\mathbf{p}$, its coordinates with respect to A are written as the coordinate vector ${}^A\mathbf{p}$. This coordinate vector represents the 3D geometric vector $\vec{r}_{\mathbf{o}_A,\mathbf{p}}$ connecting the point $\mathbf{p}$ with $\mathbf{o}_A$, pointing towards $\mathbf{p}$ and is expressed in terms of A. Formally, we can write this as

$$ {}^A\mathbf{p} \coloneqq \begin{bmatrix} \vec{r}_{\mathbf{o}_A,\mathbf{p}} \cdot \vec{x}_A \\ \vec{r}_{\mathbf{o}_A,\mathbf{p}} \cdot \vec{y}_A \\ \vec{r}_{\mathbf{o}_A,\mathbf{p}} \cdot \vec{z}_A \end{bmatrix} \in \mathbb{R}^3, \tag{2.67} $$

where $\cdot$ denotes the scalar product and $\vec{x}_A$, $\vec{y}_A$, and $\vec{z}_A$ are the unit vectors defining the orientation frame $[A]$. Given the frames A and B, we describe the coordinate transformation from B to A by use of the *rotation matrix* ${}^A\mathbf{R}_B \in \mathrm{SO}(3)$. Note that this coordinate transformation is only depending on the orientation and is independent of the positions of $\mathbf{o}_A$ and $\mathbf{o}_B$. To describe both the position and orientation of frame B relative to frame A, hereafter referred to as the *pose*, we use the 4×4 *homogeneous transformation* matrix ${}^A\mathbf{H}_B \in \mathrm{SE}(3)$ given by

$$ {}^A\mathbf{H}_B \coloneqq \begin{bmatrix} {}^A\mathbf{R}_B & {}^A\mathbf{o}_B \\ \mathbf{0} & 1 \end{bmatrix}. \tag{2.68} $$

Given a point $\mathbf{p}$, we can write the coordinate vector ${}^{B}\mathbf{p}$ into ${}^{A}\mathbf{p}$ by

$$ {}^{A}\bar{\mathbf{p}} = {}^{A}\mathbf{H}_{B}\,{}^{B}\bar{\mathbf{p}}, \tag{2.69} $$

where ${}^{A}\bar{\mathbf{p}}$ and ${}^{B}\bar{\mathbf{p}}$ represent the homogeneous representation of the coordinate vectors ${}^{A}\mathbf{p}$ and ${}^{B}\mathbf{p}$, respectively given by

$$ {}^{A}\bar{\mathbf{p}} = \begin{bmatrix} {}^{A}\mathbf{p} \\ 1 \end{bmatrix} \quad \text{and} \quad {}^{B}\bar{\mathbf{p}} = \begin{bmatrix} {}^{B}\mathbf{p} \\ 1 \end{bmatrix}. \tag{2.70} $$

2.4.2 Notation of rotational velocity

Any point $\mathbf{p}$ attached to a rigid-body with coordinate frame B follows a path in spacial coordinates given by

$$ {}^{A}\mathbf{p}(t) = {}^{A}\mathbf{R}(t)\,{}^{B}\mathbf{p}. \tag{2.71} $$

The velocity of ${}^{A}\mathbf{p}(t)$ is then given by the time derivative of (2.71), resulting in

$$ {}^{A}\dot{\mathbf{p}} = \frac{d}{dt}({}^{A}\mathbf{p}(t)) = {}^{A}\dot{\mathbf{R}}(t)\,{}^{B}\mathbf{p}. \tag{2.72} $$

It follows that we can rewrite equation (2.72) as

$$ {}^{A}\dot{\mathbf{p}} = {}^{A}\dot{\mathbf{R}}_{B}\,{}^{A}\mathbf{R}_{B}^{-1}\,{}^{A}\mathbf{R}_{B}\,{}^{B}\mathbf{p}, \tag{2.73} $$

where we omitted the time dependencies for the sake of brevity. We can now conveniently define the *instantaneous spatial angular velocity* as ${}^{A}\boldsymbol{\omega}_{A,B} \in \mathbb{R}^3$, given by

$$ {}^{A}\boldsymbol{\omega}_{A,B}^{\wedge} \coloneqq {}^{A}\dot{\mathbf{R}}_{B}\,{}^{A}\mathbf{R}_{B}^{-1}, \tag{2.74} $$

where we used $(\cdot)^{\wedge}$ as the *hat*-operator to write and element of $\mathbb{R}^3$ as a skew-symmetric matrix according to (2.49). Similar to (2.74) we define the *instantaneous body angular velocity* as ${}^{B}\boldsymbol{\omega}_{A,B} \in \mathbb{R}^3$, which is given by

$$ {}^{B}\boldsymbol{\omega}_{A,B}^{\wedge} \coloneqq {}^{A}\mathbf{R}_{B}^{-1}\,{}^{A}\dot{\mathbf{R}}_{B}. \tag{2.75} $$

2.4.3 Notation of rigid-body velocity

We now consider both angular and linear velocities of a frame. Consider the homogeneous matrix mapping a point expressed in terms of frame B to a point expressed in terms of frame A, as given by (2.68). Similar to the property that multiplying ${}^{A}\dot{\mathbf{R}}_{B}$ by ${}^{A}\mathbf{R}_{B}^{-1}$ results in a skew-symmetric matrix, we can apply a similar operation to the homogeneous matrix. Multiplying ${}^{A}\dot{\mathbf{H}}_{B}$ from the right by the inverse of ${}^{A}\mathbf{H}_{B}$ leads to

$$ \begin{aligned} {}^{A}\dot{\mathbf{H}}_{B}\,{}^{A}\mathbf{H}_{B}^{-1} &= \begin{bmatrix} {}^{A}\dot{\mathbf{R}}_{B} & {}^{A}\dot{\mathbf{o}}_{B} \\ \mathbf{0} & 0 \end{bmatrix} \begin{bmatrix} {}^{A}\mathbf{R}_{B}^{T} & -{}^{A}\mathbf{R}_{B}^{T}\,{}^{A}\mathbf{o}_{B} \\ \mathbf{0} & 1 \end{bmatrix} \\ &= \begin{bmatrix} {}^{A}\dot{\mathbf{R}}_{B}\,{}^{A}\mathbf{R}_{B}^{T} & {}^{A}\dot{\mathbf{o}}_{B} - {}^{A}\dot{\mathbf{R}}_{B}\,{}^{A}\mathbf{R}_{B}^{T}\,{}^{A}\mathbf{o}_{B} \\ \mathbf{0} & 0 \end{bmatrix} \\ &= \begin{bmatrix} {}^{A}\boldsymbol{\omega}_{A,B}^{\wedge} & {}^{A}\boldsymbol{v}_{A,B} \\ \mathbf{0} & 0 \end{bmatrix}, \end{aligned} \tag{2.76} $$

which takes the same form of (2.55), and is an element of $\mathfrak{se}(3)$. The *right trivialized* velocity of frame B with respect to frame A is then defined as

$$ {}^{A}\mathbf{v}_{A,B} \coloneqq \begin{bmatrix} {}^{A}\boldsymbol{v}_{A,B} \\ {}^{A}\boldsymbol{\omega}_{A,B} \end{bmatrix} \in \mathbb{R}^6. \tag{2.77} $$

Similarly, by multiplying on the left, one obtains

$$\begin{aligned} {}^{A}\mathbf{H}_{B}^{-1}{}^{A}\dot{\mathbf{H}}_{B} &= \begin{bmatrix} {}^{A}\mathbf{R}_{B}^{T} & -{}^{A}\mathbf{R}_{B}^{T}{}^{A}\mathbf{o}_{B} \\ \mathbf{0} & 1 \end{bmatrix} \begin{bmatrix} {}^{A}\dot{\mathbf{R}}_{B} & {}^{A}\dot{\mathbf{o}}_{B} \\ \mathbf{0} & 0 \end{bmatrix} \\ &= \begin{bmatrix} {}^{A}\mathbf{R}_{B}^{T}{}^{A}\dot{\mathbf{R}}_{B} & {}^{A}\mathbf{R}_{B}^{T}{}^{A}\dot{\mathbf{o}}_{B} \\ \mathbf{0} & 0 \end{bmatrix} \\ &= \begin{bmatrix} {}^{B}\boldsymbol{\omega}_{A,B}^{\wedge} & {}^{B}\boldsymbol{v}_{A,B} \\ \mathbf{0} & 0 \end{bmatrix}, \end{aligned} \tag{2.78}$$

which results in the *left trivialized* velocity of frame B with respect to frame A, given by

$${}^{B}\mathbf{v}_{A,B} \coloneqq \begin{bmatrix} {}^{B}\boldsymbol{v}_{A,B} \\ {}^{B}\boldsymbol{\omega}_{A,B} \end{bmatrix} \in \mathbb{R}^{6}. \tag{2.79}$$

The relation between the right and left trivialized velocity is given by the so called adjoint map [100]. We will denote this by ${}^{A}\mathbf{X}_{B}$ and define it as

$${}^{A}\mathbf{X}_{B} \coloneqq \begin{bmatrix} {}^{A}\mathbf{R}_{B} & {}^{A}\mathbf{o}_{B}^{\wedge}{}^{A}\mathbf{R}_{B} \\ \mathbf{0} & {}^{A}\mathbf{R}_{B} \end{bmatrix} \in \mathbb{R}^{6\times 6}, \tag{2.80}$$

such that the right and left trivialized velocities are related through

$${}^{A}\mathbf{v}_{A,B} = {}^{A}\mathbf{X}_{B}{}^{B}\mathbf{v}_{A,B}. \tag{2.81}$$

2.4.4 Wrench notation

We can write the coordinates of a given wrench with respect to a frame B as

$${}_{B}\mathbf{f} \coloneqq \begin{bmatrix} {}_{B}\boldsymbol{f} \\ {}_{B}\boldsymbol{\tau} \end{bmatrix} \in \mathbb{R}^{6}, \tag{2.82}$$

where ${}_{B}\boldsymbol{f}$ is the translational component of the wrench and ${}_{B}\boldsymbol{\tau}$ the angular component. This wrench can be expressed in terms of another frame A by the use of the wrench coordinate transformation ${}_{A}\mathbf{X}^{B}$ such that

$${}_{A}\mathbf{f} = {}_{A}\mathbf{X}^{B}{}_{B}\mathbf{f}. \tag{2.83}$$

This coordinate transformation is closely related to the adjoint map ${}^{A}\mathbf{X}_{B}$, given by (2.80), and is defined as

$${}_{A}\mathbf{X}^{B} \coloneqq {}^{B}\mathbf{X}_{A}^{T} = \begin{bmatrix} {}^{A}\mathbf{R}_{B} & \mathbf{0} \\ -{}^{A}\mathbf{R}_{B}{}^{B}\mathbf{o}_{A}^{\wedge} & {}^{A}\mathbf{R}_{B} \end{bmatrix} = \begin{bmatrix} {}^{A}\mathbf{R}_{B} & \mathbf{0} \\ {}^{A}\mathbf{o}_{B}^{\wedge}{}^{A}\mathbf{R}_{B} & {}^{A}\mathbf{R}_{B} \end{bmatrix}. \tag{2.84}$$

In some cases, one needs to express the wrench with respect to a frame B with the orientation in terms of frame A. This is for example the case with gravity, where the direction of the gravity force is independent of time with respect to the inertial frame A. This is possible by introducing a new frame by combining the frames A and B. This frame is given as $B[A] \coloneqq (\mathbf{o}_B, [A])$, hence, the frame whose origin coincides with the origin of B and whose orientation coincides with the orientation of A. In such a case, we can express the wrench as ${}_{B[A]}\mathbf{f}$. The relation between ${}_{B[A]}\mathbf{f}$ and ${}_{B}\mathbf{f}$ is then given by

$${}_{B}\mathbf{f} = {}_{B}\mathbf{X}^{B[A]}{}_{B[A]}\mathbf{f}, \tag{2.85}$$

where, from (2.84), we get

$${}_{B}\mathbf{X}^{B[A]} = \begin{bmatrix} {}^{A}\mathbf{R}_{B}^{T} & \mathbf{0} \\ \mathbf{0} & {}^{A}\mathbf{R}_{B}^{T} \end{bmatrix}. \tag{2.86}$$

Finally, the time derivative of a wrench coordinate transformation ${}_{A}\mathbf{X}^{B}$ is given by

$${}_{A}\dot{\mathbf{X}}^{B} = {}_{A}\mathbf{X}^{B}\ {}^{B}\mathbf{v}_{A,B}\bar{\times}^{*}, \tag{2.87}$$

where $\bar{\times}^*$ represents the dual cross-product between a twist and a wrench as in [88] and whose matrix representation is given by

$$ {}^{B}\mathbf{v}_{A,B}\bar{\times}^* = \begin{bmatrix} {}^{B}\boldsymbol{\omega}^{\wedge}_{A,B} & \mathbf{0} \\ {}^{B}\boldsymbol{v}^{\wedge}_{A,B} & {}^{B}\boldsymbol{\omega}^{\wedge}_{A,B} \end{bmatrix}. \tag{2.88} $$

With the notation described in this section, we are able to express the position, orientation, and velocity of the cuboid with respect to the camera frame A, as is illustrated in Figure 2.7. In the next section, we will show how, for a given pose of the object, we can find its representation in the image plane.

2.5 Image formation

In this section, we will first give a short review on the basics of image formation, that is, how 3D objects are observed by a camera under the pinhole camera principle. Next, this principle is applied to cuboids to give their representation in the image plane.

2.5.1 Perspective projection

The image plane is considered to be the plane containing the sensing elements of the camera. Points on this plane are often expressed in terms of pixel coordinates, where the origin of the coordinate frame is located in the upper left corner of the image plane. In Figure 2.8, the frame is indicated by K. The camera coordinate frame, however, is located at the camera center, which is positioned a distance f behind the image plane, where f denotes the *focal length* of the camera. The assumption is made that this frame is coinciding with the world coordinate frame A for simplicity. This means that the x- and y-axis of frame A span a plane parallel to the image plane. Furthermore, the z-axis of A is the *principal axis*, which coincides with the image plane at the *principal point* denoted by (u_0, v_0).

Consider the point ${}^{A}\mathbf{p}$ in Figure 2.8 whose coordinates are $\begin{bmatrix} {}^{A}\mathbf{p}_x & {}^{A}\mathbf{p}_y & {}^{A}\mathbf{p}_z \end{bmatrix}^T$. Under the pinhole camera principle, the relation between the point ${}^{A}\mathbf{p}$ and the point ${}^{K}\mathbf{q} = \begin{bmatrix} {}^{K}\mathbf{q}_x & {}^{K}\mathbf{q}_y \end{bmatrix}^T$ in the image plane, is given by the equations of perspective projection as in [101, Chapter 11] and [86]. The values for ${}^{K}\mathbf{q}_x$ and ${}^{K}\mathbf{q}_y$ can then be obtained by

$$ {}^{K}\mathbf{q}_x = f\frac{{}^{A}\mathbf{p}_x}{{}^{A}\mathbf{p}_z} \tag{2.89} $$

and

$$ {}^{K}\mathbf{q}_y = f\frac{{}^{A}\mathbf{p}_y}{{}^{A}\mathbf{p}_z}, \tag{2.90} $$

respectively. As the sensing elements in the image plane form a discrete set of pixels, the coordinates ${}^{K}\mathbf{q}_x$ and ${}^{K}\mathbf{q}_y$ need to be converted into pixel coordinates. Thereto, the x and y-values of $\mathbf{q}$ need to be rounded to integer values and translated over a distance u_0 and v_0, respectively, to express them in terms of the image coordinate frame. It is often the case that pixels are not square, such that the width and height of a pixel need to be compensated for by the scaling parameters s_x and s_y, respectively. This means that the conversion to pixel coordinates is given by

$$ u = s_x {}^{K}\mathbf{q}_x + u_0 \quad \text{and} \quad v = s_y {}^{K}\mathbf{q}_y + v_0. \tag{2.91} $$

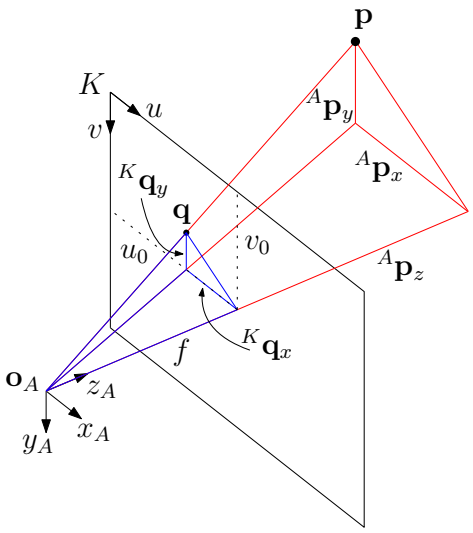


Figure 2.8: Schematic representation of a point ${}^{A}\mathbf{p}$ in the 3D space, projected to point ${}^{K}\mathbf{q}$ in the image plane.

In matrix form, the conversion from the point ${}^{A}\mathbf{p}$ to the image plane is given by

$$ {}^{A}\mathbf{p}_z \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \mathbf{K} \begin{bmatrix} {}^{A}\mathbf{p}_x \\ {}^{A}\mathbf{p}_y \\ {}^{A}\mathbf{p}_z \end{bmatrix}, \tag{2.92} $$

where $\mathbf{K}$ is known as the *camera intrinsic matrix*, given by

$$ \mathbf{K} = \begin{bmatrix} \alpha_x & 0 & u_0 \\ 0 & \alpha_y & v_0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{2.93} $$

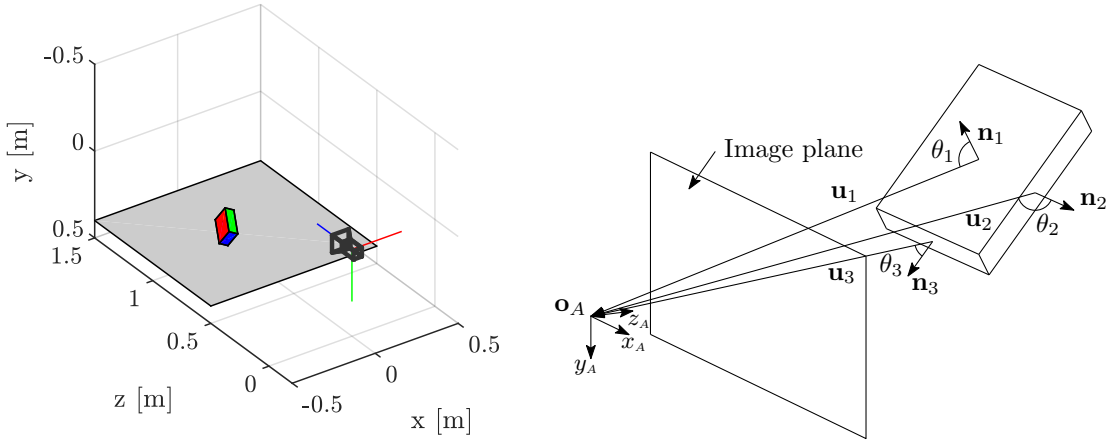
In (2.93), α_x and α_y represent the focal length in terms of pixel coordinates, which are given by $\alpha_x = f s_x$ and $\alpha_y = f s_y$, respectively.

2.5.2 Projection of the cuboid in the image plane

We will now use the theory of the previous paragraph to illustrate how the cuboid is represented in the image plane. This will be used for computing the likelihood, discussed in Section 4.2, as well as creating artificial images for simulations, discussed in Chapter 5. Consider Figure 2.9a, where the cuboid is shown together with the camera in a 3D space. In order to determine its representation in the image plane, first the visible faces of the cuboid from the camera need to be determined. This computation is done by the use of the normal vector of each face, denoted by $\mathbf{n}_i$, and the vectors pointing from each face center to the world coordinate frame, denoted by $\mathbf{u}_i$. Figure 2.9b schematically shows a possible state of the cuboid together with the image plane and the world coordinate frame A. We now state, that a face of the cuboid is visible if the angle between $\mathbf{n}_i$ and $\mathbf{u}_i$ is less than $\pm\frac{1}{2}\pi$ rad. This requirement can be written as $\cos(\theta_i) > 0$, where θ_i, the angle between the two vectors $\mathbf{n}_i$ and $\mathbf{u}_i$, is given by

$$ \cos(\theta_i) = \frac{(\mathbf{n}_i \cdot \mathbf{u}_i)}{(\|\mathbf{n}_i\| \cdot \|\mathbf{u}_i\|)}, \tag{2.94} $$

with $\|\cdot\|$ the Euclidean norm of the vector. This computation is done for each face of the cuboid, resulting in N_f number of visible faces, which are at most three faces ($N_f = 3$).



(a) **(b)**

Figure 2.9: Determining the visible faces of the cuboid. A pose of the cuboid in 3D space (a) and a schematic representation of the vectors used to compute the visible faces in the image plane (b).

An example is given for the cuboid shown in Figure 2.9a. After determining its visible faces, we can project the points on these faces to the image plane by the use of the projection equations of (2.92). This gives the image shown in Figure 2.10.

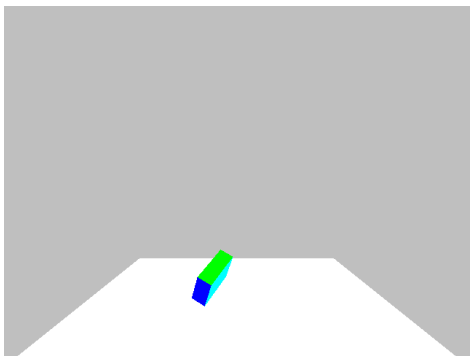


Figure 2.10: Representation of the cuboid in the image plane. The pose of the cuboid corresponds to the pose shown in Figure 2.9a.

2.6 Concluding remark

In this chapter, we provided the reader with necessary background information on particle filters, Lie group theory, multibody dynamics notation, and image formation. In the next chapter, we will show the generalization of the unscented particle filter discussed in Section 2.1 to fit with the Lie groups of $\mathcal{P}$ and $\mathcal{S}$ discussed in Section 2.2.

Chapter 3

The Geometric Unscented Particle Filter for Rigid Body Dynamics

In the previous chapter, we discussed the UPF for the *Euclidean* case, hence for state vectors of the form $\mathbf{x} \in \mathbb{R}^n$. However, as discussed in Section 2.3, we use a parametrization of the state in $\mathcal{S}$. We therefore require a generalization of the UPF in order to cope for the Lie group structure of our parametrization, and in this chapter we show how this generalization is done. This will result in the *Geometric Unscented Particle Filter* (GUPF), which is considered as one of the main contributions of this thesis.

3.1 Initialization of the particle filter

As stated in Section 2.1, we assume an initial uncertainty distribution $p(\mathbf{x}_0)$ for the state. This distribution is defined by its mean $\bar{\mathbf{x}}_0$ and covariance $\boldsymbol{P}_0^{\mathbf{x}}$ given by

$$\bar{\mathbf{x}}_0 = ({}^A\mathbf{R}_B, {}^A\mathbf{o}_B, {}^B\boldsymbol{v}_{A,B}, {}^B\boldsymbol{\omega}_{A,B}) \in \mathcal{S} \tag{3.1}$$

and

$$\boldsymbol{P}_0^{\mathbf{x}} = \begin{bmatrix} \boldsymbol{P}^{\mathbf{R}} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{P}^{\mathbf{o}} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \boldsymbol{P}^{\boldsymbol{v}} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \boldsymbol{P}^{\boldsymbol{\omega}} \end{bmatrix} \in \mathfrak{s} \otimes \mathfrak{s}, \tag{3.2}$$

where the diagonal elements of $\boldsymbol{P}_0^{\mathbf{x}}$ represent the covariances corresponding to the uncertainty in orientation, position, linear velocity, and angular velocity. As discussed in Section 2.2.4, the covariance is defined in the tangent space at the point $\bar{\mathbf{x}}_0 \in \mathcal{S}$. To initialize the particle filter, we will create a set of $i = 1, \ldots, N$ particles $\mathbf{x}^{(i)}$ around this initial state mean such that these particles represent the initial state distribution. These particles are created according to

$$\mathbf{x}_0^{(i)} = \bar{\mathbf{x}}_0 \operatorname{Exp}\left(\boldsymbol{\chi}^{(i)}\right), \tag{3.3}$$

where

$$\boldsymbol{\chi}^{(i)} = \left(\boldsymbol{\xi}^{(i)\mathbf{R}}, \boldsymbol{\xi}^{(i)\mathbf{o}}, \boldsymbol{\xi}^{(i)\boldsymbol{v}}, \boldsymbol{\xi}^{(i)\boldsymbol{\omega}}\right) \in \mathfrak{s}. \tag{3.4}$$

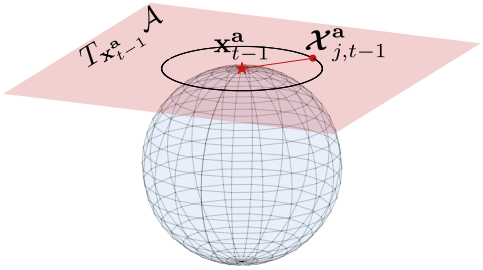


Figure 3.1: Example of one sigma point of the augmented state, in the tangent space at $\mathbf{x}^{\mathbf{a}}_{t-1}$. The radius of the circle is defined by the matrix square root of the scaled augmented covariance matrix, as in (3.11). The zeroth sigma point $\boldsymbol{\mathcal{X}}^{\mathbf{a}}_{0,t-1}$ is located at the origin of the tangent space, which represents $\mathbf{x}^{\mathbf{a}}_{t-1}$.

In (3.4), the values for $\boldsymbol{\xi}$ are sampled according to

$$\begin{aligned}
\boldsymbol{\xi}^{(i)\mathbf{R}} &\sim \mathcal{N}(\mathbf{0}, \boldsymbol{P}^{\mathbf{R}}), \qquad i = 1, \dots, N, \quad \boldsymbol{P}^{\mathbf{R}} \in \mathbb{R}^{3\times 3}, \\
\boldsymbol{\xi}^{(i)\mathbf{o}} &\sim \mathcal{N}(\mathbf{0}, \boldsymbol{P}^{\mathbf{o}}), \qquad i = 1, \dots, N, \quad \boldsymbol{P}^{\mathbf{o}} \in \mathbb{R}^{3\times 3}, \\
\boldsymbol{\xi}^{(i)\boldsymbol{v}} &\sim \mathcal{N}(\mathbf{0}, \boldsymbol{P}^{\boldsymbol{v}}), \qquad i = 1, \dots, N, \quad \boldsymbol{P}^{\boldsymbol{v}} \in \mathbb{R}^{3\times 3}, \\
\boldsymbol{\xi}^{(i)\boldsymbol{\omega}} &\sim \mathcal{N}(\mathbf{0}, \boldsymbol{P}^{\boldsymbol{\omega}}), \qquad i = 1, \dots, N, \quad \boldsymbol{P}^{\boldsymbol{\omega}} \in \mathbb{R}^{3\times 3}.
\end{aligned} \tag{3.5}$$

By substituting (3.4) into (3.3) and computing the exponential mapping, the particles $\mathbf{x}^{(i)}_0$ are thus given by

$$\mathbf{x}^{(i)}_0 = \left(\mathbf{R}\mathrm{Exp}\left(\boldsymbol{\xi}^{(i)\mathbf{R}}\right), \mathbf{o} + \boldsymbol{\xi}^{(i)\mathbf{o}}, \boldsymbol{v} + \boldsymbol{\xi}^{(i)\boldsymbol{v}}, \boldsymbol{\omega} + \boldsymbol{\xi}^{(i)\boldsymbol{\omega}}\right). \tag{3.6}$$

The particles $\mathbf{x}^{(i)}_0$ are thus created by sampling N elements of $\boldsymbol{\chi}^{(i)}$ in the tangent space at $\bar{\mathbf{x}}_0$, and mapping them to the Lie group by the use of (3.3). Each particle thus has a state mean given by (3.6) and a state covariance given by (3.1), normally distributed such that we can write $p(\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_0, \boldsymbol{P}^{\mathbf{x}}_0)$ for each particle.

3.2 Creating the sigma points

We now consider a single particle at time $t-1$ and disregard the superscript $(\cdot)^{(i)}$. As discussed in Section 2.1, the state mean and state covariance are augmented with the process noise and observation noise parameters. We therefore define the group of the augmented state as

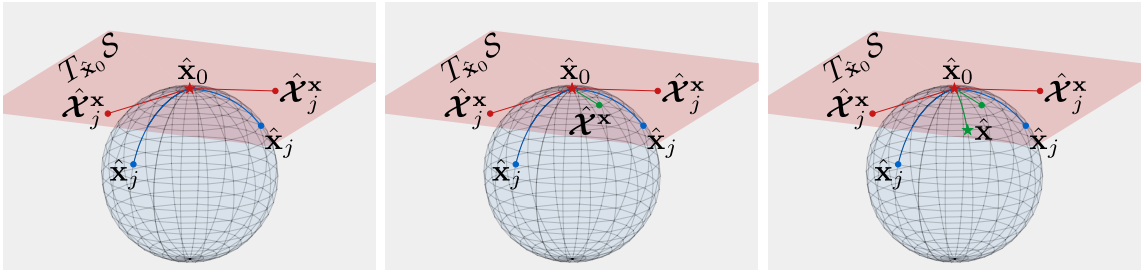
$$\mathcal{A} \coloneqq \mathcal{S} \times \mathcal{S} \times \mathcal{P}, \tag{3.7}$$

such that we can write an element $\mathbf{x}^{\mathbf{a}} \in \mathcal{A}$ as

$$\mathbf{x}^{\mathbf{a}} = (\mathbf{x}, \mathbf{m}, \mathbf{n}) \in \mathcal{A}, \tag{3.8}$$

where $\mathbf{x} \in \mathcal{S}$ is the state, $\mathbf{m} \in \mathcal{S}$ the process noise, and $\mathbf{n} \in \mathcal{P}$ the observation noise. The Lie algebra of $\mathcal{A}$, denoted by $\mathfrak{a}$, is then given by

$$\mathfrak{a} \coloneqq \mathfrak{s} \otimes \mathfrak{s} \otimes \mathfrak{p}. \tag{3.9}$$



(a) **(b)** **(c)**

Figure 3.2: Iterative procedure of computing the mean on of the predicted state $\hat{\mathbf{x}} \in \mathcal{S}$. **(a):** The propagated sigma points $\hat{\mathbf{x}}_j$ are mapped to the tangent space at $\hat{\mathbf{x}}_0$ resulting in the points $\hat{\boldsymbol{\mathcal{X}}}^{\mathbf{x}}_j$. **(b):** The mean value of the points $\hat{\boldsymbol{\mathcal{X}}}^{\mathbf{x}}_j$ is computed in the tangent space, resulting in $\hat{\boldsymbol{\mathcal{X}}}^{\mathbf{x}}$ (indicated in green), which may not be at the origin of the tangent space. **(c):** The point $\hat{\boldsymbol{\mathcal{X}}}^{\mathbf{x}}$ is mapped to the Lie group, resulting in $\hat{\mathbf{x}}$ (indicated by the green star), which is used as new mean value. The steps (a-c) are then repeated until $\|\hat{\boldsymbol{\mathcal{X}}}^{\mathbf{x}}\|$ is converged to a predefined threshold. The subscripts concerning time are omitted for the sake of readability.

We will define the augmented state mean and covariance in the tangent space at a point $\mathbf{x}^{\mathbf{a}}_{t-1} = (\mathbf{x}_{t-1}, \mathbf{m}, \mathbf{n}) \in \mathcal{A}$ as

$$\bar{\boldsymbol{\mathcal{X}}}^{\mathbf{a}}_{t-1} = \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix} \in \mathfrak{a} \quad \text{and} \quad \boldsymbol{P}^{\mathbf{a}}_{t-1} = \begin{bmatrix} \boldsymbol{P}^{\mathbf{x}} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{Q} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \boldsymbol{R} \end{bmatrix} \in \mathfrak{a} \otimes \mathfrak{a}, \tag{3.10}$$

respectively, where $\boldsymbol{P}^{\mathbf{x}} \in \mathfrak{s} \otimes \mathfrak{s}$ is the state covariance, $\boldsymbol{Q} \in \mathfrak{s} \otimes \mathfrak{s}$ the process noise covariance, and $\boldsymbol{R} \in \mathfrak{p} \otimes \mathfrak{p}$ the observation noise covariance. The sigma points are defined in the tangent space of $\mathcal{A}$ at the point $\mathbf{x}^{a}_{t-1}$ and are given by

$$\begin{aligned} \boldsymbol{\mathcal{X}}^{\mathbf{a}}_{0,t-1} &= \bar{\boldsymbol{\mathcal{X}}}^{\mathbf{a}}_{t-1}, \\ \boldsymbol{\mathcal{X}}^{\mathbf{a}}_{j,t-1} &= \bar{\boldsymbol{\mathcal{X}}}^{\mathbf{a}}_{t-1} + \left(\sqrt{(n_a + \lambda)\boldsymbol{P}^{\mathbf{a}}_{t-1}}\right)_j \quad \text{for} \quad j = 1, \dots, n_a, \\ \boldsymbol{\mathcal{X}}^{\mathbf{a}}_{j,t-1} &= \bar{\boldsymbol{\mathcal{X}}}^{\mathbf{a}}_{t-1} - \left(\sqrt{(n_a + \lambda)\boldsymbol{P}^{\mathbf{a}}_{t-1}}\right)_j \quad \text{for} \quad j = n_a + 1, \dots, 2n_a, \end{aligned} \tag{3.11}$$

where $\left(\sqrt{(n_a + \lambda)\boldsymbol{P}^{\mathbf{a}}_{t-1}}\right)_j$ denotes the $j-$th column of the matrix square root of $(n_a+\lambda)\boldsymbol{P}^{\mathbf{a}}_{t-1}$ as in [61,66,70,102]. Furthermore, $\lambda = {\alpha_k}^2(n_a+\zeta)-n_a$ where $\zeta = 0$ and $\alpha_k = 0$ are parameters to scale the sigma points as in [61] and n_a denotes the dimension of the augmented state, which in this case is $n_a = 30$. A direct result of the augmented state is that each sigma point can now be expressed as

$$\boldsymbol{\mathcal{X}}^{\mathbf{a}}_{j,t-1} = \begin{bmatrix} \left(\boldsymbol{\mathcal{X}}^{\mathbf{x}}_{j,t-1}\right)^T & \left(\boldsymbol{\mathcal{X}}^{\mathbf{m}}_{j,t-1}\right)^T & \left(\boldsymbol{\mathcal{X}}^{\mathbf{n}}_{j,t-1}\right)^T \end{bmatrix}^T, \tag{3.12}$$

where $\boldsymbol{\mathcal{X}}^{\mathbf{x}}_{j,t-1} \in \mathfrak{s}$ is the sigma point of the state, $\boldsymbol{\mathcal{X}}^{\mathbf{m}}_{j,t-1} \in \mathfrak{s}$ the sigma point of the process noise, and $\boldsymbol{\mathcal{X}}^{\mathbf{n}}_{j,t-1} \in \mathfrak{p}$ the sigma point of the observation noise. Each sigma point is weighted according to (2.25), hence similar to the Euclidean case and similar to the work presented in [70], such that each particle contains a set of sigma points denoted by $\left\{\boldsymbol{\mathcal{X}}^{\mathbf{a}}_{j,t-1}, W_j\right\}$ for $j = 0, \dots, 2n_a$.

3.3 Propagating the sigma points

We now propagate the sigma points of the state, $\boldsymbol{\mathcal{X}}^{\mathbf{x}}_{j,t-1}$, to obtain the predicted state sigma points $\hat{\mathbf{x}}_{j,t|t-1}$ on the Lie group $\mathcal{S}$. Propagation is done through the nonlinear motion model $f : \mathcal{S} \to \mathcal{S}$, describing the dynamics of the cuboid and we will elaborate on this function in Chapter 4. For now, we assume that this function is known, such that propagating the sigma points through this function yields

$$\hat{\mathbf{x}}_{j,t|t-1} = f\left(\mathbf{x}_{t-1}\operatorname{Exp}\left(\boldsymbol{\mathcal{X}}^{\mathbf{x}}_{j,t-1}\right)\right)\operatorname{Exp}\left(\boldsymbol{\mathcal{X}}^{\mathbf{m}}_{j,t-1}\right) \quad \text{for} \quad j = 0, \ldots, 2n_a, \tag{3.13}$$

where $\mathbf{x}_{t-1}$ is the state mean, $\boldsymbol{\mathcal{X}}^{\mathbf{x}}_{j,t-1}$ the sigma points of the state, and $\boldsymbol{\mathcal{X}}^{\mathbf{m}}_{j,t-1}$ the sigma points of the process noise. Note that, in (3.13), the sigma points of the state, defined in the tangent space of the mean $\mathbf{x}_{t-1}$, are first mapped to the group by the exponential mapping before they are propagated through the motion model. Where in (2.34) the sigma points of the process noise were additive, in (3.13) this operation is performed by mapping them to the Lie group. This results in $2n_a + 1$ predicted sigma points denoted by $\hat{\mathbf{x}}_{j,t|t-1}$ on the Lie group. From these sigma points we can again compute a state mean and state covariance, which will form the *prior distribution*. In order to compute this mean and covariance, we follow the procedure as discussed in Section 2.2.4.

As stated in Section 2.2.4, we need an initial guess for the mean value. Here, our first guess for the mean is given by the sigma point $\hat{\mathbf{x}}_{0,t|t-1}$. On this point on the Lie group, a tangent space is considered and all the other sigma points are mapped to this tangent space by

$$\hat{\boldsymbol{\mathcal{X}}}^{\mathbf{x}}_{j,t|t-1} = \operatorname{Log}\left(\left(\hat{\mathbf{x}}_{0,t|t-1}\right)^{-1}\hat{\mathbf{x}}_{j,t|t-1}\right) \quad \text{for} \quad j = 0, \ldots, 2n_a, \tag{3.14}$$

which is schematically depicted in Figure 3.2a. Next, the mean point parametrization on the tangent space is computed similar to (2.62) such that

$$\hat{\boldsymbol{\mathcal{X}}}^{\mathbf{x}}_{t|t-1} = \sum_{j=0}^{2n_a} W_j^{(m)}\hat{\boldsymbol{\mathcal{X}}}^{\mathbf{x}}_{j,t|t-1}. \tag{3.15}$$

This mean may differ from the origin of the tangent space, such that the mean can be updated by

$$\hat{\mathbf{x}}_{t|t-1} = \hat{\mathbf{x}}_{0,t|t-1}\operatorname{Exp}\left(\hat{\boldsymbol{\mathcal{X}}}^{\mathbf{x}}_{t|t-1}\right), \tag{3.16}$$

which is depicted in Figures 3.2b and 3.2c, respectively. Next, $\hat{\mathbf{x}}_{t|t-1}$ is used as new mean value and the steps of (3.14), (3.15), and (3.16) are repeated until $\|\hat{\boldsymbol{\mathcal{X}}}^{\mathbf{x}}_{t|t-1}\|$ is smaller than a pre-defined convergence threshold, resulting in a final estimate of the mean $\hat{\mathbf{x}}_{t|t-1}$. The state covariance can then be computed similar to (2.64) to obtain

$$\hat{\boldsymbol{P}}^{\mathbf{x}}_{t|t-1} = \sum_{j=0}^{2n_a} W_j^{(c)}\operatorname{Log}\left(\left(\hat{\mathbf{x}}_{t|t-1}\right)^{-1}\hat{\mathbf{x}}_{j,t|t-1}\right)\operatorname{Log}\left(\left(\hat{\mathbf{x}}_{t|t-1}\right)^{-1}\hat{\mathbf{x}}_{j,t|t-1}\right)^T, \tag{3.17}$$

with $\hat{\boldsymbol{P}}^{\mathbf{x}}_{t|t-1} \in \mathfrak{s} \otimes \mathfrak{s}$. The predicted state mean and covariance give, for each particle, the *prior distribution* given by

$$p(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\hat{\mathbf{x}}_{t|t-1}, \hat{\boldsymbol{P}}^{\mathbf{x}}_{t|t-1}). \tag{3.18}$$

3.4 Obtaining a measurement

As discussed in Section 2.1.4, the next step in the GUPF is to incorporate a measurement. In our case, such a measurement would give the 3D position and orientation of the cuboid at a given time. This measurement could be the result of some image-processing algorithm, point clouds or motion capture systems. In [54], the pose of an object is estimated by the use of depth images and in [85] the 3D position of a ball is reconstructed from fitting ellipses over color blobs detected in the 2D image. However, creating a similar reconstruction for cuboids was beyond the scope of this thesis and here we rely on different methods to obtain a measurement.

In this section, we will describe two different measurements. One method requires the computation of the likelihood of all the particles and computes the measurement as a weighted mean of the pose parameters of these particles. This will result in the GUPF-L, the Geometric Unscented Particle Filter using the likelihood to compute a measurement. Another option is to assume a method as in [54] or [85] is available for cuboids, and simulate this measurement by using ground truth data with additive Gaussian noise representing the measurement noise. This will result in the GUPF-Z, the Geometric Unscented Particle Filter making use of a given measurement $\mathbf{Z}$. As a result, the GUPF-Z is not directly applicable, as it would require the development of a measurement method as in [54] or [85]. However, the GUPF-Z will be considered to show its potential with respect to the GUPF-L. In the following sections, both measurement methods will be discussed.

3.4.1 Measurement from likelihood estimation

To obtain a measurement from the particles themselves, we will use the predicted set of particles $\hat{\mathbf{x}}_{t|t-1}^{(i)}$ and consider their pose parameters $\mathbf{H}^{(i)} = (\mathbf{R}^{(i)}, \mathbf{o}^{(i)}) \in \mathcal{P}$. Each pose is then evaluated by the *likelihood function* according to

$$L^{(i)} = \mathcal{L}(\mathbf{H}^{(i)}), \tag{3.19}$$

with $\mathcal{L} : \mathcal{P} \to \mathbb{R}$ the likelihood function and $L^{(i)}$ the likelihood assigned to particle i. We will explain the likelihood function in detail in Chapter 4 and assume for now this function is able to assign a likelihood to a particle by using its pose parameters and evaluating the representation of this pose in the image plane. In order to compute a weighted average, we follow the same procedure as in Section 2.2.4 but make the weight of each particle proportional to its likelihood. We again start with an initial guess for the mean value, here chosen as the pose parameters of the particle with the highest likelihood. This yields

$$\tilde{\mathbf{Z}}_t = \underset{i=1,\dots,N}{\arg\max} \left(\mathcal{L}\left(\mathbf{H}^{(i)} \right) \right) \in \mathcal{P}. \tag{3.20}$$

Following the procedure of Section 2.2.4, the pose parameters of all the other particles are mapped to the tangent space of $\mathcal{P}$ at $\tilde{\mathbf{Z}}_t$ by

$$\boldsymbol{\mathcal{H}}^{(i)} = \operatorname{Log}\left(\tilde{\mathbf{Z}}_t^{-1} \mathbf{H}^{(i)} \right). \tag{3.21}$$

The weighted average is then computed in the tangent space of $\mathcal{P}$ at $\tilde{\mathbf{Z}}_t$ by

$$\boldsymbol{\mathcal{Z}}_t = \sum_{i=1}^{N} \tilde{L}^{(i)} \boldsymbol{\mathcal{H}}^{(i)}, \tag{3.22}$$

where $\tilde{L}^{(i)}$ denotes the normalized weight of particle (i) given by

$$\tilde{L}^{(i)} = \frac{L^{(i)}}{\sum_{j=1}^{N} L^{(j)}}. \tag{3.23}$$

Again, an update might be needed if $\boldsymbol{\mathcal{Z}}_t$ is not at the origin of the tangent space such that the mean can be updated by

$$\mathbf{Z}_t = \tilde{\mathbf{Z}}_t \operatorname{Exp}(\boldsymbol{\mathcal{Z}}_t). \tag{3.24}$$

The steps of (3.21), (3.22), and (3.24) are then repeated until $\|\boldsymbol{\mathcal{Z}}_t\|$ has converged to a predefined threshold resulting in the final measurement given by (3.24).

3.4.2 Measurement from ground truth data

The second approach assumes a measurement is available, possibly obtained from an image-processing algorithm or a motion capture system. This measurement is simulated by using ground truth data of a trajectory of the cuboid, and adding noise to the pose parameters to simulate the measurement noise. Given the ground truth pose parameters at time t $\mathbf{GT}_t = (\mathbf{R}_t, \mathbf{o}_t) \in \mathcal{P}$, the measurement $\mathbf{Z}_t$ is obtained by

$$\mathbf{Z}_t = \mathbf{GT}_t \operatorname{Exp}\left(\boldsymbol{\phi}_t^{\mathbf{R}}, \boldsymbol{\phi}_t^{\mathbf{o}}\right), \tag{3.25}$$

where the values $\boldsymbol{\phi}_t$ are sampled according to

$$\begin{aligned} \boldsymbol{\phi}_t^{\mathbf{R}} &\sim \mathcal{N}(\mathbf{0}, \boldsymbol{P}^{\mathbf{R}}), \qquad \boldsymbol{P}^{\mathbf{R}} \in \mathbb{R}^{3\times 3}, \\ \boldsymbol{\phi}_t^{\mathbf{o}} &\sim \mathcal{N}(\mathbf{0}, \boldsymbol{P}^{\mathbf{o}}), \qquad \boldsymbol{P}^{\mathbf{o}} \in \mathbb{R}^{3\times 3}, \end{aligned} \tag{3.26}$$

with $\boldsymbol{P}^{\mathbf{R}}$ and $\boldsymbol{P}^{\mathbf{o}}$ representing the covariances related to the noise on orientation and position, respectively.

3.5 Update of the state

Each computed sigma point corresponding to the state, obtained in (3.13), is now evaluated through the observation model $h : \mathcal{S} \to \mathcal{P}$ together with the sigma points of the measurement noise to obtain

$$\hat{\mathbf{H}}_{j,t|t-1} = h\left(\hat{\mathbf{x}}_{j,t|t-1}\right) \operatorname{Exp}\left(\boldsymbol{\mathcal{X}}^{\mathbf{n}}_{j,t-1}\right), \quad j = 0, \ldots, 2n_a, \tag{3.27}$$

where $\hat{\mathbf{H}}_{j,t|t-1} \in \mathcal{P}$ is the predicted pose. Similar to the procedure discussed in Section 2.2.4, we can compute the mean of the points $\hat{\mathbf{H}}_{j,t|t-1}$ to obtain $\bar{\mathbf{H}}_{t|t-1}$. We hereto first consider a tangent space at $\hat{\mathbf{H}}_{0,t|t-1}$ and map the other poses $\hat{\mathbf{H}}_{j,t|t-1}$ to this tangent space by the use of the logarithmic mapping on $\mathcal{P}$. Following the procedure of Section 2.2.4, this will yield a final estimation of $\bar{\mathbf{H}}_{t|t-1}$, at which we consider a tangent space with the poses $\boldsymbol{\mathcal{H}}_{j,t|t-1} \in \mathfrak{p}$. The observation noise covariance and state-observation cross-covariance are then given by

$$\boldsymbol{P}^{\mathbf{yy}}_{t|t-1} = \sum_{j=0}^{2n_a} W_j^{(c)} \left(\boldsymbol{\mathcal{H}}_{j,t|t-1}\right)\left(\boldsymbol{\mathcal{H}}_{j,t|t-1}\right)^T \in \mathfrak{p} \otimes \mathfrak{p} \tag{3.28}$$

and

$$\boldsymbol{P}^{\mathbf{xy}}_{t|t-1} = \sum_{j=0}^{2n_a} W_j^{(c)} \left(\hat{\boldsymbol{\mathcal{X}}}^{\mathbf{x}}_{j,t|t-1}\right)\left(\boldsymbol{\mathcal{H}}_{j,t|t-1}\right)^T \in \mathfrak{s} \otimes \mathfrak{p}, \tag{3.29}$$

where $\hat{\boldsymbol{\mathcal{X}}}^{\mathbf{x}}_{j,t|t-1}$ was computed by (3.14). The innovation is then computed by incorporating the measurement at time t as in [70] such that we write

$$\boldsymbol{\rho} = \operatorname{Log}\left(\left(\bar{\mathbf{H}}_{t|t-1}\right)^{-1}\mathbf{Z}_t\right) \in \mathfrak{p}. \tag{3.30}$$

The state mean is now updated according to

$$\mathbf{x}_t = \hat{\mathbf{x}}_{t|t-1}\operatorname{Exp}\left(\left(\boldsymbol{P}^{\mathbf{xy}}_{t|t-1}\left(\boldsymbol{P}^{\mathbf{yy}}_{t|t-1}\right)^{-1}\boldsymbol{\rho}\right)\right), \tag{3.31}$$

where Exp denotes the exponential mapping on $\mathcal{S}$. The state covariance is then updated to obtain

$$\boldsymbol{P}^{\mathbf{x}}_t = \hat{\boldsymbol{P}}^{\mathbf{x}}_{t|t-1} - \boldsymbol{P}^{\mathbf{xy}}_{t|t-1}\left(\boldsymbol{P}^{\mathbf{xy}}_{t|t-1}\left(\boldsymbol{P}^{\mathbf{yy}}_{t|t-1}\right)^{-1}\right)^T \tag{3.32}$$

as in [70], such that $\boldsymbol{P}^{\mathbf{x}}_t \in \mathfrak{s}\otimes\mathfrak{s}$. Note how (3.31) and (3.32) are related to (2.41). Where in the Euclidean case of (2.41) the state update is performed by addition, this action is performed in (3.31) by the group operator and the exponential mapping, where the term $\boldsymbol{P}^{\mathbf{xy}}_{t|t-1}\left(\boldsymbol{P}^{\mathbf{yy}}_{t|t-1}\right)^{-1}$ is the Kalman gain, similar to (2.40). As a result, the updated state will give the proposal distribution for each particle, given by

$$q(\mathbf{x}_t|\mathbf{y}_{1:t}) = \mathcal{N}\left(\mathbf{x}_t, \boldsymbol{P}^{\mathbf{x}}_t\right). \tag{3.33}$$

3.6 Computing the weights for each particle

We now sample each particle from its proposal distribution according to

$$\tilde{\mathbf{x}}_t = \mathbf{x}_t\operatorname{Exp}(\boldsymbol{\chi}) \quad \text{with} \quad \boldsymbol{\chi} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{P}^{\mathbf{x}}_t), \tag{3.34}$$

where $\operatorname{Exp}: \mathfrak{s} \to \mathcal{S}$. This sampled particle is then evaluated at the prior distribution, which takes the form of (2.60), as discussed in Section 2.2.4, such that

$$p(\tilde{\mathbf{x}}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\tilde{\mathbf{x}}_t|\hat{\mathbf{x}}_{t|t-1}, \hat{\boldsymbol{P}}^{\mathbf{x}}_{t|t-1}) \tag{3.35}$$

$$= \frac{1}{\sqrt{(2\pi)^{12}|\boldsymbol{\Sigma}|}}\exp\left(-\frac{1}{2}\operatorname{Log}\left(\left(\hat{\mathbf{x}}_{t|t-1}\right)^{-1}\tilde{\mathbf{x}}_t\right)^T\boldsymbol{\Sigma}^{-1}\operatorname{Log}\left(\left(\hat{\mathbf{x}}_{t|t-1}\right)^{-1}\tilde{\mathbf{x}}_t\right)\right), \tag{3.36}$$

where we used the substitution

$$\boldsymbol{\Sigma} = \hat{\boldsymbol{P}}^{\mathbf{x}}_{t|t-1}. \tag{3.37}$$

The sampled particle $\tilde{\mathbf{x}}_t$ is in a similar way evaluated at the proposal distribution given by (3.33) and the particle is evaluated at the likelihood function $p(\mathbf{y}_t|\tilde{\mathbf{x}}_t) = \mathcal{L}(\tilde{\mathbf{x}}_t)$. Next, the unnormalized weight of each particle is given by (2.18) which are consequently normalized by the use of (2.14).

3.7 Final estimate

The final estimate of the cuboid's pose is obtained by a weighted average of all the particles. In order to compute the weighted average we again use the method discussed in Section 2.2.4. Here, we will consider the particle $\tilde{\mathbf{x}}^{(i)}_t$ with the highest weight $w^{(i)}_t$ as the initial guess for a weighted mean. At this point of the Lie group $\mathcal{S}$, a tangent space will be considered and the other particles are mapped to that tangent space, to obtain the proposed particles in the tangent space denoted by $\tilde{\boldsymbol{\mathcal{X}}}^{(i)\mathbf{x}}_t$. The mean value is computed in this tangent space

according to

$$\bar{\boldsymbol{\mathcal{X}}}_t^{\mathbf{x}} = \sum_{i=1}^{N} w_t^{(i)} \tilde{\boldsymbol{\mathcal{X}}}_t^{(i)\mathbf{x}}, \tag{3.38}$$

which may not be at the origin, such that an update is needed resulting in a final estimate of the weighted average. The particles are consequently resampled in a similar fashion as discussed in Section 2.1.3.

3.8 Conclusion

In this chapter, we generalized the UPF of Section 2.1 to the parametrization of the state in $\mathcal{S}$, resulting in the GUPF-L and the GUPF-Z. We showed how the particles are propagated through the motion model and how we use the computation of their likelihood to obtain a measurement and to determine their weights. However, we refrained from discussing these functions. In the next chapter, we will show how the likelihood can be computed from a given pose of the cuboid and explain the motion model behind the GUPF, which is considered the key contribution of this thesis.

Chapter 4

Applying the GUPF for Visual Tracking of a Cuboid

In this chapter, the *likelihood function* and the *motion model* as introduced in Sections 3.4.1 and 3.3, respectively, are described in detail. To this end, first a *geometric model* of the cuboid is created, containing information about its shape, mass, inertia, and contact points. This model will be used in both the likelihood function and the motion model, which will be discussed in Sections 4.2 and 4.3, respectively.

4.1 Geometric model of the cuboid

Prior knowledge on the cuboid's geometric shape, mass, and inertia is needed to correctly describe its dynamical behavior in the motion model. Furthermore, prior knowledge on the geometric shape of the cuboid is required to determine its representation in the image plane for a given pose, which is needed to compute the likelihood of a particle. For rigid objects with a complex shape, one might need a CAD model to obtain these properties, as for example is done in [33, 36, 103–105]. However, due to the simple geometric shape of the cuboid, we will only consider the location of the vertices of the cuboid to describe its shape. As we assume the body to be rigid, the edges of the cuboid are assumed to be straight, and the faces describing the surface of the cuboid are assumed to be flat. Similar approaches can be found in [28, 59, 63].

Since we are dealing with impacts, it is also important to define the *contact points* of the cuboid. That is, the points on the surface of the cuboid with which it can make contact with other objects or surfaces. Depending on how the contact is made, this could be one vertex of the cuboid, an edge or a face. In this work, we will assume the vertices of the cuboid to be the only contact points, thereby focusing on point-contacts only. However, the reader is referred to [106] for a framework of modeling curve-curve contact, curve-surface contact, and arbitrary shaped planar contacts.

In the remainder of this work, we will consider a cuboid whose *width*, *length*, and *height* are given as $w = 0.150$ m, $l = 0.100$ m, and $h = 0.050$ m, respectively. As discussed in Section 2.4.1, we assign a body-fixed frame B to the cuboid, located at its center of mass. The vertices are then labeled as the points $\mathbf{p}_i$, $i = 1, \ldots, 8$. This results in the model shown in Figure 4.1. We can now express the coordinates of the points $\mathbf{p}_i$ in terms of the body

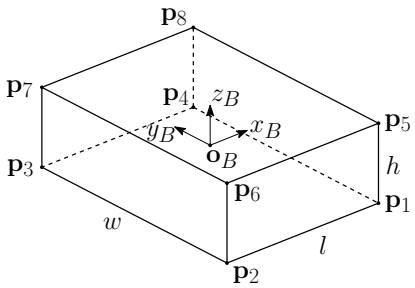


Figure 4.1: Geometric model of the cuboid with body fixed frame B. The vertices are indicated by the points $\mathbf{p}_i$.

fixed frame B. This gives

$$\begin{aligned} {}^B\mathbf{p}_1 &= 1/2\begin{bmatrix} l & -w & -h \end{bmatrix}^T, & {}^B\mathbf{p}_5 &= 1/2\begin{bmatrix} l & -w & h \end{bmatrix}^T, \\ {}^B\mathbf{p}_2 &= 1/2\begin{bmatrix} -l & -w & -h \end{bmatrix}^T, & {}^B\mathbf{p}_6 &= 1/2\begin{bmatrix} -l & -w & h \end{bmatrix}^T, \\ {}^B\mathbf{p}_3 &= 1/2\begin{bmatrix} -l & w & -h \end{bmatrix}^T, & {}^B\mathbf{p}_7 &= 1/2\begin{bmatrix} -l & w & h \end{bmatrix}^T, \\ {}^B\mathbf{p}_4 &= 1/2\begin{bmatrix} l & w & -h \end{bmatrix}^T, & {}^B\mathbf{p}_8 &= 1/2\begin{bmatrix} l & w & h \end{bmatrix}^T. \end{aligned} \tag{4.1}$$

Given the dimensions of the cuboid, the inertia tensor is explicitly given by

$${}_B\mathbb{M}_B = \begin{bmatrix} m\mathbf{I}_3 & \mathbf{0} \\ \mathbf{0} & {}_B\mathbb{I}_B \end{bmatrix}, \tag{4.2}$$

as in [88], where

$${}_B\mathbb{I}_B := \begin{bmatrix} (m/12)(w^2+h^2) & 0 & 0 \\ 0 & (m/12)(l^2+h^2) & 0 \\ 0 & 0 & (m/12)(l^2+w^2) \end{bmatrix}, \tag{4.3}$$

with m the total mass of the cuboid. In (4.2) and (4.3), the right and left superscripts B refer to the inertia tensor being expressed with respect to frame B whose origin coincides with the body center of mass. In the next section, we discuss how the geometric model is used for computation of the likelihood.

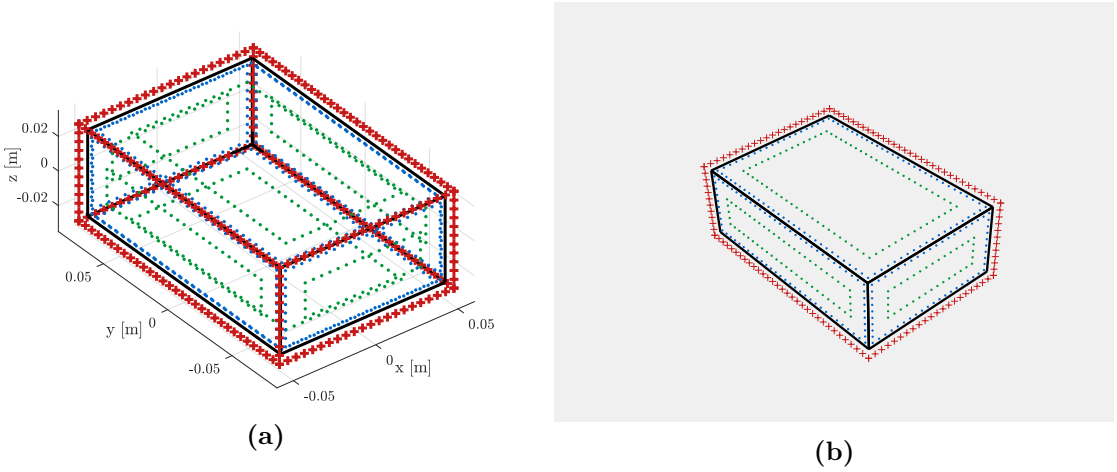
4.2 Likelihood computation from color histograms

In Section 3.4, it was shown that the likelihood function is able to assign a likelihood to a certain particle by using its pose parameters according to

$$L^{(i)} = \mathcal{L}(\mathbf{H}^{(i)}), \tag{4.4}$$

with $\mathcal{L} : \mathcal{P} \to \mathbb{R}$ the likelihood function and $L^{(i)}$ the likelihood assigned to particle i, whose value ideally becomes higher if the particle is closer to the true state of the cuboid. In this section, this likelihood function will be discussed in further detail.

Using the geometric model of Section 4.1 and the theory of Section 2.5, assigning a likelihood is done by evaluating the projection of a pose in the image plane. To this end, a method will be used similar to [28, 41], where an accurate and computationally efficient method is proposed for likelihood computation from image data. In these works, 3D points are positioned around the edges of the geometric model of the cuboid, which are then projected to the image plane such that each point falls on a certain pixel. Making use of the HSI color space, the values of that pixel (hue, saturation, and intensity) are then used for



(a) (b)

Figure 4.2: The different sets of points used to create the color histograms. The 3D sampled points around the geometric model (a) and their projection onto the image plane for a given pose $\mathbf{H}$ (b).

creation of *color histograms*. This color histogram is a representation of the distribution of the colors of the evaluated set of pixels. Each of these evaluated pixels has a hue, saturation, and intensity that fall within the fixed range of colors that span the image's color space. A color histogram thus carries statistical information of a certain set of pixels.

Computation of the likelihood is done by comparing the color histograms of different sets of points with a-priori created reference color histograms of the object. In [28, 41], only points around the edges of the cuboid are used to create color histograms. However, as noted in [103], using only the points near the edges of an object is a waste of information: pixels inside the object contour can contain a lot of information about an object's pose. We therefore extend the framework of [28] and [41] by sampling points on each surface of the cuboid as well. This is illustrated in Figure 4.2a, where three sets of points can be distinguished: a set of green points on the faces, a set of blue points located on the surfaces of the cuboids near the edges, and a third set of red points located right outside the edges. After determining the visible faces and edges of the cuboid, we can give a representation of the cuboid in the image plane for a given pose. An example of this can be seen in Figure 4.2b. Note that we disregarded all the red points laying inside the contouring edges of the cuboid.

After the visible points are projected to certain pixel points, as illustrated in Figure 4.2b, the *hue*, *saturation*, and *intensity* of these pixels are evaluated to create the color histograms. For each particle, a few histograms are computed: the green points in Figure 4.2b are used to compute a histogram of the object $\boldsymbol{H}_{obj,i}$ for each visible face i. The blue points are used to compute the histograms of the inside of the contour $\boldsymbol{H}_{cin,i}$ for each face, and the red points are used to compute the histogram of the outside of the contour $\boldsymbol{H}_{cout}$. Furthermore, each face of the cuboid has a pre-computed reference histogram $\boldsymbol{H}_{ref,i}$. This way, distinction can be made between the faces such that the orientation of the cuboid can be estimated, as long as each face has a distinguishable color histogram.

As a standard camera returns RGB images, the images are first converted into the HSI color format, which decouples the color from intensity. The color space is discretized into a number of *bins*, typically in the range of $[0 \dots 255]$, but in this case the number of bins are reduced to a range of $[0 \dots 12]$ for hue and saturation, and $[0 \dots 4]$ for intensity to reduce

computational cost. The smaller range of intensity has been selected to be more robust to illumination changes as in [51] and the number of bins are similar to the values used in [28]. Hence, we define the number of bins for hue, saturation, and intensity as $\mathbb{B}_H = 12$, $\mathbb{B}_S = 12$, and $\mathbb{B}_I = 4$, respectively, such that we can define the set of possible pixel values as $\mathcal{H} \coloneqq \mathbb{B}_H \times \mathbb{B}_S \times \mathbb{B}_I$, where $\mathcal{H} \subset \mathbb{N}^3$.

The color histograms are now computed as follows. We define $\mathbf{b}(\mathbf{d}) \in \mathcal{H}$ as the bin index at pixel location $\mathbf{d} = \begin{bmatrix} u & v \end{bmatrix}^T$ in the image plane, containing the hue, saturation, and intensity of that pixel. Each bin of the color histogram is then computed by

$$h(\mathbf{b}) = \beta \sum_{\mathbf{d} \in \mathcal{U}} \delta^3[\mathbf{b} - \mathbf{b}(\mathbf{d})] \in \mathbb{R}, \tag{4.5}$$

where $\delta^3 : \mathcal{H} \to \{0, 1\}$ is the three-dimensional Kronecker delta function, $\mathcal{U}$ the set of all the evaluated pixel points in the image, and β a normalizing constant to ensure that h is a probability distribution $\sum_{\mathbf{b} \in \mathcal{H}} h(\mathbf{b}) = 1$. The color histogram is then obtained by evaluating each bin of the total color space $\mathcal{H}$, such that

$$\boldsymbol{H} = \{h(\mathbf{b})\}_{\mathbf{b} \in \mathcal{H}}, \tag{4.6}$$

which we encode as a $12 \times 12 \times 4$ matrix, such that $\boldsymbol{H} \in \mathbb{R}^{12 \times 12 \times 4}$. After the histograms for the different sets of points are obtained, they can be used for computation of the likelihood. To this end, *similarities* between the different color histograms are computed by the use of the Bhattacharyya similarity function, as described in [107]. Given two color histograms $\boldsymbol{H}_1$ and $\boldsymbol{H}_2$, the similarity between these histograms is defined as

$$S\left(\boldsymbol{H}_1, \boldsymbol{H}_2\right) \coloneqq \sum_{i=1}^{\mathbb{B}_H} \left(\sum_{j=1}^{\mathbb{B}_S} \left(\sum_{k=1}^{\mathbb{B}_I} \sqrt{\boldsymbol{H}_1(i,j,k) \cdot \boldsymbol{H}_2(i,j,k)} \right) \right). \tag{4.7}$$

Since the histograms are normalized, S is in the range of $[0, \ldots, 1]$, where $S = 1$ refers to a perfect match. We will now compute the similarities

$$S_1 = \frac{1}{N_f} \sum_{i=1}^{N_f} S\left(\boldsymbol{H}_{cin,i}, \boldsymbol{H}_{ref,i}\right), \tag{4.8}$$

$$S_2 = \frac{1}{N_f} \sum_{i=1}^{N_f} S\left(\boldsymbol{H}_{obj,i}, \boldsymbol{H}_{ref,i}\right), \tag{4.9}$$

and

$$S_3 = \frac{1}{N_f} \sum_{i=1}^{N_f} S\left(\boldsymbol{H}_{cin,i}, \boldsymbol{H}_{cout}\right), \tag{4.10}$$

where N_f denotes the number of visible faces as described in Section 2.5. Considering these similarities, a particle should have a high likelihood when S_1 and S_2 are high, but S_3 is low. This is expressed by the "distance"

$$D = \frac{\kappa_1 \left(1 - S_1\right) + \kappa_2 \left(1 - S_2\right) + \kappa_3 S_3}{\kappa_1 + \kappa_2 + \kappa_3}, \tag{4.11}$$

where, if the particle is representing the true state, all terms in the numerator should go to zero. Furthermore, in (4.11), κ_1, κ_2, and κ_3 are weighting parameters and determine the contribution of the similarities to the distance function. The likelihood itself is then modeled

as a Laplacian distribution as in [41] to obtain

$$L = e^{-\frac{|D|}{\epsilon}}, \tag{4.12}$$

where ϵ is a scaling parameter, which in our experiments is set to 0.1.

With the content of this section, it is now possible to assign a likelihood to a certain particle. It is important that this likelihood function is able to distinguish between different particles such that a higher likelihood is assigned to particles closer to the true state. To see how the likelihood function behaves, different simulations are executed of which the results are shown in Appendix A. In these experiments, we set $\kappa_1 = 1$, $\kappa_2 = 2$, and $\kappa_3 = 5$, and it is shown that the likelihood function is able to accurately estimate the position within 1mm accuracy and the orientation within 3 degrees accuracy. This accuracy can be achieved if the particles are close to the true state. To ensure this is the case, a motion model is developed which accurately describes the true dynamics of the cuboid. This is discussed in the next section.

4.3 Motion model with impacts

In this section, we will discuss the motion model of the cuboid as used in the GUPF-L and GUPF-Z described in Chapter 3. For the sake of brevity, we will consider only one single particle i at a given time t, such that we neglect the superscript $(\cdot)^{(i)}$. This means we write its state as $\mathbf{x}_t = ({}^A\mathbf{R}(t)_B, {}^A\mathbf{o}(t)_B, {}^B\boldsymbol{v}(t)_{A,B}, {}^B\boldsymbol{\omega}(t)_{A,B})$. We will first describe the equations of motion in Section 4.3.1, and then discuss the laws for contact, impact, and friction subsequently in Sections 4.3.2, 4.3.3, and 4.3.4. Throughout this chapter we will closely follow the framework of Glocker in [108], [109], and [110], and Glocker and Pfeiffer in [111] to describe these laws and discuss the preliminary material within each section. Finally, in Sections 4.3.5 and 4.3.6, we will describe the numerical implementation.

4.3.1 Mechanical systems with unilateral constraints

We can model the equations of motion of a rigid body in persistent contact, but without impulsive loads, as

$$\mathbb{M}\dot{\mathbf{v}}(t) + \mathbf{v}(t)\bar{\times}^*\mathbb{M}\mathbf{v}(t) = \mathbf{f}(t) + \mathbf{W}_N(\mathbf{x},t)\boldsymbol{\lambda}_N + \mathbf{W}_T(\mathbf{x},t)\boldsymbol{\lambda}_T, \tag{4.13}$$

$${}^A\dot{\mathbf{H}}(t)_B = {}^A\mathbf{H}(t)_B\mathbf{v}(t), \tag{4.14}$$

where we employed the short-hand notation $\mathbb{M} = {}_B\mathbb{M}_B$, $\mathbf{v}(t) = {}^B\mathbf{v}(t)_{A,B}$, and $\mathbf{f}(t) = {}_B\mathbf{f}(\mathbf{x},t)$ for the ease of readability. In (4.13), $\mathbb{M}$ denotes the inertia tensor, $\mathbf{v}(t)$ the left trivialized velocity, and $\mathbf{f}(t)$ the generalized wrench containing the generalized forces and torques applied to the center of mass of the body [89] (except for the contact forces), which here involve the gravitational forces. We express the gravitational force with the orientation of frame A, such that ${}_{B[A]}\mathbf{f} = \begin{bmatrix}0 & mg & 0\end{bmatrix}^T$ which can rewritten into ${}_B\mathbf{f}$ by the use of (2.85), with m the total mass of the cuboid and g the gravitational acceleration. The contact forces are introduced through $\boldsymbol{\lambda}_N \in \mathbb{R}^{n_c}$ with $\mathbf{W}_N(\mathbf{x},t) \in \mathbb{R}^{n\times n_c}$ a matrix containing the generalized force directions, where n_c denotes the number of closed contact points as will be discussed in Section 4.3.2. Similarly, the friction forces are introduced through $\boldsymbol{\lambda}_T \in \mathbb{R}^{2n_c}$ with $\mathbf{W}_T(\mathbf{x},t) \in \mathbb{R}^{n\times 2n_c}$ the matrix containing the generalized force directions of the friction.

If a slip-to-stick transition occurs, the friction force $\boldsymbol{\lambda}_T(t)$ is discontinuous. At these instances in time, the acceleration $\dot{\mathbf{v}}(t)$ is not defined. Therefore, the equations of motion

as shown in (4.13) are not defined for all t, but only when $\boldsymbol{\lambda}_T(t)$ is continuous. A motion without impulses implies that the normal forces are bounded and continuous in time. At the moment of an impact, however, the normal forces are impulsive. In order to allow for the inclusion of the impulsive forces into the equations of motion (4.13), these equations are turned into measure differential equations such that

$$\mathbb{M}d\mathbf{v}(t) + \mathbf{v}(t)\bar{\times}^*\mathbb{M}\mathbf{v}(t)dt = \mathbf{f}(t)dt + \mathbf{W}_N d\mathbf{P}_N + \mathbf{W}_T d\mathbf{P}_T, \tag{4.15}$$

where we omitted the dependency on $\mathbf{x}$ and t for $\mathbf{W}_N$ and $\mathbf{W}_T$, for the sake of brevity. Equation (4.15) is now at the level of momenta (i.e., the time integral of forces), whereas (4.13) was written at the level of (generalized) forces. In (4.15), $d\mathbf{P}_N$ and $d\mathbf{P}_T$ are respectively the differential measures of the momenta associated to the contact in normal direction and tangential direction. These differential measures consists out of a Lebesgue measurable part $\boldsymbol{\lambda}dt$ which denotes the non-impulsive part of the contact/friction force and an atomic part $\boldsymbol{\Lambda}d\eta$, denoting the impulsive part of the contact/friction force [112,113]. Hence, we can write the differential measures as

$$d\mathbf{P}_N = \boldsymbol{\lambda}_N dt + \boldsymbol{\Lambda}_N d\eta \tag{4.16}$$

and

$$d\mathbf{P}_T = \boldsymbol{\lambda}_T dt + \boldsymbol{\Lambda}_T d\eta, \tag{4.17}$$

where dt is the Lebesgue measure and $d\eta$ is the atomic measure, defined as $d\eta = \delta(t-t_i)$ [114]. Here, $\delta(\cdot)$ is the Dirac delta function and t_i is the time instant at which an impulse load occurs. The atomic measure has as property that its integral over a singleton[1] t_i can be given by

$$\int_{t_i}^{t_i} d\eta = \int_{t_i}^{t_i} \delta(t-t_i)dt = 1. \tag{4.18}$$

We can see how the impulsive forces contribute to a jump in velocity by integrating (4.15) over a singleton t_i. This results in

$$\mathbb{M}\left(\mathbf{v}(t^+) - \mathbf{v}(t^-)\right) = \mathbf{W}_N\boldsymbol{\Lambda}_N + \mathbf{W}_T\boldsymbol{\Lambda}_T, \tag{4.19}$$

which thus illustrates the velocity jump at the singleton t_i as a result of the impulsive force [115,116]. In (4.19), we made use of the fact that

$$\int_{t_i}^{t_i} \mathbb{M}d\mathbf{v}(t) = \mathbb{M}\left(\mathbf{v}(t^+) - \mathbf{v}(t^-)\right), \tag{4.20}$$

where $\mathbf{v}(t^+)$ and $\mathbf{v}(t^-)$ are the right and left limits of the velocity at time t_i, respectively [116].

In this section, the dynamics of a mechanical systems with contact, impact, and friction are derived and given by (4.15). However, the constitutive laws for these forces still need to be derived, which will be done in the following sections. For more information on measure differential equations and their applications regarding nonsmooth mechanics the reader is referred to [110,112,113,117].

4.3.2 Signorini's contact law

In this section, the constitutive law for contact will be formulated. Let us thereto consider a cuboid which is about to make contact with a surface, as depicted in Figure 4.3. Recall that frame C defines the location and orientation of the contact surface with which the cuboid

[1] A singleton is an isolated discrete time instant.

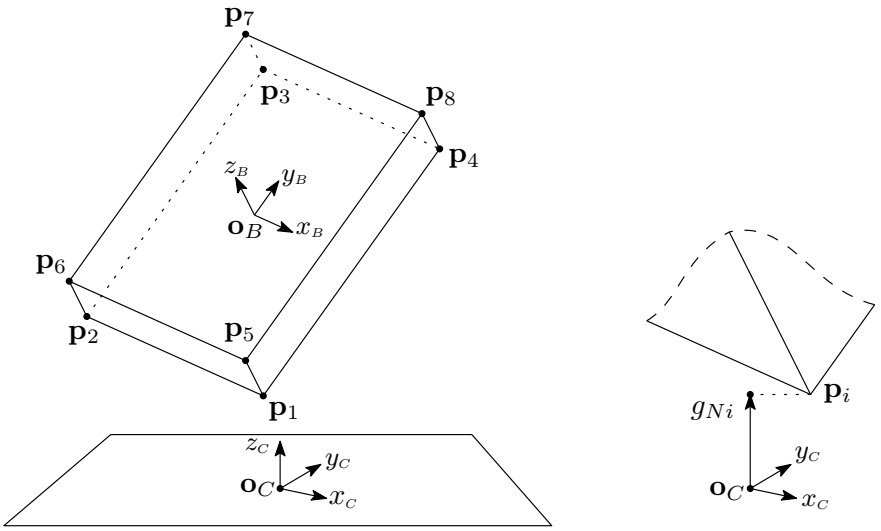


Figure 4.3: Illustration of the cuboid, about to make contact with a surface. The gap function is indicated by g_{Ni}.

is about to make contact and this frame is oriented such that the unit vector $\vec{z}_c$ defines the normal to the plane. The contact distance, in the latter referred to as the *gap function*, is defined as the distance between a contact point $\mathbf{p}_i$ on the cuboid and the contact surface, in the normal direction of that plane. We can define the point $\mathbf{p}_i$ with respect to frame A and we refer to this as the coordinate vector ${}^A\mathbf{p}_i$. The normal of the contact plane, which in Figure 4.3 is defined by the unit vector of C in z-direction, is given by ${}^C\mathbf{z}_C = \begin{bmatrix}0 & 0 & 1\end{bmatrix}^T$. Expressing this vector in terms of frame A, one obtains ${}^A\mathbf{z}_C = {}^A\mathbf{R}_C {}^C\mathbf{z}_C$. The gap function g_{Ni} is then defined as the distance in z-direction of C between the point $\mathbf{p}_i$ and the origin of C. Hence, g_{Ni} is given by

$$g_{Ni} = {}^A\mathbf{z}_C^T \left({}^A\mathbf{p}_i - {}^A\mathbf{o}_C \right), \tag{4.21}$$

which is depicted on the right side of Figure 4.3 [109, Section 5.4]. By using the fact that

$${}^A\mathbf{p}_i = {}^A\mathbf{o}_B + {}^A\mathbf{R}_B {}^B\mathbf{p}_i, \tag{4.22}$$

with ${}^B\mathbf{p}_i$ given by (4.1), (4.21) can be rewritten as

$$g_{Ni} = {}^A\mathbf{z}_C^T \Big(\left({}^A\mathbf{o}_B + {}^A\mathbf{R}_B {}^B\mathbf{p}_i \right) - {}^A\mathbf{o}_C \Big), \tag{4.23}$$

where ${}^B\mathbf{p}_i$ is independent of time. Under the assumption that the body is rigid and cannot penetrate the surface, the gap function cannot become negative. Hence, this condition is written as the unilateral constraint

$$g_{Ni} \geq 0. \tag{4.24}$$

In addition, the normal contact force λ_{Ni} can only be nonzero when the bodies are in contact, but must be zero when the bodies are separated. This force can only be non-negative, as it cannot prevent the bodies from separating after contact [118,119]. This means two situations can occur:

1. $g_{Ni} = 0$ and $\lambda_{Ni} \geq 0$: the bodies are in contact (the contact is closed),
2. $g_{Ni} > 0$ and $\lambda_{Ni} = 0$: the bodies are separated (the contact is open).

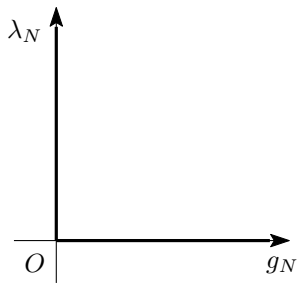


Figure 4.4: Signorini's normal contact law.

These relations result in a complementarity condition which is described by Signorini's contact law [109, 120–122] and is given as follows:

$$g_{Ni} \geq 0, \;\; \lambda_{Ni} \geq 0, \;\; g_{Ni}\lambda_{Ni} = 0, \;\; \text{for } i \in \mathcal{I}, \tag{4.25}$$

which is illustrated in Figure 4.4. In (4.25), the set $\mathcal{I} = \{1, 2, \ldots, n\}$ is the set of all possible contact points. In the sequel we will make use of a set $\mathcal{I}_c$, indicating which contacts are closed. We will now define the set

$$\mathcal{I}_c \coloneqq \{i \in \mathcal{I} \mid g_{Ni} = 0\} \tag{4.26}$$

as the set of *closed contact* points. It is clear that $\mathcal{I}_c$ is a subset of $\mathcal{I}$ and if all contacts are closed, then $\mathcal{I}_c = \mathcal{I}$. With $\mathbf{g}_N = \begin{bmatrix} g_{N,1} & g_{N,2} & \ldots & g_{N,n_c} \end{bmatrix}^T$, we will define the column of normal contact distances and with $\boldsymbol{\lambda}_N = \begin{bmatrix} \lambda_{N,1} & \lambda_{N,2} & \ldots & \lambda_{N,n_c} \end{bmatrix}^T$ the column of normal contact forces for all closed contacts. Using these formulations, Signorini's contact law for normal forces can be expressed equivalently by the so called *proximal point formulation* according to

$$\boldsymbol{\lambda}_N = \operatorname{prox}_{\mathcal{C}_N}(\boldsymbol{\lambda}_N - r\mathbf{g}_N) \quad \text{with} \quad \mathcal{C}_N = \{\boldsymbol{\lambda}_N \in \mathbb{R}^{n_c} \mid \boldsymbol{\lambda}_N \geq \mathbf{0}\} \quad \text{and} \quad r > 0, \tag{4.27}$$

where $\mathcal{C}_N$ is the set of admissible normal forces. In (4.27), we understand the inequality sign component wise, namely

$$\boldsymbol{\lambda}_N \geq \mathbf{0} \iff \lambda_i \geq 0, \quad \forall i \in \mathcal{I}_c. \tag{4.28}$$

4.3.3 Newton's impact law

There exist at least 3 main approaches to modeling impacts as instantaneous processes: Newton's kinematic approach [123], Poisson's kinetic approach [124], and Stronge's energetic definitions as discussed in [125, 126]. An extensive review on the topic can be found in [127, Chapter 4] and the reader is referred to [128] for a review on multiple impact models. In this work, we will use Newton's impact law to describe the impact between two bodies due to its simplicity. Newton's restitution law, originally stated in [123], gives a relation between the pre-impact normal velocity γ_{Ni}^- and the post-impact normal velocity γ_{Ni}^+ in contact point i according to

$$\gamma_{Ni}^+ = -e_{Ni}\gamma_{Ni}^-, \tag{4.29}$$

where e_{Ni} is referred to as *Newton's coefficient of restitution* [128] of contact point i and typically $0 \leq e_{Ni} \leq 1$, where $e_{Ni} = 0$ corresponds to a perfectly plastic impact and $e_{Ni} = 1$ corresponds to a completely elastic impact [127]. We will assume the coefficient of restitution to be constant, whereas this coefficient in general depends on the friction as well [129]. The normal contact velocity $\gamma_{Ni} \in \mathbb{R}$ is the velocity of the contact point $\mathbf{p}_i$ in the normal direction of the contact surface. The normal contact velocity is thus given by the time derivative of

the gap-function [116]. Hence,

$$\gamma_{Ni} = \dot{g}_{Ni}, \tag{4.30}$$

almost everywhere. For a contact point $\mathbf{p}_i$, the normal contact velocity is thus given as

$$\gamma_{Ni} = \frac{d}{dt}\left({}^A\mathbf{z}_C^T \left(\left({}^A\mathbf{o}_B + {}^A\mathbf{R}_B {}^B\mathbf{p}_i \right) - {}^A\mathbf{o}_C \right) \right) = {}^A\mathbf{z}_C^T {}^A\dot{\mathbf{o}}_B + {}^A\mathbf{z}_C^T {}^A\dot{\mathbf{R}}_B {}^B\mathbf{p}_i, \tag{4.31}$$

where we used the fact that ${}^B\mathbf{p}_i$ and ${}^A\mathbf{o}_C$ are independent of time. Recall that the left trivialized velocity is given by

$${}^B\mathbf{v}_{A,B} \coloneqq \begin{bmatrix} {}^B\boldsymbol{v}_{A,B} \\ {}^B\boldsymbol{\omega}_{A,B} \end{bmatrix} \in \mathbb{R}^6, \tag{4.32}$$

where ${}^B\boldsymbol{v}_{A,B}$ and ${}^B\boldsymbol{\omega}_{A,B}$ are defined such that

$${}^B\boldsymbol{v}_{A,B} \coloneqq {}^A\mathbf{R}_B^T {}^A\dot{\mathbf{o}}_B \tag{4.33}$$

and

$${}^B\boldsymbol{\omega}_{A,B}^\wedge \coloneqq {}^A\mathbf{R}_B^T {}^A\dot{\mathbf{R}}_B. \tag{4.34}$$

Now, (4.33) and (4.34) can be rewritten as

$${}^A\dot{\mathbf{o}}_B = {}^A\mathbf{R}_B {}^B\boldsymbol{v}_{A,B} \tag{4.35}$$

and

$${}^A\dot{\mathbf{R}}_B = {}^A\mathbf{R}_B {}^B\boldsymbol{\omega}_{A,B}^\wedge, \tag{4.36}$$

and substituting (4.35) and (4.36) into (4.31) gives

$$\gamma_{Ni} = {}^A\mathbf{z}_C^T {}^A\mathbf{R}_B {}^B\boldsymbol{v}_{A,B} + {}^A\mathbf{z}_C^T {}^A\mathbf{R}_B {}^B\boldsymbol{\omega}_{A,B}^\wedge {}^B\mathbf{p}_i. \tag{4.37}$$

Rewriting (4.37) yields

$$\gamma_{Ni} = {}^A\mathbf{z}_C^T {}^A\mathbf{R}_B {}^B\boldsymbol{v}_{A,B} - {}^A\mathbf{z}_C^T {}^A\mathbf{R}_B {}^B\mathbf{p}_i^\wedge {}^B\boldsymbol{\omega}_{A,B}, \tag{4.38}$$

from which the left trivialized velocity can be separated such that

$$\gamma_{Ni} = \begin{bmatrix} {}^A\mathbf{z}_C^T {}^A\mathbf{R}_B & -{}^A\mathbf{z}_C^T {}^A\mathbf{R}_B {}^B\mathbf{p}_i^\wedge \end{bmatrix} \begin{bmatrix} {}^B\boldsymbol{v}_{A,B} \\ {}^B\boldsymbol{\omega}_{A,B} \end{bmatrix}. \tag{4.39}$$

Finally, (4.39) can be written in a compact form such that

$$\gamma_{Ni} = \mathbf{w}_{Ni}^T \mathbf{v} \ \in \mathbb{R}, \tag{4.40}$$

with the short-hand notation $\mathbf{v} = {}^B\mathbf{v}_{A,B}$, and where $\mathbf{w}_{Ni}^T$ is the row-vector containing the force directions of the contact impulses of contact point $\mathbf{p}_i$, as defined by (4.39). When a contact is closed, hence its corresponding unilateral constraint is active, an impact may occur. This impact, which causes a jump in velocity, generates a normal impulsive force $\Lambda_{Ni} \geq 0$. In some cases, the contact does not participate in the impact, i.e., the impulsive force is zero, even though the contact is closed. This particularly happens for multi-contact situations. In such a case, we allow the post-impact velocity to be higher than prescribed by Newton's impact law such that $\gamma_{Ni}^+ \geq -e_{Ni}\gamma_{Ni}^-$. These contacts are referred to as *superfluous* contacts, which indicates that they can be removed from the system without changing the contact-impact process [113,115,116]. We can now formulate an impact law that relates the impulsive force Λ_{Ni} to the post-impact velocity γ_{Ni}^+ for closed contacts $i \in \mathcal{I}_c$. For a closed contact point i, two situations can occur:

1. $\Lambda_{Ni} > 0 \quad \wedge \quad \gamma_{Ni}^+ = -e_{Ni}\gamma_{Ni}^-$: The contact is actively participating in the impact process,

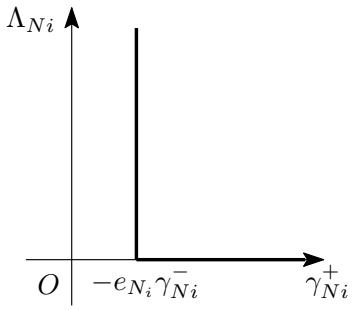


Figure 4.5: Newton's impact law.

2. $\Lambda_{Ni} = 0 \quad \wedge \quad \gamma_{Ni}^+ \geq -e_{Ni}\gamma_{Ni}^-$: The contact is superfluous.

The two situations described above express a complementarity condition given as

$$\Lambda_{Ni} \geq 0, \quad \gamma_{Ni}^+ + e_{Ni}\gamma_{Ni}^- \geq 0, \quad \left(\gamma_{Ni}^+ + e_{Ni}\gamma_{Ni}^-\right)\Lambda_{Ni} = 0, \tag{4.41}$$

and is also depicted in Figure 4.5 [115]. It is clear that in case of a non-vanishing impulse, so if $\Lambda_{Ni} > 0$, Newton's law is applied such that $\gamma_{Ni}^+ = -e_{Ni}\gamma_{Ni}^-$, which is expressed by the third condition in (4.41). Let us introduce the variable ξ_{Ni} according to

$$\xi_{Ni} = \gamma_{Ni}^+ + e_{Ni}\gamma_{Ni}^-, \tag{4.42}$$

such that (4.41) can be written as

$$\Lambda_{Ni} \geq 0, \quad \xi_{Ni} \geq 0, \quad \xi_{Ni}\Lambda_{Ni} = 0, \quad i \in \mathcal{I}_c \tag{4.43}$$

which now gives a compact form of the complementarity relation describing the set-valued force law for impact with restitution. We can now write (4.43) using the proximal point formulation, which results in

$$\Lambda_{Ni} = \text{prox}_{\mathcal{C}_N}\left(\Lambda_{Ni} - r\xi_{Ni}\right) \ \forall i \in \mathcal{I}_c \quad \text{with} \quad \mathcal{C}_N = \mathbb{R}^+ \quad \text{and} \quad r > 0. \tag{4.44}$$

We will define the column of all normal contact velocities $\boldsymbol{\gamma}_N$ and the column of all normal impulsive forces $\boldsymbol{\Lambda}_N$ for all closed contacts by

$$\boldsymbol{\gamma}_N = \text{col}\{\gamma_{Ni}\}, \ i \in \mathcal{I}_c, \tag{4.45}$$

and

$$\boldsymbol{\Lambda}_N = \text{col}\{\Lambda_{Ni}\}, \ i \in \mathcal{I}_c, \tag{4.46}$$

respectively. If for example contacts 4 and 8 are closed, hence $\mathcal{I}_c = \{4, 8\}$, then $\boldsymbol{\gamma}_N = \begin{bmatrix}\gamma_{N4} & \gamma_{N8}\end{bmatrix}^T$ and $\boldsymbol{\Lambda}_N = \begin{bmatrix}\Lambda_{N4} & \Lambda_{N8}\end{bmatrix}^T$, as in (4.15). As a result, (4.42) can be written as

$$\boldsymbol{\xi}_N = \boldsymbol{\gamma}_N^+ + \boldsymbol{e}_N\boldsymbol{\gamma}_N^-, \tag{4.47}$$

with $\boldsymbol{e}_N = \text{diag}\{e_{Ni}\}, \ i \in \mathcal{I}_c$. Now, (4.44) can be written in matrix form using the proximal point formulation which results in

$$\boldsymbol{\Lambda}_N = \text{prox}_{\mathcal{C}_N}\left(\boldsymbol{\Lambda}_N - r\boldsymbol{\xi}_N\right) \quad \text{with} \quad \mathcal{C}_N = \{\boldsymbol{\Lambda}_N \in \mathbb{R}^{n_c} \mid \boldsymbol{\Lambda}_N \geq \boldsymbol{0}\} \quad \text{and} \quad r > 0, \tag{4.48}$$

where n_c denotes the number of closed contact points and $\mathcal{C}_N$ is the set of all admissible normal impulsive forces.

4.3.4 Coulomb's friction law

A dry-friction force arises when two solid surfaces come into contact and additionally no fluids are present at the contact. This force apposes the relative lateral motion of the two

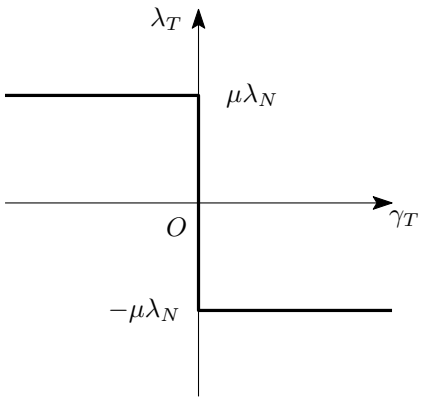


Figure 4.6: Coulomb's friction law for the planar case.

bodies, which can have a big effect on the dynamical behavior of the body under consideration. Whether a body in contact will stick or slip at its contact point, will also have a direct effect on the tilt of the body. It is therefore of high importance to have an accurate description of the friction model.

In order to model dry-friction, we will use a set-valued Coulomb friction model as derived in [109, Chapter 10], [113, Chapter 5], and [111, Chapter 6]. An extensive review on modeling Coulomb friction can be found in [118, Chapter 10]. We will first consider the planar case, in which the relative tangential velocity $\gamma_T \in \mathbb{R}$ is a scalar. We make a distinction between two situations: *static friction* and *dynamic friction* [112]. In the former, the friction force λ_T enforces zero relative tangential velocity between the contacting bodies. The latter refers to a situation where the relative tangential velocity is nonzero and the tangential force has a constant magnitude, given as $\mu\lambda_N$, opposing the tangential velocity. Here, λ_N is the normal contact force as governed by the contact law in Section 4.3.2 and μ is the friction coefficient. We will assume the friction coefficient to be constant, such that we make no distinction between a coefficient of static friction and dynamic friction. Furthermore, we assume the coefficient of friction to be independent of velocity, thereby neglecting the Stribeck effect and hysteresis. These topics are however extensively discussed in [109, Section 1.1], [113, Section 5.3], and [114, Section 4.2]. As a result, in case of static friction, the friction force must lie in the interval $-\mu\lambda_N \leq \lambda_T \leq \mu\lambda_N$. We can therefore distinguish three different cases:

$$\begin{aligned} \gamma_T = 0 \;&\Rightarrow\; |\lambda_T| \leq \mu\lambda_N \qquad \text{static friction,} \\ \gamma_T < 0 \;&\Rightarrow\; \lambda_T = \mu\lambda_N \qquad \text{negative sliding,} \\ \gamma_T > 0 \;&\Rightarrow\; \lambda_T = -\mu\lambda_N \qquad \text{positive sliding.} \end{aligned} \tag{4.49}$$

The cases described above can be expressed by the following set-valued force law:

$$\lambda_T \in -\mu\lambda_N \text{Sign}(\gamma_T), \tag{4.50}$$

where $\text{Sign}(\gamma_T)$ denotes the set-valued Sign function of γ_T [114, Section 5.2]. Alternatively, (4.50) can be written as

$$\lambda_T \in \begin{cases} -\mu\lambda_N, & \gamma_T > 0, \\ [-1,1]\mu\lambda_N, & \gamma_T = 0, \\ \mu\lambda_N, & \gamma_T < 0, \end{cases} \tag{4.51}$$

which is depicted in Figure 4.6. We can now define the set

$$\mathcal{C}_T = \{\lambda_T \mid -\mu\lambda_N \leq \lambda_T \leq \mu\lambda_N\} \tag{4.52}$$

as the set of admissible values of the tangential force λ_T. It is obvious that a tangential force can only exist for closed contacts. Hence, we can rewrite the convex set in (4.52) for each closed contact point as

$$\mathcal{C}_{Ti} = \{\lambda_{Ti} \mid -\mu_i \lambda_{Ni} \leq \lambda_{Ti} \leq \mu_i \lambda_{Ni}\} \quad \forall i \in \mathcal{I}_c. \tag{4.53}$$

If the tangential force λ_{Ti} lies in the interior of $\mathcal{C}_{Ti}$, the contact point is *sticking*, whereas if it lies on the boundary of $\mathcal{C}_{Ti}$ the contact point is *sliding* in opposing direction of the tangential force [111].

We now extend the planar case to Coulomb's isotropic spatial friction law, which requires the definition of a tangent contact plane [130]. Consider again Figure 4.3, where we defined the contact plane with the frame C. In Section 4.3.2, we defined the normal of the contact plane as ${}^C\mathbf{z}_C = \begin{bmatrix}0 & 0 & 1\end{bmatrix}$. As a result, we can now define the in-plane vectors, given by the unit vectors in x- and y-direction of frame C, as ${}^C\mathbf{x}_C = \begin{bmatrix}1 & 0 & 0\end{bmatrix}$ and ${}^C\mathbf{y}_C = \begin{bmatrix}0 & 1 & 0\end{bmatrix}$, respectively. We defined the normal contact velocity of contact point $\mathbf{p}_i$ as the velocity by which that point is approaching the contact surface, in the normal direction of the contact surface, as defined by (4.31). Likewise, we will here define the tangential velocity as the velocity of $\mathbf{p}_i$ with respect to C in x- and y-direction of the contact plane as $\gamma_{Ti,x} \in \mathbb{R}$ and $\gamma_{Ti,y} \in \mathbb{R}$, respectively. Similar to (4.31), remembering ${}^A\mathbf{x}_C$ and ${}^A\mathbf{y}_C$ are constants, we can derive these velocities as

$$\gamma_{Ti,x} = \frac{d}{dt}\left({}^A\mathbf{x}_C^T\left({}^A\mathbf{p}_i - {}^A\mathbf{o}_C\right)\right), \tag{4.54}$$

and

$$\gamma_{Ti,y} = \frac{d}{dt}\left({}^A\mathbf{y}_C^T\left({}^A\mathbf{p}_i - {}^A\mathbf{o}_C\right)\right), \tag{4.55}$$

where ${}^A\mathbf{x}_C = {}^A\mathbf{R}_C\,{}^C\mathbf{x}_C$ and ${}^A\mathbf{y}_C = {}^A\mathbf{R}_C\,{}^C\mathbf{y}_C$. We can rewrite (4.54) and (4.55) into a similar form of (4.39), obtaining

$$\gamma_{Ti,x} = \begin{bmatrix}{}^A\mathbf{x}_C^T\,{}^A\mathbf{R}_B & -{}^A\mathbf{x}_C^T\,{}^A\mathbf{R}_B\,{}^B\mathbf{p}_i^\wedge\end{bmatrix}\begin{bmatrix}{}^B\boldsymbol{v}_{A,B}\\ {}^B\boldsymbol{\omega}_{A,B}\end{bmatrix} \tag{4.56}$$

and

$$\gamma_{Ti,y} = \begin{bmatrix}{}^A\mathbf{y}_C^T\,{}^A\mathbf{R}_B & -{}^A\mathbf{y}_C^T\,{}^A\mathbf{R}_B\,{}^B\mathbf{p}_i^\wedge\end{bmatrix}\begin{bmatrix}{}^B\boldsymbol{v}_{A,B}\\ {}^B\boldsymbol{\omega}_{A,B}\end{bmatrix}. \tag{4.57}$$

We combine the two components of the tangential velocity as

$$\boldsymbol{\gamma}_{Ti} = \begin{bmatrix}\gamma_{Ti,x}\\ \gamma_{Ti,y}\end{bmatrix} = \begin{bmatrix}{}^A\mathbf{x}_C^T\,{}^A\mathbf{R}_B & -{}^A\mathbf{x}_C^T\,{}^A\mathbf{R}_B\,{}^B\mathbf{p}_i^\wedge\\ {}^A\mathbf{y}_C^T\,{}^A\mathbf{R}_B & -{}^A\mathbf{y}_C^T\,{}^A\mathbf{R}_B\,{}^B\mathbf{p}_i^\wedge\end{bmatrix}\begin{bmatrix}{}^B\boldsymbol{v}_{A,B}\\ {}^B\boldsymbol{\omega}_{A,B}\end{bmatrix}, \tag{4.58}$$

which we write in compact form as

$$\boldsymbol{\gamma}_{Ti} = \mathbf{w}_{Ti}^T\mathbf{v} \ \in \mathbb{R}^2, \tag{4.59}$$

where $\mathbf{w}_{Ti}^T \in \mathbb{R}^{2\times n}$ is the matrix containing the force directions of the friction forces acting on the contact point $\mathbf{p}_i$. The set-valued force law of (4.50) can now be extended to the spatial case to obtain

$$\boldsymbol{\lambda}_{Ti} = \text{prox}_{\mathcal{C}_{Ti}}\left(\boldsymbol{\lambda}_{Ti} - r\boldsymbol{\gamma}_{Ti}\right), \quad \text{with} \quad \mathcal{C}_{Ti} = \{\boldsymbol{\lambda}_{Ti} \mid \|\boldsymbol{\lambda}_{Ti}\| \leq \mu\lambda_{Ni}\} \quad \forall i \in \mathcal{I}_c \tag{4.60}$$

and $r > 0$. Note that in (4.60), the set of admissible values of the tangential force $\boldsymbol{\lambda}_{Ti}$ is given by the convex set $\mathcal{C}_{Ti}$, that we will refer to as the isotropic *friction disk* as in [131], illustrated in Figure 4.7.

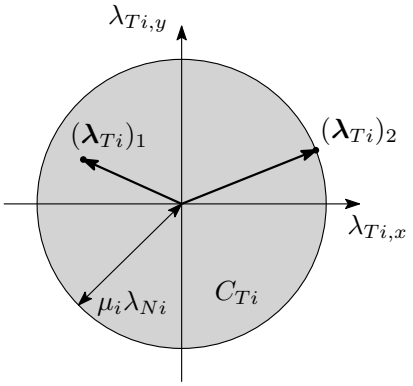


Figure 4.7: Schematic representation of Coulomb's isotropic friction law. Two situations are illustrated: the point $(\boldsymbol{\lambda}_{Ti})_1$ corresponds to a situation where $\|\boldsymbol{\gamma}_{Ti}\| = 0$, whereas the point $(\boldsymbol{\lambda}_{Ti})_2$ corresponds to a situation where $\|\boldsymbol{\gamma}_{Ti}\| > 0$.

The set-valued force laws for Coulomb-friction will now be written in terms of momenta instead of forces. This is done to incorporate Coulomb's law in the measure differential equations of (4.15) and in order to account for impulsive friction forces associated with a certain amount of restitution. Let us introduce the variable $\boldsymbol{\xi}_{Ti} \in \mathbb{R}^2$ according to

$$\boldsymbol{\xi}_{Ti} = \boldsymbol{\gamma}_{Ti}^{+} + \boldsymbol{e}_{Ti}\boldsymbol{\gamma}_{Ti}^{-}, \tag{4.61}$$

where $\boldsymbol{e}_{Ti} \in \mathbb{R}^2$ denotes the coefficient of restitution in tangential direction for which holds that $\|\boldsymbol{e}_{Ti}\| \leq 1$. In most practical cases however, its value is zero [113]. The set-valued force law for $\boldsymbol{\xi}_{Ti}$ and $\boldsymbol{\Lambda}_{Ti}$ then becomes

$$\boldsymbol{\Lambda}_{Ti} \geq 0, \quad \boldsymbol{\xi}_{Ti} \geq 0, \quad \boldsymbol{\xi}_{Ti}\boldsymbol{\Lambda}_{Ti} = 0, \quad i \in \mathcal{I}_c. \tag{4.62}$$

Note that, dimension-wise, both $\boldsymbol{\xi}_{Ti} \in \mathbb{R}^2$ and $\boldsymbol{\Lambda}_{Ti} \in \mathbb{R}^2$ as both of these variables have a component in x- and y-direction. Similar to (4.44), we will write the set-valued impact law for Coulomb's friction using the proximal point formulation to obtain

$$\boldsymbol{\Lambda}_{Ti} = \operatorname{prox}_{\mathcal{C}_{Ti}}\left(\boldsymbol{\Lambda}_{Ti} - r\boldsymbol{\xi}_{Ti}\right), \quad \text{with} \quad \mathcal{C}_{Ti} = \{\boldsymbol{\Lambda}_{Ti} \mid \|\boldsymbol{\Lambda}_{Ti}\| \leq \mu\Lambda_{Ni}\} \quad \forall i \in \mathcal{I}_c \tag{4.63}$$

and $r > 0$. Similar to (4.46), we define the column of all tangential impulsive forces $\boldsymbol{\Lambda}_T$ as

$$\boldsymbol{\Lambda}_T = \operatorname{col}\{\boldsymbol{\Lambda}_{Ti}\}, \ i \in \mathcal{I}_c, \tag{4.64}$$

such that $\boldsymbol{\Lambda}_T \in \mathbb{R}^{2n_c}$, as in (4.15). If again, for example, contacts 4 and 8 are closed, hence $\mathcal{I}_c = \{4, 8\}$, then $\boldsymbol{\Lambda}_T = \begin{bmatrix}\boldsymbol{\Lambda}_{T4}^T & \boldsymbol{\Lambda}_{T8}^T\end{bmatrix}^T$.

Now that the constitutive laws for contact, impact, and friction are discussed, we can combine them, together with the dynamics of (4.15), into a numerical model. This is done by the use of a so-called time-stepping scheme, which will be the topic of the next section. For more information on modeling friction laws, the reader is referred to [111, 131, 132].

4.3.5 Numerical integration using time-stepping

This section discusses the time-stepping method, introduced by Moreau [112] and further developed in various works such as [114, 116, 133–135]. The advantage of the time-stepping method is that, when integrating the equations of motion (4.15) over a time-interval possibly containing one or more time instants at which impulsive loads occur, the contributions of the smooth forces and impulsive forces are both taken into account. This avoids the need for

event detection of the impacts, which for event-driven schemes can cause serious problems when accumulation of impacts occurs.

The dynamics of the system can be described by the measure differential equations (4.15) and the constitutive laws for contact, impact, and friction such that the entire dynamics for both continuous and impulsive phases of motion are given as

$$\begin{aligned}
&\mathbb{M}d\mathbf{v}(t) + \mathbf{v}(t)\bar{\times}^*\mathbb{M}\mathbf{v}(t)dt = \mathbf{f}(t)dt + \mathbf{W}_N d\mathbf{P}_N + \mathbf{W}_T d\mathbf{P}_T, \\
&\boldsymbol{\lambda}_N = \operatorname{prox}_{\mathcal{C}_N}(\boldsymbol{\lambda}_N - r\mathbf{g}_N) \quad \text{with} \quad \mathcal{C}_N = \{\boldsymbol{\lambda}_N \in \mathbb{R}^{n_c} \mid \boldsymbol{\lambda}_N \geq \mathbf{0}\}, \\
&\boldsymbol{\lambda}_{Ti} = \operatorname{prox}_{\mathcal{C}_{Ti}}(\boldsymbol{\lambda}_{Ti} - r\boldsymbol{\gamma}_{Ti}) \quad \text{with} \quad \mathcal{C}_{Ti} = \{\boldsymbol{\lambda}_{Ti} \mid \|\boldsymbol{\lambda}_{Ti}\| \leq \mu\lambda_{Ni}\} \quad \forall i \in \mathcal{I}_c, \\
&\boldsymbol{\Lambda}_N = \operatorname{prox}_{\mathcal{C}_N}(\boldsymbol{\Lambda}_N - r\boldsymbol{\xi}_N) \quad \text{with} \quad \mathcal{C}_N = \{\boldsymbol{\Lambda}_N \in \mathbb{R}^{n_c} \mid \boldsymbol{\Lambda}_N \geq \mathbf{0}\}, \\
&\boldsymbol{\Lambda}_{Ti} = \operatorname{prox}_{\mathcal{C}_{Ti}}(\boldsymbol{\Lambda}_{Ti} - r\boldsymbol{\xi}_{Ti}) \quad \text{with} \quad \mathcal{C}_{Ti} = \{\boldsymbol{\Lambda}_{Ti} \mid \|\boldsymbol{\Lambda}_{Ti}\| \leq \mu\Lambda_{Ni}\} \quad \forall i \in \mathcal{I}_c,
\end{aligned} \tag{4.65}$$

with r now chosen as $r = 0.1$. The time-stepping algorithm can be thought of as a mid-point DAE-integrator. For a single time step Δt, the starting time is denoted as t_a, whereas the end time is denoted as t_e such that $t_e = t_a + \Delta t$. The time at the mid-point of the time-step is denoted as t_m such that $t_m = t_a + \frac{1}{2}\Delta t$. At the beginning of each time step, the pose ${}^A\mathbf{H}(t_a)_B$ of the object and the left trivialized velocity ${}^B\mathbf{v}(t_a)_{A,B}$ are known, either from initial conditions or computed from the last time-step. First the pose of the object at the mid-point is computed by the kinematic model, according to

$${}^A\mathbf{H}(t_m)_B = {}^A\mathbf{H}(t_a)_B \operatorname{Exp}\big((1/2\Delta t)\mathbf{v}^\wedge(t_a)\big), \tag{4.66}$$

where $\operatorname{Exp} : \mathfrak{se}(3) \to \mathrm{SE}(3)$. At this point, the gap function of each contact point is evaluated to determine if a contact has been closed. Hence, we check whether the condition of (4.24) still holds. Therefore, we define $g_{Ni,m}$ as the gap-function at the mid-point for each contact point, given as

$$g_{Ni,m} = {}^A\mathbf{z}_C^T \left(\big({}^A\mathbf{o}(t_m)_B + {}^A\mathbf{R}(t_m)_B {}^B\mathbf{p}_i\big) - {}^A\mathbf{o}_C \right) \quad i \in \mathcal{I}, \tag{4.67}$$

where ${}^A\mathbf{o}(t_m)_B$ and ${}^A\mathbf{R}(t_m)_B$ follow directly from ${}^A\mathbf{H}(t_m)_B$ in (4.66). The matrices of generalized force directions for the contact/impact forces and the friction forces at t_m are now given by

$$\mathbf{W}_{N,m} = \{\mathbf{w}_{Ni,m}\},\ i \in \mathcal{I}_c \quad \text{and} \quad \mathbf{W}_{T,m} = \{\mathbf{w}_{Ti,m}\},\ i \in \mathcal{I}_c, \tag{4.68}$$

respectively, with $\mathbf{w}_{Ni.m}$ given by (4.39) and (4.40) and $\mathbf{w}_{Ti,m}$ given by (4.58) and (4.59), evaluated at the mid-point. Dimension-wise, this means that $\mathbf{W}_{N,m} \in \mathbb{R}^{n \times n_c}$ and $\mathbf{W}_{T,m} \in \mathbb{R}^{n \times 2n_c}$. Next, the equations of motion as given by the first line in (4.65) are integrated using the mid-point rule to obtain

$$\mathbb{M}\big(\mathbf{v}(t_e) - \mathbf{v}(t_a)\big) + \mathbf{v}(t_a)\bar{\times}^*\mathbb{M}\mathbf{v}(t_a)\Delta t = \mathbf{f}_m \Delta t + \mathbf{W}_{N,m}\mathbf{P}_N + \mathbf{W}_{T,m}\mathbf{P}_T, \tag{4.69}$$

where we used the approximations

$$\begin{aligned}
\int_{\Delta t} \mathbb{M} d\mathbf{v}(t) &\approx \mathbb{M}\big(\mathbf{v}(t_e) - \mathbf{v}(t_a)\big),\\
\int_{\Delta t} \mathbf{v}(t) \bar{\times}^* \mathbb{M}\mathbf{v}(t) dt &\approx \mathbf{v}(t_a) \bar{\times}^* \mathbb{M}\mathbf{v}(t_a)\Delta t,\\
\int_{\Delta t} \mathbf{f}(t) dt &\approx \mathbf{f}_m \Delta t,\\
\int_{\Delta t} \mathbf{W}_N d\mathbf{P}_N &\approx \mathbf{W}_{N,m}\mathbf{P}_N,\\
\int_{\Delta t} \mathbf{W}_T d\mathbf{P}_T &\approx \mathbf{W}_{T,m}\mathbf{P}_T,
\end{aligned} \tag{4.70}$$

with

$$\mathbf{P}_N = \boldsymbol{\Lambda}_N + \boldsymbol{\lambda}_N \Delta t \quad \text{and} \quad \mathbf{P}_T = \boldsymbol{\Lambda}_T + \boldsymbol{\lambda}_T \Delta t, \tag{4.71}$$

as in [114,136]. The last four equations of (4.65) are now included to finally obtain the total set of equations needed to be solved in the time-stepping scheme:

$$\begin{aligned}
\mathbf{v}(t_e) &= \mathbf{v}(t_a) + \mathbb{M}^{-1}\Big(\mathbf{f}_m \Delta t - \mathbf{v}(t_a) \bar{\times}^* \mathbb{M}\mathbf{v}(t_a)\Delta t + \mathbf{W}_{N,m}\mathbf{P}_N + \mathbf{W}_{T,m}\mathbf{P}_T\Big),\\
\boldsymbol{\lambda}_N &= \text{prox}_{\mathcal{C}_N}\left(\boldsymbol{\lambda}_N - r\mathbf{g}_N\right) \quad \text{with} \quad \mathcal{C}_N = \{\boldsymbol{\lambda}_N \in \mathbb{R}^{n_c} \mid \boldsymbol{\lambda}_N \geq \mathbf{0}\},\\
\boldsymbol{\lambda}_{Ti} &= \text{prox}_{\mathcal{C}_{Ti}}\left(\boldsymbol{\lambda}_{Ti} - r\boldsymbol{\gamma}_{Ti}\right) \quad \text{with} \quad \mathcal{C}_{Ti} = \{\boldsymbol{\lambda}_{Ti} \mid \|\boldsymbol{\lambda}_{Ti}\| \leq \mu\lambda_{Ni}\} \quad \forall i \in \mathcal{I}_c,\\
\boldsymbol{\Lambda}_N &= \text{prox}_{\mathcal{C}_N}\left(\boldsymbol{\Lambda}_N - r\boldsymbol{\xi}_N\right) \quad \text{with} \quad \mathcal{C}_N = \{\boldsymbol{\Lambda}_N \in \mathbb{R}^{n_c} \mid \boldsymbol{\Lambda}_N \geq \mathbf{0}\},\\
\boldsymbol{\Lambda}_{Ti} &= \text{prox}_{\mathcal{C}_{Ti}}\left(\boldsymbol{\Lambda}_{Ti} - r\boldsymbol{\xi}_{Ti}\right) \quad \text{with} \quad \mathcal{C}_{Ti} = \{\boldsymbol{\Lambda}_{Ti} \mid \|\boldsymbol{\Lambda}_{Ti}\| \leq \mu\Lambda_{Ni}\} \quad \forall i \in \mathcal{I}_c,
\end{aligned} \tag{4.72}$$

where $r = 0.1$, and $\boldsymbol{\xi}_N$ and $\boldsymbol{\xi}_T$ are given by

$$\boldsymbol{\xi}_N = \boldsymbol{\gamma}_{N,e} + \boldsymbol{e}_N \boldsymbol{\gamma}_{N,a} \quad \text{and} \quad \boldsymbol{\xi}_T = \boldsymbol{\gamma}_{T,e} + \boldsymbol{e}_T \boldsymbol{\gamma}_{T,a}, \tag{4.73}$$

where

$$\begin{aligned}
\boldsymbol{\gamma}_{N,a} &= \mathbf{W}_{N,a}^T \mathbf{v}(t_a), \qquad \boldsymbol{\gamma}_{N,e} = \mathbf{W}_{N,m}^T \mathbf{v}(t_e),\\
\boldsymbol{\gamma}_{T,a} &= \mathbf{W}_{T,a}^T \mathbf{v}(t_a), \qquad \boldsymbol{\gamma}_{T,e} = \mathbf{W}_{T,m}^T \mathbf{v}(t_e).
\end{aligned} \tag{4.74}$$

In (4.74), $\mathbf{W}_{N,a}$ and $\mathbf{W}_{N,m}$ are the matrices containing the generalized force directions of the contact/impact at te beginning of the time step and at the mid-point, respectively. These are given by

$$\mathbf{W}_{N,a} = \left\{ \begin{bmatrix} {}^A\mathbf{z}_C^T {}^A\mathbf{R}(t_a)_B & -{}^A\mathbf{z}_C^T {}^A\mathbf{R}(t_a)_B {}^B\mathbf{p}_i^\wedge \end{bmatrix}^T \right\} \quad i \in \mathcal{I}_c \tag{4.75}$$

and

$$\mathbf{W}_{N,m} = \left\{ \begin{bmatrix} {}^A\mathbf{z}_C^T {}^A\mathbf{R}(t_m)_B & -{}^A\mathbf{z}_C^T {}^A\mathbf{R}(t_m)_B {}^B\mathbf{p}_i^\wedge \end{bmatrix}^T \right\} \quad i \in \mathcal{I}_c. \tag{4.76}$$

Similarly, $\mathbf{W}_{T,a}$ and $\mathbf{W}_{T,m}$ are the matrices containing the generalized force directions of the friction at te beginning of the time step and at the mid-point respectively, given by

$$\mathbf{W}_{T,a} = \left\{ \begin{bmatrix} {}^A\mathbf{x}_C^T {}^A\mathbf{R}(t_a)_B & -{}^A\mathbf{x}_C^T {}^A\mathbf{R}(t_a)_B {}^B\mathbf{p}_i^\wedge \\ {}^A\mathbf{y}_C^T {}^A\mathbf{R}(t_a)_B & -{}^A\mathbf{y}_C^T {}^A\mathbf{R}(t_a)_B {}^B\mathbf{p}_i^\wedge \end{bmatrix}^T \right\} \quad i \in \mathcal{I}_c, \tag{4.77}$$

and

$$\mathbf{W}_{T,m} = \left\{ \begin{bmatrix} {}^A\mathbf{x}_C^T {}^A\mathbf{R}(t_m)_B & -{}^A\mathbf{x}_C^T {}^A\mathbf{R}(t_m)_B {}^B\mathbf{p}_i^\wedge \\ {}^A\mathbf{y}_C^T {}^A\mathbf{R}(t_m)_B & -{}^A\mathbf{y}_C^T {}^A\mathbf{R}(t_m)_B {}^B\mathbf{p}_i^\wedge \end{bmatrix}^T \right\} \quad i \in \mathcal{I}_c. \tag{4.78}$$

Note that the unknown velocity $\mathbf{v}(t_e)$ is introduced in (4.72) through (4.73) and (4.74). The

equations (4.72) must be solved to find $\mathbf{v}(t_e)$, $\boldsymbol{\lambda}_N$, $\boldsymbol{\lambda}_T$, $\boldsymbol{\Lambda}_N$, and $\boldsymbol{\Lambda}_T$. These equations can be solved by a NCP formulation, as discussed in [114]. There are multiple solvers possible to solve an NCP, as discussed in [137] and [118, Chapter 10]. Often used methods are Lemke's algorithm or pivoting methods as discussed in [138, Section 3]. We follow the approach of Leine and Nijmeijer as proposed in [114, Chapter 6], due to its simplicity and effectiveness in our specific case of a single rigid body, and this will be discussed in Section 4.3.6. This is a modified Newton method and similar methods are used in [139] and [137, Section 5].

When the velocity at the end of the time step $\mathbf{v}(t_e)$ is known, the pose of the object can be updated from the mid-point to the end of the time step. Hence,

$$ {}^{A}\mathbf{H}(t_e)_B = {}^{A}\mathbf{H}(t_m)_B \operatorname{Exp}\big((1/2\Delta t)\mathbf{v}^{\wedge}(t_e)\big). \tag{4.79} $$

In summary, the time-stepping scheme will have the following steps:

1. The pose ${}^{A}\mathbf{H}(t_a)_B$ of the object and the left trivialized velocity $\mathbf{v}(t_a)$ are known at the beginning of the time step t_a;
2. The midpoint is computed according to (4.66);
3. The gap-function is evaluated at the mid-point to see whether a contact has been closed; hence we check whether $g_{Ni,m} > 0$, for $i \in \mathcal{I}$, to determine the set $\mathcal{I}_c$ given by (4.26);
4. The system matrices at the mid-point, $\mathbf{f}_m$, $\mathbf{W}_{N,m}$, and $\mathbf{W}_{T,m}$ are computed by (2.85), (4.75), and (4.77) respectively;
5. The values for $\mathbf{v}(t_e)$, $\boldsymbol{\lambda}_N$, $\boldsymbol{\lambda}_T$, $\boldsymbol{\Lambda}_N$, and $\boldsymbol{\Lambda}_T$ at the end of the time step are computed by solving (4.72) using fixed point iteration (NCP solver);
6. The pose of the object is updated to the next time step according to (4.79).

What remains is to understand how the velocity $\mathbf{v}(t_e)$, normal forces $\boldsymbol{\lambda}_N$, tangential forces $\boldsymbol{\lambda}_T$, normal impulsive forces $\boldsymbol{\Lambda}_N$, and tangential impulsive forces $\boldsymbol{\Lambda}_T$ are computed. This will be explained in the next section.

4.3.6 Fixed-point iteration for solving the nonlinear algebraic equations

Fixed point iteration is used in order to find the values for $\mathbf{v}(t_e)$, $\boldsymbol{\lambda}_N$, $\boldsymbol{\lambda}_T$, $\boldsymbol{\Lambda}_N$, and $\boldsymbol{\Lambda}_T$ at the end of the time step. In general, fixed-point iteration is a numerical method to find the roots of a function $f(\boldsymbol{x})$, hence to find the points $\boldsymbol{x}$ for which holds that

$$ f(\boldsymbol{x}) = 0. \tag{4.80} $$

Finding the roots of the function $f(\boldsymbol{x})$ is similar to finding the so-called fixed points of the function $g(\boldsymbol{x})$ [140, Chapter 3], hence to find the points $\boldsymbol{x}$ for which holds that

$$ \boldsymbol{x}_{k+1} = g(\boldsymbol{x}_k), \tag{4.81} $$

where the function $g(\boldsymbol{x})$ is a mapping of the nonlinear functions (4.80), which may be non-unique. In our case, the unknown variables of (4.80) are given by

$$ \boldsymbol{x} = \begin{bmatrix} \mathbf{v}(t_e)^T & \boldsymbol{\lambda}_N^T & \boldsymbol{\lambda}_T^T & \boldsymbol{\Lambda}_N^T & \boldsymbol{\Lambda}_T^T \end{bmatrix}^T. \tag{4.82} $$

Note that (4.72) is already written in the same form of (4.81) such that the set of nonlinear functions can be solved using fixed-point iteration, making use of (4.73) and (4.74). In pseudo code, the fixed-point iteration is written as in Algorithm 1. The prox-functions in

Algorithm 1 Fixed-point Iteration

$\mathbf{P}_N = \mathbf{0}$, $\mathbf{P}_T = \mathbf{0}$, tol $= 1 \cdot 10^{-5}$, $r = 0.1$
repeat
  Compute the velocity at the end of the time step:
  $\mathbf{v}(t_e) \leftarrow \mathbf{v}(t_a) + \mathbb{M}^{-1}\left(\mathbf{f}_m \Delta t - \mathbf{v}(t_a)\bar{\times}^* \mathbb{M}\mathbf{v}(t_a)\Delta t + \mathbf{W}_{N,m}\mathbf{P}_N + \mathbf{W}_{T,m}\mathbf{P}_T\right)$

  Compute normal and tangential velocities at the contact points:
  $\boldsymbol{\gamma}_{N,a} \leftarrow \mathbf{W}_{N,a}^T \mathbf{v}(t_a)$
  $\boldsymbol{\gamma}_{N,e} \leftarrow \mathbf{W}_{N,m}^T \mathbf{v}(t_e)$
  $\boldsymbol{\gamma}_{T,a} \leftarrow \mathbf{W}_{T,a}^T \mathbf{v}(t_a)$
  $\boldsymbol{\gamma}_{T,e} \leftarrow \mathbf{W}_{T,m}^T \mathbf{v}(t_e)$

  $\boldsymbol{\xi}_N \leftarrow \boldsymbol{\gamma}_{N,e} + \boldsymbol{e}_N \boldsymbol{\gamma}_{N,a}$
  $\boldsymbol{\xi}_T \leftarrow \boldsymbol{\gamma}_{T,e} + \boldsymbol{e}_T \boldsymbol{\gamma}_{T,a}$

  Update the momenta:
  $\mathbf{P}_{N,old} \leftarrow \boldsymbol{\Lambda}_N + \boldsymbol{\lambda}_N \Delta t$
  $\mathbf{P}_{T,old} \leftarrow \boldsymbol{\Lambda}_T + \boldsymbol{\lambda}_T \Delta t$

  $\boldsymbol{\Lambda}_N \leftarrow \text{prox}_{\mathcal{C}_N}(\boldsymbol{\Lambda}_N - r\boldsymbol{\xi}_N)$
  $\boldsymbol{\Lambda}_{Ti} \leftarrow \text{prox}_{\mathcal{C}_{Ti}}(\boldsymbol{\Lambda}_{Ti} - r\boldsymbol{\xi}_{Ti}) \quad \forall i \in \mathcal{I}_c$
  $\boldsymbol{\lambda}_N \leftarrow \text{prox}_{\mathcal{C}_N}(\boldsymbol{\lambda}_N - r\mathbf{g}_N)$
  $\boldsymbol{\lambda}_{Ti} \leftarrow \text{prox}_{\mathcal{C}_{Ti}}(\boldsymbol{\Lambda}_{Ti} - r\boldsymbol{\xi}_{Ti}) \quad \forall i \in \mathcal{I}_c$

  $\mathbf{P}_N \leftarrow \boldsymbol{\Lambda}_N + \boldsymbol{\lambda}_N \Delta t$
  $\mathbf{P}_{Ti} \leftarrow \boldsymbol{\Lambda}_{Ti} + \boldsymbol{\lambda}_{Ti} \Delta t$

  Compute the error:
  error $\leftarrow \|(\mathbf{P}_N - \mathbf{P}_{N,old})\|_2 + \|(\mathbf{P}_T - \mathbf{P}_{T,old})\|_2$
until error $<$ tol

Algorithm 1 are given as in [136, Chapter 2]. Since $\boldsymbol{\Lambda}_N$ can only be positive, the function $\text{prox}_{\mathcal{C}_N}(\boldsymbol{x})$ with $\boldsymbol{x} \in \mathbb{R}^{n_c}$ can be written as

$$\text{prox}_{\mathcal{C}_N}(\boldsymbol{x}) = \begin{cases} \mathbf{0} & \text{for} \quad \boldsymbol{x} \leq \mathbf{0} \\ \boldsymbol{x} & \text{for} \quad \boldsymbol{x} > \mathbf{0} \end{cases} \qquad \text{for} \quad \mathcal{C}_N = (\mathbb{R}^{n_c})^+, \tag{4.83}$$

where we used that $\boldsymbol{x} = \boldsymbol{\Lambda}_N - r\boldsymbol{\xi}_N$. The function $\text{prox}_{\mathcal{C}_{Ti}}(x)$ with $\boldsymbol{x} \in \mathbb{R}^2$, as given in the last line of (4.72), can be modeled as

$$\text{prox}_{\mathcal{C}_{Ti}}(\boldsymbol{x}) = \begin{cases} \boldsymbol{x} & \text{for} \quad \|\boldsymbol{x}\| \leq a \\ a\frac{\boldsymbol{x}}{\|\boldsymbol{x}\|} & \text{for} \quad \|\boldsymbol{x}\| > a \end{cases} \quad \text{for} \quad \mathcal{C}_{Ti} = \{\boldsymbol{x} \in \mathbb{R}^2 \mid \|\boldsymbol{x}\| \leq a\}, \tag{4.84}$$

where we used that $a = \mu\Lambda_{Ni}$ and $\boldsymbol{x} = \boldsymbol{\Lambda}_{Ti} - r\boldsymbol{\xi}_{Ti}$.

4.3.7 Conclusion on the motion model

What remains, is the link between the motion model described in this section and the notation of the motion model $f(\mathbf{x})$ in the scope of the GUPF discussed in Chapter 3. Recall that the state of a particle at the start time of a time step t_a is given as $\mathbf{x}_t = ({}^{A}\mathbf{R}(t_a)_B, {}^{A}\mathbf{o}(t_a)_B, {}^{B}\boldsymbol{v}(t_a)_{A,B}, {}^{B}\boldsymbol{\omega}(t_a)_{A,B})$. In Section 4.3.5, we showed that the pose is updated by the use of (4.66) and (4.79). This results in the homogeneous transformation matrix ${}^{A}\mathbf{H}(t_e)_{A,B}$, which contains the rotation matrix ${}^{A}\mathbf{R}(t_e)_B$ and position ${}^{A}\mathbf{o}(t_e)_B$ of the cuboid at the end of the time step. The velocity at the end of the time step is then computed by the first line of (4.72), which contains the linear velocity ${}^{B}\boldsymbol{v}(t_e)_{A,B}$ and angular velocity ${}^{B}\boldsymbol{\omega}(t_e)_{A,B}$ of the cuboid at the end of the time step. The state $\mathbf{x}_t$ can thus be propagated by the use of the motion model described in this section to obtain the state at the end of the time step, $\mathbf{x}_{t+1}$. This is then summarized in the function $f(\cdot)$, as used in Chapter 3.

For further readings on iterative solvers for non-linear functions using proximal function, the reader is referred to [130]. See also [122,136,141,142] for further readings on time stepping schemes for multibody dynamics with contact and friction. For modeling these systems, the reader is referred to [113, 114, 118, 121, 127, 143].

4.4 Conclusion

In this chapter, the likelihood function and the motion model were discussed. By the use of the geometric model of the cuboid, described in Section 4.1, the likelihood function is able to locate the object in the image plane. As tracking is the combination of both estimating the pose in the image and following the motion, Section 4.3 discusses the motion model used within the GUPF. In the next chapter, the proposed tracking algorithm will be compared to state-of-the-art methods by the use of simulations.

Chapter 5

Numerical Validation

In Chapters 3 and 4, we discussed the GUPF-L and the GUPF-Z, and the corresponding likelihood function and motion model. In this chapter, we will validate these algorithms by the use of simulations and compare their performance to that of state-of-the-art tracking algorithms.

Two different scenarios are considered in which a cuboid is tossed onto a surface. In the first scenario, the motion is parallel to the image plane of the camera and, in the second scenario, the cuboid is tossed towards the camera. This means that, in the first scenario, the appearance of the object in the image plane is relatively constant over time, while in the second scenario, the object size in the image increases as the object moves towards the camera. Typically, it is harder to track the object when it is far away from the camera, therefore appearing smaller in the image. These different scenarios therefore allow us to compare the performance of the different tracking algorithms with respect to this problem. Ground truth trajectories are created corresponding to these two scenarios to numerically test the tracking algorithms in simulation. Artificial images are created corresponding to these trajectories, as if they are obtained from a real RGB camera, and these images serve as the input to the tracking algorithms. In Section 5.1, we will discuss how the ground truth trajectories and the artificial images are created. In Section 5.2, it is discussed how the virtual measurement $\mathbf{Z}$, as used in the GUPF-Z, is obtained by adding noise to the ground truth trajectories. Finally, Section 5.3 covers the numerical evaluation of the different tracking algorithms and gives a comparison in their performance.

5.1 Ground truth trajectories for simulation

In simulation, the geometric model of the cuboid as discussed in Section 4.1 is used. The ground truth trajectories are then created using both this geometric model and the content of Section 4.3. In Figure 5.1a, the ground truth trajectory is shown for the first scenario, where the cuboid's initial pose is given by

$$ {}^{A}\mathbf{H}_{B} = \begin{bmatrix} 1 & 0 & 0 & 0.5 \\ 0 & \cos(\theta) & -\sin(\theta) & 0 \\ 0 & \sin(\theta) & \cos(\theta) & 0.8 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{5.1} $$

with $\theta = 110$ degrees. The initial velocity is given by

$$ {}^{B}\mathbf{v}_{A,B} = \begin{bmatrix} -2 & 0 & 0 & 5 & 2 & 0 \end{bmatrix}^{T}. \tag{5.2} $$

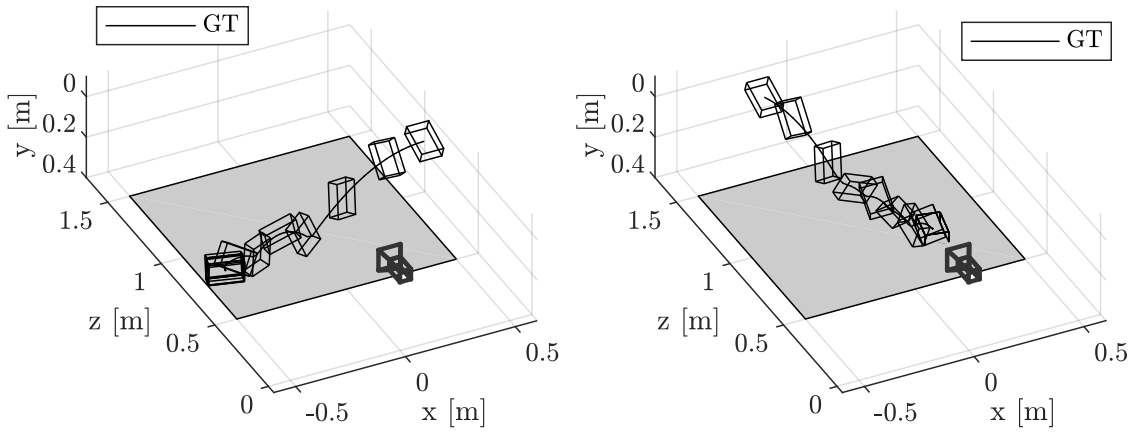


Figure 5.1: The ground truth trajectories. First scenario (a) and second scenario (b).

From this initial state, the cuboid follows a trajectory, impacting the contact surface multiple times, as shown in Figure 5.1a. We will refer to this trajectory as $\mathbf{GT}_t = ({}^A\mathbf{R}(t)_B^{GT}, {}^A\mathbf{o}(t)_B^{GT})$. In Figure 5.1a, the trajectory of ${}^A\mathbf{o}(t)_B^{GT}$ is indicated by the black line and the pose of the cuboid is shown for the frames shown in Figure 5.2. The duration of this trajectory is $T = 1.1$ seconds. Assuming to have a 60 fps camera, a total of 65 artificial images corresponding to this trajectory is obtained, where these artificial images are created using the content of Section 2.5. Several snapshots taken from this video sequence are shown in Figure 5.2.

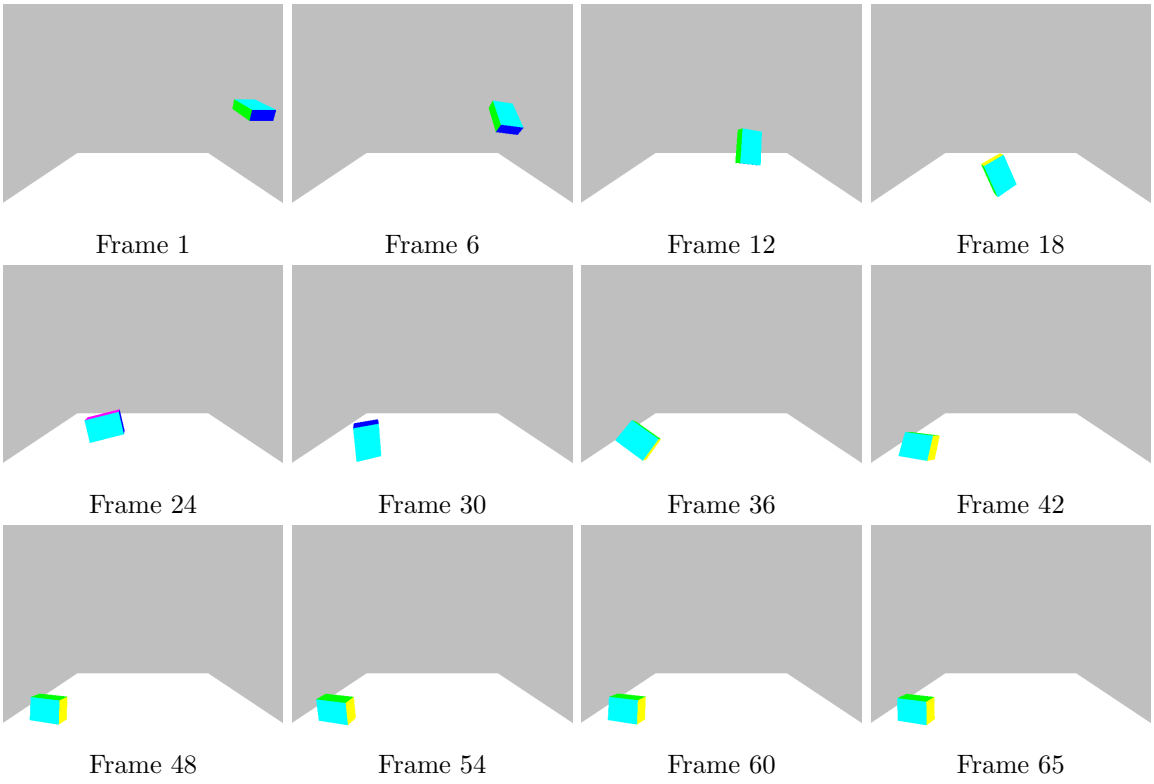


Figure 5.2: The artificially created images of the ground truth trajectory as seen from the camera. Images corresponding to the trajectory of Figure 5.1a.

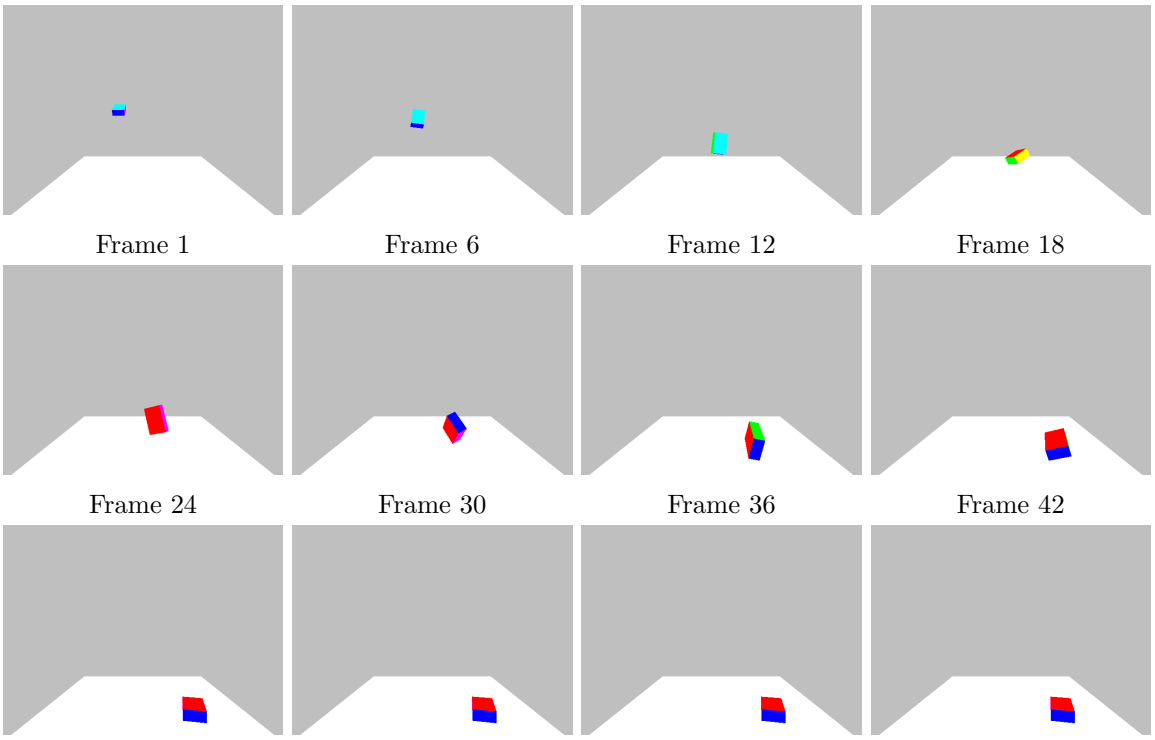


Figure 5.3: The artificially created images of the ground truth trajectory as seen from the camera. Images corresponding to the trajectory of Figure 5.1b.

In Figure 5.1b, the ground truth trajectory is shown for the second scenario, where the cuboid's initial pose is given by

$$ {}^{A}\mathbf{H}_{B} = \begin{bmatrix} 1 & 0 & 0 & -0.2 \\ 0 & \cos(\theta) & -\sin(\theta) & 0 \\ 0 & \sin(\theta) & \cos(\theta) & 1.5 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{5.3} $$

with $\theta = 110$ degrees, and the initial velocity is given by

$$ {}^{B}\mathbf{v}_{A,B} = \begin{bmatrix} 1 & 1.2 & 0 & 5 & 2 & 0 \end{bmatrix}^{T}. \tag{5.4} $$

Similar to the first scenario, the second scenario also contains 65 images, and several snapshots taken from this video sequence are shown in Figure 5.3, corresponding to the poses shown in Figure 5.1b.

5.2 Measurement data

As discussed in Section 3.4, two types of measurements are used for simulation, one resulting in the GUPF-Z and the other resulting in the GUPF-L. In the former, Gaussian noise is added to the ground truth data to simulate a measurement using the content of Section 3.4.2. For both scenarios, a trajectory is obtained by setting the covariance matrices in orientation and position as

$$ \boldsymbol{P}^{\mathbf{R}} = 5 \cdot 10^{-3} \mathrm{diag}\left(\begin{bmatrix} 1 & 1 & 1 \end{bmatrix}\right) \text{ rad}^2 \quad \text{and} \quad \boldsymbol{P}^{\mathbf{o}} = 1 \cdot 10^{-3} \mathrm{diag}\left(\begin{bmatrix} 1 & 1 & 1 \end{bmatrix}\right) \text{ m}^2, \tag{5.5} $$

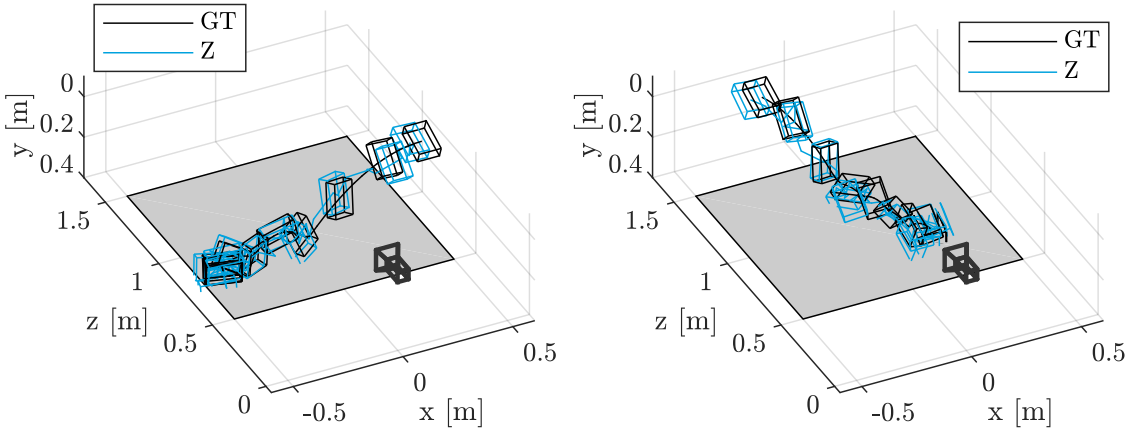


Figure 5.4: Trajectory of the measurement **Z** shown together with the ground truth **GT** for the first scenario (a) and the second scenario (b).

respectively, resulting in a measurement which we denote as $\mathbf{Z}_t = ({}^{A}\mathbf{R}(t)_B^Z, {}^{A}\mathbf{o}(t)_B^Z)$. In Figure 5.4, these measurements are shown in light blue, together with the ground truth of both scenarios. The origin, ${}^{A}\mathbf{o}(t)_B^Z$, is shown by the light blue line, and the poses corresponding to the snapshots of Figures 5.2 and 5.3 are visualized. Figure 5.4 shows that the measurements are noisy: at each time instant of the trajectories there is a position error between the origin of the measurement and the ground truth trajectory. Besides, an error in orientation can be observed. We define the error in position between the measurement and the ground truth as

$$e_{\mathbf{o}} = {}^{A}\mathbf{o}_B^Z - {}^{A}\mathbf{o}_B^{GT}, \tag{5.6}$$

where the dependency on time is omitted for the sake of brevity. We define e_x, e_y, and e_z as the errors in x, y, and z-direction, respectively, which are the components of $e_{\mathbf{o}}$. We define the error in orientation as

$$e_{\mathbf{R}} = \log\left(\left(\mathbf{R}^{GT}\right)^{-1}\mathbf{R}^{Z}\right) \in \mathfrak{so}(3), \tag{5.7}$$

where we used the short-hand notation $\mathbf{R}^{GT} = {}^{A}\mathbf{R}(t)_B^{GT}$ and $\mathbf{R}^{Z} = {}^{A}\mathbf{R}(t)_B^{Z}$. Since (5.7) gives the error as an element of $\mathfrak{so}(3)$, we can take its norm, denoted by $\|e_{\mathbf{R}}\|$, to obtain the

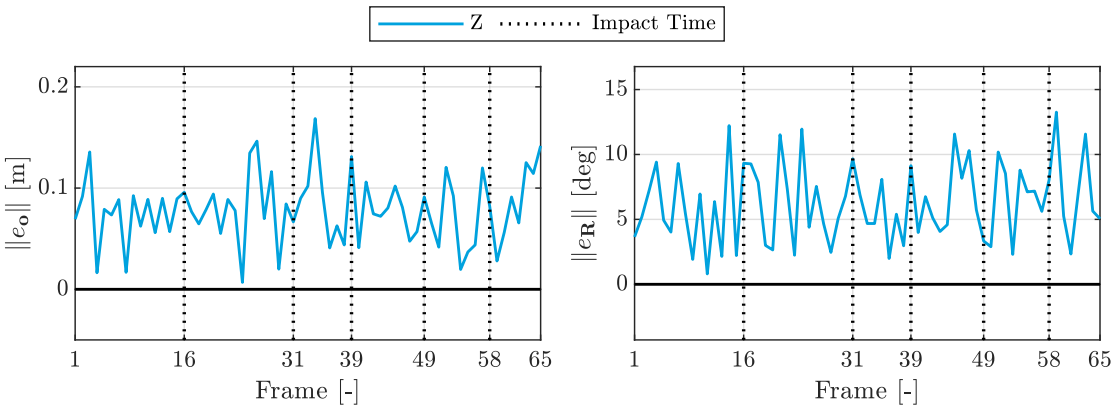


Figure 5.5: Position error of the measurement (left) and angular error of the measurement (right) for the first scenario.

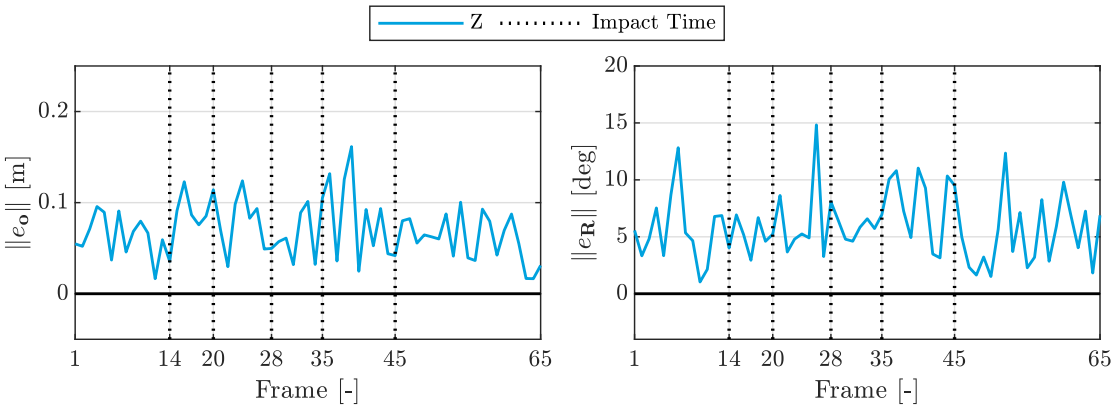


Figure 5.6: Position error of the measurement (left) and angular error of the measurement (right) for the second scenario.

angular error in radians, which can then easily be converted into degrees. The norm of the position error and angular error of the measurement are shown in Figures 5.5 and 5.6 for the first and second scenario, respectively. For the first scenario, the average of the norm of the position error is 0.079 m and the average angular error is 6.24 degrees. For the second scenario, the average of the norm of the position error is 0.070 m and the average angular error is 5.91 degrees.

5.3 Numerical evaluation

In this section, we will evaluate the GUPF-Z and the GUPF-L by the use of the simulations described in Section 5.1 and compare their performance to the works presented in [65] and [66]. In these works, a Particle Filter (PF) with a constant velocity model is used, and therefore differ from the GUPF-Z and GUPF-L both in filtering technique and motion model. To validate the comparison, we introduce the GUPF-CV in Section 5.3.1, which uses the proposed filtering technique described in Chapter 3, but with a motion model adopted from [65] and [66]. This allows us to see the effect of the proposed GUPF with respect to state-of-the-art tracking algorithms. Next, the GUPF-CV is compared to the GUPF-Z and GUPF-L, where the last two filters now use the motion model as described in Section 4.3. This allows us to see the effect of changing the motion model, and shows the benefit of modeling impacts and friction on the tracking performance.

In the simulations described in this section, we use the initial ground truth position as the initial state mean and set the elements of the initial state covariance as

$$\boldsymbol{P}^{\mathbf{R}} = 5 \cdot 10^{-4} \mathrm{diag}\left(\begin{bmatrix} 1 & 1 & 1 \end{bmatrix}\right) \text{ rad}^2, \qquad \boldsymbol{P}^{\mathbf{o}} = 1 \cdot 10^{-4} \mathrm{diag}\left(\begin{bmatrix} 1 & 1 & 1 \end{bmatrix}\right) \text{ m}^2, \tag{5.8}$$

$$\boldsymbol{P}^{\boldsymbol{v}} = 1 \cdot 10^{-7} \mathrm{diag}\left(\begin{bmatrix} 1 & 1 & 1 \end{bmatrix}\right) \text{ m}^2/\text{s}^2, \qquad \boldsymbol{P}^{\boldsymbol{\omega}} = 1 \cdot 10^{-7} \mathrm{diag}\left(\begin{bmatrix} 1 & 1 & 1 \end{bmatrix}\right) \text{ rad}^2/\text{s}^2, \tag{5.9}$$

from which the initial set of 50 particles is sampled according to (3.3), (3.4), and (3.5). In the following section, we will show the comparison between the GUPF-CV and the works presented in [65] and [66]. Next, in Sections 5.3.2 and 5.3.3, we show the comparison between the GUPF-CV and the GUPF-Z and GUPF-L for the two scenarios described in Section 5.1, respectively.

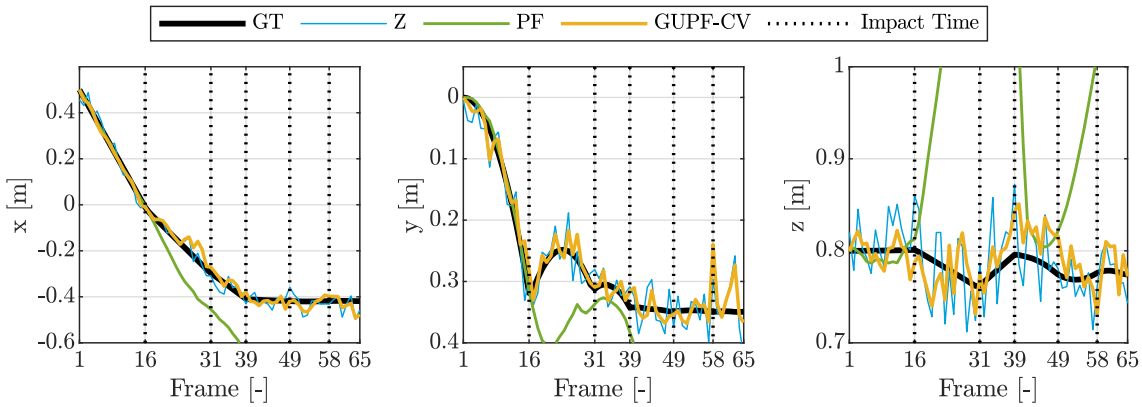


Figure 5.7: Results of the various tracking algorithms. The x-position (left), y-position (middle), and z-position (right) of the results are plotted together with the ground truth trajectory and the measurement.

5.3.1 Comparing the GUPF with the state-of-the-art

In [65] and [66], a constant velocity model is used to describe the state transition on SE(3). In this constant velocity model, no preference is given to any direction of motion such that the equations of motion can be written as

$$\mathbf{H}(t_e) = \mathbf{H}(t_a)\mathrm{Exp}\Big(\Delta t \mathbf{v}^\wedge(t) + \sqrt{\Delta t}\mathbf{a}^\wedge\Big), \tag{5.10}$$

$$\mathbf{v}^\wedge(t_e) = \frac{1}{\Delta t}\mathrm{Log}\Big(\mathbf{H}(t_a)^{-1}\mathbf{H}(t_e)\Big), \tag{5.11}$$

where $\mathbf{a} \in \mathbb{R}^6$ is a zero-mean white noise vector and $\Delta t = 1/60$, corresponding to the assumption of a 60fps camera. Note that in (5.10), we used the short-hand notation $\mathbf{v} = {}^B\mathbf{v}_{A,B}$ and $\mathbf{H} = {}^A\mathbf{H}_B$. Using this motion model in the framework of the GUPF, means that we have to write the state as an element of $\mathcal{S}$. This is a straightforward task, as at each time t, ${}^A\mathbf{R}(t)_B$ and ${}^A\mathbf{o}(t)_B$ directly follow from ${}^A\mathbf{H}(t)_B$, and ${}^A\boldsymbol{v}_{A,B}$ and ${}^A\boldsymbol{\omega}_{A,B}$ directly follow from ${}^B\mathbf{v}_{A,B}$. This allows us to directly implement this motion model in the GUPF which, together with the measurement $\mathbf{Z}$, results in the GUPF-CV.

We now compare the GUPF-CV with the PF of [65] and [66]. To this end, the first scenario is considered in which the motion of the cuboid is parallel to the image plane. The output of each algorithm is the position ${}^A\mathbf{o}_B$ and the orientation ${}^A\mathbf{R}_B$ of the cuboid for each frame. In Figure 5.7, the x, y, and z-components of ${}^A\mathbf{o}_B$ are shown as results of the different tracking algorithms, as well as for the ground truth trajectory and the measurement. The time instants at which an impact occurs are indicated as well. In Figure 5.7, we observe that the PF is able to closely follow the cuboid up to the first impact, as the tracking results are close to the ground truth. However, right after the first impact, the PF loses track of the cuboid. This is also clearly visible in Figure 5.8, where the norm of the position error and the norm of the orientation error are shown. Figure 5.8, shows that the error of the PF up to the first impact is small, but rapidly grows after the first impact occurs. The GUPF-CV, on the other hand, is able to stay closer to the ground truth trajectory. This also becomes clear from Table 5.1, where the root mean square (RMS) of the errors shown in Figure 5.8 are given. Since the GUPF-CV uses the measurement $\mathbf{Z}$, the RMS of the position and orientation errors of the measurement are shown as well. Besides, the error reduction is

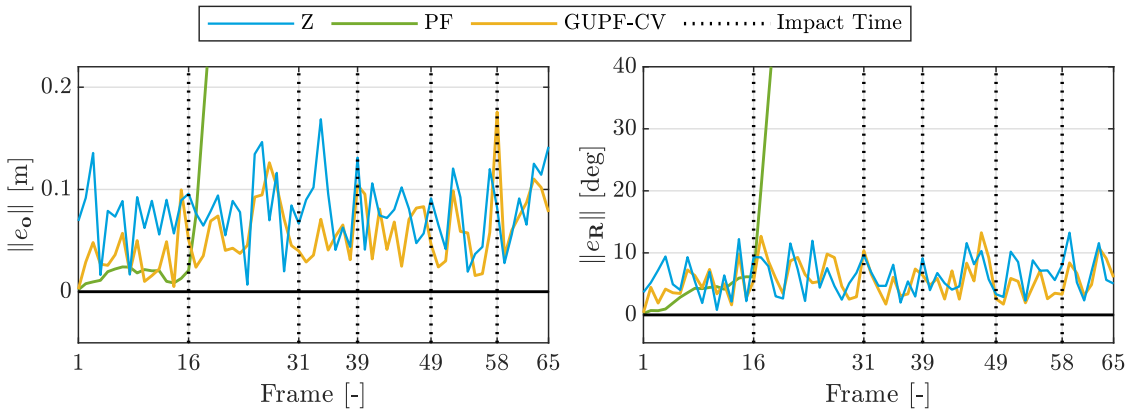


Figure 5.8: Norm of the position error (left) and the norm of the orientation error (right) as result of the different tracking algorithms.

Table 5.1: Resulting RMS values and error reduction of $\|e_{\mathbf{o}}\|$ and $\|e_{\mathbf{R}}\|$.

Method	RMS of $\|e_{\mathbf{o}}\|$ [m]	Reduction [%]	RMS of $\|e_{\mathbf{R}}\|$ [deg]	Reduction [%]
PF	0.86	-904	122.8	-1672
GUPF-CV	0.0646	24.6	6.2	9.8
Z	0.0856	-	6.9	-

shown, which indicate the percentage with which the RMS of the measurement is decreased by the different algorithms. Table 5.1 shows that the GUPF-CV is able to improve upon the noisy measurement, as the error in position and orientation is reduced with respect to the measurement with 24.6% and 9.8%, respectively. Even though the underlying motion model of the PF and the GUPF-CV is the same, the GUPF-CV is able to use the measurement to move the particles towards the area with higher likelihood, as discussed in Section 2.1. The increased performance of the GUPF-CV over the measurement can then be found in the fact that the GUPF-CV uses multiple particles to obtain an estimation of the pose, which is more accurate than the measurement.

5.3.2 Comparing the motion models: Results of the first scenario

We now consider the GUPF-CV as explained in the previous section, the GUPF-Z, and the GUPF-L, where the last two filters now make use of the motion model described in Section 4.3. We evaluate these algorithms for the first scenario, where the motion is parallel to the image plane. In Figure 5.9, the x, y, and z-components of ${}^{A}\mathbf{o}_{B}$ are shown as results of the different tracking algorithms, shown together with the ground truth. The time instants at which an impact occurs are indicated as well. Figure 5.10 shows the tracking results regarding the orientation of the cuboid. To visualize the orientation, we plotted ${}^{A}\mathbf{R}(t)_{B}v_j$ for $t \in [0, \mathrm{T}]$, where v_j, with $j = 1, 2, 3$, is given by

$$v_1 = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}^T, \quad v_2 = \begin{bmatrix} 0 & 1 & 0 \end{bmatrix}^T, \quad v_3 = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T. \tag{5.12}$$

In other words, given the orientation frame $[B]$, Figure 5.10 shows the trajectory of the tip of its unit vectors.

Considering Figure 5.9, one can see that the tracking performance of the GUPF-L and the GUPF-Z is better than the performance of the GUPF-CV. Especially in y-direction, their

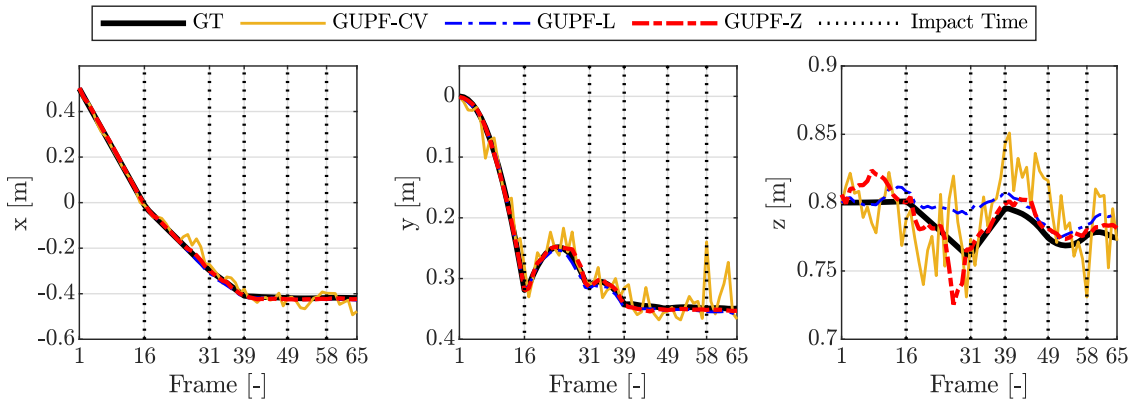


Figure 5.9: Results of the various tracking algorithms. The x-position (left), y-position (middle), and z-position (right) of the results are plotted together with the ground truth trajectory.

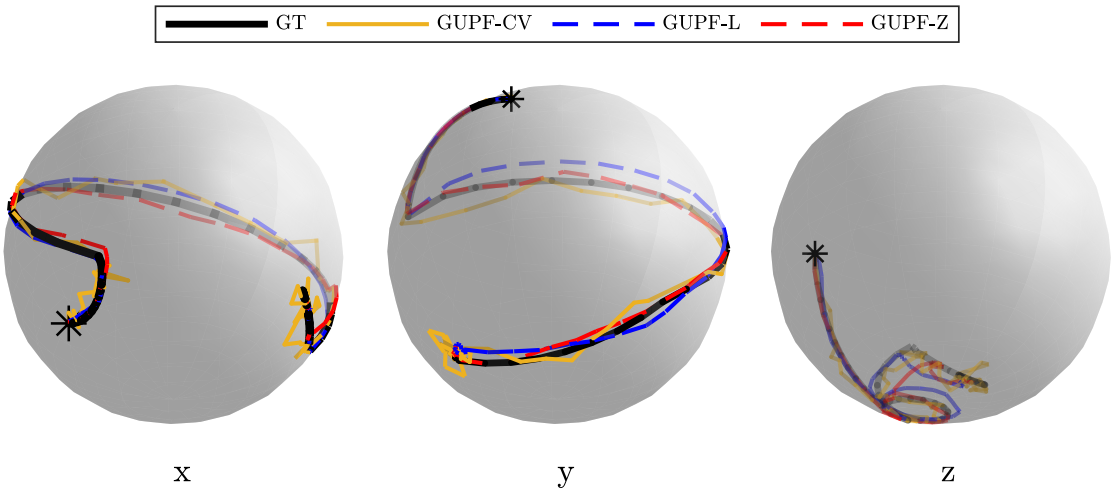


Figure 5.10: Numerical solutions of the orientation for the different algorithms. To visualize the orientation ${}^{A}\mathbf{R}(t)_B$, $t \in [0, T]$, we have plotted ${}^{A}\mathbf{R}(t)_B v_j$ on the unit sphere with $\{v_j\}_{j=1}^{3}$ given by (5.12). The figures correspond to v_1 (left), v_2 (middle), and v_3 (right). The asterisks indicate the starting point of the simulation.

resulting trajectory is much closer to the ground truth. Only in z-direction, all filters have trouble locating the cuboid accurately. A reason for this can be that keeping track of the distance of the object from the camera in z-direction is much more complicated compared to the x- and y-direction. This is because changes in z-direction of the cuboid's position only result in small differences in its appearance in the image plane. Furthermore, the impacts ensure the motion in z-direction is not smooth, as can be seen by the abrupt changes in the right image of Figure 5.9. This makes estimating the z-position of the cuboid more complicated. Although the impacts have a high influence on the orientation of the object, as can be seen by the quick changes of direction in Figure 5.10, all filters are able to closely track the orientation. Especially around x- and y-direction, the result of the GUPF-Z is close to the ground truth.

To see how much the position and orientation as outcome of the different algorithms

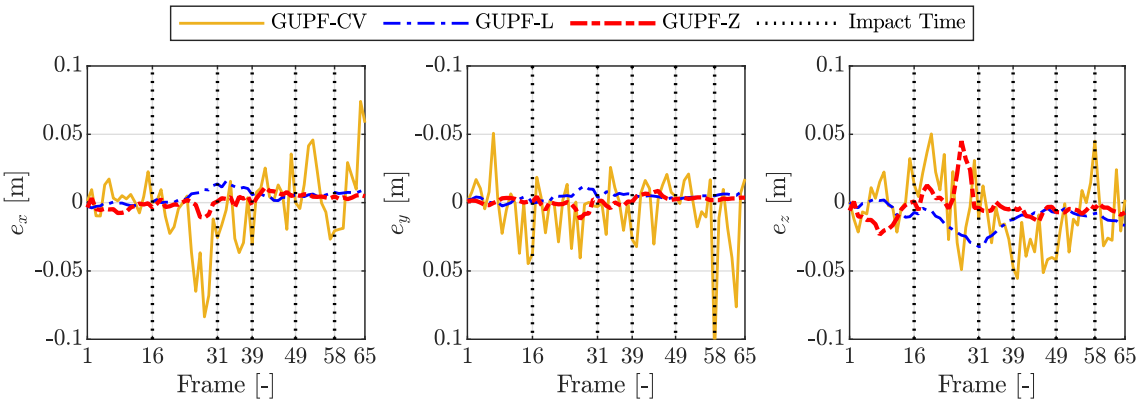


Figure 5.11: Position errors in x-direction (left), y-direction (middle), and z-direction (right).

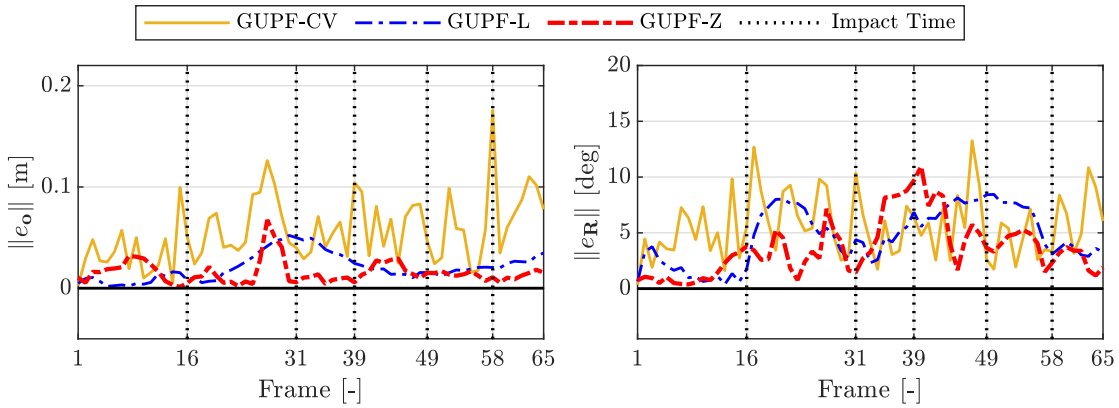


Figure 5.12: Norm of the position error (left) and the norm of the orientation error (right) as result of the different tracking algorithms.

differ from the ground truth, we computed the position and angular errors. In Figure 5.11, the position errors in x, y, and z-direction of the different algorithms are shown, computed in a similar way as in (5.6). The norm of the position error and the norm of the orientation error are then shown in Figure 5.12. From Figure 5.11 and 5.12, it becomes clear that the GUPF-L and GUPF-Z outperform the GUPF-CV in estimating the position of the cuboid. Especially in x- and y-direction, the error as result of the GUPF-L and the GUPF-Z is much smaller. Since the GUPF-L and the GUPF-Z take the impacts into account, no error-jumps are observed at the moments of impacts. However, as the right plot of Figure 5.11 shows, estimation of in the z-direction is still noisy, and the errors as result of the GUPF-L and GUPF-Z are not much smaller than that as result of the GUPF-CV. The same can be observed by considering the right plot of Figure 5.12, from which can be concluded that the the results of all algorithms are similar.

To see how much the different algorithms differ in performance, the RMS of the errors shown in Figure 5.12 are taken. This results in the values shown in Table 5.2. Since the GUPF-CV and the GUPF-Z make use of the measurement $\mathbf{Z}$, the RMS of the position and orientation errors of the measurement are shown as well. Besides, the error reduction is shown, which indicate the percentage with which the RMS of the measurement is decreased by the different algorithms. From Table 5.2, it becomes clear that that the GUPF-L and

Table 5.2: Resulting RMS values and error reduction for the first scenario.

Method	RMS of $\|e_{\mathbf{o}}\|$ [m]	Reduction [%]	RMS of $\|e_{\mathbf{R}}\|$ [deg]	Reduction [%]
GUPF-CV	0.065	24.6	6.2	9.8
GUPF-L	0.025	71.1	5.3	23.4
GUPF-Z	0.019	77.5	4.5	35.5
Z	0.086	-	6.9	-

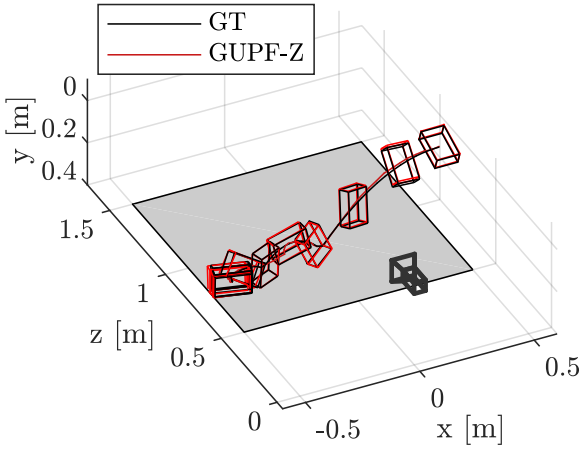


Figure 5.13: Trajectory as result of the UPF-Z shown together with the ground truth.

GUPF-Z show similar performance in estimating the position of the cuboid. For both tracking algorithms, the error is around three times smaller than that of the GUPF-CV. Regarding the orientation, the GUPF-L and and GUPF-Z also show better tracking results than the GUPF-CV, although the differences here are smaller. Concluding from Table 5.2, we say that the GUPF-Z shows the best result for this scenario. In order to visualize how much the result of the UPF-Z still differs from the ground truth, its resulting trajectory is shown in Figure 5.13, together with the ground truth.

This section described the results of the first scenario, where the motion of the cuboid is parallel to the image plane. In the next section, we will discuss the performance under a different scenario, where the cuboid is moving towards the camera.

5.3.3 Comparing the motion models: Results of the second scenario

Similar to the first scenario, the estimated values of ${}^{A}\mathbf{o}_{B}$ as result of the different algorithms are shown in Figure 5.14 and the results regarding the orientation of the cuboid are shown in Figure 5.15. Again, the GUPF-L and the GUPF-Z show a significant improvement regarding the tracking performance with respect to the GUPF-CV. In this scenario, also the results of the GUPF-L and the GUPF-Z in z-direction are clearly better than the results of the GUPF-CV. A reason for this could be, that the motion of the cuboid in z-direction is now more constant over time, and less affected by the impacts. Where in the first scenario the GUPF-L and the GUPF-Z had trouble estimating the z-position of the cuboid, this is not the case for the second scenario. Also regarding the orientation, the results of the GUPF-L and GUPF-Z stay close to the ground truth trajectory, as can be concluded from Figure 5.15.

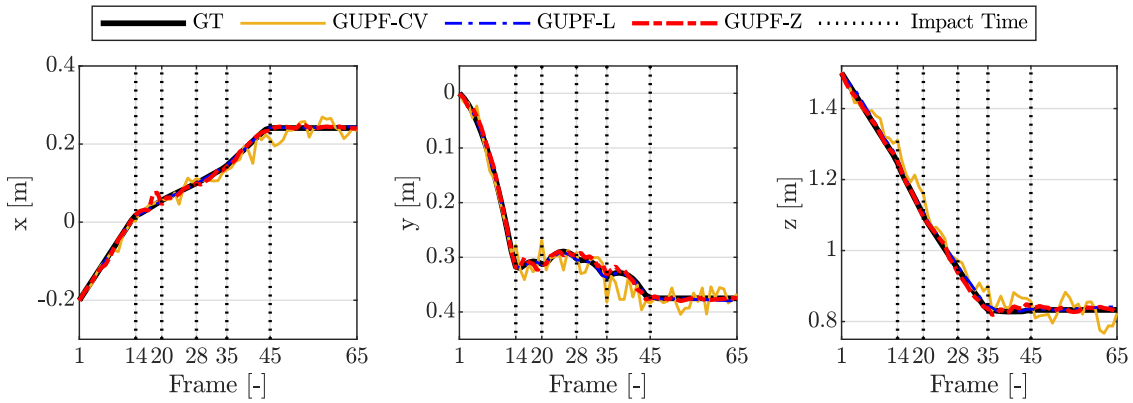


Figure 5.14: Results of the various tracking algorithms. The x-position (left), y-position (middle), and z-position (right) of the results are plotted together with the ground truth trajectory.

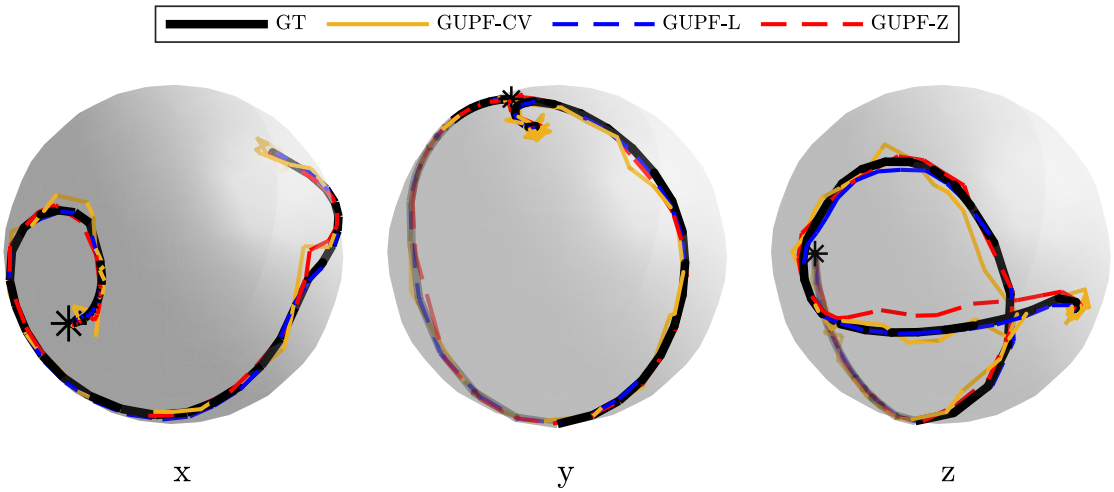


Figure 5.15: Numerical solutions of the orientation for the different algorithms. To visualize the orientations ${}^{A}\mathbf{R}(t)_{B}$, $t \in [0, T]$, we have plotted ${}^{A}\mathbf{R}(t)_{B} v_j$ on the unit sphere with $\{v_j\}_{j=1}^{3}$ given by (5.12). The figures correspond to v_1 (left), v_2 (middle), and v_3 (right). The asterisks indicate the starting point of the simulation.

To get a better view on how much the different algorithms perform, the error in position and orientation are again considered. Figure 5.16 shows the position error in x, y, and z-direction, while Figure 5.17 shows the norm of the position error and the norm of the orientation error in the left and right plot, respectively. From Figure 5.16, it is indeed clear that the GUPF-L and GUPF-Z now perform better regarding the estimation of the cuboid in z-direction, compared to the first scenario. Despite that, the errors are still higher in this direction compared to the x- and y-direction. From Figure 5.17, it becomes clear how well the GUPF-L is performing, as the norm of the position error stays close to zero.
Consider the right plot of Figure 5.17, which shows the norm of the orientation error. It is clear, that the GUPF-L also performs better than the GUPF-Z regarding the estimation of the orientation. This can also be concluded from Table 5.3, where the RMS values of the errors are shown. Considering these RMS values, the GUPF-Z seems to perform similar in to the first scenario. The GUPF-L, however, shows an increase in performance. A reason for

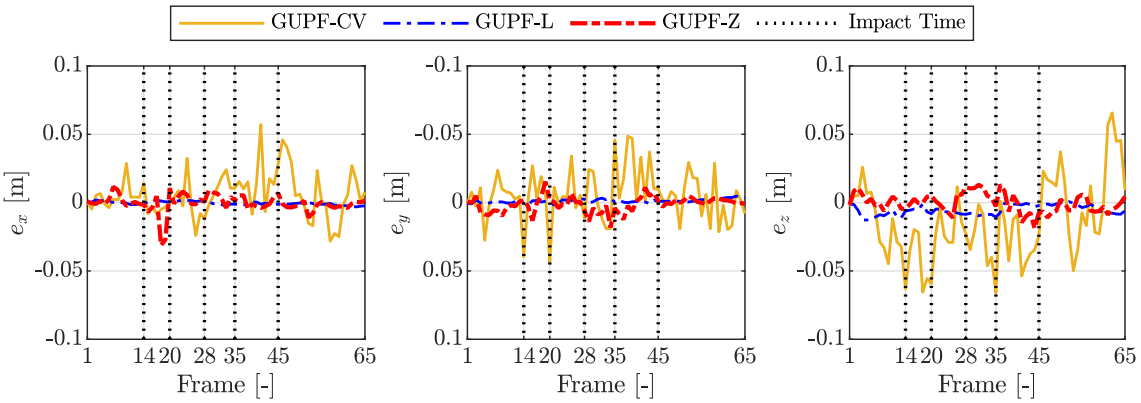


Figure 5.16: Position errors in x-direction (left), y-direction (middle), and z-direction (right).

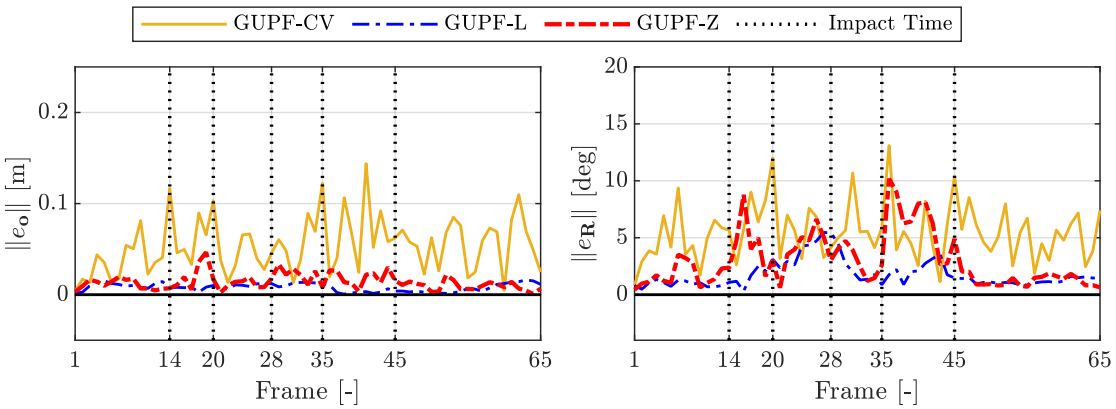


Figure 5.17: Tracking errors of the different algorithms for the second scenario.

this could be found in the way the measurement is obtained. Note that the GUPF-Z shows some high error peaks, for example, at frame 16 and 36, and not necessarily at the impact times. These peaks seem to overlap with the error peaks of the GUPF-CV, which also makes use of the measurement $\mathbf{Z}$. This could indicate that the performance of the GUPF-Z is in this case suffering from the noisy measurement. The weighted average of the particles, used as the measurement for the GUPF-L, is more accurate than the measurement $\mathbf{Z}$, used in the GUPF-Z. Where in the first scenario the particles of the GUPF-L were more spread, in the second scenario the particles stay closer to the true state of the cuboid, resulting in a better measurement for the GUPF-L. We can thus conclude that, as long as the particles stay close to the ground truth, the GUPF-L can outperform the GUPF-Z. To show how close the result of the GUPF-L is to the ground truth, its trajectory is plotted together with the ground truth in Figure 5.18.

5.4 Conclusion

In this chapter, the proposed GUPF was validated by the use of simulations. Two scenarios were considered where the cuboid is tossed on a surface and the tracking performance of the different algorithms was evaluated and compared to the works of [65] and [66]. First of all, we concluded that the GUPF showed better tracking results than the PF, due to

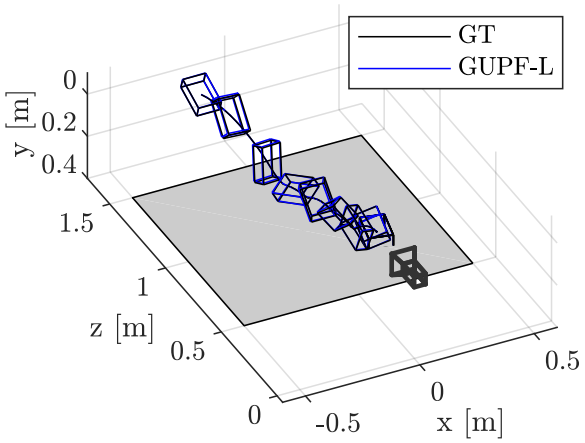


Figure 5.18: Trajectory of the resulting simulation of the GUPF-L for the second scenario.

Table 5.3: Resulting RMS values and error reduction for the second scenario.

Method	RMS of $\|e_{\mathbf{o}}\|$ [m]	Reduction [%]	RMS of $\|e_{\mathbf{R}}\|$ [deg]	Reduction [%]
GUPF-CV	0.062	19.1	6.0	9.1
GUPF-L	0.009	87.9	2.1	67.4
GUPF-Z	0.017	78.3	3.8	42.3
Z	0.076	-	6.6	-

the fact that the GUPF uses a measurement to move the particles to areas with higher likelihood. By comparing the GUPF-CV with the GUPF-L and the GUPF-Z, we saw that, for the first scenario, the GUPF-Z was outperforming the other algorithms in terms of both position and orientation error. In the second scenario, the GUPF-L was outperforming the GUPF-Z and the GUPF-CV, estimating the position and orientation of the cuboid close to the ground truth position in each frame. We argued that the measurement in the GUPF-L is more accurate than the measurement $\mathbf{Z}$, resulting in a higher tracking performance, but only when the particles stay close to the true state of the object. In the next chapter, we will further elaborate on the conclusions that can be drawn from this research and several recommendations are proposed for further research.

Chapter 6

Conclusions and Recommendations

As stated in Section 1.4, the objective of this thesis is to develop an algorithm able to track objects, in particular cuboids, that after being tossed, can collide with surfaces. The approach taken to achieve this goal included computer vision tracking techniques, including object detection based on histogram comparison and object tracking using the proposed *Geometric Unscented Particle Filter*. This is done in combination with a motion model describing the dynamics of the cuboid both during free flight as well as across the impacts. Part of the tracking algorithm is the use of a measurement, giving an indication of the cuboid's pose in 3D. Two different approaches were taken to obtain this measurement: one method based on a likelihood estimation (GUPF-L) and a second approach, where the pose is assumed to be measured by other computer vision sensors, with a Gaussian zero-mean estimation error (GUPF-Z). In Chapter 5, both these methods were compared in simulation to state-of-the-art tracking methods as the standard particle filter (PF). In this chapter, conclusions are drawn based upon the results of Chapter 5 and recommendations are given for further research on this topic.

6.1 Conclusions

This thesis is divided into two main parts: *detection*, where the 3D pose of the cuboid is estimated by the use of image data, and *tracking*, where the goal is to maintain an estimate of the cuboid's state along time. In the following sections, we will reflect on these two topics.

6.1.1 Detection of the cuboid

Formally, the research objective related to object detection is stated as:

Estimate the state, i.e., the pose and velocity, of the cuboid in 3D space, based on images from a RGB camera.

To achieve this research objective, multiple approaches are considered. In Chapter 1, a distinction is made between feature-based and model-based detection methods, where, in the former, localization of the object is based on 2D information only, and in the latter a 3D model is used for object detection in the image. Although feature-based methods are found to be computationally more efficient than model-based methods, they often allow for localization of the object in 2D only. In other words, the object is detected in the image without knowing its corresponding 3D pose. As estimating the 3D pose of the object is one of the objectives of this thesis, a model based approach is taken, where a geometric model of

the cuboid is created. Hypothesized states of the cuboid are then evaluated in each image. By the use of perspective projection, the relation between the pose of the cuboid in 3D and its representation in the image plane can be found. Considering the various challenges appearing in object detection, as discussed in Section 1.2.1.4, a detection method is created, similar to the work presented in [28]. In this detection method, several color histograms are computed for each hypothesis by evaluating the representation of a given pose in the image plane. These color histograms are then used to compute a likelihood, which should be high when the hypothesis is close to the true state of the cuboid. In Appendix A, the behavior of the likelihood function is shown, from which is concluded that the cuboid can be detected in an image up to 1mm accuracy. However, detecting the distance of the cuboid from the camera turned out to be very hard, especially when the object was far away from the camera. The same conclusion is drawn from the numerical evaluating of Chapter 5, where is shown that the errors related to estimating the z-position are relatively high. As the size of the object in the image plane is directly related to the distance from the camera, detecting the object is harder when it is further away from the camera, and thereby represented by less pixels in the image. In that sense, there is always a trade off between distance and image resolution. Regarding the orientation of the cuboid, the results in Appendix A show that the likelihood function is able to correctly estimate the orientation up to 3 degrees accuracy, and the same results can be observed from the simulation results in Chapter 5. Considering the velocity of the cuboid, we are able to estimate the linear and angular velocity of the cuboid by the use of the particle filter, which we discuss in the next section. We can thus conclude that the research objective is reached and we are able to estimate the position within 1mm accuracy and the orientation within 3 degrees accuracy.

6.1.2 Tracking of the cuboid

Formally, the research objective related to object tracking is stated as:

Develop a tracking algorithm that, based on the estimate of the previous state, gives a prediction of the current state of the cuboid.

Several filtering techniques are considered to achieve this research objective, like the Kalman filter and the extended Kalman filter. These filters assume the state variables to be Gaussian, and therefore give poor estimations for non-Gaussian state variables. Considering the nonsmooth mechanics introduced by impacts and friction, the choice is made to use an unscented particle filter as filtering technique. The advantage of the particle filter is that, at each time instant, it maintains several hypotheses of the state. These hypotheses are evaluated in the image to estimate the state, where the weighted mean gives an estimation of the pose and the velocity of the cuboid. However, often a large number of particles is needed to obtain good tracking results, leading to high computation time. This especially happens when the motion of the particles is poorly defined and, as discussed in [55], tracking performance could be improved if the motion model would be closer to the true dynamics of the object. Therefore, in this research the goal is to fill this gap in the literature by providing a motion model able to describe the cuboid's motion during free flight as well as across the impacts. In Chapter 4, this motion model is discussed, where the theory of nonsmooth mechanics is used to describe the effects of impacts and friction on the cuboid's motion. To combine this motion model with the unscented particle filter, this filter is generalized to fit with the Lie group structure of state of the cuboid. This generalization is discussed in Chapter 3, leading to the proposed *Geometric Unscented Particle Filter*.

The performance of the GUPF is tested in simulation and compared to state-of-the-art methods. The advantage of the GUPF as tracking algorithm, is that uses the current observation to move the particles towards the area with higher likelihood. Therefore, the GUPF shows increased performance results with respect to standard particle filters. Furthermore, modeling the impacts and friction shows great improvements in the tracking results. State-of-the-art methods use a constant velocity model, giving no preference to any direction of motion. As a result, particles are constantly spread around the last estimated position, resulting in a relatively poor tracking performance. By incorporating a motion model giving a more restricted direction of motion of the particles, the particles stay closer to the true position of the cuboid resulting in higher tracking performance. The simulation results of the GUPF-Z and the GUPF-L show great improvements over state-of-the-art methods, as can be concluded from Table 5.2 and Table 5.3. We can thus conclude that the research objective is reached and we are able to accurately estimate the current state of the cuboid by the use of the GUPF.

6.2 Recommendations

The research objectives haven been reached, as concluded in the previous section. However, during the process of this research, several thoughts have come to mind for possible improvements of the tracking algorithm. In the following paragraphs, we discuss some of the possibilities to extend this research.

6.2.1 Object detection

In the likelihood function, we use a fixed distance between the points to compute the histograms of the inside of the contour $\boldsymbol{H}_{cin}$ and the outside of the contour $\boldsymbol{H}_{cout}$, embedded in the geometric model of the cuboid. In [103], however, this distance is changed as function of the object size in the image resulting in accurate detection of the object. As the results of Appendix A show that estimating the distance of the object to the camera is hard, the method of Zhong and Zhang in [103] could potentially improve this estimation. Furthermore, the number of pixel points used to compute the histograms could be changed according to the object size in the image. This is for example done in [41]. In the simulations done in Appendix A, we noticed that multiple points of the cuboid's model may end up at the same pixel location in the image when the object is far away from the camera. This could lead to a histogram which is not representative for the cuboid's appearance. Changing the number of points to compute the histograms as function of the distance could prevent this. Finally, in [144], Perez et al. use the squared distance D for the likelihood, resulting in a more 'peaky' distribution, leading to a better distinction between different particles. These different approaches to the likelihood function could potentially improve the detection of the object in the image plane.

6.2.2 Object tracking

In Chapter 3, we assumed a given measurement $\mathbf{Z}$. This measurement was obtained by adding Gaussian noise to the ground truth. A measuring technique should be developed, as for example is done in [35], for implementation of the GUPF-Z. This would improve the simulation and could give a better idea of how the algorithm would work in real-life scenarios.

Furthermore, the simulations executed in Chapter 5 show various conversion rates of the fixed-point iteration as discussed in Section 4.3.6. In [118], the Lemke algorithm is proposed to solve the NCP due to its robustness and fast conversion rate. Using this pivoting method to solve the NCP in Section 4.3.6, could be an improvement of the algorithm. However, in [137] is stated that the Lemke algorithm should be avoided, as the calculation time would grow exponentially with the number of contact points. As a result, further investigation is needed to find an optimal NCP solver. Besides, existing software, like Algoryx [145], could be used for more complex simulations with more complex shaped objects.

Finally, the simulations of Chapter 5 were executed by using a set of 50 particles in each simulation. In general, this is very low number of particles. In [28] for example, experiments where executed where a set of 10000 particles were used. By the law of large numbers, increasing the set of particles would always give a better tracking result, as with a growing set of particles the estimation of the posterior distribution will improve, but this also leads to higher computation time. In this work, the computation time is neglected, as implementation into real-time scenarios is not considered. However, implementation of the algorithm into `C` or `C++` would be necessary to speed up the computation time.

6.2.3 Validation by simulations and experiments

Several points can be addressed to improve the validation of Chapter 5. In Chapter 1, the choice for color histograms to detect the object was made based on its robustness for illumination changes and motion blur. The simulations of Chapter 5, however, do not show these phenomena. Tracking is performed by following a cuboid of which the faces have very distinctive colors and a background is used with a uniform color, all of which may not be representatives for real-life scenarios, making the tracking less challenging. Furthermore, adding noise to the pose of the cuboid, the camera parameters, and the geometric model of the cuboid, could give a better representation of a real-life scenario. By adding these noises and introducing motion blur, the tracking algorithm could be tested in more challenging scenarios. Furthermore, the assumption is made that the object is rigid and does not deform upon the impacts, while in real-life scenarios, these deformations may occur. Therefore, the effect of deformations should be taken into account as well. Real-life experiments should be executed to see how the algorithm performs under these circumstances and the motion model and likelihood function should be adapted according to the results of these experiments. Finally, an open database could be created where the ground truth scenarios are stored, for example by the use of OptiTrack [146]. This allows other researchers to reproduce the tracking results or use the scenarios to develop other tracking algorithms.

Bibliography

[1] A. Iriondo, E. Lazkano, L. Susperregi, J. Urain, A. Fernandez, and J. Molina, "Pick and Place Operations in Logistics Using a Mobile Manipulator Controlled with Deep Reinforcement Learning," *Applied Sciences*, vol. 9, Jan. 2019.

[2] SmartRobotics, "Smart robotics in het technisch weekblad." https://www.smart-robotics.nl/smart-robotics-in-het-technisch-weekblad/. Accessed: 12-02-2020.

[3] KUKA, "Kuka case study: Human-robot-collaboration at siemens." https://www.kuka.com/en-at/industries/solutions-database/2016/07/solution-robotics-siemens. Accessed: 12-02-2020.

[4] Vanderlande, "Smart item robotics." https://www.vanderlande.com/warehousing/innovative-systems/picking/smart-item-robotics/. Accessed: 12-02-2020.

[5] A. Zeng, S. Song, J. Lee, A. Rodriguez, and T. Funkhouser, "TossingBot: Learning to Throw Arbitrary Objects with Residual Physics," in *Proceedings of Robotics: Science and Systems (RSS)*, 2019.

[6] K. Smith, D. Gatica-Perez, J.-M. Odobez, and S. Ba, "Evaluating Multi-Object Tracking," in *Proceedings IEEE Conference on Computer Vision and Pattern Recognition*, Jul. 2005.

[7] A. Yilmaz, O. Javed, and M. Shah, "Object Tracking: A Survey," *ACM Computer Surveys*, vol. 38, Dec. 2006.

[8] P. Lanvin, J. Noyer, and M. Benjelloun, "An hardware architecture for 3D object tracking and motion estimation," in *2005 IEEE International Conference on Multimedia and Expo*, Jul. 2005.

[9] L. Masson, M. Dhome, and F. Jurie, "Robust real time tracking of 3D objects," in *Proceedings of the 17th International Conference on Pattern Recognition*, pp. 252–255, Sep. 2004.

[10] E. Marchand, P. Bouthemy, F. Chaumette, and V. Moreau, "Robust real-time visual tracking using a 2D-3D model-based approach," in *Proceedings of the Seventh IEEE International Conference on Computer Vision*, vol. 1, pp. 262–268, Sep. 1999.

[11] V. Lepetit and P. Fua, "Monocular Model-based 3D Tracking of Rigid Objects," *Foundations and Trends in Computer Graphics and Vision*, vol. 1, pp. 1–89, Jan. 2005.

[12] K. X. Yang and M. H. Sheu, "Edge-based moving object tracking algorithm for an embedded system," in *2016 IEEE Asia Pacific Conference on Circuits and Systems (APCCAS)*, pp. 153–155, Oct. 2016.

[13] Q. Xiao, X. Liu, and M. Liu, "Object Tracking Based on Local Feature Matching," in *2012 Fifth International Symposium on Computational Intelligence and Design*, vol. 1, pp. 399–402, Oct. 2012.

[14] Y. Yang, X. Wang, J. Wu, H. Chen, and Z. Han, "An improved mean shift object tracking algorithm based on ORB feature matching," in *The 27th Chinese Control and Decision Conference (2015 CCDC)*, pp. 4996–4999, May 2015.

[15] L. Zhang and R. Wu, "Tracking Object Using Object-strips Color Feature," in *2010 WASE International Conference on Information Engineering*, vol. 1, pp. 212–215, Aug. 2010.

[16] M. Vincze, M. Ayromlou, and W. Kubinger, "An Integrating Framework for Robust Real-Time 3D Object Tracking," in *Computer Vision Systems*, (Berlin, Heidelberg), pp. 135–150, Springer Berlin Heidelberg, 1999.

[17] M. Sarfraz and A. Taimur, "Real Time Object Detection and Motion," in *2009 Second International Conference in Visualisation*, pp. 235–240, Jul. 2009.

[18] P. Tian, "A particle filter object tracking based on feature and location fusion," in *2015 6th IEEE International Conference on Software Engineering and Service Science (ICSESS)*, pp. 762–765, Sep. 2015.

[19] Y. Shirai, Y. Mae, and S. Yamamoto, "Object Tracking by Using Optical Flows and Edges," in *Robotics Research* (G. Giralt and G. Hirzinger, eds.), (London), pp. 440–447, Springer London, 1996.

[20] T. F. Syeda-Mahmood, "Data and Model-Driven Selection Using Color Regions," in *International Journal of Computer Vision*, vol. 21, pp. 9–36, Jan. 1997.

[21] J. Barandiaran and D. Borro, "Edge-Based Markerless 3D Tracking of Rigid Objects," in *17th International Conference on Artificial Reality and Telexistence (ICAT 2007)*, pp. 282–283, Nov. 2007.

[22] P. Wunsch and G. Hirzinger, "Real-time visual tracking of 3D objects with dynamic handling of occlusion," in *Proceedings of International Conference on Robotics and Automation*, vol. 4, pp. 2868–2873, Apr. 1997.

[23] T. Peter, "Effizientes Eliminieren verdeckter Kanten in polyedrischen Szenen zur Simulation von Kamerabildern," tech. rep., Institute of Robotics and System Dynamics, German Aerospace Center (DLR), 1996.

[24] D. G. Lowe, "Robust model-based motion tracking through the integration of search and estimation," *International Journal of Computer Vision*, vol. 8, pp. 113–122, Aug. 1992.

[25] C. Harris and C. Stennett, "RAPID - A video rate object tracker," in *BMVC*, 1990.

[26] A. D. Worrall, G. D. Sullivan, and K. D. Baker, "Pose refinement of active models using forces in 3D," in *European Conference on Computer Vision - ECCV '94*, pp. 341–350, Springer Berlin Heidelberg, 1994.

[27] M. Taiana, J. Santos, J. Gaspar, J. Nascimento, A. Bernardino, and P. Lima, "Color 3D model-based tracking with arbitrary projection models," in *International Conference on Simulation, Modeling and Programming for Autonomous Robots*, 2008.

[28] M. Taiana, J. Santos, J. Gaspar, J. Nascimento, A. Bernardino, and P. Lima, "Tracking Objects with Generic Calibrated Sensors: An Algorithm Based on Color and 3D Shape Features," *Robot. Auton. Syst.*, vol. 58, pp. 784–795, Jun. 2010.

[29] L. Min, Y. Yang, Y. Liu, and Y. Leng, "Robust 3D human tracking based on Kinect," in *2015 34th Chinese Control Conference (CCC)*, pp. 4615–4619, Jul. 2015.

[30] T. Nakamura, "Real-time 3-D object tracking using Kinect sensor," in *2011 IEEE International Conference on Robotics and Biomimetics*, pp. 784–788, Dec. 2011.

[31] Y. Liu, "Tracking objects in 3D space with back-projected correlation filter," in *2018 33rd Youth Academic Annual Conference of Chinese Association of Automation (YAC)*, pp. 475–478, May 2018.

[32] L. Guo, G. Fan, and W. Sheng, "Robust object detection by cuboid matching with local plane optimization in indoor RGB-D images," in *2017 IEEE Visual Communications and Image Processing (VCIP)*, pp. 1–4, Dec. 2017.

[33] C. Choi and H. I. Christensen, "RGB-D object tracking: A particle filter approach on GPU," in *International Conference on Intelligent Robots and Systems*, pp. 1084–1091, Nov. 2013.

[34] G. Ning, Z. Zhang, C. Huang, X. Ren, H. Wang, C. Cai, and Z. He, "Spatially supervised recurrent convolutional neural networks for visual object tracking," in *IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1–4, May 2017.

[35] D. Dwibedi, T. Malisiewicz, V. Badrinarayanan, and A. Rabinovich, "Deep Cuboid Detection: Beyond 2D Bounding Boxes," *CoRR*, Nov. 2016.

[36] T. Drummond and R. Cipolla, "Real-time visual tracking of complex structures," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, pp. 932–946, Jul. 2002.

[37] C. Choi and H. I. Christensen, "Robust 3D visual tracking using particle filtering on the special Euclidean group: A combined approach of keypoint and edge features," *The International Journal of Robotics Research*, vol. 31, no. 4, pp. 498–519, 2012.

[38] G. Klein and D. W. Murray, "Full-3D Edge Tracking with a Particle Filter," in *Proceedings of the British Machine Vision Conference 2006*, pp. 1119–1128, Jan. 2006.

[39] M. Armstrong and A. Zisserman, "Robust Object Tracking," in *Asian Conference on Computer Vision*, vol. 1, pp. 58–61, 1995.

[40] E. Falotico, M. Taiana, D. Zambrano, A. Bernardino, J. Santos-Victor, P. Dario, and C. Laschi, "Predictive tracking across occlusions in the iCub robot," in *9th IEEE-RAS International Conference on Humanoid Robots*, pp. 486–491, Dec 2009.

[41] M. Taiana, J. Nascimento, J. Gaspar, and A. Bernardino, "Sample-Based 3D Tracking of Colored Objects: A Flexible Architecture," in *British Machine Vision Conference (BMVC)*, Jan. 2008.

[42] T. Shibata, H. Tabata, S. Schaal, and M. Kawato, "A model of smooth pursuit in primates based on learning the target dynamics," *Neural Networks: the official journal of the International Neural Network Society*, vol. 18, pp. 213 – 224, 2005.

[43] M. J. Swain and D. H. Ballard, "Color indexing," in *International Journal of Computer Vision*, vol. 7, pp. 11–32, Nov. 1991.

[44] M. A. Fischler and R. C. Bolles, "Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography," *Communications of the Association for Computing Machinery (ACM)*, vol. 24, pp. 381–395, jun. 1981.

[45] K. Pauwels and D. Kragic, "SimTrack: A simulation-based framework for scalable real-time object pose detection and tracking," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1300–1307, Sep. 2015.

[46] P. Perez, J. Vermaak, and A. Blake, "Data fusion for visual tracking with particles," in *Proceedings of the IEEE*, vol. 92, pp. 495–513, Mar. 2004.

[47] K. Okuma, A. Taleghani, N. de Freitas, J. J. Little, and D. G. Lowe, "A Boosted Particle Filter: Multitarget Detection and Tracking," in *Computer Vision - ECCV 2004*, (Berlin, Heidelberg), pp. 28–39, Springer Berlin Heidelberg, 2004.

[48] G. Healey and D. Slater, "Using illumination invariant color histogram descriptors for recognition," in *Proceedings IEEE Conference on Computer Vision and Pattern Recognition*, pp. 355 – 360, Jun. 1994.

[49] Stern H and B. Efros, "Adaptive color space switching for face tracking in multi-colored lighting environments," in *Proceedings of Fifth IEEE International Conference on Automatic Face Gesture Recognition*, pp. 249–254, May 2002.

[50] Y. Park, V. Lepetit, and W. Woo, "Handling Motion-Blur in 3D Tracking and Rendering for Augmented Reality," *IEEE Transactions on Visualization and Computer Graphics*, vol. 18, pp. 1449–1459, Sep. 2012.

[51] S. Olufs, F. Adolf, R. Hartanto, and P. Plöger, "Towards Probabilistic Shape Vision in RoboCup: A Practical Approach," in *RoboCup 2006: Robot Soccer World Cup X* (G. Lakemeyer, E. Sklar, D. G. Sorrenti, and T. Takahashi, eds.), (Berlin, Heidelberg), pp. 171–182, Springer Berlin Heidelberg, 2007.

[52] E. R. AlBasiouny, A. Sarhan, and T. Medhat, "Mean-shift-FAST Algorithm to Handle Motion-Blur with Tracking Fiducial Markers," in *2015 Tenth International Conference on Computer Engineering Systems (ICCES)*, pp. 286–292, Dec. 2015.

[53] W. Wilson, "Visual Servo Control of Robots Using Kalman Filter Estimates of Relative Pose," in *IFAC Proceedings Volumes*, vol. 26, pp. 633 – 638, 1993. 12th Triennal Wold Congress of the International Federation of Automatic control. Volume 3 Applications I, Sydney, Australia, 18-23 July.

[54] T. Hervier, S. Bonnabel, and F. Goulette, "Accurate 3D maps from depth images and motion sensors via nonlinear Kalman filtering," *IEEE International Conference on Intelligent Robots and Systems*, May 2012.

[55] G. Taylor and L. Kleeman, "Fusion of Multimodal Visual Cues for Model-Based Object Tracking," in *Australasian Conference on Robotics and Automation (ACRA)*, 2003.

[56] W. J. Wilson, C. C. Williams Hulls, and G. S. Bell, "Relative end-effector control using Cartesian position based visual servoing," *IEEE Transactions on Robotics and Automation*, pp. 684–696, Oct. 1996.

[57] X. Deng, A. Mousavian, Y. Xiang, F. Xia, T. Bretl, and D. Fox, "PoseRBPF: A Rao-Blackwellized Particle Filter for 6D Object Pose Tracking," in *Robotics Science and Systems*, May 2019.

[58] P. Barrera, J. Cañas Plaza, and V. Matellán, "Visual Object Tracking in 3D with Color Based Particle Filter," in *World Enformatika Conference (WEC)*, pp. 200–203, Jan. 2005.

[59] D. Kragic and H. I. Christensen, "Model based techniques for robotic servoing and grasping," in *International Conference on Intelligent Robots and Systems*, vol. 1, pp. 299–304, Sep. 2002.

[60] T. Shibata and S. Schaal, "Biomimetic smooth pursuit based on fast learning of the target dynamics," in *Proceedings IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 1, pp. 278–285, Oct. 2001.

[61] R. Van Der Merwe, A. Doucet, N. De Freitas, and E. Wan, "The Unscented Particle Filter," in *Proceedings of the 13th International Conference on Neural Information Processing Systems*, NIPS '00, (Cambridge, USA), pp. 563–569, Aug. 2000.

[62] D. Ding, Z. Jiang, and C. Liu, "Object tracking algorithm based on particle filter with color and texture feature," in *35th Chinese Control Conference (CCC)*, pp. 4031–4036, Jul. 2016.

[63] J. Noyer, P. Lanvin, and M. Benjelloun, "Model-based tracking of 3D objects based on a sequential Monte-Carlo method," in *Conference Record of the Thirty-Eighth Asilomar Conference on Signals, Systems and Computers, 2004.*, vol. 2, pp. 1744–1748, Nov. 2004.

[64] Y. Salih and A. S. Malik, "3D Tracking using particle filters," in *IEEE International Instrumentation and Measurement Technology Conference*, pp. 1–4, May 2011.

[65] J. Kwon and F. Park, "Visual Tracking via Particle Filtering on the Affine Group," *International Journal of Robotic Research - IJRR*, vol. 29, pp. 997 – 1002, Jul. 2008.

[66] J. Kwon and K. Lee, "Monocular SLAM with Locally Planar Landmarks via Geometric Rao-Blackwellized Particle Filtering on Lie Groups," in *Proceedings IEEE Conference on Computer Vision and Pattern Recognition*, vol. 1, pp. 1522–1529, Jun. 2010.

[67] A. Chiuso and S. Soatto, "Monte Carlo filtering on Lie groups," in *Proceedings of the IEEE Conference on Decision and Control*, vol. 1, pp. 304 – 309, Feb. 2000.

[68] J. Kwon, M. Choi, F. C. Park, and C. Chun, "Particle filtering on the Euclidean group: framework and applications," *Robotica*, vol. 25, p. 725–737, Jun. 2007.

[69] G. Loianno, M. Watterson, and V. Kumar, "Visual Inertial Odometry for Quadrotors on SE(3)," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1544–1551, May 2016.

[70] M. Brossard, S. Bonnabel, and J. Condomines, "Unscented Kalman filtering on Lie groups," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2485–2491, Sep. 2017.

[71] M. S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, "A Tutorial on Particle Filters for Online Nonlinear/Non-Gaussian Bayesian Tracking," *IEEE Transactions on Signal Processing*, vol. 50, pp. 174–188, Feb. 2002.

[72] A. Doucet and A. Johansen, "A Tutorial on Particle Filtering and Smoothing: Fifteen years later," *Handbook of Nonlinear Filtering*, vol. 12, Jan. 2009.

[73] N. Bergman, *Recursive Bayesian Estimation : Navigation and Tracking Applications*. PhD thesis, Linköping University, Linköping, 1999.

[74] A. Doucet, S. Godsill, and C. Andrieu, "On sequential Monte Carlo sampling methods for Bayesian filtering," *Statistics and Computing*, vol. 10, pp. 197–208, Jul. 2000.

[75] A. Doucet, N. De Freitas, and N. Gordon, *Sequential Monte Carlo Methods in Practice*. Springer New York, 1st. ed., 2001.

[76] A. Doucet and N. J. Gordon, "Simulation-based optimal filter for maneuvering target tracking," in *SPIE Signal and Data Processing of Small Targets*, pp. 241–255, Oct. 1999.

[77] J. D. Hol, T. B. Schon, and F. Gustafsson, "On Resampling Algorithms for Particle Filters," in *2006 IEEE Nonlinear Statistical Signal Processing Workshop*, pp. 79–82, Sep. 2006.

[78] R. Douc, "Comparison of Resampling Schemes for Particle Filtering," in *4th International Symposium on Image and Signal Processing and Analysis (ISPA)*, pp. 64–69, 2005.

[79] J.D.Hol, "Resampling in particle filters," tech. rep., Linköpings universitet, Department of Electrical Engineering, May 2004.

[80] M. Wüthrich, J. Bohg, D. Kappler, C. Pfreundt, and S. Schaal, "The Coordinate Particle Filter - A novel Particle Filter for High Dimensional Systems," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2454–2461, May 2015.

[81] D. Crisan and A. Doucet, "A Survey of Convergence Results on Particle Filtering Methods for Practitioners," *IEEE Transactions on Signal Processing*, vol. 50, pp. 736–746, Mar. 2002.

[82] S. J. Julier and J. K. Uhlmann, "A New Extension of the Kalman Filter to Nonlinear Systems," in *Signal Processing, Sensor Fusion, and Target Recognition VI*, vol. 3068, pp. 182 – 193, SPIE, Jul. 1997.

[83] E. Wan and R. van der Merwe, "The Unscented Kalman Filter for Nonlinear Estimation," in *Symposium on Adaptive Systems for Signal Processing, Communication and Control*, pp. 153–158, IEEE, 2000.

[84] Yuanxin Wu, Dewen Hu, Meiping Wu, and Xiaoping Hu, "Unscented Kalman filtering for additive noise case: augmented versus nonaugmented," *IEEE Signal Processing Letters*, vol. 12, pp. 357–360, May 2005.

[85] N. Greggio, J. Gaspar, A. Bernardino, and J. Santos-Victor, "Monocular vs binocular 3D real-time ball tracking from 2D ellipses," in *ICINCO 2011 - Proceedings of the 8th International Conference on Informatics in Control, Automation and Robotics*, vol. 2, pp. 67–73, Jul. 2011.

[86] Y. Ma, S. Soatto, J. Kosecka, and S. Sastry, *An Invitation to 3-D Vision - From Images to Geometric Models*, vol. 26. Springer-Verlag New York, 2004.

[87] V. Varadarajan, *Lie Groups, Lie Algebras, and Their Representations*, vol. 102. Spinger-Verlag New York, 1984.

[88] S. Traversaro and A. Saccon, *Multibody dynamics notation (version 2).* Technische Universiteit Eindhoven, Nov. 2019. DC2019.100.

[89] R. M. Murray, S. S. Sastry, and L. Zexiang, *A Mathematical Introduction to Robotic Manipulation.* CRC Press, Inc., 1994.

[90] Y. Wang and G. S. Chirikjian, "Nonparametric Second-order Theory of Error Propagation on Motion Groups," *The International Journal of Robotics Research*, vol. 27, no. 11-12, pp. 1258–1273, 2008.

[91] A. Mueller, "Coordinate Mappings for Rigid Body Motions," *Journal of Computational and Nonlinear Dynamics*, vol. 12, Sep. 2016.

[92] M. Zeestraten, *Programming by Demonstration on Riemannian Manifolds.* PhD thesis, University of Genova, 2017.

[93] T. D. Barfoot and P. T. Furgale, "Associating Uncertainty With Three-Dimensional Poses for Use in Estimation Problems," *IEEE Transactions on Robotics*, vol. 30, pp. 679–693, Jun. 2014.

[94] G. S. Chirikjian, *Stochastic Models, Information Theory, and Lie Groups*, vol. 2. Birkhäuser New York, NY, USA, 2012.

[95] G. Dubbelman, *Intrinsic statistical techniques for robust pose estimation.* PhD thesis, University of Amsterdam, Sep. 2014.

[96] M. Zefran, V. Kumar, and C. Croke, "Metrics and Connections for Rigid-Body Kinematics," *The International Journal of Robotics Research*, vol. 18, Feb. 1999.

[97] R. Featherstone, *Rigid Body Dynamics Algorithms.* Springer US, 1st. ed., 2008.

[98] R. Brégier, F. Devernay, L. Leyrit, and J. L. Crowley, "Defining the Pose of Any 3D Rigid Object and an Associated Distance," *International Journal of Computer Vision*, pp. 571–596, Jun. 2018.

[99] F. C. Park, "Distance Metrics on the Rigid-Body Motions with Applications to Mechanism Design," *ASME Journal of Mechanism Design*, vol. 117, pp. 48–54, Jan. 1995.

[100] K. M. Lynch and F. C. Park, *Modern Robotics: Mechanics, Planning, and Control.* New York, NY, USA: Cambridge University Press, 1st ed., 2017.

[101] M. Spong, S. Hutchinson, and M. Vidyasagar, *Robot Modeling and Control.* Wiley, 2005.

[102] S. Hauberg, F. Lauze, and K. S. Pedersen, "Unscented Kalman Filtering on Riemannian Manifolds," *Journal of Mathematical Imaging and Vision*, vol. 46, pp. 103–120, May 2013.

[103] L. Zhong and L. Zhang, "A Robust Monocular 3D Object Tracking Method Combining Statistical and Photometric Constraints," *International Journal of Computer Vision*, vol. 127, no. 8, pp. 973–992, 2019.

[104] V. A. Prisacariu and I. D. Reid, "PWP3D: Real-Time Segmentation and Tracking of 3D Objects," *International Journal of Computer Vision*, vol. 98, pp. 335–354, 2012.

[105] P. Azad, T. Asfour, and R. Dillmann, "Combining Appearance-based and Model-based Methods for Real-Time Object Recognition and 6D Localization," in *International Conference on Intelligent Robots and Systems*, pp. 5339 – 5344, Nov. 2006.

[106] M. Anitescu, J. F. Cremer, and F. A. Potra, "Formulating 3D Contact Dynamics Problems," *Reports on computational mathematics*, Oct. 1995. Department of Mathematics, University of Iowa.

[107] D. Comaniciu, V. Ramesh, and P. Meer, "Real-time tracking of non-rigid objects using mean shift," in *Proceedings IEEE Conference on Computer Vision and Pattern Recognition*, vol. 2, pp. 142–149, 2000.

[108] C. Glocker, *Dynamik von Starrkörpersystemen mit Reibung und Stößen*, vol. 18. Düsseldorf: VDI-Verlag, 1995.

[109] C. Glocker, *Set-Valued Force Laws: Dynamics of Non-Smooth Systems*, vol. 1. Springer-Verlag Berlin Heidelberg, Jan. 2001.

[110] C. Glocker, *An Introduction to Impacts*, pp. 45–101. Vienna: Springer Vienna, 2006.

[111] F. Pfeiffer and C. Glocker, *Multibody Dynamics with Unilateral Contacts*. Wiley, New York, 1st. ed., 1996.

[112] J. J. Moreau, *Unilateral Contact and Dry Friction in Finite Freedom Dynamics*, vol. 302 of *International Centre for Mechanical Sciences (Courses and Lectures)*, pp. 1–82. Springer, 1988.

[113] R. Leine and N. van de Wouw, *Stability and convergence of mechanical systems with unilateral constraints*. Lecture notes in applied and computational mechanics, Germany: Springer-Verlag Berlin Heidelberg, 2008.

[114] R. Leine and H. Nijmeijer, *Dynamics and bifurcations of non-smooth mechanical systems*, vol. 18 of *Lecture Notes in Applied and Computational Mechanics*. Germany: Springer-Verlag Berlin Heidelberg, 2004.

[115] C. Glocker, "Newton's and Poisson's Impact Law for the Non-Convex Case of Reentrant Corners," in *Complementarity, Duality and Symmetry in Nonlinear Mechanics*, (Dordrecht), pp. 101–125, Springer Netherlands, 2004.

[116] C. Glocker and F. Pfeiffer, "On frictionless impact models in rigid-body systems," *Philosophical Transactions of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, vol. 359, no. 1789, pp. 2385–2404, 2001.

[117] B. Piccoli, "Measure Differential Equations," *Archive for Rational Mechanics and Analysis*, p. 1289–1317, Aug. 2017.

[118] C. Lacoursière, *Ghosts and machines : regularized variational methods for interactive simulations of multibodies with dry frictional contacts*. PhD thesis, Umeå University, Computing Science, 2007.

[119] C. Glocker and P. Panagiotopoulos, "Analytical Mechanics. Addendum I: Inequality Constraints with Elastic Impacts. The Convex Case," *ZAMM - Journal of Applied Mathematics and Mechanics / Zeitschrift für Angewandte Mathematik und Mechanik*, vol. 78, no. 4, pp. 219–229, 1998.

[120] C. Duriez, C. Andriot, and A. Kheddar, "Signorini's contact model for deformable objects in haptic simulations," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, vol. 4, pp. 3232–3237, Sep. 2004.

[121] C. Glocker, "Formulation of spatial contact situations in rigid multibody systems," *Computer Methods in Applied Mechanics and Engineering*, vol. 177, pp. 199 – 214, Apr. 1999.

[122] M. Anitescu and F. A. Potra, "A time-stepping method for stiff multibody dynamics with contact and friction," *International Journal for Numerical Methods in Engineering*, vol. 55, no. 7, pp. 753–784, 2002.

[123] I. Newton, *Philosophiae naturalis principia methematica*. Jussu Societatis Regiae ac Typis Josephi Streater, Jul. 1687.

[124] S. Poisson, *Traité de mécanique*. Bachelier, 2nd ed., 1833.

[125] V. Lubarda, "The Bounds on the Coefficients of Restitution for the Frictional Impact of Rigid Pendulum Against a Fixed Surface," *Journal of Applied Mechanics*, vol. 77, Jan. 2010.

[126] W. Stronge, "Rigid Body Collisions with Friction," in *Proceedings of The Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 431, pp. 169–181, Oct. 1990.

[127] B. Brogliato, *Nonsmooth Mechanics*. Springer International Publishing Switzerland, 3th ed., 2016.

[128] N. S. Nguyen and B. Brogliato, *Multiple Impacts in Dissipative Granular Chains*, vol. 72. Springer-Verlag Berlin Heidelberg, 2014.

[129] W. Stronge, "Friction in collisions: Resolution of a paradox," *Journal of Applied Physics*, vol. 69, no. 2, pp. 610–612, 1991.

[130] C. Studer and C. Glocker, "Representation of Normal Cone Inclusion Problems in Dynamics Via Non-linear Equations," *Archive of Applied Mechanics*, vol. 76, pp. 327–348, Nov. 2006.

[131] C. Glocker and R. Leine, "A set-valued force law for spatial Coulomb–Contensou friction," *European Journal of Mechanics*, vol. 22, no. 2, pp. 193 – 216, 2003.

[132] L. X. Anh, *Dynamics of Mechanical Systems with Coulomb Friction*. Springer, Berlin, Heidelberg, 1st ed., 2003.

[133] M. Anitescu, F. A. Potra, and D. E. Stewart, "Time-stepping for three-dimensional rigid body dynamics," *Computer Methods in Applied Mechanics and Engineering*, vol. 177, no. 3, pp. 183 – 197, 1999.

[134] D. E. Stewart, "Convergence of a Time-Stepping Scheme for Rigid-Body Dynamics and Resolution of Painlevé's Problem," *Archive for Rational Mechanics and Analysis*, vol. 145, pp. 215–260, Dec. 1998.

[135] D. Stewart and J. C. Trinkle, "An Implicit Time-Stepping Scheme for Rigid Body Dynamics with Inelastic Collisions and Coulomb Friction," *International Journal for Numerical Methods in Engineering*, vol. 39, no. 15, pp. 2673–2691, 1996.

[136] C. Studer, *Numerics of Unilateral Contacts and Friction*, vol. 47 of *Lecture Notes in Applied and Computational Mechanics*. Springer-Verlag Berlin Heidelberg, Jan. 2009.

[137] T. Schindler, "Consistent Time-Integration Schemes for Flexible Multibody Systems with Friction and Impacts," habilitation, Technische Universität München, Schriesheim, 2017.

[138] J. J. Júdice, "Algorithms for Linear Complementarity Problems," in *Algorithms for Continuous Optimization: The State of the Art*, (Dordrecht), pp. 435–474, Springer Netherlands, 1994.

[139] T. Schindler, S. Rezaei, J. Kursawe, and V. Acary, "Half-explicit timestepping schemes on velocity level based on time-discontinuous Galerkin methods," *Computer Methods in Applied Mechanics and Engineering*, vol. 290, pp. 250 – 276, 2015.

[140] J. D. Hoffman, *Numerical Methods for Engineers and Scientists*. New York, NY: McGraw-Hill, 1992.

[141] M. Anitescu and F. A. Potra, "Formulating Dynamic Multi-Rigid-Body Contact Problems with Friction as Solvable Linear Complementarity Problems," *Nonlinear Dynamics*, vol. 14, pp. 231–247, Nov. 1997.

[142] V. Acary and B. Brogliato, *Numerical Methods for Nonsmooth Dynamical Systems*, vol. 35. Springer-Verlag Berlin Heidelberg, 1th ed., 2008.

[143] D. E. Stewart, "Rigid-Body Dynamics with Friction and Impact," *SIAM Rev.*, vol. 42, pp. 3–39, Mar. 2000.

[144] P. Pérez, C. Hue, J. Vermaak, and M. Gangnet, "Color-Based Probabilistic Tracking," in *Proceedings of the 7th European Conference on Computer Vision-Part I*, ECCV '02, (London, UK), pp. 661–675, Springer-Verlag, 2002.

[145] Algoryx, "Agx dynamics." https://www.algoryx.se/products/agx-dynamics/. Accessed: 15-03-2020.

[146] Optitrack, "Motion capture systems." https://optitrack.com/. Accessed: 15-03-2020.

Appendix A

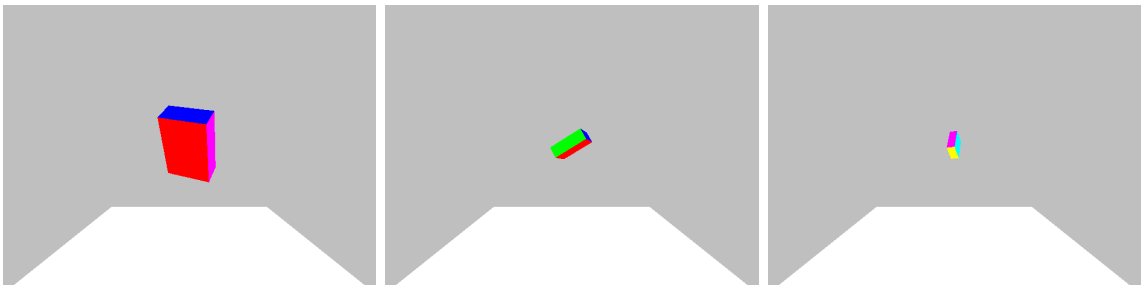
Behavior of the Likelihood Function

In this section, we will illustrate the behavior of the likelihood function discussed in Section 4.2. To this end, we consider three images, shown in Figure A.1, where the cuboid appears in different poses. The distance from the camera is different in each image; in Figure A.1a the cuboid is 0.5m in front of the camera, whereas in Figures A.1b and A.1c, the cuboid is respectively 1m and 1.5m in front of the camera. This allows us to test the performance of the likelihood function when the object is represented by less pixels in the image.

Recall that the camera coordinate frame is oriented, such that the distance from the camera is defined in the z-direction, and the x- and y-axis of that frame span a plane parallel to the image plane. First, we evaluate the likelihood function in the xy-plane, by varying the x- and y-position of the cuboid around the ground truth. This is discussed in Section A.1. Next, we evaluate the performance of the likelihood function regarding the z-direction, by varying the position in this direction around the ground truth. This is discussed in Section A.2. Finally, we vary the orientation of the cuboid around its ground truth and evaluate the performance, which will be discussed in Section A.3.

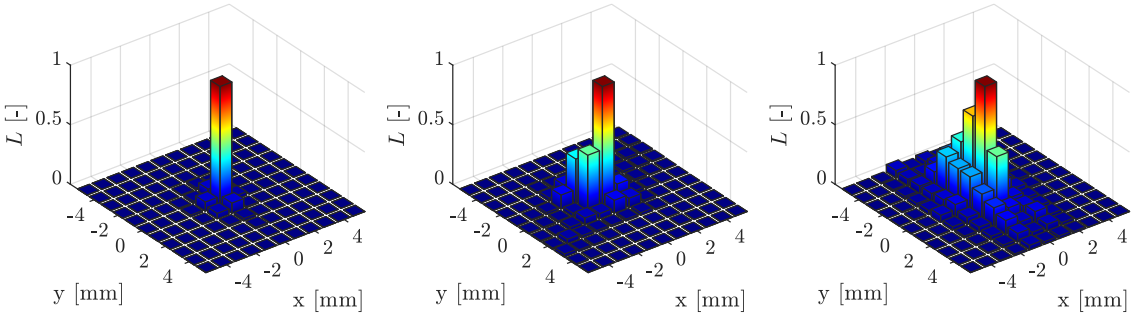
A.1 Varying the x- and y-position of the cuboid

A set of particles is created in the xy-plane around the ground truth locations of the cuboids shown in Figure A.1. These particles are all spaced 1mm apart, in a range of -5mm to 5mm, in both x- and y-direction. This creates a total set of 121 particles of which the likelihood is evaluated according to (4.12). These values are then normalized to be in the range $[0, 1]$.



(a) **(b)** **(c)**

Figure A.1: Images used to compute the convergence of the likelihood function.



(a) **(b)** **(c)**

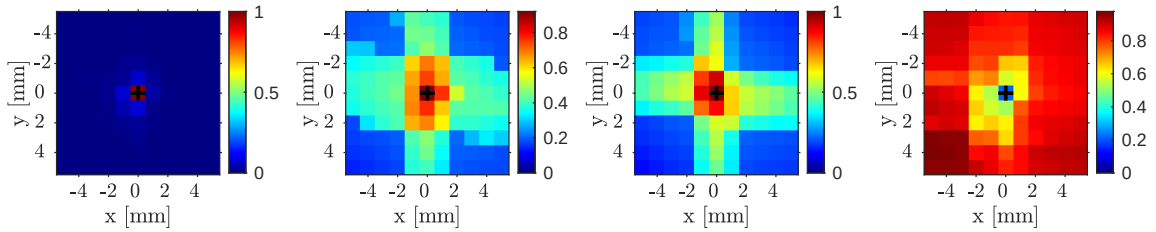
Figure A.2: Bar plots showing the likelihood for different positions around the ground truth. Labels correspond to the images of Figure A.1.

This results in the bar-plots shown in Figure A.2 which, from left to right, correspond to the images of Figure A.1. In Figure A.2, the ground truth position corresponds to $x = 0$ and $y = 0$ for each image. It is clear that, for each figure, the ground truth position yields the highest likelihood and that the particles positioned around the ground truth have a very low likelihood. Furthermore, when the object is further away, the likelihood becomes less discriminate, which is a result of the object being represented in the image by less pixels. It that case, it may happen that the particles placed 1mm apart, may give the same representation in the image plane. However, the likelihood function is still able to locate the cuboid up to 1mm accuracy when the cuboid is at 1.5m in front of the camera, as can be concluded from the right image of Figure A.2.

Recall that the likelihood function (4.12) directly depends on the three similarities computed by (4.8), (4.9), and (4.10), and a particle should have a high likelihood when S_1 and S_2 are high, but S_3 is low. In order to see how these similarities contribute to the likelihood, they are shown in Figures A.3, A.4, and A.5, for each image of Figure A.1, respectively. In each image of Figure A.3, the black plus sign indicates the maximum or minimum value.

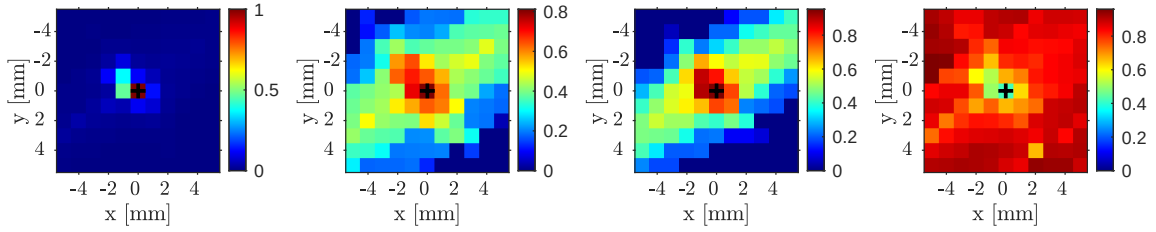
Considering Figure A.3, Figure A.4, and Figure A.5, it can be seen that S_3 has a dominating contribution to the likelihood computation, as this similarity is more discriminating between particles. We therefore set its weighting parameter κ_3, which corresponds to its contribution in the distance function (4.11), to $\kappa_3 = 5$. The weighting parameters κ_1 and κ_2 were then set to $\kappa_1 = 1$ and $\kappa_2 = 2$. The weighting on S_2 is thus set higher than the weighting on S_1. This choice is made since we see that S_2 seems to be more discriminating when the object is further away, as can be concluded by comparing Figure A.5b with Figure A.5c.

We can also see that, when the object is further away, it becomes harder to distinguish the different particles. This especially becomes clear when one compares Figure A.3b and A.5b. In Figure A.3b, the particle related to the ground truth position has a much higher value than the particles around it, while, in Figure A.5b, some of the particles around the ground truth end up having the same value, and can therefore not be distinguished from the ground truth position. However, considering the likelihood itself, one can see, by considering Figure A.5a, that even if the object is far away from the camera, the likelihood of the ground truth position is still much higher than particles positioned close around it.



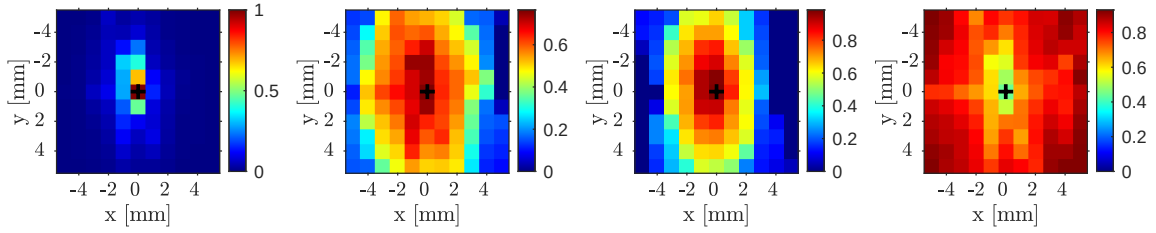
(a) **(b)** **(c)** **(d)**

Figure A.3: Likelihood (a) and similarities S_1 (b), S_2 (c), and S_3 (d) as a result of evaluating Figure A.1a.



(a) **(b)** **(c)** **(d)**

Figure A.4: Likelihood (a) and similarities S_1 (b), S_2 (c), and S_3 (d) as a result of evaluating Figure A.1b.



(a) **(b)** **(c)** **(d)**

Figure A.5: Likelihood (a) and similarities S_1 (b), S_2 (c), and S_3 (d) as a result of evaluating Figure A.1c.

The results in this section were obtained by considering variation of the particles in x- and y-direction around the ground truth position. However, they do not take variation in z-direction into account. This will be discussed in the next section.

A.2 Varying the z-direction

In this section, the likelihood function is evaluated by varying the z-direction of the particles around the ground truth. Therefore, particles are sampled between 50mm in front of the ground truth position and 50mm behind the ground truth positions, with increments of 1mm. Both the likelihood and the similarities S_1, S_2, and S_3 are evaluated for all images of Figure A.1, of which the results are shown in Figure A.6. The result for A.1a, shown in the left image of Figure A.6, show that if the object is close to the camera, it is possible to estimate its z-position accurately, as the likelihood peaks at the ground truth position (at $z = 0$), and is small when $z \neq 0$. However, if the object is further away, it becomes hard to estimate the z-position of the cuboid: the middle and right image of Figure A.6 show

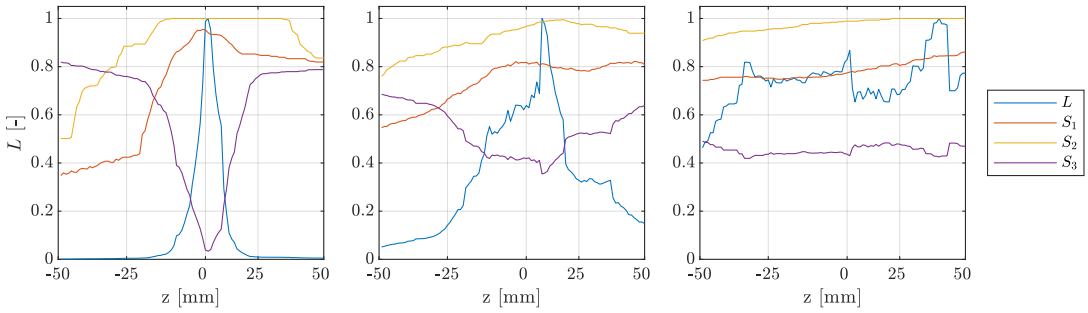


Figure A.6: Results of varying in z-direction. The results are plotted for the images shown in Figure A.1 from left to right, respectively. The ground truth position for each plot is at $z = 0$.

high likelihood values for a big range of z-positions. As in these cases the object is far away, and therefore represented by a limited amount of pixels in the image plane, changing the z-direction will not affect its representation in the image plane much. Therefore, estimating the z-position will be hard if the cuboid is far away from the camera.

A.3 Varying the orientation

In this section, we evaluate how the likelihood function behaves by changing the orientation of the particles with respect to the ground truth. Therefore, the particles are rotated between -180 degrees and 180 degrees around the ground truth, located at 0 degrees. Again, for each of the images in Figure A.1, the likelihood and the similarities S_1, S_2, and S_3 are evaluated. We do this by rotating particles around x, y, and z, and finally around x and y simultaneously. In Figures A.7, A.8, and A.9, the results are shown.

Considering the results, we see that the likelihood is only high within a few degrees around the ground truth. For the results of Figure A.1a, so when the cuboid is only 0.5m in front of the camera, we see that the highest likelihood is found within 3 degrees around the ground truth. For the results of Figures A.1b and A.1c, so when the cuboid is respectively 1m and 1.5m in front of the camera, we observe the same accuracy. However, we also see that the likelihood function in these cases is less discriminating, as the likelihood is high in a wider range around the ground truth. Furthermore, we observe that, similar to the results of Section A.1, the contribution of S_3 is dominant over S_1 and S_2. However, we now see that S_1 is performing better than S_2, being more discriminating and resulting in lower values for particles further away from the ground truth.

The observant reader will note that, in all the results, the likelihood has an increase around -180 and 180 degrees. A reason for this is that, at these points, the cuboid is completely flipped along one axis, such that the contouring edges are aligned exactly with the ground truth. However, since the colors of the faces do not match the ground truth, the likelihood stays nonetheless low. We also observe that S_1 and S_2 do not continuously decrease when the orientation is further from the ground truth. For example, in the most left image of Figure A.9, we see that S_1 and S_2 show small increases around -70 or $+110$ degrees. We do however not know what causes this phenomenon, and we do not observe the same behavior in S_3. However, these small increases do not affect the overall likelihood, and are therefore neglected.

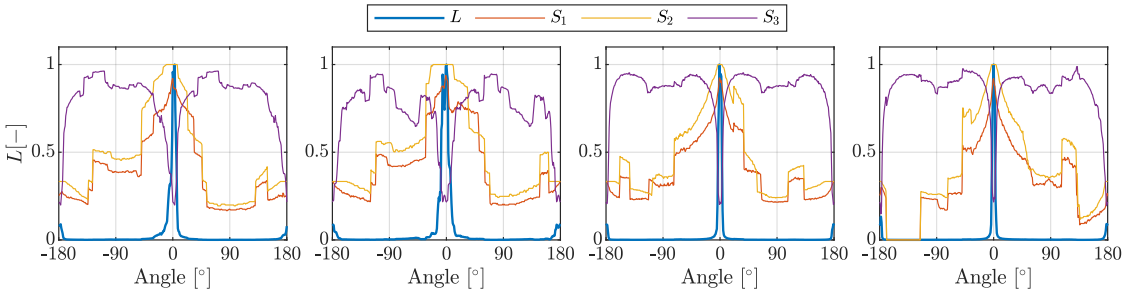


Figure A.7: Likelihood for various orientation around the ground truth of Figure A.1a. From left to right: rotating around x, y, z, and x, y, respectively. The ground truth orientation is at 0° for each plot.

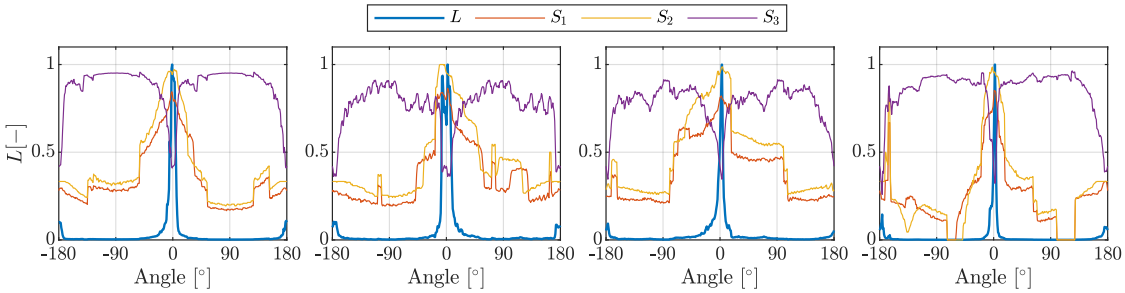


Figure A.8: Likelihood for various orientation around the ground truth of Figure A.1b. From left to right: rotating around x, y, z, and x, y, respectively. The ground truth orientation is at 0° for each plot.

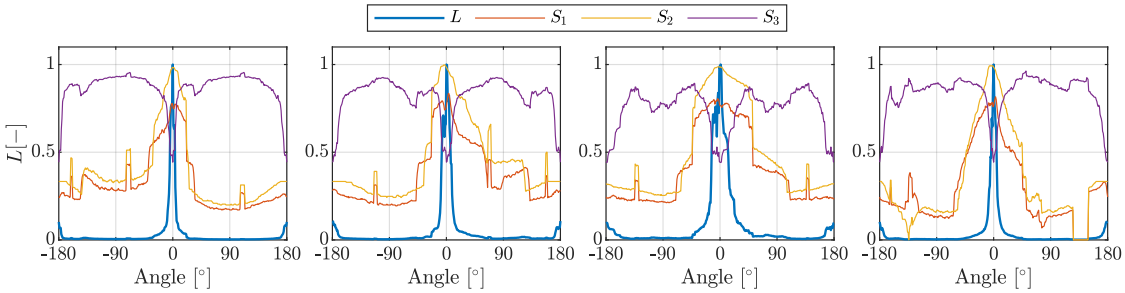


Figure A.9: Likelihood for various orientation around the ground truth of Figure A.1c. From left to right: rotating around x, y, z, and x, y, respectively. The ground truth orientation is at 0° for each plot.

A.4 Conclusion

The results of this section show that the likelihood function discussed in Chapter 3 is able to locate the cuboid from a set of particles around the ground truth. From Section A.1, it can be seen that, for each scenario of Figure A.1, the likelihood function is able to find the ground truth position up to 1mm accuracy, even when the object is far away. Section A.2 showed that finding the right z-position of the cuboid is difficult. When the cuboid is close to the camera this is still possible, but when the cuboid is 1.5 meters in front of the camera, no distinction can be made between particles 50mm in front or behind the ground truth. This is unavoidable, and results in a trade-off between image resolution and depth estimation. Finally, Section A.3 showed that the likelihood function is able to find the right pose of the cuboid within 3 degrees accuracy.

Appendix B

Group Operations for the Pose and States Lie Groups

In Section 2.3 we discussed the group of the poses $\mathcal{P} = \mathrm{SO}(3) \times \mathbb{R}^3$ and the group of the states $\mathcal{S} = \mathrm{SO}(3) \times \mathbb{R}^3 \times \mathbb{R}^3 \times \mathbb{R}^3_\times$. In this section, we will give the identity element, inverse, and exponential and logarithmic mappings for both groups.

B.1 The group of the poses

In Section 2.3, we defined the group of the poses as $\mathcal{P} = \mathrm{SO}(3) \times \mathbb{R}^3$, which is the direct product of $\mathrm{SO}(3)$ and $\mathbb{R}^3$. As a result, the identity element $e \in \mathcal{P}$ is given by

$$e = (\mathbf{I}_3, \mathbf{0}), \tag{B.1}$$

where $\mathbf{I}_3$ is the identity element of $\mathrm{SO}(3)$ and $\mathbf{0}$ the identity of $\mathbb{R}^3$. The inverse of an element $\mathbf{H}_1 = (\mathbf{R}_1, \mathbf{o}_1) \in \mathcal{P}$, is then given by

$$(\mathbf{R}_1, \mathbf{o}_1)^{-1} = \left(\mathbf{R}_1^{-1}, -\mathbf{o}_1\right), \tag{B.2}$$

where $\mathbf{R}_1^{-1}$ is given by the matrix inverse. As a result, by the definitions discussed in 2.2.1, this gives

$$(\mathbf{R}_1, \mathbf{o}_1)(\mathbf{R}_1, \mathbf{o}_1)^{-1} = \left(\mathbf{R}_1 {\mathbf{R}_1}^{-1}, \mathbf{o}_1 - \mathbf{o}_1\right) = e, \tag{B.3}$$

and

$$(\mathbf{R}_1, \mathbf{o}_1)^{-1}(\mathbf{R}_1, \mathbf{o}_1) = \left({\mathbf{R}_1}^{-1}\mathbf{R}_1, -\mathbf{o}_1 + \mathbf{o}_1\right) = e. \tag{B.4}$$

The Lie algebra corresponding to $\mathcal{P}$, denoted by $\mathfrak{p}$, is now defined as $\mathfrak{p} \coloneqq \mathfrak{so}(3) \times \mathbb{R}^3$, which we can write as $\mathfrak{p} = \mathbb{R}^3_\times \times \mathbb{R}^3$, as discussed in Section 2.2.3. Let us now consider the elements $\mathbf{H}_1 = (\mathbf{R}_1, \mathbf{o}_1) \in \mathcal{P}$ and $\mathbf{H}_2 = (\mathbf{R}_2, \mathbf{o}_2) \in \mathcal{P}$, and a tangent space at $\mathbf{H}_1$, denoted by $T_{\mathbf{H}_1}\mathcal{P}$. We can than map the point $\mathbf{H}_2$ to this tangent space by the use of the logarithmic mapping $\mathrm{Log} : \mathcal{P} \to \mathfrak{p}$, which, by using (2.46), results in

$$\begin{aligned}
\boldsymbol{\mathcal{H}} &= \mathrm{Log}\left(\mathbf{H}_1^{-1}\mathbf{H}_2\right), && \text{(B.5)}\\
&= \mathrm{Log}\left((\mathbf{R}_1, \mathbf{o}_1)^{-1}(\mathbf{R}_2, \mathbf{o}_2)\right), && \text{(B.6)}\\
&= \mathrm{Log}\left(\left({\mathbf{R}_1}^{-1}\mathbf{R}_2, -\mathbf{o}_1 + \mathbf{o}_2\right)\right), && \text{(B.7)}\\
&= \left(\left(\boldsymbol{\xi}^{\mathbf{R}}\right)^\wedge, \boldsymbol{\xi}^{\mathbf{o}}\right), && \text{(B.8)}
\end{aligned}$$

which results in an element in $\mathfrak{so}(3) \times \mathbb{R}^3$. Note that in (B.8), we used the substitution of $\boldsymbol{\xi}^{\mathbf{o}} = -\mathbf{o}_1 + \mathbf{o}_2$ and the fact that $\left(\boldsymbol{\xi}^{\mathbf{R}}\right)^\wedge$ is the result of the logarithmic mapping of $\left({\mathbf{R}_1}^{-1}\mathbf{R}_2\right)$ on SO(3), as discussed in Section 2.2.2. Using the fact that we can write an element of $\mathfrak{so}(3)$ as an element of $\mathbb{R}^3_\times$, we can rewrite the last line of (B.5) to obtain

$$\boldsymbol{\mathcal{H}} = \left(\boldsymbol{\xi}^{\mathbf{R}}, \boldsymbol{\xi}^{\mathbf{o}}\right) \in \mathfrak{p}. \tag{B.9}$$

The exponential mapping is the inverse operation, and can be used to map the point $\boldsymbol{\mathcal{H}} \in T_{\mathbf{H}_1}\mathcal{P}$ back to the group. Using (2.45), this yields

$$\mathbf{H}_2 = \mathbf{H}_1 \operatorname{Exp}\left(\boldsymbol{\mathcal{H}}\right), \tag{B.10}$$

$$= \mathbf{H}_1 \operatorname{Exp}\left(\left(\boldsymbol{\xi}^{\mathbf{R}}, \boldsymbol{\xi}^{\mathbf{o}}\right)\right). \tag{B.11}$$

By using the fact that we can write $\boldsymbol{\xi}^{\mathbf{R}} \in \mathbb{R}^3_\times$ as $\left(\boldsymbol{\xi}^{\mathbf{R}}\right)^\wedge \in \mathfrak{so}(3)$, we can use the exponential mapping on SO(3), as given by (2.50), to map $\boldsymbol{\xi}^{\mathbf{R}}$ back to SO(3). As the exponential mapping on $\mathbb{R}^3$ maps the element to its own group, we can write (B.11) as

$$\mathbf{H}_2 = \mathbf{H}_1 \left(\mathbf{R}_1^{-1}\mathbf{R}_2, -\mathbf{o}_1 + \mathbf{o}_2\right), \tag{B.12}$$

$$= \left(\mathbf{R}_1, \mathbf{o}_1\right) \left(\mathbf{R}_1^{-1}\mathbf{R}_2, -\mathbf{o}_1 + \mathbf{o}_2\right), \tag{B.13}$$

$$= \left(\mathbf{R}_1\mathbf{R}_1^{-1}\mathbf{R}_2, \mathbf{o}_1 - \mathbf{o}_1 + \mathbf{o}_2\right), \tag{B.14}$$

$$= \left(\mathbf{R}_2, \mathbf{o}_2\right). \tag{B.15}$$

B.2 The group of the states

In Section 2.3, we defined the group of the states as $\mathcal{S} = \mathrm{SO}(3) \times \mathbb{R}^3 \times \mathbb{R}^3 \times \mathbb{R}^3_\times$, which is the direct product of SO(3), $\mathbb{R}^3$, $\mathbb{R}^3$, and $\mathbb{R}^3_\times$. As a result, the identity element $e \in \mathcal{S}$ is given by

$$e = \left(\mathbf{I}_3, \mathbf{0}, \mathbf{0}, \mathbf{0}\right), \tag{B.16}$$

where $\mathbf{I}_3$ is the identity element of SO(3) and $\mathbf{0}$ the identity of $\mathbb{R}^3$ and $\mathbb{R}^3_\times$, respectively. The inverse of an element $\mathbf{x}_1 = \left(\mathbf{R}_1, \mathbf{o}_1, \mathbf{v}_1, \boldsymbol{\omega}_1\right) \in \mathcal{S}$, is then given by

$$\left(\mathbf{R}_1, \mathbf{o}_1, \mathbf{v}, \boldsymbol{\omega}_1\right)^{-1} = \left(\mathbf{R}_1^{-1}, -\mathbf{o}_1, -\mathbf{v}_1, -\boldsymbol{\omega}_1\right), \tag{B.17}$$

where $\mathbf{R}_1^{-1}$ is given by the matrix inverse. As a result, by the definitions discussed in 2.2.1, this gives

$$\left(\mathbf{R}_1, \mathbf{o}_1, \mathbf{v}_1, \boldsymbol{\omega}_1\right) \left(\mathbf{R}_1, \mathbf{o}_1, \mathbf{v}_1, \boldsymbol{\omega}_1\right)^{-1} = \left(\mathbf{R}_1{\mathbf{R}_1}^{-1}, \mathbf{o}_1 - \mathbf{o}_1, \mathbf{v}_1 - \mathbf{v}_1, \boldsymbol{\omega}_1 - \boldsymbol{\omega}_1\right) = e, \tag{B.18}$$

and

$$\left(\mathbf{R}_1, \mathbf{o}_1, \mathbf{v}_1, \boldsymbol{\omega}_1\right)^{-1} \left(\mathbf{R}_1, \mathbf{o}_1, \mathbf{v}_1, \boldsymbol{\omega}_1\right) = \left({\mathbf{R}_1}^{-1}\mathbf{R}_1, -\mathbf{o}_1 + \mathbf{o}_1, -\mathbf{v}_1 + \mathbf{v}_1, -\boldsymbol{\omega}_1 + \boldsymbol{\omega}_1\right) = e. \tag{B.19}$$

The Lie algebra corresponding to $\mathcal{S}$ is defined as $\mathfrak{s} \coloneqq \mathbb{R}^3_\times \times \mathbb{R}^3 \times \mathbb{R}^3 \times \mathbb{R}^3_\times$. If we consider a tangent space at the point $\mathbf{x}_1 \in \mathcal{S}$, denoted by $T_{\mathbf{x}_1}\mathcal{S}$, and an element $\boldsymbol{\chi} = \left(\boldsymbol{\xi}^{\mathbf{R}}, \boldsymbol{\xi}^{\mathbf{o}}, \boldsymbol{\xi}^{\mathbf{v}}, \boldsymbol{\xi}^{\boldsymbol{\omega}}\right) \in \mathfrak{s}$, we can define the exponential mapping $\exp : \mathfrak{s} \mapsto \mathcal{S}$ such that

$$\mathbf{x}_2 = \mathbf{x}_1 \operatorname{Exp}\left(\boldsymbol{\chi}\right), \tag{B.20}$$

$$= \mathbf{x}_1 \left(\operatorname{Exp}\left(\left(\boldsymbol{\xi}^{\mathbf{R}}\right)^\wedge\right), \boldsymbol{\xi}^{\mathbf{o}}, \boldsymbol{\xi}^{\mathbf{v}}, \boldsymbol{\xi}^{\boldsymbol{\omega}}\right), \tag{B.21}$$

$$= \left(\mathbf{R}_1 \operatorname{Exp}\left(\left(\boldsymbol{\xi}^{\mathbf{R}}\right)^\wedge\right), \mathbf{o}_1 + \boldsymbol{\xi}^{\mathbf{o}}, \mathbf{v}_1 + \boldsymbol{\xi}^{\mathbf{v}}, \boldsymbol{\omega}_1 + \boldsymbol{\xi}^{\boldsymbol{\omega}}\right), \tag{B.22}$$

where, in (B.21) and (B.22), Exp denotes the exponential mapping on SO(3), given by (2.50). The logarithmic mapping is defined as the inverse operation such that $\operatorname{Log} : \mathcal{S} \mapsto \mathfrak{s}$. Given the elements $\mathbf{x}_1 \in \mathcal{S}$ and $\mathbf{x}_2 \in \mathcal{S}$, the logarithmic mapping maps the point $\mathbf{x}_2$ to the tangent

space at $\mathbf{x}_1$ such that

$$\boldsymbol{\chi} = \mathrm{Log}\left(\mathbf{x}_1^{-1}\mathbf{x}_2\right), \tag{B.23}$$

$$= \mathrm{Log}\left((\mathbf{R}_1, \mathbf{o}_1, \mathbf{v}_1, \boldsymbol{\omega}_1)^{-1}, (\mathbf{R}_2, \mathbf{o}_2, \mathbf{v}_2, \boldsymbol{\omega}_2)\right), \tag{B.24}$$

$$= \mathrm{Log}\left(\mathbf{R}_1^{-1}\mathbf{R}_2, -\mathbf{o}_1 + \mathbf{o}_2, -\mathbf{v}_2 + \mathbf{v}_2, -\boldsymbol{\omega}_1 + \boldsymbol{\omega}_2\right), \tag{B.25}$$

$$= \left(\left(\boldsymbol{\xi}^{\mathbf{R}}\right)^{\wedge}, \boldsymbol{\xi}^{\mathbf{o}}, \boldsymbol{\xi}^{\mathbf{v}}, \boldsymbol{\xi}^{\boldsymbol{\omega}}\right), \tag{B.26}$$

which results in an element of $\mathfrak{so}(3) \times \mathbb{R}^3 \times \mathbb{R}^3 \times \mathbb{R}^3_{\times}$, which can then be written as an element of $\mathfrak{s}$ to finally obtain

$$\boldsymbol{\chi} = \left(\boldsymbol{\xi}^{\mathbf{R}}, \boldsymbol{\xi}^{\mathbf{o}}, \boldsymbol{\xi}^{\mathbf{v}}, \boldsymbol{\xi}^{\boldsymbol{\omega}}\right) \in \mathfrak{s}. \tag{B.27}$$