Visual Object Tracking with Collision Models

Validation of Impact-Aware Particle Filters on Real-Life Videos

Dingemans, S.

Award date:
2021

EINDHOVEN UNIVERSITY OF TECHNOLOGY

Department of Mechanical Engineering
Dynamics and Control Section

Visual Object Tracking with Collision Models:

Validation of Impact-Aware Particle Filters on Real-Life Videos

MSc Graduation Project

S. Dingemans (1002724)

DC 2021.034

Project coaches: dr. ir. A. Saccon
dr. ir. A. Bernardino (IST, Lisbon, Portugal)
ir. M.J. Jongeneel

Project supervisor: prof. dr. ir. N. van de Wouw

Committee member: dr. G. Dubbelman

Eindhoven, August 19, 2021

Abstract

The process of following objects through the frames of a recording is called *visual object tracking*. Visual object tracking is used in many different applications and is a widely studied subject in literature. State-of-the-art object tracking algorithms are able to accurately track objects whose motion is smooth, but when an object changes directions abruptly, for example during a collision, a loss of tracking is observed. The tracking of objects that collide with a surface is still an uncharted territory and is an essential skill to make impact-aware manipulation possible for robots.

We will use visual object tracking techniques in this research to keep track of the 3D position and orientation of parcels in time after being tossed on a surface. The execution of the tossing motion and the flight of the parcel are recorded with a single RGB camera. Tossing and grabbing objects, instead of slowly picking and placing them, are possible solutions to increase the throughput of robotic arms. This is beneficial for logistic companies, which have to cope with the ever-growing e-commerce market and labor scarcity. Another advantage of tossing is that objects can be placed outside the reach of the robotic arm.

In previous work, an object tracking algorithm has been developed that is capable of tracking a cuboid colliding with a surface in a synthetic environment. A cuboid is a common shape for objects in logistic applications. The developed algorithm is called the *Geometric Unscented Particle Filter* and takes into account complex dynamics in the motion model, whereas state-of-the-art object tracking techniques use a constant velocity motion model. Complex dynamics, emerging from impacts and friction, generate nonsmooth behaviour or, in other words, jumps in the velocity. By incorporating a nonsmooth motion model, the object tracking performance is increased significantly in comparison to existing methods. The Geometric Unscented Particle Filter has only been tested on synthetic images with simulated physics, so it is unknown whether the algorithm is able to accurately track cuboid-shaped objects that collide with a surface in real-life videos.

In this research, the developed object tracking algorithm is validated on real-life videos and modifications are implemented to further increase the tracking performance. The videos show a parcel colliding with a conveyor belt, after being tossed by hand. First, the performance of one of the main components of the algorithm is evaluated, which is the approximate likelihood function. Then, a new image-processing routine is proposed to obtain the 3D position and orientation of the parcel from a 2D image. Finally, the performance of the Geometric Unscented Particle Filter is compared to the performance of state-of-the-art object tracking algorithms. To be able to draw conclusions regarding the accuracy of the position and orientation estimation, the results are compared to ground truth data measured via an external motion capture system. In correspondence with the results in simulation, implementing a nonsmooth motion model in the algorithm leads to a much better tracking performance than that of an algorithm with a constant velocity motion model.

Preface

At the start of this project, I had never worked on topics related to visual object tracking and collision models before. Fortunately, Maarten Jongeneel, who is my predecessor in this project, wrote a very clear and high level report from which I learned a lot about different filtering techniques and theories I had never heard of, such as Lie group theory.

During the project, there were many moments I got stuck or needed help. Maarten was luckily always available to answer questions via e-mail or in a short Teams meeting. Once in two weeks, also Alessandro Saccon and Alexandre Bernardino were present to help me and discuss the progress of my project. Due to the many useful discussions and feedback in these meetings, I constantly got new ideas and motivation to keep improving my work. Next to Maarten, Alessandro, and Alexandre, Nathan van de Wouw also has helped me by giving helpful feedback during our monthly meetings. It was always nice to hear that you are on the right track, even though it sometimes felt that I had not progressed a lot.

Besides my supervisors, I also want to thank Vanderlande for providing the boxes and letting me make use of the test setup located at the Innovation Lab. I am especially grateful for their effort to make it possible for me to visit the lab during the COVID-19 pandemic, which has prevented a huge delay in my project.

Last but not least, I want to thank my family and friends. At the times that it did not go very well, my mother, who also worked at home every day, was there to encourage me. For simple questions or just some fun, one of my friends was always available for a Teams meeting. Without their support during this difficult time of not being able to leave the house, I could not have delivered this work.

Sander Dingemans
Eindhoven, August 2021

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 $\mathbf{x} \in \mathfrak{so}(3)$
$\mathrm{SO}(3)$	The Special Orthogonal group in 3 dimensions
$\mathfrak{so}(3)$	The Lie algebra of SO(3)
$\mathrm{SE}(3)$	The Special Euclidean group in 3 dimensions
$\mathfrak{se}(3)$	The Lie algebra of SE(3)
$\mathcal{P}$	The Lie group of the poses: $\mathcal{P} = \mathrm{SO}(3) \times \mathbb{R}^3$
$\mathcal{S}$	The Lie group of the states: $\mathcal{S} = \mathcal{P} \times \mathfrak{p}$
$\mathfrak{p}$	The Lie algebra of $\mathcal{P}$
$\mathfrak{s}$	The 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	Camera sensor frame
$[A]$	Orientation frame associated to A
$\mathbf{b}$	Bin index
$\mathbf{b_d}$	Bin index at pixel location $\mathbf{d}$
B	Body fixed frame, located at the center of mass
$B[A]$	Frame with origin $\mathbf{o}_B$ and orientation $[A]$
$\mathbb{B}_H$	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
C	Contact surface frame
$\mathbf{d}$	General location of a pixel in the image
D	Distance between histograms
$\boldsymbol{e}_N$	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
$\mathbb{E}$	Expected value
f	Focal length
${}_B\mathbf{f}$	Coordinates of the wrench $\mathbf{f}$ w.r.t. B
F	Frame of the base of the robot
g	Gravitational acceleration
g_{Ni}	Gap-function of contact point i, denoting the distance in normal direction between the contact point and the contact surface
G	Camera casing frame
h	Height of the box

$\boldsymbol{H}$	Color histogram
$\boldsymbol{\mathcal{H}}$	Pose parameters of the object, element of $\mathfrak{p}$
${}^{A}\mathbf{H}_{B}$	Homogeneous transformation from B to A
${}_{B}\mathbb{I}_{B}$	Inertia matrix w.r.t. frame B
k	Radial distortion parameter
K	Image coordinate frame
$\mathbf{K}$	Camera intrinsic matrix
$\boldsymbol{K}$	Kalman gain
l	Length of the box
$L^{(i)}$	Approximate likelihood assigned to particle i
$\mathbf{L}_t$	Measurement at time t, obtained with the approximate likelihood function
m	Mass of the box
${}_{B}\mathbb{M}_{B}$	6×6 inertia tensor expressed w.r.t. frame B
n_a	Dimension of the augmented state
n_c	Number of closed contacts
n_x	Dimension of the state
N	Number of particles
N_f	Number of visible faces
${}^{A}\mathbf{o}_{B}$	Coordinates of the origin $\mathbf{o}_B$ w.r.t. A
$\mathbf{p}_i$	Contact point i
${}^{A}\mathbf{p}_{i}$	Coordinates of point $\mathbf{p}_i$ w.r.t. A
$\mathbf{P}_N$	Momentum associated to the contact in normal direction
$\mathbf{P}_T$	Momentum associated to the contact in tangential direction
$\boldsymbol{P}^{a}$	Augmented state covariance
$\boldsymbol{P}^{m}$	Process noise covariance
$\boldsymbol{P}^{n}$	Measurement noise covariance
$\boldsymbol{P}^{x}$	Covariance of the state
$\boldsymbol{P}^{xy}$	State-observation cross-covariance
$\boldsymbol{P}^{y}$	Covariance of the observation
$\boldsymbol{Q}$	Process noise covariance, element of $\mathfrak{s} \otimes \mathfrak{s}$
$\boldsymbol{R}$	Measurement noise covariance, element of $\mathfrak{p} \otimes \mathfrak{p}$
${}^{A}\mathbf{R}_{B}$	Rotation matrix from B to A
r	Convergence parameter for proximal functions
s	Skew parameter
s_x	Number of pixels per world unit in the horizontal direction
s_y	Number of pixels per world unit in the vertical direction
S	Bhattacharyya similarity between two histograms
t	Global time
u	Pixel coordinate in the horizontal direction
u_0	x-coordinate of the principal point in pixels
v	Pixel coordinate in the vertial direction
v_0	y-coordinate of the principal point in pixels
${}^{B}\mathbf{v}_{A,B}$	Twist expressing the velocity of B w.r.t. 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 and width of the box
$\mathbf{w}^{T}_{Ni}$	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
W_j	Weight of the sigma point j
$\mathbf{W}_N$	Matrix containing the normal force directions of all closed contacts
$\mathbf{W}_T$	Matrix containing the tangential force directions of all closed contacts
$\mathbf{x}_t$	State of the box at time t
$\boldsymbol{x}_d$	Vector of distorted normalized image coordinates

$\boldsymbol{x}_u$	Vector of undistorted normalized image coordinates
X^a	Sigma point of the augmented state
X^x	Sigma point of the state
X^m	Sigma point of the process noise
X^n	Sigma point of the observation noise
${}_B\mathbf{X}^A$	Wrench transformation from A to B
$\mathbf{y}_t$	Observation at time t
$\mathbf{z}_t$	Measurement at time t
$\mathbf{Z}_t$	Measurement at time t, obtained with the object detector and pose estimator

Greek symbols

α_k	Scaling parameter of the unscented Kalman filter with $0 < \alpha < 1$
α_x	Focal length in terms of pixels in the horizontal direction
α_y	Focal length in terms of pixels in the vertical direction
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
δ	Dirac delta function
δ_k	Kronecker delta function
Δt	Time interval of a time step
ϵ	Scaling parameter of the approximate likelihood function
ζ	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
$\boldsymbol{\Sigma}$	General covariance matrix
${}^B\boldsymbol{\omega}_{A,B}$	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 that 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 $\mathcal{C}_{Ti}$
$\otimes$	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 on 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 $\boldsymbol{x} = \begin{bmatrix} x_1 & x_2 & ... & x_n \end{bmatrix}^T$

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

Contents

1 Introduction **1**
- 1.1 Visual object tracking . . . 1
- 1.2 State of the art in visual object tracking . . . 4
  - 1.2.1 2D and 3D object tracking . . . 4
  - 1.2.2 Object tracking-by-detection and recursive object tracking . . . 4
  - 1.2.3 Marker-based and natural feature-based object tracking . . . 6
  - 1.2.4 Vision sensors . . . 7
- 1.3 Problem definition . . . 7
  - 1.3.1 Object detection and 3D pose estimation . . . 8
  - 1.3.2 Likelihood computation . . . 8
  - 1.3.3 Evaluation and improvement of the state estimation . . . 9
- 1.4 Structure of the report . . . 9

2 Preliminaries **10**
- 2.1 Visual object tracking using particle filters . . . 10
  - 2.1.1 Sequential Monte Carlo . . . 11
  - 2.1.2 Resampling . . . 12
  - 2.1.3 Choice of proposal distribution . . . 13
  - 2.1.4 Using an unscented Kalman filter to create a proposal distribution . . . 14
- 2.2 Lie groups: basic notation and probability distributions . . . 17
  - 2.2.1 Preliminaries on Lie groups . . . 17
  - 2.2.2 The Special Orthogonal group SO(3) . . . 19
  - 2.2.3 The Special Euclidean group SE(3) . . . 19
  - 2.2.4 Gaussian distributions on Lie groups . . . 20
- 2.3 Group structure of the state . . . 21
- 2.4 Multibody dynamics notation . . . 22
  - 2.4.1 Coordinate frames and points . . . 22
  - 2.4.2 Rigid-body velocity . . . 23
  - 2.4.3 Wrench notation . . . 25
- 2.5 Image formation . . . 25
  - 2.5.1 Perspective projection . . . 25
  - 2.5.2 Projection of the box onto the image plane . . . 26
- 2.6 Conclusion . . . 27

3 Geometric and dynamic model of boxes **28**
- 3.1 Experimental setup and data processing . . . 28
  - 3.1.1 Hardware elements . . . 28
  - 3.1.2 Recording instruments . . . 29
  - 3.1.3 Projection optimization . . . 32
  - 3.1.4 Synchronization MoCap data and RGB images . . . 34
- 3.2 Geometric model of the boxes . . . 36
- 3.3 Approximate likelihood computation . . . 37
- 3.4 Motion model . . . 39
  - 3.4.1 Equations of motion . . . 40

3.4.2 Contact law . . . 40
3.4.3 Impact law . . . 41
3.4.4 Friction law . . . 43
3.4.5 Time-stepping algorithm . . . 44
3.4.6 Fixed-point iteration . . . 45
3.5 Conclusion . . . 46

4 Pose estimator and experimental validation of the GUPF **47**
4.1 Performance evaluation of the approximate likelihood computation . . . 47
4.1.1 Reference color histogram . . . 48
4.1.2 Position estimation . . . 48
4.1.3 Orientation estimation . . . 51
4.1.4 Conclusion approximate likelihood computation . . . 53
4.2 Obtaining a pose measurement . . . 54
4.2.1 Measurement in previous work . . . 54
4.2.2 Measurement from newly proposed method . . . 55
4.2.3 Conclusion measurement . . . 59
4.3 Performance evaluation of the GUPF . . . 59
4.3.1 Settings and general subjects . . . 59
4.3.2 Results Video 1 (Box 5) . . . 63
4.3.3 Results Video 2 (Box 3) . . . 69
4.3.4 Conclusion performance evaluation GUPF . . . 75
4.4 Conclusion . . . 75

5 Conclusion and recommendations **76**
5.1 Conclusion . . . 76
5.1.1 Approximate likelihood computation . . . 76
5.1.2 Object detection and 3D pose estimation . . . 77
5.1.3 Evaluation and improvement of the state estimation . . . 78
5.2 Recommendations . . . 79
5.2.1 Approximate likelihood computation . . . 79
5.2.2 Object detection and 3D pose estimation . . . 80
5.2.3 State estimation with the GUPF . . . 80

Bibliography **81**

A Performance approximate likelihood under motion blur **86**
A.1 Position estimation . . . 86
A.2 Orientation estimation . . . 88
A.3 Conclusion . . . 89

B Annotation of images **90**
B.1 Goal of the annotation . . . 90
B.2 Bounding boxes and labels . . . 90

C Results Video 3 (Box 4) **91**
C.1 Recorded video and 3D trajectory . . . 91
C.2 Measurement results . . . 92
C.3 Comparing the particle filters . . . 93

Chapter 1

Introduction

1.1 Visual object tracking

In the past decade, the amount of recorded and stored video data has increased considerably all over the world. Manually analyzing and processing this data to understand the content of the videos are challenging and time consuming tasks. Therefore techniques that automatically interpret video content are in great demand [1]. An important process in the interpretation of video content is localizing and determining the position and orientation of objects in each frame of a recording, which is called visual object tracking. Visual object tracking is already used in many different sectors, some examples are defense, healthcare, and security [2]. It is, for instance, possible to track missiles, vehicles, molecules, and people in public areas. By monitoring people, the crowd dynamics can be studied, but it is also possible to check whether people keep their distance from each other, which is relevant during the current COVID-19 pandemic.

Visual tracking techniques are not only used in the previously mentioned sectors, they are also popular in the industrial sector, for example in warehousing companies where goods are stored before they are sold or distributed. With the help of computer vision, a lot of tasks that are currently performed by human operators in warehouses have the potential of being fully automated. Some of these tasks include recognizing the marking on goods, staff tracking, and item management [3]. When a shipment arrives at a warehouse, the first task is identifying all the goods. The barcode, or other types of marking, can be read automatically with a camera, after which the obtained information is sent to a database. Staff tracking can be useful in maintaining the safety in a warehouse. Cameras can locate workers being at a dangerous area, it can be checked whether the workers wear a helmet, and it is possible to recognize unauthorized people. In a lot of these warehouses, an often performed task is the picking of objects from a conveyor belt and the placing on a pallet or into a bin, or vice versa. This operation is part of the item management process. The picking and placing of objects is a non-ergonomic task for human operators, so more and more warehousing companies introduce robots to do this kind of labor. In Figure 1.1, a robot is shown that picks a parcel from a conveyor belt.

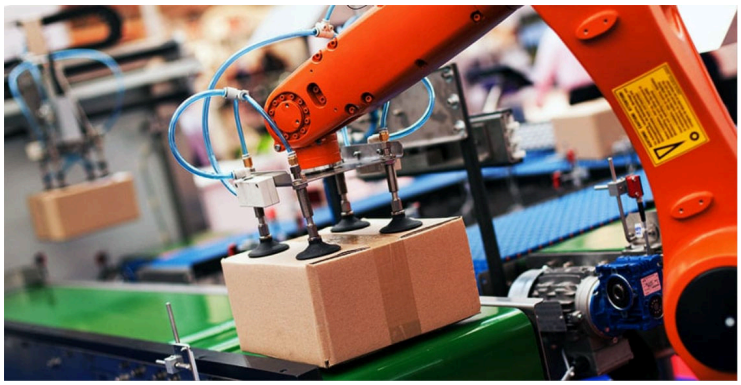


Figure 1.1: Robot picking parcels from a conveyor belt. Image taken from [4].

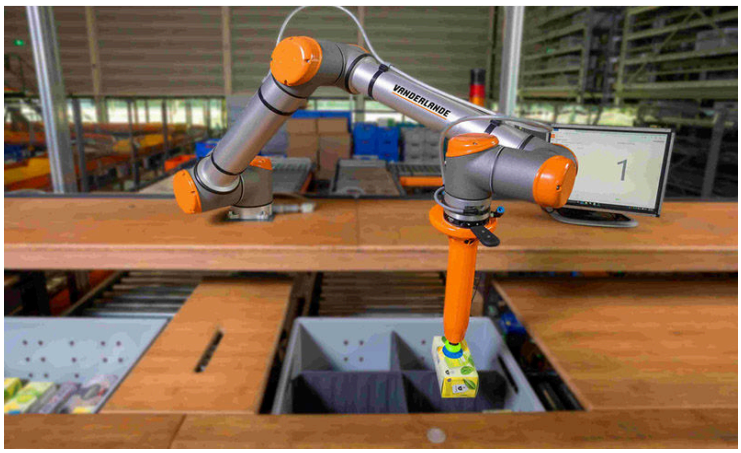


Figure 1.2: Smart Item Robotics solution that makes use of a vision system to identify and locate products. Image taken from [5].

Robotic systems do not only ease the labor of employees, they can also do the same work for a long time without losing consistency, accuracy, and speed. Another reason why robots are introduced is to cope with labor scarcity. In order to create a more flexible work environment, these robots can be equipped with vision systems to handle a wide variety of products, without human intervention. The Smart Item Robotics (SIR) solution, which can be seen in Figure 1.2, is an example of such a robotic system. This robot can identify different products by making use of its perceptive sensors. Furthermore, the SIR is able to take the dimensions of a product into account, before it places the product on a conveyor belt or into a bin.

Over the last years, the number of orders to be processed by warehousing companies has increased significantly due to the growing e-commerce market. As there is a shortage of personnel, it is hard for these companies to keep up with the demands from industry. Increasing the throughput of the pick-and-place operation performed by the robotic arms may be a way to boost the productivity. Currently, the picking and placing of parcels happens at zero relative velocity between the end effector of the robot and the parcel, which is a relatively slow operation. A possible solution to speed up the pick-and-place operation is by grabbing and tossing parcels, instead of slowly picking and placing them. Not only the throughput improves by tossing parcels, but it is also possible to place objects outside the reach of the robotic arm. For this application, visual object tracking can be used to monitor the execution of the tossing motion and the flight of the parcel.

After being tossed, the parcels collide with a surface before coming to rest. In the case of a warehousing company, this surface can be a conveyor belt or the sides of a tote box. A camera is used to record the trajectory of the parcel and to estimate its pose, i.e., the position and orientation, in each frame of the video, an object tracking algorithm is employed. The tracking of objects that collide with surfaces is, however, still an uncharted territory in the literature. State-of-the-art object tracking techniques often describe the motion of the object with a constant velocity motion model, which is not sufficient when collisions occur. The reason for this is that jumps in the velocity are introduced due to the collisions. This means that existing methods thus have great difficulties tracking objects where such complex dynamics are involved.

In previous research, Jongeneel [6] has developed an algorithm that is able to track the motion of a cuboid, colliding with a surface, in a simulated environment. More specifically, the novel impact-aware Geometric Unscented Particle Filter (GUPF) is used to estimate the state of the cuboid in each frame of the video. The Geometric Unscented Particle Filter is a combination of the unscented particle filter applied on a state that evolves in a Lie group and a motion model that takes into account complex dynamics, such as friction and impacts. A geometric approach has been taken to

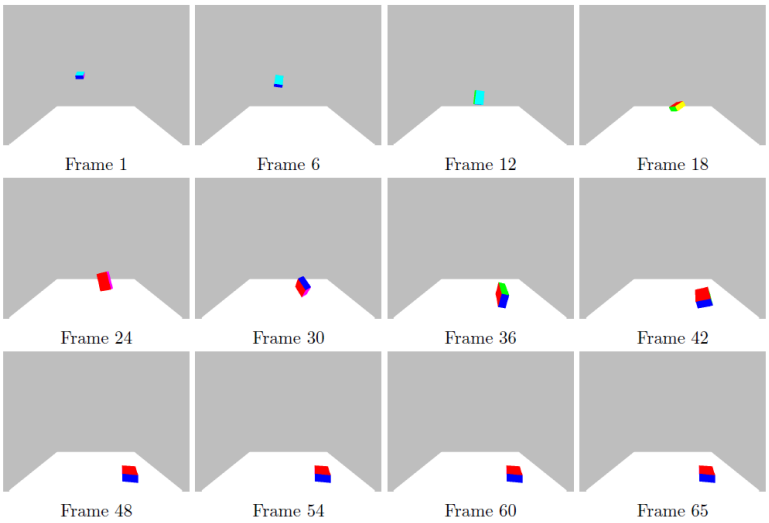


Figure 1.3: A sequence of synthetic images, which has been used to test and develop the object tracking algorithm. Image taken from [6].

model the dynamics, which clarifies the name of the object tracking algorithm. The algorithm can accurately estimate the state of the cuboid, however it has only been tested on synthetic images with simulated physics. A sequence of these synthetic images is shown in Figure 1.3 for illustration.

Before this algorithm can be used in real-life applications, some open issues must be addressed. Since the synthetic images do not contain realistic effects such as motion blur, background clutter, and changing illumination levels, it is unknown whether the developed algorithm is sufficiently robust to cope with these effects and to provide good pose and velocity estimates. It is thus necessary to validate the algorithm on real-life videos. Moreover, the movement of the object in the simulated environment, described by the motion model, needs to be validated on experimental data. To achieve good tracking performance, the motion model must describe the movement of the object as accurately as possible. Poort [7] has already used the dynamical model proposed in [6] to determine the expected motion of a parcel in free flight and across the impacts given a known initial state until the package comes to rest. Finally, the algorithm must be able to handle uncertainties in the object properties. In [6], it is assumed that the object is a cuboid with fixed dimensions and mass, and that each face has a different distinctive color, as can be seen in Figure 1.3. In this report, we consider tossing scenarios within warehouses where a lot of parcels with different dimensions and masses have to be processed. These parcels are similar to the one shown in Figure 1.1, which usually have a uniform color and possibly some printing on its faces.

The goal of this research is to validate the developed object tracking algorithm in the work of Jongeneel [6] on real-life videos and to implement modifications to further increase the tracking performance of the algorithm. The tossing of parcels on a conveyor belt by a robotic arm is a promising application to develop and validate the object tracking algorithm with collision models, but in the end the goal is to be able to track objects colliding with surfaces in many different applications. The tracking of objects that collide with surfaces is namely an essential skill to make impact-aware manipulation possible for robots and can, for example, thus be used to develop robots that can catch objects.

The remainder of this chapter discusses state-of-the-art techniques for visual object tracking in Section 1.2, the research goals in Section 1.3, and the structure of the report in Section 1.4. State-of-the-art visual object tracking techniques are discussed to provide the reader with useful background information in the field of computer vision and to motivate the choice for the Geometric Unscented Particle Filter to track cuboid shaped objects that collide with a surface.

1.2 State of the art in visual object tracking

The goal of visual object tracking is following the movement of an object from frame to frame in a video, recorded with a camera. The trajectory of the object is projected onto a 2D plane, which is called the image plane. Different techniques exist for the tracking of an object, that depend on the environment and application. Since the amount of literature on the subject of object tracking is quite extensive, only the most important techniques and approaches are discussed in the following sections. A survey that gives a good overview of the different techniques is [8].

1.2.1 2D and 3D object tracking

A distinction can be made between 2D and 3D visual object tracking. 2D object tracking focuses on following an object in the image plane in a video sequence. However, the position and orientation, also referred to as the *pose*, of the object in 3D space remains unknown. In [9], 2D object tracking is used to track persons in a public space, in that case a subway platform. Figure 1.4 shows three frames of an image sequence where a person wearing a white jacket is tracked. With this technique, the 3D position and orientation of the person is not recovered. When it is necessary to follow an object in subsequent frames, and simultaneously keep track of the pose of an object, 3D object tracking techniques must be used.

With 3D object tracking, the actual position and orientation of the object in 3D space is recovered continuously. An example of an application where 3D object tracking is used is Position-Based Visual Servoing [10]. In this application, the motion of a robotic arm is controlled based on computer vision data, such that it is possible to manipulate an object. In [11], the task of a robotic system is to track and grasp a moving object, by making use of two cameras. It is important to continuously estimate the pose of an object at each time instant, to make it possible for the robot to grasp the object.

This research focuses on exploiting object tracking techniques to estimate the pose of an object in 3D space. Therefore, in the remainder of this section, 3D object tracking techniques are discussed and 2D object tracking techniques are not considered any further.

1.2.2 Object tracking-by-detection and recursive object tracking

The goal of object detection is localizing and identifying a target object in the images of a recording. How often detection is performed in a tracking problem depends on the technique that is used. There are two main techniques for object tracking; object tracking-by-detection and recursive object tracking [8]. With the object tracking-by-detection technique, the object is detected in each frame of a video and the current pose is estimated without making use of the pose in previous frame [12]. On the other hand, in the recursive object tracking technique, object detection is only applied once,

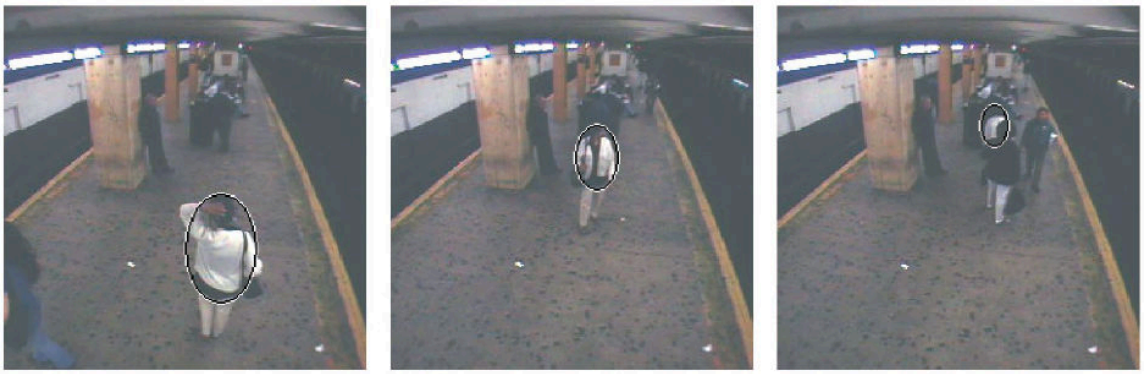


Figure 1.4: 2D tracking of a person on a subway platform. The person can be followed through the frames, but the 3D position and orientation remains unknown. Image taken from [9].

namely to initialize or re-initialize the object tracking algorithm [13]. After the tracking algorithm is initialized, the pose is predicted in the current frame based on knowledge of the pose in the previous frame. This prediction is done with the help of a motion model, which describes the expected motions an object can undergo in a particular environment. The benefit of the recursive object tracking technique is that image features can be found much easier when a priori knowledge about the pose is used, in comparison to the object tracking-by-detection technique. Besides that, the tracking accuracy of the recursive method is better, because the motion model filters noise and prevents gross detection errors. It is, however, important that the motion model is a good approximation of the object's motion, otherwise target loss may occur. Recursive object tracking is discussed more elaborately below, by introducing several filtering methods that fuse the image information with the temporal motion models of the object.

In this work, the *state* of an object is the position, orientation, linear velocity, and the angular velocity [14]. The recursive object tracking technique is called this way, because the dominant approaches typically use a recursive Bayesian estimation of the state distribution. In the remainder of this section, six different filters of this type are discussed.

Bayes filter

The Bayes filter [15] estimates the state at the current time step by making use of the state estimate of the previous time step and the available measurement. This process can be divided into two steps. In the first step, the state estimate of the previous time step and the state transition model, also known as the motion model, are used to make a prediction of the current state estimate. This predicted state estimate is called the *a priori* state estimate. In the second step, the current measurement is used to update the a priori state estimate, resulting in an improved state estimate, called the *a posteriori* state estimate. This process is repeated for each time step. In the context of visual object tracking, the state of the object in the current frame is first estimated with the prediction step and after that by using the current image. The prediction of the current state estimate can be updated, leading to a better state estimate of the object.

Kalman filters

The Kalman filter [16] is a variant of the Bayes filter, which assumes that the state transitions and the observation model are linear. However, when either the state transitions, observation model, or both are nonlinear, the Kalman filter cannot be used. The extended Kalman filter (EKF) and the unscented Kalman filter (UKF) [17], on the other hand, are able to handle nonlinear state transitions and observation models. In all types of Kalman filters, the probability distribution of the state is approximated by a Gaussian random variable. The difference between the extended and unscented Kalman filter is that for the extended Kalman filter the Gaussian random variable is propagated through the first-order linearization of the nonlinear system, while for the unscented Kalman filter a set of sample points, capturing the true mean and covariance of the Gaussian random variable, is propagated through the true nonlinear system [17]. When the Gaussian random variable is propagated through the first-order linearization of the nonlinear system, as is done with the extended Kalman filter, only first-order accuracy of the true mean and covariance can be achieved. By propagating through the true nonlinear system, third-order accuracy can be achieved. This means that the unscented Kalman filter is an improvement compared to the extended Kalman filter in terms of accuracy.

Particle filters

Besides the Bayes filter and ordinary, extended, and unscented Kalman filter, another way to recursively estimate the probability distribution of the state is by making use of sequential Monte Carlo methods, also called particle filters. Particle filters can handle any nonlinearity [18] and the main advantage is that the state distribution does not have to be Gaussian, it can be anything. Instead of representing the state distribution as a continuous distribution, it is represented by a set

of samples, which are called particles. These particles are sampled from a proposal distribution, which must be able to approximate the posterior distribution accurately enough. Different proposal distributions can be chosen to sample from. Often the transition prior is used as a proposal distribution. The transition prior is the probability distribution of the state, which is directly computed from the previous state, so without taking observations into account. Examples of particle filters that choose the transition prior as the proposal are the Bootstrap Filter and Condensation Algorithm [19].

It is also possible to create the proposal distribution with the unscented Kalman filter. Particle filters that use the unscented Kalman filter to create the proposal distribution are called unscented particle filter (UPF) [18]. The advantage of using the unscented Kalman filter to create the proposal distribution is that the particles sampled from this proposal distribution are closer to the area with high likelihood, resulting in an improved tracking performance.

1.2.3 Marker-based and natural feature-based object tracking

As explained in the previous section, before the pose can be estimated, the object needs to be detected. The detection and pose estimation can be simplified by relying on markers instead of natural features of the object [8]. Markers of different shapes and colors exist, which makes them easy to identify in an image. In [20], a Concentric Contrasting Circle is used as marker. This marker is created by placing a black ring on a white background. Next to circular markers, also planar square markers exist. This type of marker is used in [21] and also has a white inside and black contour. As it is known where the markers are located on the object, the pose can be estimated easily. Under certain circumstances, like the processing of parcels in a warehouse, one needs to rely on natural features, because no markers are present and often the parcels differ in size and appearance.

The detection and tracking of objects by making use of their natural features instead of markers is a lot more difficult. For natural feature-based object tracking two different methods can be used, one is model-based object tracking and the other one is feature-based object tracking. The model-based object tracking method uses 3D knowledge of the object to help with the detection and tracking of the object through the frames [22]. This technique is very useful when the dimensions and shape of the object are known. In Figure 1.5, an example of model-based object tracking is shown, where the known shape and color of the target object are used for tracking. Feature-based object tracking methods do not make use of any known geometric information. In that case, the 3D motion of the object is estimated by the detection and tracking of 2D primitives, such as points and lines [23].

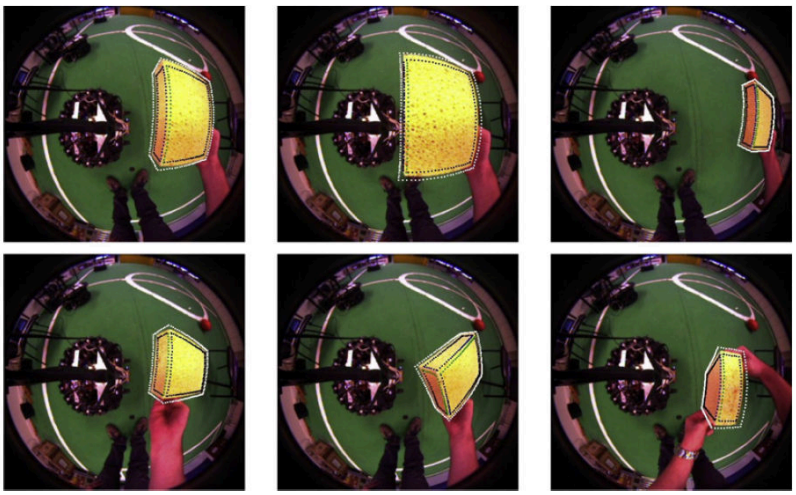


Figure 1.5: 3D model-based tracking of a cuboid in images with significant distortion due to the use of a wide-angle lens. Knowledge about the shape and color of the object is used to achieve accurate tracking. Image taken from [14].

1.2.4 Vision sensors

Different types of vision sensors can be used to record the movement of objects. A common choice for computer vision related problems is a simple standard RGB camera, which provides color information about the environment. From the calibrated camera images, information about the pose of the object can be extracted when its size and shape are known. When no geometric model of the object is available, multiple cameras or an RGB-D camera can be used to determine the pose of objects in space. RGB-D cameras combine the color information from RGB camera images with depth information for the estimation of the pose [24]. However, it is only possible to use RGB-D cameras in very specific environments, because of the sensitivity to light, limited reach, and sensitivity to noise.

As shown in [11], it is also possible to use multiple cameras to track an object. One camera is fixed in the environment, while the other camera is mounted at the end effector of a robot. This is especially handy in situations where an object is often occluded during the process. When the object is not visible for one camera, it can still be seen by the other camera. It is thus very important to position the camera, or multiple cameras, in such a way that the target object stays in the field of view of at least one camera.

As for the different object tracking techniques, the camera choice depends on the application and environment. There are a couple of important parameters that should be taken into account while choosing a camera, two of them are the resolution and frame rate. When an object travels at high speed and abruptly changes direction, a higher frame rate is necessary to fully capture the trajectory of the object. At the same time, a high resolution is desired to get accurate images of the object. A trade-off should be made between the frame rate and resolution. Typically, a high resolution means a low frame rate, and a high frame rate means a low resolution. When a high resolution and low frame rate is used, the computational cost will be large [25]. On the other hand, a high frame rate and low resolution results in a loss of spatial accuracy. Other camera parameters that play a crucial role in recording the movement of an object are the lens distortion and view angle of the camera.

1.3 Problem definition

As introduced in Section 1.1, visual object tracking is used in many different applications. Instead of tracking objects that move with a constant velocity, which is often done in literature, the focus of this project is on applying visual tracking techniques to track objects that collide with environment surfaces. When an object collides with a surface, jumps in the velocity of the object occur due to the impacts. The application that is considered in this project is the tossing of parcels by a robotic arm, but there are lot more applications where the tracking of objects that collide with surfaces can be used, for example robots that have to grasp or manipulate objects.

After being tossed, the parcels collide with a surface, for example a conveyor belt or the sides of a tote box. With an RGB camera, a video is made of the motion of the parcel, on which an object tracking algorithm is applied to follow the parcel from frame to frame. Since the parcels collide with a surface, the impact dynamics must be taken into account in the algorithm to achieve accurate object tracking. Jongeneel [6] has developed an object tracking algorithm that can track the motion of a cuboid shaped object colliding with a surface. The algorithm is called the Geometric Unscented Particle Filter, which is an unscented particle filter with a nonsmooth motion model, instead of a constant velocity motion model. As discussed in Section 1.2, the unscented particle filter is a 3D, recursive object tracking technique and it can handle non-Gaussian state variables. The GUPF is thus very suitable for the state estimation of a parcel that collides with a surface, however, it has only been tested on synthetic images with simulated physics. It is therefore unknown whether the algorithm can also keep track of cuboid shaped objects colliding with surfaces in realistic, more challenging environments.

The main goal of this research is defined as:

Validate the developed object tracking algorithm in [6] on real-life videos of parcels colliding with a surface and implement modifications to increase the tracking performance, where a motion capture system is used for recording the ground truth and evaluating the accuracy of the state estimation.

In the following sections, several subgoals are formulated, which need to be reached before the main goal of the research can be accomplished. The subgoals are based on the different components of the visual object tracking process.

1.3.1 Object detection and 3D pose estimation

Important components in visual object tracking are object detection and pose estimation. Object detection is used to localize and identify the target object in the image and pose estimation is used to obtain information about the position and orientation of the object in 3D space, after it is detected. Without the use of a detection and pose estimation routine, it is necessary to initialize the object tracking algorithm by explicitly using preknowledge of the pose at the start, which can be impractical in automated solutions. Besides this, recursive object tracking techniques, like particle filters, are fragile techniques when no use is made of a detection algorithm [8]. When the object is lost due to a complete occlusion or movement out of the camera view, it is possible that the object tracking algorithm is unable to recover. When the object becomes visible again, the loss of tracking can be solved by re-initializing the tracking algorithm by means of object detection routine. Furthermore, with object detection and pose estimation a measurement $\mathbf{z}_t$ of the pose parameters can be obtained in each frame of the video, which can be incorporated in the unscented Kalman filter to create a better proposal distribution for each particle.

In this work, we assume that the initial pose of the parcel is known and that it is visible in the image from the beginning to the end of the video, as is also the case in [6]. Regarding the measurement, Jongeneel has used two different methods to obtain $\mathbf{z}_t$. The first method makes use of the motion model and the second method adds Gaussian noise to the ground truth data. Employing the motion model to obtain the measurement is only a robust method when the motion of the particles is described accurately. If this is not the case, the particles may move away from the object in the image and the measurement will be inaccurate. The problem with the second method is that the ground truth is not known in a realistic setting, which makes this method unusable. Using an object detector and afterwards a pose estimator to obtain the measurement will prevent the situation where particles move away from the area with high likelihood, leading to a more robust method. This is the reason why the object tracking algorithm in [6] will be extended with an object detector and 3D pose estimator.

The performance of the detection algorithm may be affected by external influences or by camera-related issues. These external influences are changing illumination levels and background clutter. Common camera-related issues are motion blur and lens distortion. Also, the pose of the object can make the detection a lot harder. The object may, for example, be close or far away from the camera. Regardless of these factors, it should be possible to detect the object, which is in our case a parcel, in the image. When the parcel is detected, the next task is to estimate its pose. The main challenges are to draw a conclusion on the orientation of the parcel, as we have to deal with symmetry, and to accurately estimate the pose when the parcel is at different distances from the camera. The following subgoal is defined for object detection and pose estimation:

Extend the developed object tracking algorithm in [6] with a state-of-the-art object detector and 3D pose estimator, which are used to obtain a measurement of the pose parameters of the parcel.

1.3.2 Likelihood computation

Another important component in visual object tracking is the computation of the likelihood for each particle. In the work of Jongeneel [6], the likelihood for each particle is computed in synthetic images.

These synthetic images contain a cuboid with distinctive colors on each face and background with a uniform color. The likelihood is computed by making use of color histograms and the similarities between them. In real life, however, each face of a parcel usually has the same color, we have to deal with symmetry, and the background consists of different colors. Besides this, motion blur may also make the computation of the likelihood more complicated. It should thus be checked whether the method with color histograms can also be used to compute the likelihood for each particle in real images containing real parcels. If the likelihood function is not sufficiently discriminative, meaning that also a high likelihood is assigned to particles that are not close to the true pose of the parcel, it is necessary to come up with solutions to increase the performance of this method or to come up with a completely different method that has a better performance. The following subgoal is formulated for the likelihood computation:

Evaluate the performance of the likelihood computation based on color histograms, as in [6], on real images containing parcels with a uniform color and, if necessary, come up with solutions or another method to increase performance.

1.3.3 Evaluation and improvement of the state estimation

When the object detection algorithm, pose estimator, and likelihood computation work on real images, the total object tracking algorithm can be applied on a real-life video. The state of the parcel, i.e., the position, orientation, linear velocity, and angular velocity, will be estimated in each frame of the video and by comparing the results with the ground truth data, conclusions can be drawn regarding the accuracy of the estimation. The ground truth will be obtained by making use of an accurate and fast motion capture system. After evaluating the results, it is possible to identify which component or components of the object tracking algorithm cause inaccuracies in the state estimation. Subsequently, these components can be improved. The following subgoal is formulated for the evaluation and improvement of the state estimation:

Estimate the state of the parcel in 3D space at any point in time from a real-life video obtained with a single RGB camera and implement modifications in case tracking is unsatisfying in the continuous and/or in the discrete portions.

1.4 Structure of the report

This report has the following structure. The relevant background material is discussed in Chapter 2. The background material consists of an explanation of the unscented particle filter, Lie group theory, multibody dynamics notation, and the image formation process. In Chapter 3, the geometric model, the approximate likelihood function, and the nonsmooth motion model are covered, which are the key elements of the Geometric Unscented Particle Filter. Furthermore, the experimental setup and its components are mentioned in Chapter 3. Chapter 4 subsequently discusses the main contributions of this research. The main contributions are the performance evaluation of the approximate likelihood computation, proposing a new method to obtain a measurement of the pose parameters of the parcel, and the evaluation of the GUPF on real-life videos. Finally, a conclusion and recommendations are given in Chapter 5, which are based on the results obtained in Chapter 4.

Chapter 2

Preliminaries

This chapter discusses the relevant background information for this research. In Section 2.1, the theory behind the unscented particle filter is explained, which is the filter that is used to estimate the state of the box in each frame of a recording. Section 2.2 covers the Lie group theory and its corresponding notation. Then, the group structure of the state is given in Section 2.3, followed by an overview of the multibody dynamics notation in Section 2.4. Finally, Section 2.5 describes the image formation process and the projection of the box onto the image plane.

The unscented particle filter, Lie group theory, and the group structure of the state are already extensively and well explained in the work of Jongeneel [6], and are therefore concisely repeated in Section 2.1, 2.2, and 2.3. Only small changes are made to these parts, which are mostly references, cross-references, layout-related issues, and typographical errors. Larger alterations are indicated by '(red.)'.

2.1 Visual 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 $y_{1:t}$. 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 [26, 27]. Substituting the result of (2.4) 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*), a 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

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

with δ the Dirac delta function and N a chosen number of particles. It is, however, not possible to sample from the filtering density, as the exact distribution is unknown [18, 26–28]. However, if it is possible to evaluate the filtering density, one can sample particles from a distribution that is assumed to be 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 [28]. 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

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

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

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

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

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

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 [29].

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 [19, p. 368], [30, 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 [29]. 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 [31–34]. In this work, the choice is made for systematic resampling due to its simplicity, high resampling quality, and low computational cost. Formally, the goal of resampling is to convert the weighted distribution into the unweighted density. Hence, the distribution

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

is replaced by

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

with n_i the number of children of particle $\mathbf{x}_t^{(i)}$ in the new set of particles $\{\mathbf{x}_t^{*(k)}\}_{k=1}^N$ [35]. The children n_i should be chosen such that the density $\hat{p}(\mathbf{x}_t|\mathbf{y}_{1:t})$ is close to $p(\mathbf{x}_t|\mathbf{y}_{1:t})$ [19, 31] in the sense that for

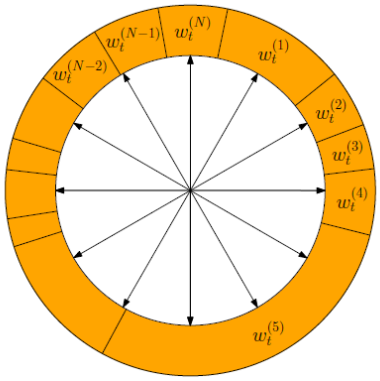


Figure 2.1: Illustration of systematic resampling.

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 [31,33]. 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 three new particles, whereas the particle corresponding to $w_t^{(3)}$ will be discarded from 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 [13, 14]. This yields the Bootstrap filter as discussed in [19, 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

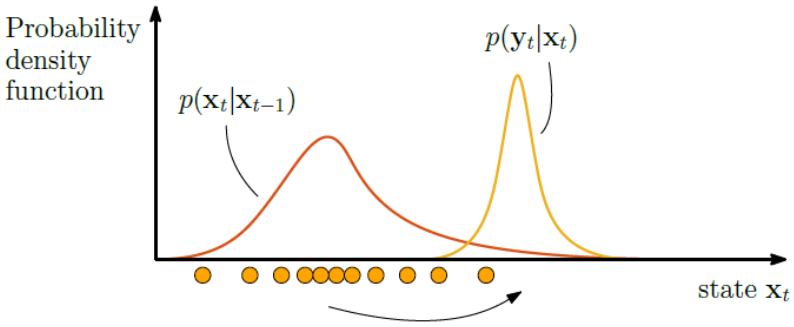


Figure 2.2: Choosing the prior as the proposal distribution. If we sample from the transition prior, the particles (here represented by the orange dots) can move away from the area with high likelihood, resulting in degeneracy of the particle set.

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 [18], 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.

2.1.4 Using an unscented Kalman filter to create a proposal distribution

The unscented transformation

The key idea behind the unscented Kalman filter 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 [36]. 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

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

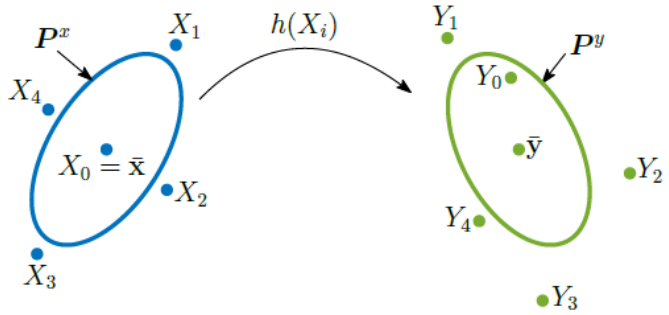


Figure 2.3: Schematic representation of the unscented transformation for a two-dimensional state, with $n_x = 2$. The sigma points X_i of the state $\mathbf{x}$ are propagated through the non-linear model h to obtain the points Y_i. These points are then used to compute the mean $\bar{\mathbf{y}}$ and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

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

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

respectively. In (2.24) and (2.25), λ is chosen as $\alpha_k^2(n_x + \zeta) - n_x$ as in [17]. 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

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

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

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

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

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

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

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 [37], 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

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

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

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

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

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

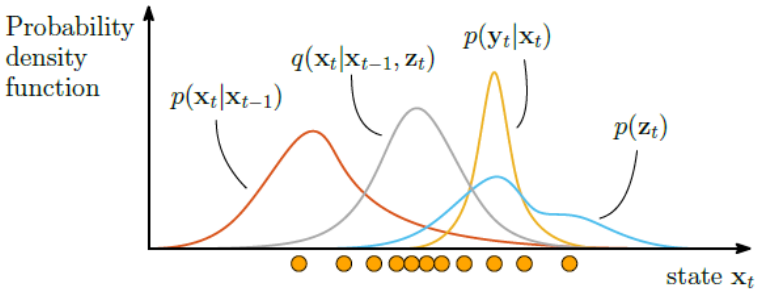


Figure 2.4: Using the UKF to create a proposal distribution. By incorporating a measurement $\mathbf{z}_t$ at time t, the transition prior $p(\mathbf{x}_t|\mathbf{x}_{t-1})$ can be updated by the use of an UKF to create a better proposal distribution $q(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{z}_t)$ which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

where $\mathbf{z}_t$ is the measurement obtained at time t. This measurement can be obtained in several ways. In this work, object detection is used to localize the box in the image and afterwards the likelihood function is employed on the resulting bounding box to obtain the measurement $\mathbf{z}_t$ of the pose of the box. Section 4.2 discusses the subjects of object detection and 3D pose estimation more elaborately (red.). 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

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

and

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

Van de Merwe et al. proposed in [18] 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 $p(\mathbf{y}_t|\mathbf{x}^{(i)}_t)$, the *transition prior* given as $p(\mathbf{x}^{(i)}_t|\mathbf{x}^{(i)}_{t-1})$, and the *proposal distribution* given by $q(\mathbf{x}^{(i)}_t|\mathbf{x}^{(i)}_{t-1}, \mathbf{z}_t)$. The likelihood function will be based on image data, and requires projection of the 3D position/orientation of the particle onto the image plane, which we will discuss in Section 3.3. 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. 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 an operator $\circ$, such that the following properties hold [38]:

- 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 $(g_1 \circ g_2) \circ g_3 = g_1 \circ (g_2 \circ g_3)$.

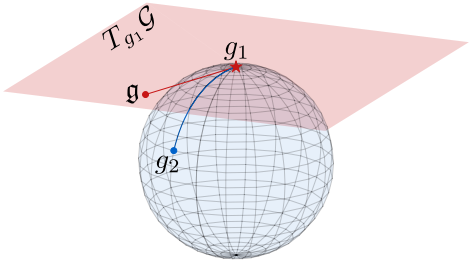


Figure 2.5: The exponential and logarithmic mappings on a Lie group transfer points between the group and the tangent space. Here, g_2 is mapped to $\mathfrak{g}$ in the tangent space at g_1 by the logarithmic mapping. The inverse operation is accomplished by the exponential mapping.

- There exists 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 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

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

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

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

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 [39].

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 $\mathbb{R}^{3\times 3}$ orthogonal matrices with determinant equal to 1 given by

$$\mathrm{SO}(3) \coloneqq \{\mathbf{R} \in \mathbb{R}^{3\times 3} \mid \mathbf{R}^T\mathbf{R} = \mathbf{I}_3, \det(\mathbf{R}) = 1\}, \tag{2.48}$$

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

$$\boldsymbol{\xi}^\wedge \coloneqq \begin{bmatrix} 0 & -\xi_z & \xi_y \\ \xi_z & 0 & -\xi_x \\ -\xi_y & \xi_x & 0 \end{bmatrix} \in \mathfrak{so}(3), \tag{2.49}$$

where $\boldsymbol{\xi} = \begin{bmatrix}\xi_x & \xi_y & \xi_z\end{bmatrix}^T \in \mathbb{R}^3$ and $(\cdot)^\wedge$ is known as the *hat*-operator as in [40, 41]. The exponential mapping for SO(3) can efficiently be computed using Rodrigues' formula as in [42, 43] such that

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

where $\|\boldsymbol{\xi}\| = \sqrt{\xi_x^2 + \xi_y^2 + \xi_z^2}$. It is convenient to know that ξ_x, ξ_y, and ξ_z form the direction of rotation and $\theta = \|\boldsymbol{\xi}\|$ 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 $\boldsymbol{\xi}$ are not uniquely covering SO(3) [44]. 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 SE(3)

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

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

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 [41], 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 a 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 [45], for example, any point on the uniform distribution on SO(3) could be called a mean. To this end, we follow the approach of [44, 46], 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 [45, 47], 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 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.

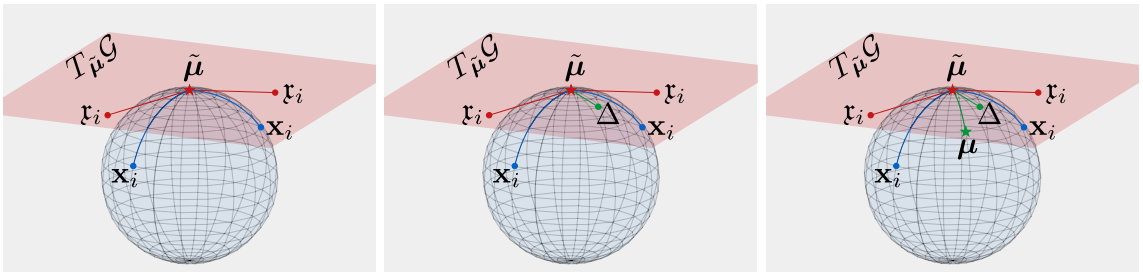


Figure 2.6: Schematic visualization of the iterative procedure of computing the mean $\boldsymbol{\mu}$ of a set of points $\mathbf{x}_i$ on a Lie group $\mathcal{G}$. The points $\mathbf{x}_i$ are projected onto the tangent space at $\tilde{\boldsymbol{\mu}}$ using the logarithmic mapping resulting 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.

Computing the mean and covariance on a Lie group

As stated in [44], 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 [44,46], 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 $\boldsymbol{\mu}$. 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 $\mathrm{Log}(\boldsymbol{\mu}^{-1}\mathbf{x}_i)$ 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 $\mathbf{H} \in \mathrm{SE}(3)$. This approach is, for example, taken in [48–50], 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 [51], 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 [44, 52], 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 $\mathrm{SO}(3) \times \mathbb{R}^3$. 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 $\mathrm{SO}(3) \times \mathbb{R}^3$ 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 $\mathrm{SO}(3) \times \mathbb{R}^3$ the rotational part and translational part are treated separately. As stated in [44, 46], 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 $\mathrm{SO}(3) \times \mathbb{R}^3$ instead of SE(3) as group structure to represent the pose. As a result, we will use $\mathrm{SO}(3) \times \mathbb{R}^3$ in our statistical framework to represent the pose of the box. In the remainder of this work, we will refer to this group as the *group of the poses*, defined as $\mathcal{P} \coloneqq \mathrm{SO}(3) \times \mathbb{R}^3$ with the corresponding Lie algebra denoted by $\mathfrak{p}$. The state of the box will be defined as a combination of the pose, the linear, and the angular velocity of the box. 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 $\mathcal{S} \coloneqq \mathrm{SO}(3) \times \mathbb{R}^3 \times \mathbb{R}^3 \times \mathbb{R}^3_\times$, 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 [6, Appendix B].

2.4 Multibody dynamics notation

In this section, the rigid-body dynamics notation of [40], which is employed in this report, is presented. Section 2.4.1 describes the notation of the position and orientation of a rigid body. The notation of rigid-body velocity is discussed in Section 2.4.2 and finally the wrench notation is given in Section 2.4.3.

2.4.1 Coordinate frames and points

The position and orientation of a rigid body in 3D space can be described by a *coordinate frame*. A coordinate frame, denoted by a capital letter, consists of an *origin* and an *orientation frame*. The origin and orientation frame of a coordinate frame A are written as $\mathbf{o}_A$ and $[A]$ respectively, such that we can write $A = (\mathbf{o}_A, [A])$.

In Figure 2.7, the most important coordinate frames in this report are shown. Frame A is called the camera sensor frame, as it is located near the plane containing the sensing elements of the camera. Frame A is especially important for the image formation. Frame B is assigned to the center of mass of the box and Frame C is the frame of the contact surface, which is in our case the conveyor belt. Frame F, located at the base of the UR10 robot, is used as the inertial frame, which means that all motions are expressed with respect to this frame. In Section 3.1.1, a description of the physical setup can be found.

The coordinates of a point $\mathbf{p}$ with respect to frame F can be written as the coordinate vector ${}^F\mathbf{p}$,

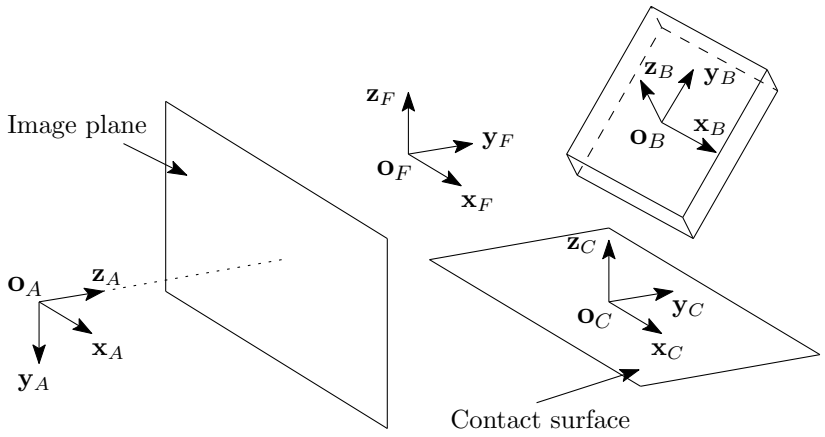


Figure 2.7: Illustration showing the most important frames in this report. Camera sensor frame A, the frame of the center of mass of the box B, the contact surface frame C, and the frame of the foundation of the robot F are shown. Adaptation from Jongeneel [6].

which represents the coordinates of the 3D geometric vector $\vec{r}_{\mathbf{o}_F,\mathbf{p}}$. This vector starts at the origin $\mathbf{o}_F$, points towards point $\mathbf{p}$, and is expressed in the orientation frame $[F]$. A formal notation of ${}^F\mathbf{p}$ is

$$ {}^F\mathbf{p} \coloneqq \begin{bmatrix} \vec{r}_{\mathbf{o}_F,\mathbf{p}} \cdot \vec{x}_F \\ \vec{r}_{\mathbf{o}_F,\mathbf{p}} \cdot \vec{y}_F \\ \vec{r}_{\mathbf{o}_F,\mathbf{p}} \cdot \vec{z}_F \end{bmatrix}, \tag{2.67} $$

where $\vec{x}_F$, $\vec{y}_F$, and $\vec{z}_F$ are mutually orthogonal unit vectors defining the orientation frame $[F]$ and $\cdot$ is the scalar product. Suppose we have two coordinate frames B and F with coinciding origins, then the coordinate transformation from B to F is described by the rotation matrix ${}^F\mathbf{R}_B \in \mathrm{SO}(3)$. When the two origins do not coincide, the position and orientation of frame B with respect to frame F is described by the 4×4 homogeneous transformation matrix ${}^F\mathbf{H}_B \in \mathrm{SE}(3)$ given by

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

Mapping a coordinate vector ${}^B\mathbf{p}$ to ${}^F\mathbf{p}$ can be done by making use of the homogeneous transformation matrix ${}^F\mathbf{H}_B$. Let ${}^B\bar{\mathbf{p}}$ and ${}^F\bar{\mathbf{p}}$ denote the homogeneous representation of ${}^B\mathbf{p}$ and ${}^F\mathbf{p}$, respectively. Mathematically, ${}^B\bar{\mathbf{p}}$ and ${}^F\bar{\mathbf{p}}$ are written as

$$ {}^B\bar{\mathbf{p}} \coloneqq \begin{bmatrix} {}^B\mathbf{p} \\ 1 \end{bmatrix} \quad \text{and} \quad {}^F\bar{\mathbf{p}} \coloneqq \begin{bmatrix} {}^F\mathbf{p} \\ 1 \end{bmatrix}, \tag{2.69} $$

from which follows that

$$ {}^F\bar{\mathbf{p}} = {}^F\mathbf{H}_B {}^B\bar{\mathbf{p}}. \tag{2.70} $$

Alternatively, we can also write

$$ {}^F\mathbf{p} = {}^F\mathbf{R}_B {}^B\mathbf{p} + {}^F\mathbf{o}_B. \tag{2.71} $$

2.4.2 Rigid-body velocity

By making use of the time derivatives of the rotation matrix and homogeneous transformation matrix we can derive expressions for the relative velocity between coordinate frames. The time derivative of the rotation matrix ${}^F\mathbf{R}_B$ is defined as

$$ {}^F\dot{\mathbf{R}}_B \coloneqq \frac{d}{dt}\left({}^F\mathbf{R}_B\right) \tag{2.72} $$

and the time derivative of the homogeneous transformation matrix ${}^{F}\mathbf{H}_{B}$ is defined as

$$ {}^{F}\dot{\mathbf{H}}_{B} \coloneqq \frac{d}{dt}\left({}^{F}\mathbf{H}_{B}\right) = \begin{bmatrix} {}^{F}\dot{\mathbf{R}}_{B} & {}^{F}\dot{\mathbf{o}}_{B} \\ \mathbf{0} & 0 \end{bmatrix}. \tag{2.73} $$

In (2.73), ${}^{F}\dot{\mathbf{o}}_{B}$ is the time derivative of ${}^{F}\mathbf{o}_{B}$, given by

$$ {}^{F}\dot{\mathbf{o}}_{B} \coloneqq \frac{d}{dt}\left({}^{F}\mathbf{o}_{B}\right). \tag{2.74} $$

In order to obtain a more compact representation of ${}^{F}\dot{\mathbf{H}}_{B}$, we pre- or post-multiply ${}^{F}\dot{\mathbf{H}}_{B}$ by ${}^{F}\mathbf{H}_{B}^{-1}$. This results in an element of $\mathfrak{se}(3)$ in both cases, which are called twists. Pre-multiplying leads to

$$ \begin{aligned} {}^{F}\mathbf{H}_{B}^{-1}{}^{F}\dot{\mathbf{H}}_{B} &= \begin{bmatrix} {}^{F}\mathbf{R}_{B}^{T} & -{}^{F}\mathbf{R}_{B}^{T}{}^{F}\mathbf{o}_{B} \\ \mathbf{0} & 1 \end{bmatrix} \begin{bmatrix} {}^{F}\dot{\mathbf{R}}_{B} & {}^{F}\dot{\mathbf{o}}_{B} \\ \mathbf{0} & 0 \end{bmatrix} \\ &= \begin{bmatrix} {}^{F}\mathbf{R}_{B}^{T}{}^{F}\dot{\mathbf{R}}_{B} & {}^{F}\mathbf{R}_{B}^{T}{}^{F}\dot{\mathbf{o}}_{B} \\ \mathbf{0} & 0 \end{bmatrix}, \end{aligned} \tag{2.75} $$

where the term ${}^{F}\mathbf{R}_{B}^{T}{}^{F}\dot{\mathbf{R}}_{B}$ is skew symmetric and the inverse ${}^{F}\mathbf{H}_{B}^{-1}$ is determined by making use of (2.56). As defined by (2.49), the $(\cdot)^{\wedge}$ hat-operator is used to write an element of $\mathbb{R}^3$ as a skew-symmetric matrix, so we can define

$$ {}^{B}\boldsymbol{v}_{F,B} \coloneqq {}^{F}\mathbf{R}_{B}^{T}{}^{F}\dot{\mathbf{o}}_{B}, \tag{2.76} $$

$$ {}^{B}\boldsymbol{\omega}_{F,B}^{\wedge} \coloneqq {}^{F}\mathbf{R}_{B}^{T}{}^{F}\dot{\mathbf{R}}_{B}. \tag{2.77} $$

Combining these two equations leads to the *left trivialized velocity* of frame B with respect to frame F, given by

$$ {}^{B}\mathbf{v}_{F,B} \coloneqq \begin{bmatrix} {}^{B}\boldsymbol{v}_{F,B} \\ {}^{B}\boldsymbol{\omega}_{F,B} \end{bmatrix} \in \mathbb{R}^{6}. \tag{2.78} $$

To obtain the *right trivialized velocity*, we post-multiply ${}^{F}\dot{\mathbf{H}}_{B}$ with ${}^{F}\mathbf{H}_{B}^{-1}$, resulting in

$$ \begin{aligned} {}^{F}\dot{\mathbf{H}}_{B}{}^{F}\mathbf{H}_{B}^{-1} &= \begin{bmatrix} {}^{F}\dot{\mathbf{R}}_{B} & {}^{F}\dot{\mathbf{o}}_{B} \\ \mathbf{0} & 0 \end{bmatrix} \begin{bmatrix} {}^{F}\mathbf{R}_{B}^{T} & -{}^{F}\mathbf{R}_{B}^{T}{}^{F}\mathbf{o}_{B} \\ \mathbf{0} & 1 \end{bmatrix} \\ &= \begin{bmatrix} {}^{F}\dot{\mathbf{R}}_{B}{}^{F}\mathbf{R}_{B}^{T} & {}^{F}\dot{\mathbf{o}}_{B} - {}^{F}\dot{\mathbf{R}}_{B}{}^{F}\mathbf{R}_{B}^{T}{}^{F}\mathbf{o}_{B} \\ \mathbf{0} & 0 \end{bmatrix}. \end{aligned} \tag{2.79} $$

From this we can define

$$ {}^{F}\boldsymbol{v}_{F,B} \coloneqq {}^{F}\dot{\mathbf{o}}_{B} - {}^{F}\dot{\mathbf{R}}_{B}{}^{F}\mathbf{R}_{B}^{T}{}^{F}\mathbf{o}_{B}, \tag{2.80} $$

$$ {}^{F}\boldsymbol{\omega}_{F,B}^{\wedge} \coloneqq {}^{F}\dot{\mathbf{R}}_{B}{}^{F}\mathbf{R}_{B}^{T}, \tag{2.81} $$

which eventually leads to the right trivialized velocity, given by

$$ {}^{F}\mathbf{v}_{F,B} \coloneqq \begin{bmatrix} {}^{F}\boldsymbol{v}_{F,B} \\ {}^{F}\boldsymbol{\omega}_{F,B} \end{bmatrix} \in \mathbb{R}^{6}. \tag{2.82} $$

The left and right trivialized velocity are related through

$$ {}^{F}\mathbf{v}_{F,B} = {}^{F}\mathbf{X}_{B}{}^{B}\mathbf{v}_{F,B}, \tag{2.83} $$

where ${}^{F}\mathbf{X}_{B}$ is called the adjoint map [53], which is defined as

$$ {}^{F}\mathbf{X}_{B} \coloneqq \begin{bmatrix} {}^{F}\mathbf{R}_{B} & {}^{F}\mathbf{o}_{B}^{\wedge}{}^{F}\mathbf{R}_{B} \\ \mathbf{0} & {}^{F}\mathbf{R}_{B} \end{bmatrix} \in \mathbb{R}^{6\times 6}. \tag{2.84} $$

2.4.3 Wrench notation

The forces and torques applied to a rigid body can be combined into a *wrench*, which is defined as

$$ {}_B\mathbf{f} \coloneqq \begin{bmatrix} {}_B\boldsymbol{f} \\ {}_B\boldsymbol{\tau} \end{bmatrix} \in \mathbb{R}^6. \tag{2.85} $$

The coordinates of a wrench can be changed from frame B to frame F by making use of the mapping ${}_F\mathbf{X}^B$, resulting in

$$ {}_F\mathbf{f} = {}_F\mathbf{X}^B {}_B\mathbf{f}. \tag{2.86} $$

The mapping ${}_F\mathbf{X}^B$ is closely related to the adjoint map given in (2.84). In fact, we only have to compute the transpose of ${}^B\mathbf{X}_F$, so we obtain

$$ {}_F\mathbf{X}^B \coloneqq {}^B\mathbf{X}_F^T = \begin{bmatrix} {}^F\mathbf{R}_B & \mathbf{0} \\ {}_F\mathbf{o}_B^\wedge {}^F\mathbf{R}_B & {}^F\mathbf{R}_B \end{bmatrix} = \begin{bmatrix} {}^F\mathbf{R}_B & \mathbf{0} \\ -{}^F\mathbf{R}_B {}^B\mathbf{o}_F^\wedge & {}^F\mathbf{R}_B \end{bmatrix}. \tag{2.87} $$

It is also possible to express a wrench with respect to frame B with the orientation in terms of frame F. This is very useful for forces that are independent of time with respect to a specific frame, for instance the gravity force with respect to the inertial frame F. To this end, a new frame is created by combining frame B and frame F, written as $B[F] \coloneqq (\mathbf{o}_B, [F])$. In other words, the origin of this new frame coincides with the origin of B and the orientation coincides with the orientation of F. The wrench is then written as ${}_{B[F]}\mathbf{f}$ and its relation with ${}_B\mathbf{f}$ is given by

$$ {}_B\mathbf{f} = {}_B\mathbf{X}^{B[F]} {}_{B[F]}\mathbf{f}. \tag{2.88} $$

In (2.88), ${}_B\mathbf{X}^{B[F]}$ is given by

$$ {}_B\mathbf{X}^{B[F]} = \begin{bmatrix} {}^F\mathbf{R}_B^T & \mathbf{0} \\ \mathbf{0} & {}_F\mathbf{R}_B^T \end{bmatrix}. \tag{2.89} $$

The time derivative of the wrench coordinate transformation ${}_F\mathbf{X}^B$ is defined as

$$ {}_F\dot{\mathbf{X}}^B \coloneqq {}_F\mathbf{X}^B {}_B\mathbf{v}_{F,B} \bar{\times}^*, \tag{2.90} $$

where $\bar{\times}^*$ is called the dual cross-product, whose matrix representation is

$$ {}_B\mathbf{v}_{F,B} \bar{\times}^* \coloneqq \begin{bmatrix} {}_B\boldsymbol{\omega}_{F,B}^\wedge & \mathbf{0} \\ {}_B\boldsymbol{v}_{F,B}^\wedge & {}_B\boldsymbol{\omega}_{F,B}^\wedge \end{bmatrix}. \tag{2.91} $$

2.5 Image formation

In this section, the process behind the image formation is discussed. Section 2.5.1 explains the pinhole camera principle and the perspective projection equations are given. With these equations, we can determine the visible faces of the box in the image, which is discussed in Section 2.5.2.

2.5.1 Perspective projection

The image formation process can be modeled using the pinhole camera principle [54]. Instead of a lens, a small aperture is used through which the light rays enter. At a distance f behind the pinhole an inverted image is projected onto the image plane, where f is the *focal length* of the camera. We will not consider the real image plane, but the *virtual image plane* at a distance f in front of the pinhole. The virtual image plane contains the upright image instead of the inverted image.

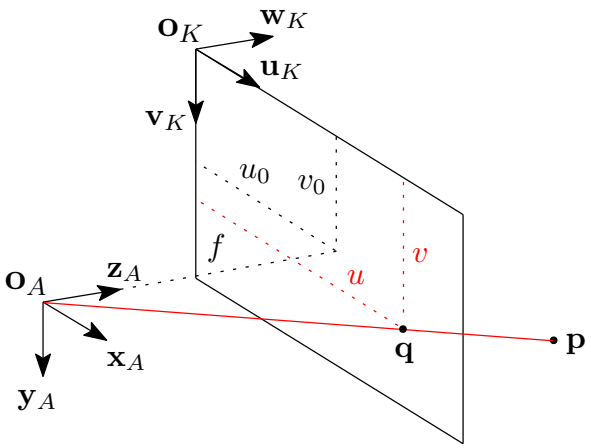


Figure 2.8: Schematic representation of a point $\mathbf{p}$ being projected onto the virtual image plane, resulting in point $\mathbf{q}$. Adaptation from Jongeneel [6].

It is assumed that the image plane contains the photo sensors of the camera. *Pixel coordinates* are often used to describe the position of points on the image plane. In Figure 2.8, the image plane is shown with in its upper left corner the origin of the image coordinate frame K. The z-axis of camera sensor frame A is called the *principal axis* and the point where the principal axis intersects the image plane is called the *principal point,* which is denoted by (u_0, v_0). The plane that is spanned by the x- and y-axis of frame A is parallel to the image plane.

Consider point $\mathbf{p}$ in Figure 2.8, which is being projected onto the image plane, resulting in point $\mathbf{q}$. Then ${}^K\mathbf{q} = \begin{bmatrix} {}^K\mathbf{q}_u & {}^K\mathbf{q}_v & {}^K\mathbf{q}_w \end{bmatrix}^T$ represents the coordinates of $\mathbf{q}$ with respect to frame K, in terms of pixels. We can now write

$$ {}^K\mathbf{q}_u\mathbf{u}_K + {}^K\mathbf{q}_v\mathbf{v}_K \coloneqq u\mathbf{u}_K + v\mathbf{v}_K, \tag{2.92} $$

which is the vector from $\mathbf{o}_K$ to $\mathbf{q}$, in terms of distance. In the remainder of this report, we will use (u, v) instead of $({}^K\mathbf{q}_u, {}^K\mathbf{q}_v)$ to denote the pixel coordinates.

The pixel coordinates of point ${}^A\mathbf{p}$ with coordinates $\begin{bmatrix} {}^A\mathbf{p}_x & {}^A\mathbf{p}_y & {}^A\mathbf{p}_z \end{bmatrix}^T$ can be determined with

$$ {}^A\mathbf{p}_z \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \mathbf{K} \begin{bmatrix} {}^A\mathbf{p}_x \\ {}^A\mathbf{p}_y \\ {}^A\mathbf{p}_z \end{bmatrix}, \tag{2.93} $$

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

$$ \mathbf{K} = \begin{bmatrix} \alpha_x & s & u_0 \\ 0 & \alpha_y & v_0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{2.94} $$

In this matrix $\mathbf{K}$, α_x and α_y represent the focal length in terms of pixels, which can be calculated with $\alpha_x = fs_x$ and $\alpha_y = fs_y$, where s_x and s_y are the number of pixels per world unit, often millimeters. Moreover, s is the skew parameter, which is equal to zero when the image axes are perpendicular. This is the case for almost all cameras.

2.5.2 Projection of the box onto the image plane

The projection of the box onto the image plane is important for the computation of the approximate likelihood, which is discussed in Section 3.3. The visible faces of the box with respect to the camera can be determined by making use of the normal vector of each face pointing to the outside of the box,

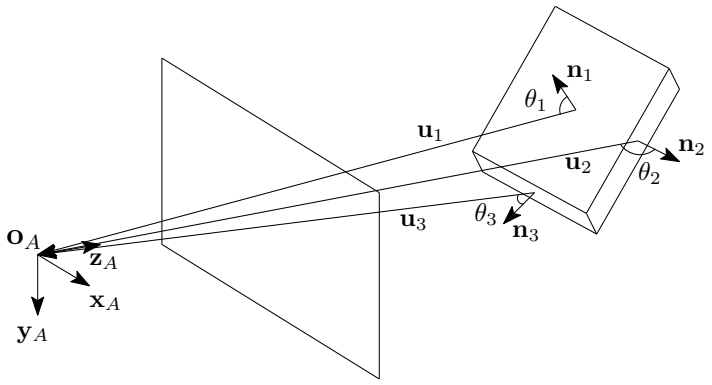


Figure 2.9: Schematic representation of vectors $\mathbf{n}_i$ and $\mathbf{u}_i$, which are used to determine the visible faces of the box with respect to the camera. Image taken from Jongeneel [6].

$\mathbf{n}_i$, and the vector $\mathbf{u}_i$, that starts at the center of each face and points towards frame A. In Figure 2.9, the vectors $\mathbf{n}_i$ and $\mathbf{u}_i$ are depicted for a specific pose of the box. When the angle θ_i between $\mathbf{n}_i$ and $\mathbf{u}_i$ is smaller than $\pm\frac{1}{2}\pi$, the face is visible for the camera. To compute θ_i, we use

$$\cos(\theta_i) = \frac{(\mathbf{n}_i \cdot \mathbf{u}_i)}{(\|\mathbf{n}_i\| \cdot \|\mathbf{u}_i\|)}, \tag{2.95}$$

where $\|\cdot\|$ indicates the Euclidean norm of the vector and $i = \{1, \ldots, 6\}$, as there are six faces of the box. The number of visible faces N_f can at most be three faces.

2.6 Conclusion

The necessary background information for visual object tracking with collision models is discussed in this chapter. In Chapter 3, we will use this information to explain the likelihood computation of particles in RGB images. Before this is done, first the camera and the other components of the experimental setup are introduced. Furthermore, we will elaborate on the nonsmooth motion model, which is used to propagate the state of a colliding box as a function of time. For the likelihood computation and the motion model a geometric model of the box is needed. This geometric model is also discussed in Chapter 3.

Chapter 3

Geometric and dynamic model of boxes

In Chapter 2, the unscented particle filter for the *Euclidean* case is introduced. In [6], the UPF has been generalized in order to be able to cope with the Lie group structure of the parametrization of the state in $\mathcal{S}$, which is called the *Geometric Unscented Particle Filter* (GUPF). The generalization of the UPF into the GUPF is not repeated again in this report, so the reader is referred to [6, Chapter 3] for an elaborate explanation regarding this subject. In the remainder of this report, we use the GUPF as the object tracking algorithm to estimate the state of the box over time.

In this chapter, the same subjects as in [6, Chapter 4] are discussed, however instead of using a cuboid in simulation, we now make use of real boxes and real images. Section 3.1 first mentions the components of the experimental setup that are used to obtain RGB images and the ground truth pose of the relevant objects. Furthermore, subjects related to the processing of the data obtained with the recording instruments are discussed. Then, in Section 3.2, the *geometric model* of the boxes is given, which is used in the *likelihood function* and the *motion model*. The likelihood function and the motion model are explained in Section 3.3 and 3.4, respectively.

3.1 Experimental setup and data processing

Before we can apply and evaluate the performance of the likelihood function and the GUPF, first RGB images and the ground truth data have to be obtained. Section 3.1.1 introduces all elements of the test setup that are used during this research to perform these tasks. In Section 3.1.2, the focus is more on the recording instruments and their specifications. By making use of the ground truth data, camera sensor frame A can be estimated, which is discussed in Section 3.1.3. In the end, Section 3.1.4 explains the process of synchronizing the RGB images and ground truth data.

3.1.1 Hardware elements

The main goal of this research is to validate the GUPF on real-life videos instead of on synthetic videos. To this end, we consider the scenario where boxes are being tossed on a conveyor belt. The execution of the tossing motion and the flight of the box are recorded with a RGB camera and the ground truth pose of the relevant objects is obtained by making use of a motion capture system. In Figure 3.1, the complete test setup can be seen.

The conveyor consists of a rubber belt and two drums, of which one is powered to move the belt. To make a fair comparison between the results in the work of Jongeneel [6] and a real-life scenario, the conveyor belt will not move during the movement of the box. On the left side next to the conveyor in Figure 3.1, a robotic manipulator is shown, called the UR10 robot. The robot itself is not used during this research, however the blue tool located at its base is of great importance. As discussed in Section 2.4, all the coordinate frames of the relevant objects are expressed with respect to frame F, which is defined by this blue tool with its corresponding set of OptiTrack markers. Because the tool is fixed to the workbench on which the UR10 robot is mounted, frame F serves perfectly as the inertial frame.

In order to obtain the ground truth pose of an object, an OptiTrack motion capture system [55]

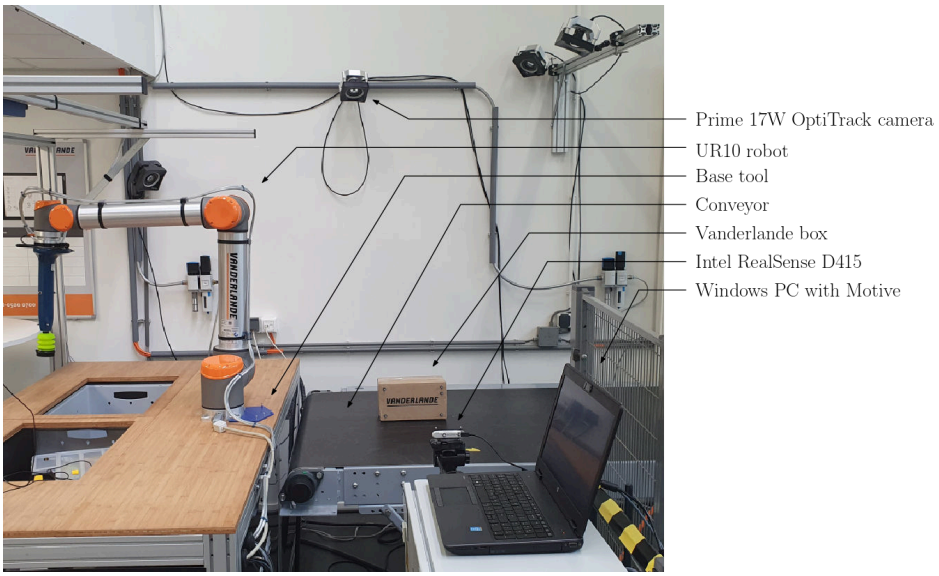


Figure 3.1: Test setup located at the Innovation Lab of Vanderlande on TU/e campus.

is employed. The OptiTrack motion capture system consists of several Prime 17W cameras, its associated software platform called Motive, and reflective markers. A pattern of reflective markers is placed on an object and the Prime 17W cameras are able to detect the markers. After combining all the 2D images of the individual cameras, the position of the markers in 3D space is obtained, which is used to derive the ground truth pose of the object. Because the motion capture system estimates the pose with a sub-millimeter accuracy, the OptiTrack data is considered to be the ground truth data. Conclusions regarding the performance of the state estimation by the GUPF can be drawn by comparing the results with the ground truth data.

We use the Intel RealSense D415 camera [56] to capture single RGB images of stationary boxes and to record videos of boxes that collide with the conveyor belt. Even though the RealSense D415 camera contains depth technology, we only make use of the RGB information. In Section 3.1.2, the OptiTrack motion capture system and the RealSense D415 camera are explained in more detail.

3.1.2 Recording instruments

The OptiTrack motion capture system and the RealSense D415 camera have been introduced shortly in Section 3.1.1. In this section, these two recording instruments are discussed more elaborately.

OptiTrack's motion capture system

In this research, we use the OptiTrack motion capture system to obtain the ground truth pose of the three boxes given in Section 3.2 and the RealSense D415 camera. The supporting software of the motion capture system is called OptiTrack's Motive and runs on a Windows computer.

Before the pose can be estimated, OptiTrack markers should be placed on the objects. The Prime 17W cameras, shown in Figure 3.2, are able to detect these markers. More specifically, the Prime 17W cameras emit infrared light with their 20 high-intensity IR LEDS, which is reflected by the markers. The reflected IR light is recorded at a resolution of 1.7 MP and with a FOV (field of view) of 70° [57]. Two different types of OptiTrack markers exist, namely active markers and passive markers. We make use of the passive markers, which reflect the light emitted by the Prime 17W cameras, in contrast to the active markers that emit light themselves. For the passive markers, the choice needs to be made between the flat or spherical variants. Flat markers would be ideal for tumbling boxes, however often a loss of tracking is observed when the view angle of the markers with respect

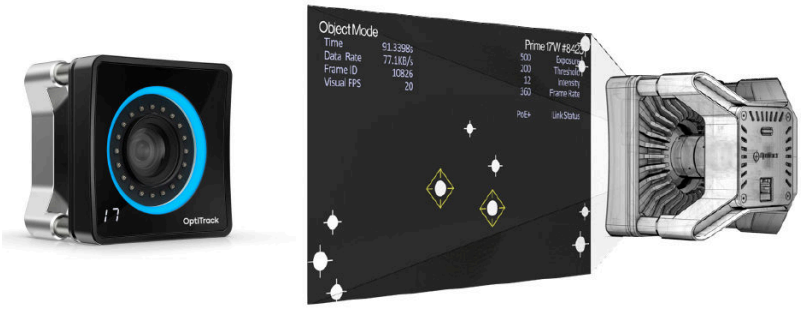


Figure 3.2: Prime 17W OptiTrack camera, which records the reflected IR light from the markers at a resolution of 1.7 MP, with a FOV of 70°, and a frame rate of 360 Hz [57]. Image taken from Poort [7].

to the OptiTrack cameras changes. Therefore, spherical markers are placed on both the boxes and RealSense D415 camera.

By pressing the record button in Motive, the Prime 17W cameras start capturing 2D images with a frame rate of 360 Hz. When a marker is identified in the image, its position in 3D space with respect to the camera is determined. In Motive, a visible marker is indicated by a ray starting at the corresponding camera and ending at the location of the marker. A 3D position is only assigned to a marker when multiple rays cross in space, meaning that the marker is visible for more than one camera. In order to achieve an accurate position estimation of the marker, it is important to know the relative pose of the cameras with respect to each other. The relative pose is determined during the calibration of the cameras, where the pose of each camera is expressed with respect to the hand-placed calibration square. Usually, the 3D position of the markers are determined with sub-millimeter accuracy, depending on the quality of the calibration.

In Motive, rigid bodies can be created from sets of at least four markers. It is necessary to place the markers in an asymmetric pattern on the objects, to have a clear distinction of orientations. After a

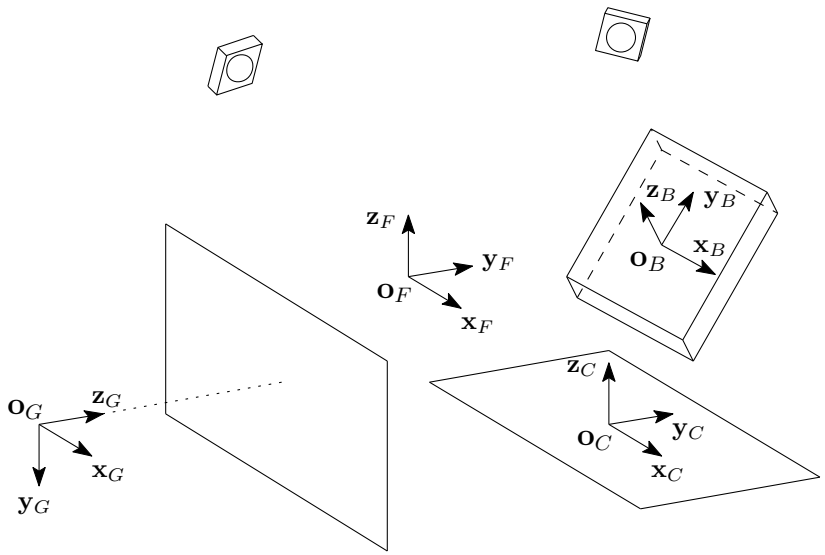


Figure 3.3: Rigid bodies defined in Motive, where frames F, B, and C are the same as in Figure 2.7, however now expressed with respect to the calibration square, whose frame is equal to frame C. Note that frame A has been changed to frame G. Only two OptiTrack cameras are shown in this figure for clarity.

rigid body has been created, also a coordinate frame is created automatically, which is initially located at the geometric center of the pattern of markers with the orientation of the calibration square. The coordinate frame of the rigid body can be placed at any desired location with any orientation. In Figure 3.3, the defined frames in Motive are shown. The advantage of a rigid body in Motive is that not all markers have to be visible for the cameras to recognize and track the object. If three markers are visible, the marker-based or ray-based algorithm fit the rigid body to the marker positions. The marker positions and poses associated to the tracked rigid bodies are saved to a .tak file, and to be able to process the results, the .tak file is exported to a .csv file.

Intel RealSense D415 camera

The Intel RealSense D415 camera, shown in Figure 3.4, is used to obtain single RGB images of stationary boxes and videos of tumbling boxes. Before any photos or videos are made, the camera needs to be calibrated, which is done with the MATLAB Camera Calibrator app [58]. A checkerboard pattern is held in front of the camera at different distances and with different angles. The Camera Calibrator app detects the corners of the checkerboard squares in all the images and performs the calibration by making use of the method described in [59]. As a result, we obtain the intrinsic, extrinsic, and lens distortion parameters of the camera.

The camera intrinsic matrix $\mathbf{K}$ is already introduced in Section 2.5.1 and for the RealSense D415 camera it is found to be

$$\mathbf{K} = \begin{bmatrix} 1390.69 & 0 & 938.52 \\ 0 & 1386.36 & 546.73 \\ 0 & 0 & 1 \end{bmatrix}. \tag{3.1}$$

It has to be mentioned that a resolution of 1920×1080 has been used for the images of the checkerboard during the calibration. When a different resolution is used, the camera intrinsic matrix $\mathbf{K}$ changes. It is not necessary to recalibrate the camera, as the new matrix $\mathbf{K}$ can be determined easily with

$$\mathbf{K} = \begin{bmatrix} \frac{res_u}{1920} & 0 & 0 \\ 0 & \frac{res_v}{1080} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1390.69 & 0 & 938.52 \\ 0 & 1386.36 & 546.73 \\ 0 & 0 & 1 \end{bmatrix}, \tag{3.2}$$

where res_u and res_v are the new resolutions in the $\mathbf{u}_k$ and $\mathbf{v}_k$ direction, respectively.

Every image captured with a camera has imperfections, which are caused by lens distortion. Two types of distortion are radial distortion and tangential distortion. Radial distortion occurs due to the fact that light rays bend more near the edge of a lens and tangential distortion occurs when the image plane and lens are not parallel [60]. We will only discuss radial distortion, as it is assumed that the image plane and lens are exactly parallel. The undistorted *normalized* image coordinates can be determined with

$$x_u = x_d\big(1 + k_1 p^2 + k_2 p^4\big) \tag{3.3}$$

and

$$y_u = y_d\big(1 + k_1 p^2 + k_2 p^4\big), \tag{3.4}$$

Figure 3.4: Intel RealSense D415 camera, which is used to obtain RGB videos of tumbling boxes.

where k_1 and k_2 are the radial distortion parameters, x_d and y_d are the distorted normalized image coordinates, and $p = \sqrt{x_d^2 + y_d^2}$. The conversion from normalized image coordinates (x, y) to pixel coordinates (u, v) is given by

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \alpha_x x + u_0 \\ \alpha_y y + v_0 \end{bmatrix}. \tag{3.5}$$

The radial distortion parameters that follow from the calibration are $k_1 = 0.0567$ and $k_2 = -0.1050$, however the influence of the radial distortion on the image is considered small enough that it is neglected in the remainder of this research. The results for the extrinsic parameters computed by the MATLAB Camera Calibrator app are also not taken into account, because the pose of the camera has been changed after the calibration.

In order to be able to capture images and to record videos with the RealSense D415 camera, a software tool called RealSense Viewer is installed on the computer that is connected to the camera via a USB cable. Different settings of the camera can be adjusted through the RealSense Viewer, such as the resolution, the frame rate, and the exposure time. The highest possible resolution and frame rate of the camera are 1920×1080 and 60 Hz, but these settings cannot be used simultaneously while recording, due to hardware limitations. When using a frame rate of 60 Hz, the resolution is automatically reduced to 640×480. As we want to record in a higher resolution, the second highest frame rate of 30 Hz is chosen, which can be used in combination with every resolution. Capturing images with a frame rate of 30 Hz unfortunately also has a downside, which is the occurrence of frame drops throughout a recording. The reason for this is that the computer is connected to the motion capture system and the RealSense D415 camera at the same time, hence computationally overloaded. Especially for a resolution of 1920×1080, the computer is not able to keep up with the frame rate of the camera. Lowering the resolution to 1280×720 reduces the occurrence of frame drops significantly, so this is the resolution that will be used during the recording of the RGB videos. Another solution to reduce frame drops is using a frame rate of 15 Hz; however, this would not be beneficial for the computational time of the GUPF, as the time between frames increases. In Section 4.3.1, the influence of several parameters on the computational time of the GUPF is discussed.

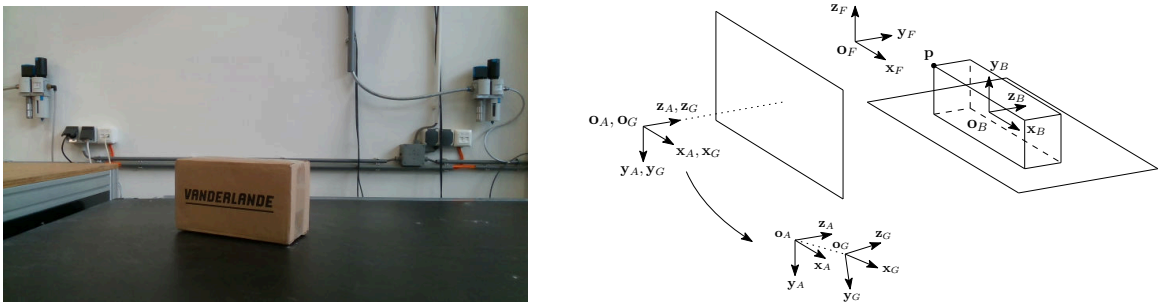
When a video is recorded with the RealSense Viewer, not only images, but also the frame rate and timestamps of the frames are saved. All this information is stored in a .bag file, which can be opened in MATLAB.

3.1.3 Projection optimization

As mentioned in Section 3.1.2, we place OptiTrack markers on the RealSense D415 camera to determine its absolute position and orientation. In Motive, a rigid body is created for the camera and its coordinate frame is denoted by the letter G, which is also depicted in Figure 3.3. Because this frame is defined by the markers that are placed on the casing of the camera, frame G is called the camera casing frame. The origin $\mathbf{o}_G$ of the frame is placed at the location where the camera sensor is thought to be, with the same orientation as frame A. Important to note is that it is impossible to have a frame G that exactly corresponds to frame A, as the true location and orientation of frame A are unknown.

Consider the image of a box in Figure 3.5a and its schematic representation containing the involved coordinate frames in Figure 3.5b. In the schematic representation, a visualization of the difference between frame A and frame G is given, where the difference is exaggerated for clarity. It can be seen that frame G has a slightly different position and orientation in comparison to frame A. Even though the difference is small between the coordinate frames, the projection of the box onto the image plane is influenced heavily.

The pose of the box and the camera are expressed in frame F by default, so from the motion capture system we obtain the homogeneous transformation matrices ${}^F\mathbf{H}_B$ and ${}^F\mathbf{H}_G$. Suppose we express the coordinates of the vertices of the box with respect to camera casing frame G with ${}^G\bar{\mathbf{p}}_i = {}^G\mathbf{H}_F {}^F\bar{\mathbf{p}}_i$



(a) **(b)**

Figure 3.5: Image of a box **(a)** and its schematic representation containing the involved coordinate frames **(b)**. The difference between frame A and frame G is exaggerated.

and afterwards project the vertices onto the image plane with

$$\begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} = {}^{G}\mathbf{p}_{i,z}^{-1}\mathbf{K} \begin{bmatrix} {}^{G}\mathbf{p}_{i,x} \\ {}^{G}\mathbf{p}_{i,y} \\ {}^{G}\mathbf{p}_{i,z} \end{bmatrix}, \tag{3.6}$$

then, after connecting the projected vertices, the result in Figure 3.6a is obtained. It can clearly be seen that the projected contour does not correspond to the true contour of the box. This shows the importance of using camera sensor frame A for the projection of points and contours onto the image plane. However, since the pose of frame A is unknown, it is necessary to come up with a method that estimates the position and orientation of this frame.

To tackle this problem, we will employ an optimization procedure. Through this optimization procedure, the homogeneous transformation matrix ${}^{A}\mathbf{H}_{G}$ is obtained, for which the difference between the calculated pixel coordinates (u, v) and the true pixel coordinates (u_{true}, v_{true}) of the vertices of the box is minimized. The true pixel coordinates of the vertices of the box are determined by hand in 30 different images to have more than enough points for the optimization. When this is done, the true pixel coordinates of each vertex i are converted into normalized image coordinates by making use of a rewritten form of (3.5), given by

$$\begin{aligned} x_{true_i} &= \frac{u_{true,i} - u_0}{\alpha_x}, \\ y_{true_i} &= \frac{v_{true,i} - v_0}{\alpha_y}, \end{aligned} \tag{3.7}$$

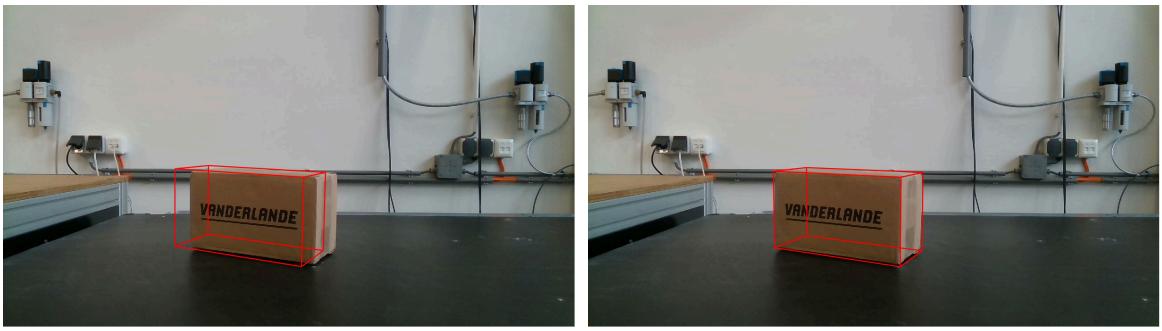
which are combined in the vector $\boldsymbol{x}_{true,i} = \begin{bmatrix} x_{true,i} & y_{true,i} \end{bmatrix}^T$. Also the coordinates of these vertices with respect to frame G are determined, denoted by ${}^{G}\mathbf{p}_i$.

Before the optimization process can be started, a parametrization for the rotation must be chosen. We choose to make use of the Euler vector parametrization. As mentioned in Section 2.2.2, the vector of exponential coordinates is given by $\boldsymbol{\xi} = \theta\mathbf{n}$, where θ is the rotation angle and $\mathbf{n} \in \mathbb{R}^3$ a unit vector along the rotation axis. Applying the hat-operator to $\boldsymbol{\xi}$ results in (2.49), with $\boldsymbol{\xi} = \begin{bmatrix} \xi_x & \xi_y & \xi_z \end{bmatrix}^T \in \mathbb{R}^3$. To obtain a rotation matrix, Rodrigues' formula is used, leading to

$$\exp(\boldsymbol{\xi}^{\wedge}) = \mathbf{I}_3 + \frac{\sin\left(\|\boldsymbol{\xi}\|\right)}{\|\boldsymbol{\xi}\|}\boldsymbol{\xi}^{\wedge} + \frac{\left(1 - \cos\left(\|\boldsymbol{\xi}\|\right)\right)}{\|\boldsymbol{\xi}\|^2}\left(\boldsymbol{\xi}^{\wedge}\right)^2, \tag{3.8}$$

where it holds that $\exp(\boldsymbol{\xi}^{\wedge}) = \mathbf{R}$. To express everything with respect to frame A, we rename $\boldsymbol{\xi}$ to ${}^{A}\boldsymbol{\xi}_{G}$, so that $\exp({}^{A}\boldsymbol{\xi}_{G}) = {}^{A}\mathbf{R}_{G}$.

Now, the optimization can be performed. We are dealing with a multiobjective, unconstrained optimization problem, so the `lsqnonlin` (nonlinear least squares) function with the trust-region-reflective



(a) **(b)**

Figure 3.6: Projection of the contour of the box onto the image plane before optimization **(a)** and after optimization **(b)**.

optimization algorithm from the MATLAB Optimization Toolbox can be used. The nonlinear least squares problem is defined as

$$\min_{{}^{A}\boldsymbol{\xi}_G, {}^{A}\mathbf{o}_G} \sum_{i=1}^{m} ||P({}^{A}\mathbf{R}_G {}^{G}\mathbf{p}_i + {}^{A}\mathbf{o}_G) - \boldsymbol{x}_{true_i}||^2, \tag{3.9}$$

where

$$P({}^{A}\mathbf{p}_i) = \begin{bmatrix} {}^{A}\mathbf{p}_{i,x}/{}^{A}\mathbf{p}_{i,z} \\ {}^{A}\mathbf{p}_{i,y}/{}^{A}\mathbf{p}_{i,z} \end{bmatrix} = \begin{bmatrix} x_i \\ y_i \end{bmatrix} \tag{3.10}$$

and m the total number of vertices. The function returns the minimizer $\mathbf{x}^* = \begin{bmatrix} {}^{A}\boldsymbol{\xi}_G^* & {}^{A}\mathbf{o}_G^* \end{bmatrix}$ for which the reprojection errors are minimized and the homogeneous transformation ${}^{A}\mathbf{H}_G$ is then given by

$${}^{A}\mathbf{H}_G = \begin{bmatrix} {}^{A}\mathbf{R}_G & {}^{A}\mathbf{o}_G \\ \mathbf{0} & 1 \end{bmatrix}. \tag{3.11}$$

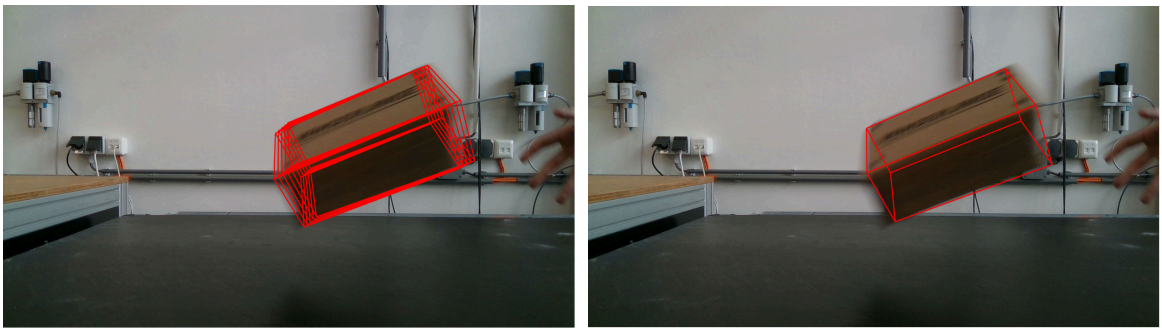
If we now express the coordinates of the vertices of the box with respect to frame A with ${}^{A}\bar{\mathbf{p}}_i = {}^{A}\mathbf{H}_G {}^{G}\bar{\mathbf{p}}_i$ and again project the vertices onto the image plane by making use of (3.6), the contour of the projected box almost completely overlaps the contour of the true box, as can be seen in Figure 3.6b.

It is important that the estimation of ${}^{A}\mathbf{H}_G$, and thus the projection of objects onto the image plane, is as accurate as possible, since it will be used in the synchronization of the MoCap data and RGB images in Section 3.1.4 and the performance evaluation of the approximate likelihood computation in Section 4.1.

3.1.4 Synchronization MoCap data and RGB images

By making use of the RealSense Viewer tool, we can capture images and make videos with the RealSense D415 camera. To be able to draw a conclusion regarding the performance of the likelihood computation and state estimation by the GUPF, which will be discussed in Section 4.1 and 4.3, we need to know the representation of the ground truth pose in the image. The RealSenseD415 camera and OptiTrack motion capture system both have an internal clock, which means that the data of these two sensors has to be aligned in time. This is referred to as the synchronization of the MoCap data and camera images. Two different scenarios are discussed in this section, which are determining the ground truth pose of a stationary box in a single image, and determining the ground truth pose of a moving box in multiple subsequent images.

To obtain the ground truth pose of a stationary box in a single image, the following chronological steps are taken:



(a) **(b)**

Figure 3.7: Box in free flight with several projected contours following from different MoCap data points **(a)** and the box with its corresponding ground truth contour after synchronization **(b)**.

1. Place the box on the conveyor belt and capture an image with the RealSense D415 camera.
2. Without moving the box, start a recording with the OptiTrack motion capture system and end the recording after a short amount of time.

The box may not move after capturing an image with the RealSense D415 camera, otherwise the ground truth data does not correspond to the box in the image. It is impossible to record only one frame with the motion capture system, therefore a very short recording is done. The length of this recording has no influence on the obtained ground truth data, as the homogeneous transformation matrix ${}^{F}\mathbf{H}_{B}$ is the same at each time instant. Because the box does not move, we can simply take the mean of ${}^{F}\mathbf{H}_{B}$ to obtain the ground truth pose. Expressing the pose of the box in the image with respect to camera sensor frame A is done with ${}^{A}\mathbf{H}_{B} = {}^{A}\mathbf{H}_{F}{}^{F}\mathbf{H}_{B}$, where ${}^{A}\mathbf{H}_{F} = {}^{A}\mathbf{H}_{G}{}^{G}\mathbf{H}_{F}$.

As we want to validate the GUPF on images containing boxes that collide with the conveyor belt, we have to record the movement of the boxes with the RealSense D415 camera and motion capture system simultaneously. The following chronological steps are taken before the synchronization can be performed:

1. Start a recording with the RealSense D415 and the OptiTrack motion capture system.
2. Toss the box by hand on the conveyor belt, while making sure that the box does not land on the OptiTrack markers.
3. Stop the with the RealSense D415 and the OptiTrack motion capture system.

The boxes should be tossed carefully to prevent that the OptiTrack markers make contact with the conveyor belt. When the box lands on the markers, the dynamics are influenced and the markers may get damaged. Given the RGB images of the colliding box, the task is now to find the corresponding true poses, which can be retrieved from the MoCap data. At each MoCap data point the transformation matrix ${}^{F}\mathbf{H}_{B}$ is given, from which the vertices of the box with respect to camera sensor frame A can be determined with ${}^{A}\bar{\mathbf{p}}_{i} = {}^{A}\mathbf{H}_{F}{}^{F}\bar{\mathbf{p}}_{i}$. Similar to Section 3.1.3, we project the contours following from each MoCap data point onto the image plane and we save the index of the data point for which the projected contour overlaps the true contour of the box in the image. In Figure 3.7a, an image of a box in free flight with several projected contours can be seen. Figure 3.7b shows the contour that corresponds the most to the true box. This is done for all images, such that we know the ground truth pose of the box in each image of a recording.

Since the synchronization is performed manually, it may be possible that the determined ground truth pose differs a couple of millimeters from the true ground truth pose for a specific image. In future work, it is better to let the software do the synchronization automatically.

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

Figure 3.8: Box 3 **(a)**, Box 4 **(b)**, and Box 5 **(c)**, provided by Vanderlande. These boxes are representative for boxes in the parcel industry.

Table 3.1: Properties of Box 3, Box 4, and Box 5.

Parameter	**Unit**	**Box 3**	**Box 4**	**Box 5**
Dimensions (l×w×h)	mm	165×143×117	198×93×129	272×155×114
Mass	kg	0.316	0.297	0.566
Inertia $\{I_{xx}, I_{yy}, I_{zz}\}$	kg mm^2	{899, 1077, 1255}	{626, 1382, 1184}	{1746, 4103, 4623}

3.2 Geometric model of the boxes

In this research, we make use of three boxes with different properties. If the GUPF is able to estimate the state of these three boxes accurately in all frames of a recording, it is assumed that the algorithm works for all boxes with similar properties. The boxes are called Box 3, Box 4, and Box 5 and are shown in Figure 3.8. Their properties, i.e., shape, dimensions, mass, and inertia, are given in Table 3.1. The reason why the boxes are given these names is that Box 1 and Box 2 are already defined in the work of Poort [7]. The boxes in Figure 3.8 are provided by Vanderlande and are therefore representative for boxes in the parcel industry in the range up to 1 kg. To make sure that the boxes are filled uniformly, a hard foam block is used, bought under the name Sculpture Block [61]. Spherical OptiTrack markers have been placed on one of the faces of the box, which allow us to obtain the ground truth position and orientation of the boxes in 3D space, as discussed in Section 3.1.2.

A geometric model of the boxes is used in the likelihood function and in the motion model. It is important that the properties of the geometric model are close to the properties of the real box to correctly determine its representation in the image plane and to correctly describe its motion after being tossed on a surface. In comparison to the cuboid used in [6], it is possible that the vertices of the real boxes are dented, the edges may not be straight, and the faces may not be flat. Moreover,

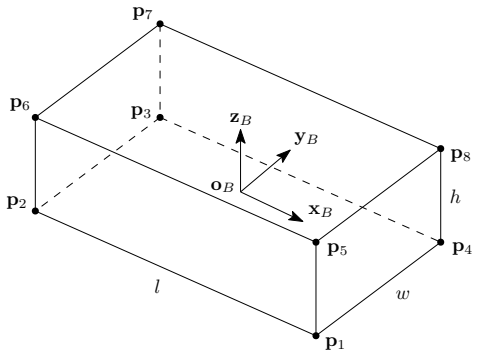


Figure 3.9: Geometric model of Box 5 with body-fixed frame B and vertices $\mathbf{p}_i$.

the real boxes are not completely rigid. The geometric model has the shape of a perfect cuboid and is assumed to be rigid, which means that the geometric model is slightly different from the real box.

As mentioned in [6], the vertices of the box are used to define the shape of the geometric model. A body-fixed frame B is assigned to the center of mass of the boxes and the vertices are indicated by $\mathbf{p}_i$, with $i = \{1, \ldots, 8\}$. In Figure 3.9, the geometric model of Box 5 is shown, seen from the same perspective as the boxes in Figure 3.8. As each box has different dimensions, the vertices are located at different distances from the center of mass. For all boxes, it holds that

$$\begin{aligned}
{}^B\mathbf{p}_1 &= 1/2\begin{bmatrix} l & -w & -h \end{bmatrix}^T, & {}^B\mathbf{p}_5 &= 1/2\begin{bmatrix} l & -w & h \end{bmatrix}^T, \\
{}^B\mathbf{p}_2 &= 1/2\begin{bmatrix} -l & -w & -h \end{bmatrix}^T, & {}^B\mathbf{p}_6 &= 1/2\begin{bmatrix} -l & -w & h \end{bmatrix}^T, \\
{}^B\mathbf{p}_3 &= 1/2\begin{bmatrix} -l & w & -h \end{bmatrix}^T, & {}^B\mathbf{p}_7 &= 1/2\begin{bmatrix} -l & w & h \end{bmatrix}^T, \\
{}^B\mathbf{p}_4 &= 1/2\begin{bmatrix} l & w & -h \end{bmatrix}^T, & {}^B\mathbf{p}_8 &= 1/2\begin{bmatrix} l & w & h \end{bmatrix}^T.
\end{aligned} \tag{3.12}$$

By making use of the dimensions and mass of the boxes, the inertia matrix can be determined. The inertia matrix of the boxes with respect to frame B is given by

$${}_B\mathbb{I}_B \coloneqq \begin{bmatrix} (m/12)(w^2+h^2) & 0 & 0 \\ 0 & (m/12)(l^2+h^2) & 0 \\ 0 & 0 & (m/12)(l^2+w^2) \end{bmatrix}, \tag{3.13}$$

where l, w, h, and m are the length, width, height, and mass of the box, respectively. The inertia tensor is then defined as

$${}_B\mathbb{M}_B = \begin{bmatrix} m\mathbf{I}_3 & \mathbf{0} \\ \mathbf{0} & {}_B\mathbb{I}_B \end{bmatrix}. \tag{3.14}$$

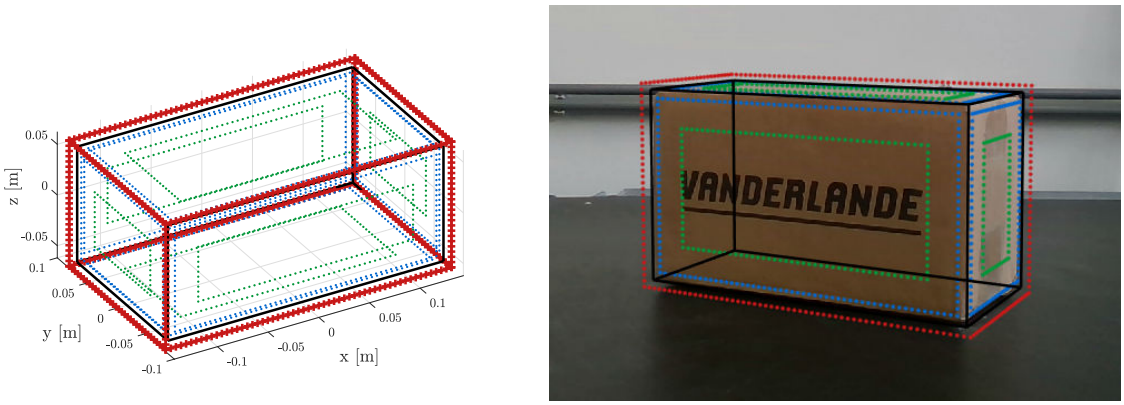
The values for the moments of inertia in Table 3.1 are calculated with (3.13). The OptiTrack markers are not included in these calculations, as their weight and dimensions are negligible.

3.3 Approximate likelihood computation

In Section 2.1.1, it is discussed that the weight assigned to a particle depends on the value of the likelihood of that particle. The closer the particle is to the true pose of the box, the higher the value for the likelihood. In the remainder of this report, we will use the term *approximate likelihood* instead of the term likelihood, since it is impossible to determine the true likelihood. The approximate likelihood computation is only based on the pose and not on the velocity of the particle, as it is difficult to estimate velocity from single images.

To compute the approximate likelihood of a particle, the theory in Section 2.5 and the geometric model given in Section 3.2 are used. A method similar to [14,62] is employed, which is both accurate and computationally efficient. Sets of 3D points are positioned on and around the geometric model of the box, which can be seen in Figure 3.10a. Given a pose of the geometric model, only the visible points are projected onto the image plane, as is shown in Figure 3.10b. Each projected point falls on a specific pixel and the information of these pixels is used to create *color histograms*. Instead of using the standard RGB values of the pixels, we make use of the HSI color space. The pixel values are now represented by the *hue*, *saturation*, and *intensity*, so color is decoupled from intensity.

In [14, 62], two sets of points near the edges of the geometric model are used for the creation of color histograms. The faces of the object also contain information about the pose; however, this information is lost when only points near the edges are used [63]. Therefore, we will sample an extra set of points on each face of the geometric model, besides the two sets near the edges. Figure 3.10 shows the three sets of points, which are distinguished by different colors, namely green, blue, and red. The set of green points is located on the faces of the model, the set of blue points is also located



(a) **(b)**

Figure 3.10: Geometric model with three sets of points **(a)** and projection of the points onto the image plane for a given pose of the geometric model **(b)**. In this case, the particle is located at the true pose of the box.

on the faces, but close to the edges, and the set of red points can be found outside the contour, again close to the edges.

Multiple color histograms are created for each particle, which are used to compute the approximate likelihood. For each visible face i, the histograms $\boldsymbol{H}_{obj,i}$ and $\boldsymbol{H}_{cin,i}$ are computed with the green points and blue points, respectively. The histogram on the outside of the contour, $\boldsymbol{H}_{cout}$, is computed with the red points. Besides the color histograms that are computed with the sets of points, also a reference histogram is needed, denoted by $\boldsymbol{H}_{ref}$. In comparison to [6], we only use a single reference color histogram, instead of multiple reference histograms for each face of the box. This is done because our boxes do not have a different color on each face.

As already mentioned, we make use of the HSI color space, instead of the RGB color model. The hue, saturation, and intensity can typically take an integer value from 0 to 255, in other words, each range has been discretized into 256 so-called *bins*. The set of possible pixel values is defined $\mathcal{H} \coloneqq \mathbb{B}_H \times \mathbb{B}_S \times \mathbb{B}_I$, where $\mathcal{H} \subset \mathbb{N}^3$, and $\mathbb{B}_H$, $\mathbb{B}_S$, and $\mathbb{B}_I$ are the number of bins for the hue, saturation, and intensity, respectively. In order to reduce computional cost, we use $\mathbb{B}_H = 12$, $\mathbb{B}_S = 12$, and $\mathbb{B}_I = 4$, which results in a range of $[0 \dots 11]$ for the hue and saturation and a range of $[0 \dots 3]$ for the intensity. By using 4 bins for the intensity, robustness against illumination changes is achieved [64].

Before the color histogram $\boldsymbol{H}$ can be computed, first the bins of the histogram have to be obtained. To this end, we define the bin index as $\mathbf{b} \in \mathcal{H}$ and the bin index at $\mathbf{d}$ as $\mathbf{b_d} \in \mathcal{H}$, where $\mathbf{d} = \begin{bmatrix} u & v \end{bmatrix}^T$ is the location of the pixel in the image plane, as explained in Section 2.5.1. The bin index $\mathbf{b_d}$ contains the values for the hue, saturation, and intensity of a pixel at location $\mathbf{d}$. The bins of the color histogram can now be determined with

$$h(\mathbf{b}) = \beta \sum_{\mathbf{d} \in \mathcal{U}} \delta_k^3[\mathbf{b} - \mathbf{b_d}] \in \mathbb{R}, \tag{3.15}$$

where β is a normalizing constant, $\mathcal{U}$ is the set of all evaluated pixels, and δ_k^3 is the three-dimensional Kronecker delta function. Subsequently, the color histogram is computed with

$$\boldsymbol{H} = \{h(\mathbf{b})\}_{\mathbf{b} \in \mathcal{H}}, \tag{3.16}$$

so each bin of the total color space $\mathcal{H}$ is evaluated and the corresponding values of $h(\mathbf{b})$ are saved in a $12 \times 12 \times 4$ matrix. Suppose that there is no pixel with the pixel values $\begin{bmatrix} 1 & 1 & 1 \end{bmatrix}$, then element $(1, 1, 1)$ of $\boldsymbol{H}$ is equal to zero.

When all the color histograms are obtained for a particle, the approximate likelihood can be computed. This is done by making use of *similarities* between the histograms, which are calculated with the Bhattacharyya similarity function [65], given as

$$S(\boldsymbol{H}_1, \boldsymbol{H}_2) \coloneqq \sum_{i=1}^{\mathbb{B}_H} \left(\sum_{j=1}^{\mathbb{B}_S} \left(\sum_{k=1}^{\mathbb{B}_I} \sqrt{\boldsymbol{H}_1(i,j,k) \cdot \boldsymbol{H}_2(i,j,k)} \right) \right). \tag{3.17}$$

This function thus returns the similarity between histograms $\boldsymbol{H}_1$ and $\boldsymbol{H}_2$. In our case, we determine the similarities

$$S_1 = \frac{1}{N_f} \sum_{i=1}^{N_f} S(\boldsymbol{H}_{cin,i}, \boldsymbol{H}_{ref}), \tag{3.18}$$

$$S_2 = \frac{1}{N_f} \sum_{i=1}^{N_f} S(\boldsymbol{H}_{obj,i}, \boldsymbol{H}_{ref}), \tag{3.19}$$

and

$$S_3 = \frac{1}{N_f} \sum_{i=1}^{N_f} S(\boldsymbol{H}_{cin,i}, \boldsymbol{H}_{cout}), \tag{3.20}$$

where N_f is the number of visible faces. As the histograms are normalized, S always takes a value in the range of $[0 \dots 1]$, where a value of 1 means that the histograms are exactly the same. We now use the distance D, given by

$$D = \frac{\kappa_1(1 - S_1) + \kappa_2(1 - S_2) + \kappa_3 S_3}{\kappa_1 + \kappa_2 + \kappa_3}, \tag{3.21}$$

to compute the approximate likelihood L with

$$L = e^{-\frac{|D|}{\epsilon}}. \tag{3.22}$$

In these equations, κ_1, κ_2, and κ_3, are weighting parameters and ϵ is a scaling parameter. Particles close to the true pose of the box should get a high value for the approximate likelihood, which is the case when S_1 and S_2 are high, and S_3 is low.

In [6, Appendix A], it is shown that the approximate likelihood function is able to estimate the position and orientation of the cuboid within 1 mm and 3 degrees accuracy, respectively. However, in reality we have to deal with noise, background clutter, and motion blur, which makes it harder to accurately determine the pose of the box. Moreover, the boxes considered in this research have a uniform color, instead of different colors for each face. Therefore, in order to see whether the pose of the box can be estimated accurately in challenging circumstances, it is necessary to evaluate the performance of the approximate likelihood computation on real images with real boxes. The results of this evaluation are discussed in Section 4.1.

3.4 Motion model

The motion model is used to obtain the state of a particle $\mathbf{x}_{t+1}$ at time $t+1$, given the state $\mathbf{x}_t$. The state of a particle is mathematically denoted by $\mathbf{x}_t = \left({}^F\mathbf{R}_B(t), {}^F\mathbf{o}_B(t), {}^B\boldsymbol{v}_{F,B}(t), {}^B\boldsymbol{\omega}_{F,B}(t)\right)$, where the superscript $(\cdot)^{(i)}$ is omitted, since we only consider a single particle i to avoid heavy notation. To be able to describe the movement of a particle that collides with a surface as accurately as possible, friction and impacts are taken into account. In Section 3.4.1, the equations of motion are introduced, followed by the explanation of the laws of contact, impact, and friction in Section 3.4.2, 3.4.3, and 3.4.4, respectively. Combining the equations of motion and the laws of contact, impact, and friction results in a set of equations describing the entire dynamics of the system. To propagate the state from t to $t+1$, a time-stepping scheme and a fixed-point iteration scheme are used, which are discussed in Section 3.4.5 and 3.4.6.

3.4.1 Equations of motion

The equations of motion of a rigid body including the effects of unilateral contacts and friction, but without impacts, are given by

$$\mathbb{M}\dot{\mathbf{v}}(t) + \mathbf{v}(t)\bar{\times}^*\mathbb{M}\mathbf{v}(t) = \mathbf{f}(t) + \mathbf{W}_N(\mathbf{x},t)\boldsymbol{\lambda}_N + \mathbf{W}_T(\mathbf{x},t)\boldsymbol{\lambda}_T, \tag{3.23}$$

where $\mathbb{M} = {}_B\mathbb{M}_B$ is the inertia tensor, $\mathbf{v}(t) = {}^B\mathbf{v}(t)_{F,B}$ is the left trivialized velocity, and $\mathbf{f}(t) = {}_B\mathbf{f}(\mathbf{x},t)$ is the wrench with respect to body-fixed frame B. The wrench contains the forces and torques applied to a rigid body, as discussed in Section 2.4.3. In our case, the wrench only contains the gravitational force, as the contact forces are represented by $\boldsymbol{\lambda}_N$ and $\boldsymbol{\lambda}_T$. Because the gravitational force is independent of time with respect to inertial frame F, we use ${}_{B[F]}\mathbf{f} = \begin{bmatrix}0 & 0 & -mg\end{bmatrix}^T$, which means that the wrench is expressed with respect to frame B with the orientation in terms of frame F. To rewrite ${}_{B[F]}\mathbf{f}$ to ${}_B\mathbf{f}$, (2.88) is used.

Since we are modeling the contact between a 3D object and a planar surface, the contact forces are divided in a normal and tangential component, denoted by $\boldsymbol{\lambda}_N$ and $\boldsymbol{\lambda}_T$. The normal component $\boldsymbol{\lambda}_N$ represents the normal contact force and the tangential component $\boldsymbol{\lambda}_T$ represents the friction force. The matrices containing the generalized force directions of the normal contact force and friction force are denoted by $\mathbf{W}_N$ and $\mathbf{W}_T$, respectively. The equations of motion (3.23) only describe the continuous state evolution as a function of time and do not describe the impact events. Therefore, to include the impact effects, (3.23) is rewritten as the measure differential equations, given by

$$\mathbb{M}d\mathbf{v}(t) + \mathbf{v}(t)\bar{\times}^*\mathbb{M}\mathbf{v}(t)dt = \mathbf{f}(t)dt + \mathbf{W}_N d\mathbf{P}_N + \mathbf{W}_T d\mathbf{P}_T, \tag{3.24}$$

which can handle the impulsive forces, in contrast to the equations of motion in (3.23). The differential measures of the momenta associated to the contact in normal and tangential direction, $d\mathbf{P}_N$ and $d\mathbf{P}_T$, are given by

$$d\mathbf{P}_N = \boldsymbol{\lambda}_N dt + \boldsymbol{\Lambda}_N d\eta \tag{3.25}$$

and

$$d\mathbf{P}_T = \boldsymbol{\lambda}_T dt + \boldsymbol{\Lambda}_T d\eta, \tag{3.26}$$

and both consist of an impulsive and a non-impulsive part of the contact/friction force. The impulsive part is denoted by $\boldsymbol{\Lambda}d\eta$ and the non-impulsive part by $\boldsymbol{\lambda}dt$, where $d\eta$ is the atomic measure, and dt the Lebesgue measure [66, 67]. The atomic measure is defined as $d\eta = \delta(t - t_i)$ [68], with $\delta(\cdot)$ the Dirac delta function and t_i the time instant of an impulsive load.

3.4.2 Contact law

In this research, it is assumed that the vertices of the box are the only *contact points* between body and contact surface. In Figure 3.11, a box is depicted just before the occurrence of an impact. The vector ${}^C\mathbf{z}_C$ is the normal vector of the contact surface, and the distance between a vertex $\mathbf{p}_i$ of the box and the contact surface is called the *gap function*, denoted by g_{Ni}. We express everything with respect to frame F, so we write

$$\begin{aligned} g_{Ni} &= {}^C\mathbf{z}_C^T {}^C\mathbf{p}_i \\ &= {}^F\mathbf{z}_C^T\big({}^F\mathbf{p}_i - {}^F\mathbf{o}_C\big) \\ &= {}^F\mathbf{z}_C^T\big({}^F\mathbf{o}_B + {}^F\mathbf{R}_B\,{}^B\mathbf{p}_i - {}^F\mathbf{o}_C\big). \end{aligned} \tag{3.27}$$

Since it is assumed that the box is rigid and unable to penetrate the surface, the gap function can only be non-negative, hence $g_{Ni} \geq 0$. The associated normal contact force λ_{Ni} is zero when bodies are separated and can only be nonzero when bodies are in contact. Moreover, the normal contact force can also only be non-negative since it cannot prevent the bodies from separating after contact [69]. There are two situations that can occur; the bodies are in contact or the bodies are separated. This can be written formally as:

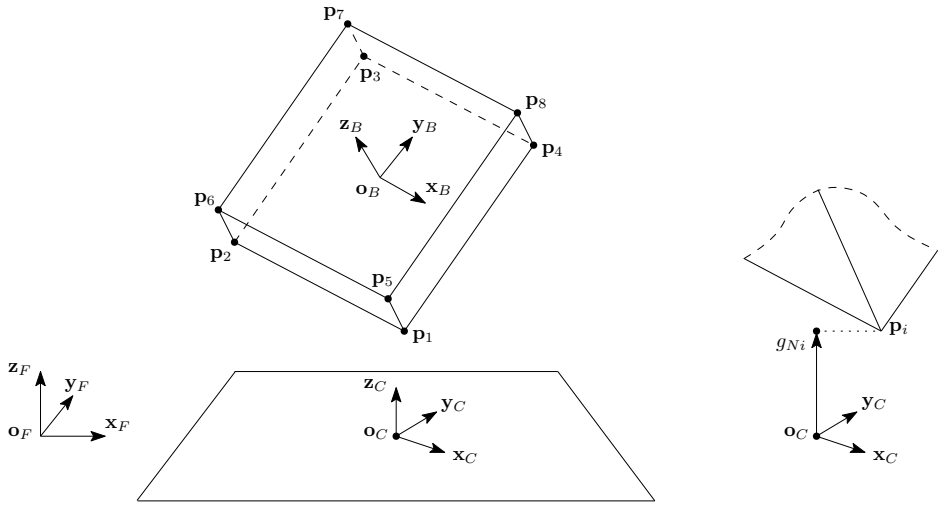


Figure 3.11: Box just before the occurrence of an impact, where the gap function is denoted by g_{Ni}. Adaptation from Jongeneel [6] and Poort [7].

1. $g_{Ni} = 0$ and $\lambda_{Ni} \geq 0$, indicating that the bodies are in contact,
2. $g_{Ni} > 0$ and $\lambda_{Ni} = 0$, indicating that the bodies are separated.

A contact is called *closed* when the bodies are in contact and *open* when the bodies are separated. Signorini's contact [70–73] law describes the complementarity condition resulting from these relations and is given by

$$g_{Ni} \geq 0, \quad \lambda_{Ni} \geq 0, \quad g_{Ni}\lambda_{Ni} = 0, \quad \text{for } i \in \mathcal{I}, \tag{3.28}$$

where the set $\mathcal{I} = \{1, 2, \ldots, n\}$ is the set of all possible contact points, with a maximum value of 8 for n. After combining all gap functions $\mathbf{g}_N = \begin{bmatrix} g_{N_1} & g_{N_2} & \ldots & g_{N_n} \end{bmatrix}^T$ and normal contact forces $\boldsymbol{\lambda}_N = \begin{bmatrix} \lambda_{N_1} & \lambda_{N_2} & \ldots & \lambda_{N_n} \end{bmatrix}^T$, we can rewrite the complementarity condition (3.28), by making use of the *proximal point formulation*, as

$$\boldsymbol{\lambda}_N = \text{prox}_{\mathcal{C}_N}(\boldsymbol{\lambda}_N - r\mathbf{g}_N), \quad \text{with } \mathcal{C}_N = \{\boldsymbol{\lambda}_N \in \mathbb{R}^n \mid \boldsymbol{\lambda}_N \geq \mathbf{0}\} \text{ and } r > 0, \tag{3.29}$$

where $\mathcal{C}_N$ is the set of admissible normal forces.

3.4.3 Impact law

In this research, the impact between two bodies is described by Newton's impact law [74]. An impact may occur when two bodies make contact, so when the contact is closed. Newton's impact law relates the normal velocity in a contact point $\mathbf{p}_i$ after an impact to the normal velocity before the impact through the coefficient of restitution, according to

$$\gamma_{Ni}^{+} = -e_{Ni}\gamma_{Ni}^{-}, \tag{3.30}$$

where e_{Ni} is Newton's coefficient of restitution, γ_{Ni}^{-} the pre-impact velocity, and γ_{Ni}^{+} the post-impact velocity. The coefficient of restitution e_{Ni} typically has a value in the range of $[0, 1]$. When $e_{Ni} = 0$, the two bodies stick together after an impact, which is called a perfectly plastic impact and when $e_{Ni} = 1$, no energy is lost during an impact and the post-impact velocity is equal to the pre-impact velocity, which is called a perfectly elastic impact [75].

The normal velocity of a contact point γ_{Ni} is defined by the derivative of the gap function, so

$$\begin{aligned} \gamma_{Ni} &= \dot{g}_{Ni} \\ &= {}^F\mathbf{z}_C^T\big({}^F\dot{\mathbf{o}}_B + {}^F\dot{\mathbf{R}}_B\,{}^B\mathbf{p}_i\big), \end{aligned} \tag{3.31}$$

where ${}^F\dot{\mathbf{o}}_B$ and ${}^F\dot{\mathbf{R}}_B$, after rewriting (2.76) and (2.77), are given by

$$ {}^F\dot{\mathbf{o}}_B = {}^F\mathbf{R}_B {}^B\boldsymbol{v}_{F,B} \tag{3.32} $$

and

$$ {}^F\dot{\mathbf{R}}_B = {}^F\mathbf{R}_B {}^B\boldsymbol{\omega}^{\wedge}_{F,B}. \tag{3.33} $$

Substituting (3.32) and (3.33) into (3.31) and rewriting the resulting expression leads to

$$ \gamma_{Ni} = \begin{bmatrix} {}^F\mathbf{z}_C^T {}^F\mathbf{R}_B & -{}^F\mathbf{z}_C^T {}^F\mathbf{R}_B {}^B\mathbf{p}_i^{\wedge} \end{bmatrix} \begin{bmatrix} {}^B\boldsymbol{v}_{F,B} \\ {}^B\boldsymbol{\omega}_{F,B} \end{bmatrix}, \tag{3.34} $$

which can be written compactly as

$$ \gamma_{Ni} = \mathbf{w}_{Ni}^T \mathbf{v} \in \mathbb{R}. \tag{3.35} $$

In (3.35), $\mathbf{w}_{Ni}^T$ is the row-vector containing the normal force directions of contact point $\mathbf{p}_i$ and $\mathbf{v} = {}^B\mathbf{v}_{F,B}$ is the left trivialized velocity, as defined by (2.78). Next to the set of all possible contact points $\mathcal{I}$, also a set of n_c closed contact points can be defined: $\mathcal{I}_c \coloneqq \{i \in \mathcal{I} \mid g_{Ni} = 0\}$. We can now relate $\mathbf{w}_{Ni}^T$ to matrix $\mathbf{W}_N$ in (3.24), which is the matrix containing the normal force directions of all closed contacts, through

$$ \mathbf{W}_N = \{\mathbf{w}_{Ni}\}, \quad \text{for } i \in \mathcal{I}_c. \tag{3.36} $$

Due to an impact, a jump in velocity occurs and a normal impulsive momentum Λ_{Ni} is generated. However, for multi-contact situations, it is possible that a closed contact does not participate in the impact. This means that no impulsive momentum is generated during the impact. When this is the case, the post-impact velocity is allowed to be higher than when the contact participates in the impact, which is denoted by $\gamma_{Ni}^+ \geq -e_{Ni}\gamma_{Ni}^-$. Contacts that do not participate in the impact are called *superfluous* contacts and do not influence the contact-impact process, so they can be removed without changing this process [74]. As for the contact between bodies, here also two situations can occur; the contact participates in the impact or the contact is superfluous. In both cases the contact is closed. Formally this can be written as:

1. $\Lambda_{Ni} > 0$ and $\gamma_{Ni}^+ = -e_{Ni}\gamma_{Ni}^-$, indicating that the contact participates in the impact,
2. $\Lambda_{Ni} = 0$ and $\gamma_{Ni}^+ \geq -e_{Ni}\gamma_{Ni}^-$, indicating that the contact is superfluous.

This can again be combined into a complementarity condition, which is

$$ \Lambda_{Ni} \geq 0, \quad \gamma_{Ni}^+ + e_{Ni}\gamma_{Ni}^- \geq 0, \quad (\gamma_{Ni}^+ + e_{Ni}\gamma_{Ni}^-)\Lambda_{Ni} = 0. \tag{3.37} $$

Now, we first define the variable ξ_{Ni} as

$$ \xi_{Ni} = \gamma_{Ni}^+ + e_{Ni}\gamma_{Ni}^-, \tag{3.38} $$

such that the complementarity condition in (3.37) is given by

$$ \Lambda_{Ni} \geq 0, \quad \xi_{Ni} \geq 0, \quad \xi_{Ni}\Lambda_{Ni} = 0, \quad \text{for } i \in \mathcal{I}_c. \tag{3.39} $$

Then, by using the proximal point formulation, (3.39) is written as

$$ \Lambda_{Ni} = \text{prox}_{\mathcal{C}_N}(\Lambda_{Ni} - r\xi_{Ni}), \quad \text{with } \mathcal{C}_N = \mathbb{R}^+ \text{ and } r > 0, \text{ for } i \in \mathcal{I}_c. \tag{3.40} $$

Just as for (3.29), we can define column vectors containing the normal contact velocities $\boldsymbol{\gamma}_N$ and normal impulsive momenta $\boldsymbol{\Lambda}_N$ for all closed contacts. The column representation of ξ_{Ni} is then denoted by

$$ \boldsymbol{\xi}_N = \boldsymbol{\gamma}_N^+ + \boldsymbol{e}_N\boldsymbol{\gamma}_N^- \tag{3.41} $$

and (3.40) is rewritten to

$$ \boldsymbol{\Lambda}_N = \text{prox}_{\mathcal{C}_N}(\boldsymbol{\Lambda}_N - r\boldsymbol{\xi}_N), \quad \text{with } \mathcal{C}_N = \{\boldsymbol{\Lambda}_N \in \mathbb{R}^{n_c} \mid \boldsymbol{\Lambda}_N \geq 0\} \text{ and } r > 0, \tag{3.42} $$

where $\mathcal{C}_N$ is the set of admissible normal impulsive momenta.

3.4.4 Friction law

When the surfaces of two bodies make contact with nonzero relative velocity, friction arises that always works in the opposite direction of the relative motion of the two bodies at the contact point. Coulomb's isotropic spatial friction law is used to model the friction, however we will first discuss the planar case. In the planar case, the relative tangential velocity $\gamma_T \in \mathbb{R}$ is a scalar, in contrast to the spatial case where it has two components, namely $\gamma_{Ti,x}$ and $\gamma_{Ti,y}$. Two types of friction can occur, namely *static friction* and *dynamic friction* [66]. For a situation where static friction occurs, the relative tangential velocity is zero, while for a situation with dynamic friction, the relative tangential velocity is nonzero. Just as in [6], the friction coefficient μ for the static and dynamic friction is assumed to be the same and constant, therefore the following three cases can occur, where λ_T is the friction force and λ_N the normal contact force:

$$\begin{aligned} \gamma_T = 0 \Rightarrow \quad & |\lambda_T| \leq \quad \mu\lambda_N, \text{ indicating static friction,} \\ \gamma_T < 0 \Rightarrow \quad & \lambda_T \;\; = \quad \mu\lambda_N, \text{ indicating negative sliding,} \\ \gamma_T > 0 \Rightarrow \quad & \lambda_T \;\; = -\mu\lambda_N, \text{ indicating positive sliding.} \end{aligned} \tag{3.43}$$

These three cases can be combined in one set-valued force law, given by

$$\lambda_T \in -\mu\lambda_N \text{Sign}(\gamma_T). \tag{3.44}$$

Using this information and the knowledge that a tangential force λ_T only exists for closed contacts, the set of admissible values of the tangential force for each closed contact point can be defined as

$$\mathcal{C}_{Ti} = \{\lambda_{Ti} \mid -\mu_i\lambda_{Ni} \leq \lambda_{Ti} \leq \mu_i\lambda_{Ni}\}, \quad \text{for } i \in \mathcal{I}_c. \tag{3.45}$$

As introduced above, the relative tangential velocity has two components in the spatial case, denoted by $\gamma_{Ti,x} \in \mathbb{R}$ and $\gamma_{Ti,y} \in \mathbb{R}$. More specifically, $\gamma_{Ti,x}$ and $\gamma_{Ti,y}$ indicate the tangential velocity of contact point $\mathbf{p}_i$ in the x- and y-direction of frame C, located at the contact surface. These velocities are derived in the same way as γ_{Ni} in (3.31), so

$$\gamma_{Ti,x} = \frac{d}{dt}\Big({}^F\mathbf{x}_C^T\big({}^F\mathbf{o}_B + {}^F\mathbf{R}_B\,{}^B\mathbf{p}_i - {}^F\mathbf{o}_C\big)\Big) \tag{3.46}$$

and

$$\gamma_{Ti,y} = \frac{d}{dt}\Big({}^F\mathbf{y}_C^T\big({}^F\mathbf{o}_B + {}^F\mathbf{R}_B\,{}^B\mathbf{p}_i - {}^F\mathbf{o}_C\big)\Big), \tag{3.47}$$

where again (3.32) and (3.33) are substituted to obtain

$$\boldsymbol{\gamma}_{Ti} = \begin{bmatrix}\gamma_{Ti,x}\\ \gamma_{Ti,y}\end{bmatrix} = \begin{bmatrix}{}^F\mathbf{x}_C^T\,{}^F\mathbf{R}_B & -{}^F\mathbf{x}_C^T\,{}^F\mathbf{R}_B\,{}^B\mathbf{p}_i^\wedge \\ {}^F\mathbf{y}_C^T\,{}^F\mathbf{R}_B & -{}^F\mathbf{y}_C^T\,{}^F\mathbf{R}_B\,{}^B\mathbf{p}_i^\wedge\end{bmatrix}\begin{bmatrix}{}^B\boldsymbol{v}_{F,B}\\ {}^B\boldsymbol{\omega}_{F,B}\end{bmatrix}. \tag{3.48}$$

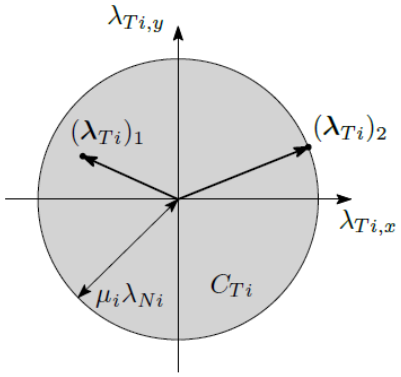


Figure 3.12: Schematic representation of Coulomb's isotropic friction law. Two situations are illustrated: the point $(\boldsymbol{\lambda}_{Ti})_1$ corresponds to a situation where $\|\boldsymbol{\gamma}_{Ti}\| = 0$, whereas the point $(\boldsymbol{\lambda}_{Ti})_2$ corresponds to a situation where $\|\boldsymbol{\gamma}_{Ti}\| > 0$. Figure and caption taken from Jongeneel [6].

In a compact form this is written as

$$\boldsymbol{\gamma}_{Ti} = \mathbf{w}_{Ti}^T \mathbf{v} \in \mathbb{R}^2, \tag{3.49}$$

where $\mathbf{w}_{Ti}^T$ is the matrix containing the directions of the friction force acting on contact point $\mathbf{p}_i$. Matrix $\mathbf{w}_{Ti}^T$ is related to $\mathbf{W}_T$ in (3.24), which is the matrix containing the friction force directions of all closed contacts, through

$$\mathbf{W}_T = \{\mathbf{w}_{Ti}\}, \quad \text{for } i \in \mathcal{I}_c. \tag{3.50}$$

The set-valued force law of (3.44) for the spatial case is now denoted by

$$\boldsymbol{\lambda}_{Ti} = \text{prox}_{\mathcal{C}_{Ti}}(\boldsymbol{\lambda}_{Ti} - r\boldsymbol{\gamma}_{Ti}), \quad \text{with } \mathcal{C}_{Ti} = \{\boldsymbol{\lambda}_{Ti} \mid \|\boldsymbol{\lambda}_{Ti}\| \leq \mu_i \lambda_{Ni}\} \text{ and } r > 0, \text{ for } i \in \mathcal{I}_c. \tag{3.51}$$

Figure 3.12 shows a schematic representation of $\mathcal{C}_{Ti}$, which is called the isotropic *friction disk* [76]. We introduce the variable $\boldsymbol{\xi}_{Ti} \in \mathbb{R}^2$, similar to $\boldsymbol{\xi}_{Ni}$ discussed in Section 3.4.3, according to

$$\boldsymbol{\xi}_{Ti} = \boldsymbol{\gamma}_{Ti}^+ + \boldsymbol{e}_{Ti}\boldsymbol{\gamma}_{Ti}^-, \tag{3.52}$$

where $\boldsymbol{e}_{Ti} \in \mathbb{R}^2$ is the coefficient of restitution is tangential direction. With the variable $\boldsymbol{\xi}_{Ti}$, the set-valued force law for Coulomb's friction can be written in terms of momenta, resulting in the set-valued impact law for Coulomb's friction. This law is written as

$$\boldsymbol{\Lambda}_{Ti} = \text{prox}_{\mathcal{C}_{Ti}}(\boldsymbol{\Lambda}_{Ti} - r\boldsymbol{\xi}_{Ti}), \quad \text{with } \mathcal{C}_{Ti} = \{\boldsymbol{\Lambda}_{Ti} \mid \|\boldsymbol{\Lambda}_{Ti}\| \leq \mu_i \Lambda_{Ni}\} \text{ and } r > 0, \text{ for } i \in \mathcal{I}_c. \tag{3.53}$$

3.4.5 Time-stepping algorithm

The equations in previous sections are now used to obtain the state $\mathbf{x}_{t+1}$, given the state $\mathbf{x}_t$ at time instant t. A time-stepping algorithm [66] is applied, which takes the contributions of smooth and impulsive forces into account when integrating the equations of motion over a time-interval. This time-interval may contain one or more time instants at which an impulsive load occurs. A time step Δt has a start time t_a, an end time $t_e = t_a + \Delta t$, and a midpoint $t_m = t_a + \frac{1}{2}\Delta t$. The first step of the algorithm is to compute the pose of the box at the midpoint, which is done with

$${}^F\mathbf{H}_B(t_m) = {}^F\mathbf{H}_B(t_a)\text{Exp}\big((1/2\Delta t)\mathbf{v}^\wedge(t_a)\big), \tag{3.54}$$

where ${}^F\mathbf{H}_B(t_a)$ and ${}^B\mathbf{v}_{F,B}(t_a)$ are known from previous time step or from the initial conditions and $\text{Exp} : \mathfrak{se}(3) \to \text{SE}(3)$. Then, at the midpoint, it is checked whether a contact is closed by evaluating the gap function $g_{Ni,m}$ of each contact point, given by

$$g_{Ni,m} = {}^F\mathbf{z}_C^T\big({}^F\mathbf{o}_B(t_m) + {}^F\mathbf{R}_B(t_m){}^B\mathbf{p}_i - {}^F\mathbf{o}_C\big), \quad i \in \mathcal{I}. \tag{3.55}$$

When a contact is closed, so $g_{Ni,m} = 0$, the matrices containing the generalized force directions for the contact forces and friction forces at t_m are given by

$$\mathbf{W}_{N,m} = \left\{ \begin{bmatrix} {}^F\mathbf{z}_C^T{}^F\mathbf{R}_B(t_m) & -{}^F\mathbf{z}_C^T{}^F\mathbf{R}_B(t_m){}^B\mathbf{p}_i^\wedge \end{bmatrix}^T \right\}, \quad i \in \mathcal{I}_c \tag{3.56}$$

and

$$\mathbf{W}_{T,m} = \left\{ \begin{bmatrix} {}^F\mathbf{x}_C^T{}^F\mathbf{R}_B(t_m) & -{}^F\mathbf{x}_C^T{}^F\mathbf{R}_B(t_m){}^B\mathbf{p}_i^\wedge \\ {}^F\mathbf{y}_C^T{}^F\mathbf{R}_B(t_m) & -{}^F\mathbf{y}_C^T{}^F\mathbf{R}_B(t_m){}^B\mathbf{p}_i^\wedge \end{bmatrix}^T \right\}, \quad i \in \mathcal{I}_c, \tag{3.57}$$

which are both used to obtain the velocity at t_e with

$$\mathbb{M}\big(\mathbf{v}(t_e) - \mathbf{v}(t_a)\big) + \mathbf{v}(t_a)\bar{\times}^*\mathbb{M}\mathbf{v}(t_a)\Delta t = \mathbf{f}_m\Delta t + \mathbf{W}_{N,m}\mathbf{P}_N + \mathbf{W}_{T,m}\mathbf{P}_T. \tag{3.58}$$

The total set of equations that needs to be solved by the time-stepping algorithm is thus

$$\begin{aligned}
\mathbf{v}(t_e) &= \mathbf{v}(t_a) + \mathbb{M}^{-1}\Big(\mathbf{f}_m\Delta t - \mathbf{v}(t_a)\bar{\times}^*\mathbb{M}\mathbf{v}(t_a)\Delta t + \mathbf{W}_{N,m}\mathbf{P}_N + \mathbf{W}_{T,m}\mathbf{P}_T\Big), \\
\mathbf{P}_N &= \boldsymbol{\lambda}_N\Delta t + \boldsymbol{\Lambda}_N, \\
\mathbf{P}_T &= \boldsymbol{\lambda}_T\Delta t + \boldsymbol{\Lambda}_T, \\
\boldsymbol{\lambda}_N &= \text{prox}_{\mathcal{C}_N}(\boldsymbol{\lambda}_N - r\boldsymbol{g}_N) \quad \text{with} \quad \mathcal{C}_N = \{\boldsymbol{\lambda}_N \in \mathbb{R}^{n_c} \mid \boldsymbol{\lambda}_N \geq 0\}, \\
\boldsymbol{\Lambda}_N &= \text{prox}_{\mathcal{C}_N}(\boldsymbol{\Lambda}_N - r\boldsymbol{\xi}_N) \quad \text{with} \quad \mathcal{C}_N = \{\boldsymbol{\Lambda}_N \in \mathbb{R}^{n_c} \mid \boldsymbol{\Lambda}_N \geq 0\}, \\
\boldsymbol{\lambda}_{Ti} &= \text{prox}_{\mathcal{C}_{Ti}}(\boldsymbol{\lambda}_{Ti} - r\boldsymbol{\gamma}_{Ti}) \quad \text{with} \quad \mathcal{C}_{Ti} = \{\boldsymbol{\lambda}_{Ti} \mid \|\boldsymbol{\lambda}_{Ti}\| \leq \mu_i\lambda_{Ni}\} \geq 0\}, \quad \forall i \in \mathcal{I}_c, \\
\boldsymbol{\Lambda}_{Ti} &= \text{prox}_{\mathcal{C}_{Ti}}(\boldsymbol{\Lambda}_{Ti} - r\boldsymbol{\xi}_{Ti}) \quad \text{with} \quad \mathcal{C}_{Ti} = \{\boldsymbol{\Lambda}_{Ti} \mid \|\boldsymbol{\Lambda}_{Ti}\| \leq \mu_i\Lambda_{Ni}\} \geq 0\}, \quad \forall i \in \mathcal{I}_c,
\end{aligned} \tag{3.59}$$

where $\boldsymbol{\xi}_N$ and $\boldsymbol{\xi}_T$ are given by

$$\boldsymbol{\xi}_N = \boldsymbol{\gamma}_{N,e} + \boldsymbol{e}_N \boldsymbol{\gamma}_{N,a} \quad \text{and} \quad \boldsymbol{\xi}_T = \boldsymbol{\gamma}_{T,e} + \boldsymbol{e}_T \boldsymbol{\gamma}_{T,a}, \tag{3.60}$$

with

$$\begin{aligned} \boldsymbol{\gamma}_{N,a} &= \mathbf{W}_{N,a}^T \mathbf{v}(t_a), \quad \boldsymbol{\gamma}_{N,e} = \mathbf{W}_{N,m}^T \mathbf{v}(t_e), \\ \boldsymbol{\gamma}_{T,a} &= \mathbf{W}_{T,a}^T \mathbf{v}(t_a), \quad \boldsymbol{\gamma}_{T,e} = \mathbf{W}_{T,m}^T \mathbf{v}(t_e). \end{aligned} \tag{3.61}$$

In (3.61), $\mathbf{W}_{N,a}$ and $\mathbf{W}_{T,a}$ are similar to $\mathbf{W}_{N,m}$ and $\mathbf{W}_{T,m}$ in (3.56) and (3.57), the only difference is the rotation matrix ${}^F\mathbf{R}_B$ at time t_a instead of t_m.

The state $\mathbf{x}_{t+1}$ is equal to the state at end time t_e, which is denoted by

$$\mathbf{x}_{t_e} = \big({}^F\mathbf{R}_B(t_e), {}^F\mathbf{o}_B(t_e), {}^B\boldsymbol{v}_{F,B}(t_e), {}^B\boldsymbol{\omega}_{F,B}(t_e)\big), \tag{3.62}$$

where ${}^B\boldsymbol{v}_{F,B}(t_e)$ and ${}^B\boldsymbol{\omega}_{F,B}(t_e)$ are already determined, since $\mathbf{v}(t_e) = {}^B\mathbf{v}_{F,B}$, and the pose of the box, ${}^F\mathbf{R}_B(t_e)$ and ${}^F\mathbf{o}_B(t_e)$, can be obtained through

$${}^F\mathbf{H}_B(t_e) = {}^F\mathbf{H}_B(t_m)\mathrm{Exp}\big((1/2\Delta t)\mathbf{v}^\wedge(t_e)\big). \tag{3.63}$$

3.4.6 Fixed-point iteration

The set of equations given in (3.59) cannot be solved directly, therefore a numerical method called fixed-point iteration [68, Chapter 6] is used. In Algorithm 1, the pseudo-code is given with which 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 can be determined.

Algorithm 1 Fixed-point iteration

$\mathbf{P}_N = 0$, $\mathbf{P}_T = 0$, tol$=1 \cdot 10^{-5}$, $r = 0.01$
repeat
  Compute the velocity at the end of the time step:
  $\mathbf{v}(t_e) \leftarrow \mathbf{v}(t_a) + \mathbb{M}^{-1}\Big(\mathbf{f}_m \Delta t - \mathbf{v}(t_a) \bar{\times}^* \mathbb{M}\mathbf{v}(t_a)\Delta t + \mathbf{W}_{N,m}\mathbf{P}_N + \mathbf{W}_{T,m}\mathbf{P}_T\Big)$

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

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

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

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

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

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

The convergence parameter for the proximal functions is set to $r = 0.01$ in this research. The higher the value of r, the quicker the convergence, however no convergence may be achieved when the value of r is chosen too high. From empirical observations it can be concluded that a value of $r = 0.01$ always leads to convergence and a relatively short computational time, hence the choice for this value. The tolerance for the error is chosen to be $\text{tol} = 1 \cdot 10^{-5}$, which is the same as in [6] and [7].

3.5 Conclusion

In this chapter, the geometric model, the approximate likelihood computation, and the motion model are discussed for real boxes, similar to the ones used in logistic companies. Also, the experimental setup and its components are covered. The approximate likelihood function and the geometric model are used to assign values for the approximate likelihood to particles. The geometric model and motion model are required to propagate the state of the box in time. Chapter 4 presents the main contributions of this research, which are the evaluation of the performance of the approximate likelihood computation on real images, the proposal of a method to obtain the measurement $\mathbf{z}_t$, by making use of object detection and the approximate likelihood function, and the performance evaluation of the GUPF on real-life videos.

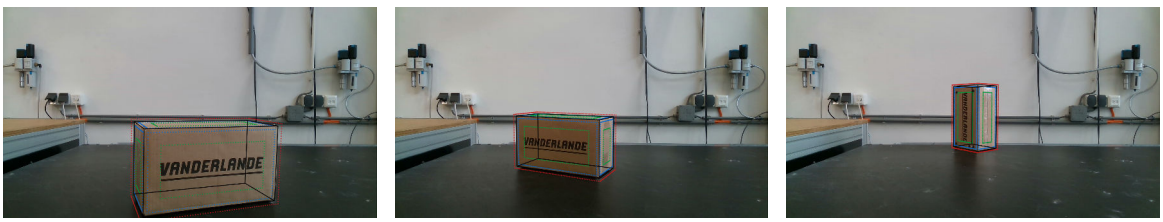
Chapter 4

Pose estimator and experimental validation of the GUPF

In this chapter, the main contributions of this research are presented. Section 4.1 discusses the performance evaluation of the approximate likelihood function on real images. This evaluation is done to conclude whether the approximate likelihood function can be used in the framework of the GUPF as it is, or if it necessary to make adjustments before the function can be implemented. Developing an image-processing algorithm to obtain a measurement $\mathbf{z}_t$ of the pose parameters of the box was beyond the scope of the work of Jongeneel [6]; however, in Section 4.2, such an algorithm is proposed. Object detection and computation of the approximate likelihood are vital parts of this algorithm. Finally, the GUPF is applied on real-life videos in Section 4.3 and its performance is evaluated and compared to the state-of-the-art particle filter.

4.1 Performance evaluation of the approximate likelihood computation

In this section, the performance of the approximate likelihood computation is evaluated on real images. We consider three test images, each with a resolution of 1920×1080 and a box at a specific distance from the camera with a certain orientation. The three test images are shown in Figure 4.1, together with their corresponding ground truth *pose particle*, which is a representation of the true pose of the object in 3D space, projected onto the image plane. We use the term pose particle to emphasize that these particles do not have a linear and angular velocity, in contrast to the particles used in a particle filter. The distance from the camera to the box is approximately 0.55 m, 0.8 m, and 1.2 m for the images in Figure 4.1, respectively. The approximate likelihood will be computed for varying positions and orientations of pose particles around the ground truth in the three test images. From the results, we can conclude whether the pose of the box can be estimated accurately by making use of the approximate likelihood function. In Appendix A, the approximate likelihood is computed for an image with motion blur. This is done to show that the applied method is also able to deal with motion blur.



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

Figure 4.1: Three test images used for the evaluation of the approximate likelihood computation. The distance from the camera to the box is 0.55 m **(a)**, 0.8 m **(b)**, and 1.2 m **(c)**.

During this performance evaluation, solely Box 5 is considered, since the results for Box 3 and 4 are similar. In Section 3.3, it is discussed that sets of green, blue, and red points are used to compute color histograms. The distances of these points with respect to the edges of the geometric model have a significant influence on how discriminating the approximate likelihood function is. To achieve an accurate estimation of the pose for both images with and without motion blur, these distances are tuned for all boxes.

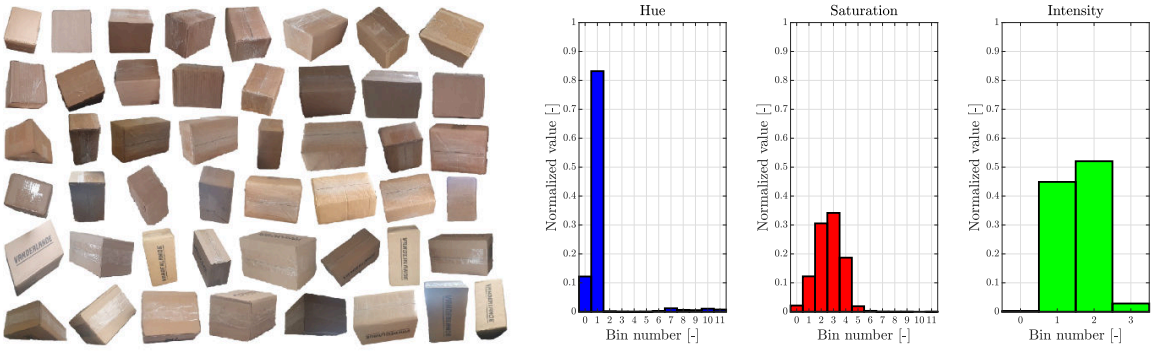
First, the reference color histogram is computed in Section 4.1.1. Then, the performance of the approximate likelihood computation is evaluated for a varying x-, y-, and z-position of the pose particles in Section 4.1.2. Finally, the orientation of the pose particles is varied in Section 4.1.3 and the performance of the approximate likelihood computation is evaluated again. Note that we express the results with respect to camera sensor frame A, whose x- and y-axis span a plane parallel to the image plane. The z-axis is then perpendicular to the image plane and indicates the distance from the camera.

4.1.1 Reference color histogram

In Section 3.3, it is explained that we compute color histograms to determine the approximate likelihood for a particle. One of these color histograms is the reference histogram $\boldsymbol{H}_{ref}$. To compute the reference histogram, first an image has to be created that contains multiple boxes with different poses in different lighting conditions. This image can be seen in Figure 4.2a. All the pixels, except the white ones in-between the boxes, are taken into account during the computation of the reference histogram. The result is depicted in Figure 4.2b, where three histograms are shown to visualize $\boldsymbol{H}_{ref}$, which is a three-dimensional matrix with size $12\times12\times4$. If we take the histogram for the hue as an example, the bar corresponding to bin number 0 is computed by adding all the elements of $\boldsymbol{H}_{ref}$ that have 0 as the bin number for the hue, regardless of the bin number for the saturation and intensity. The same is done for all the other bars. In the end, the result will be a histogram consisting of 12 bars which show how many of the considered pixels have a specific value for the hue. Adding the bars always leads to a value of 1, since each pixel has a value for the hue. For the saturation and intensity, the exact same computations are done.

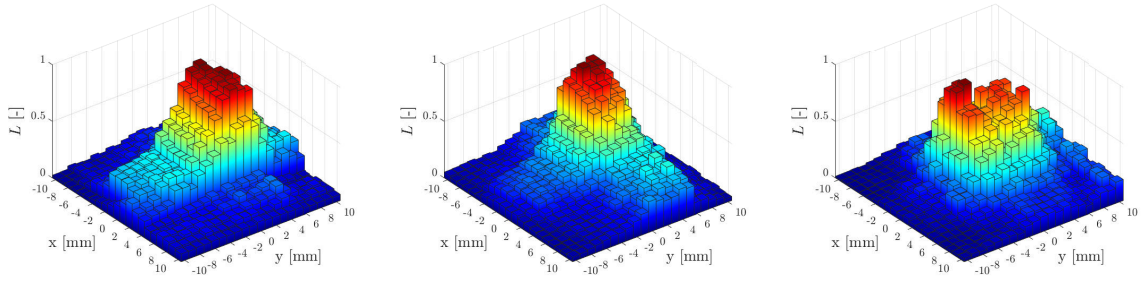
4.1.2 Position estimation

We create two sets of pose particles to draw conclusions regarding the accuracy of the position estimation with the approximate likelihood function. One set of pose particles is created in the xy-plane at a fixed distance z from the camera and the other set is created by varying the z-position of the pose particles, while the x- and y-position are fixed. The approximate likelihood values for the pose particles are normalized to be in the range $[0, 1]$.



(a) **(b)**

Figure 4.2: Reference image containing boxes with different poses in different lighting conditions **(a)** and a visualization of the computed reference color histogram $\boldsymbol{H}_{ref}$ **(b)**.



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

Figure 4.3: Approximate likelihood L for the set of pose particles in the xy-plane at a fixed distance z. Labels correspond to the images of Figure 4.1, so the distance from the camera to the box is 0.55 m **(a)**, 0.8 m **(b)**, and 1.2 m **(c)**.

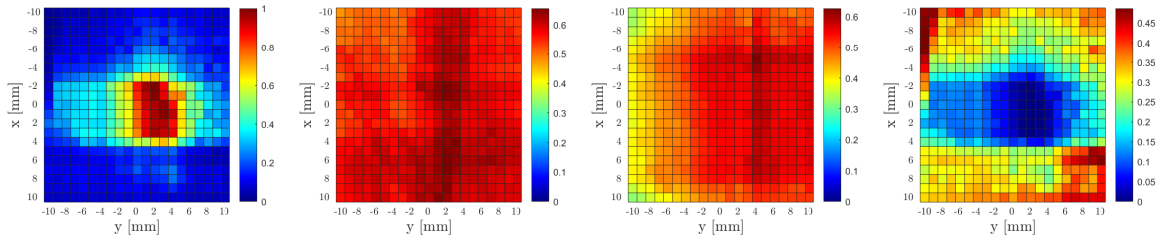
We will first consider the set of pose particles in the xy-plane at a fixed distance z. In the x- and y-direction, pose particles are created that are spaced 1 mm apart in a range of -10 mm to 10 mm. This set thus consists of 441 pose particles. The approximate likelihood is now computed by making use of (3.22) and the results are shown in Figure 4.3 for the images in Figure 4.1.

From these figures, it can be concluded that a high value for the approximate likelihood is assigned to pose particles that are close to the ground truth position, which is the point $x = 0$, $y = 0$. The further away the pose particle is from the ground truth position, the lower the value for the approximate likelihood. The accuracy of the estimation of the x- and y-position for the box in Figure 4.1a and 4.1b is similar; however, when the box is further away from the camera, which is the case for the box in Figure 4.1c, the accuracy is slightly worse. This can be derived from Figure 4.3c, where the peak for the approximate likelihood is less distinctive than in Figure 4.3a and 4.3b. The reason for this is that the representation of the pose particles in the image only changes a little bit when they are far away from the camera, leading to multiple similar color histograms and thus to similar values for the approximate likelihood. However, it can still be observed that the weighted mean is at the right position, namely $x = 0$, $y = 0$.

To get a better understanding about the bar plots in Figure 4.3, we will consider the similarities S_1, S_2, and S_3, which are used to compute the distance D as given by (3.21). In Figure 4.4, 4.5, and 4.6, the approximate likelihood and the similarities are shown for each image in Figure 4.1, respectively. The weighting parameters κ_1, κ_2, and κ_3 in the distance function and the scaling parameter ϵ in (3.22) are chosen to be $\kappa_1 = 1$, $\kappa_2 = 2$, $\kappa_3 = 5$, and $\epsilon = 0.1$, in correspondence with [6]. In the figures, it can clearly be seen that S_3 is more discriminating between pose particles than S_1 and S_2. Therefore, its weighting parameter κ_3 is set to a higher value than κ_1 and κ_2. The weighting parameter κ_2 is chosen to be higher than κ_1, as S_2 is more discriminating than S_1 when the box is closer to the camera. In our case, the box is often located at distances of less than 1 m, which are considered to be relatively close to the camera.

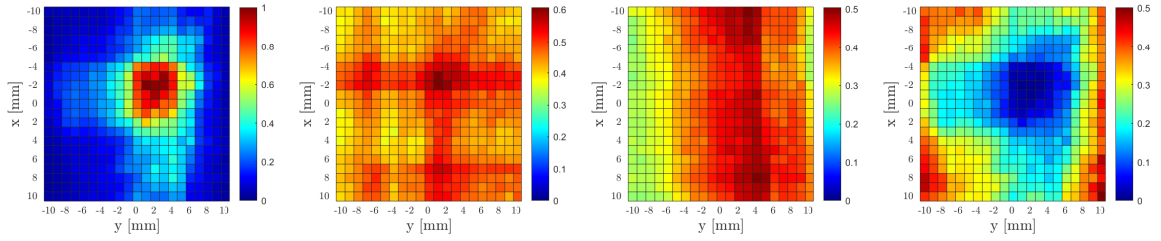
The fact that Figure 4.6a looks different from Figure 4.4a and 4.5a can be explained by comparing Figure 4.4d with 4.6d. In Figure 4.4d, the pose particles with the lowest values for S_3 are concentrated around one point, while the pose particles with the lowest values for S_3 in Figure 4.6d are distributed more. As the box is far away from the camera, it is represented by fewer pixels. The sample points that are close to each other will then fall on the same pixel, resulting in the same similarity values.

From Figure 4.4a can be concluded that the x- and y-position of the box can be estimated with an accuracy of ± 3 mm when it is located at a distance of 0.55 m in front of the camera. The accuracy for the box at 0.8 m in front of the camera is also ± 3 mm and for the box at a distance of 1.2 m, the accuracy is ± 4 mm, which can be derived from Figure 4.5a and 4.6a respectively.



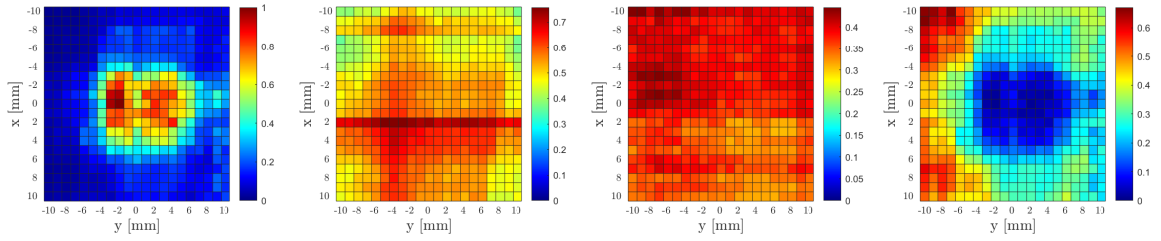
(a) **(b)** **(c)** **(d)**

Figure 4.4: Approximate likelihood L **(a)** and similarities S_1 **(b)**, S_2 **(c)**, and S_3 **(d)** for the box in Figure 4.1a, located at a distance of 0.55 m from the camera.



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

Figure 4.5: Approximate likelihood L **(a)** and similarities S_1 **(b)**, S_2 **(c)**, and S_3 **(d)** for the box in Figure 4.1b, located at a distance of 0.8 m from the camera.



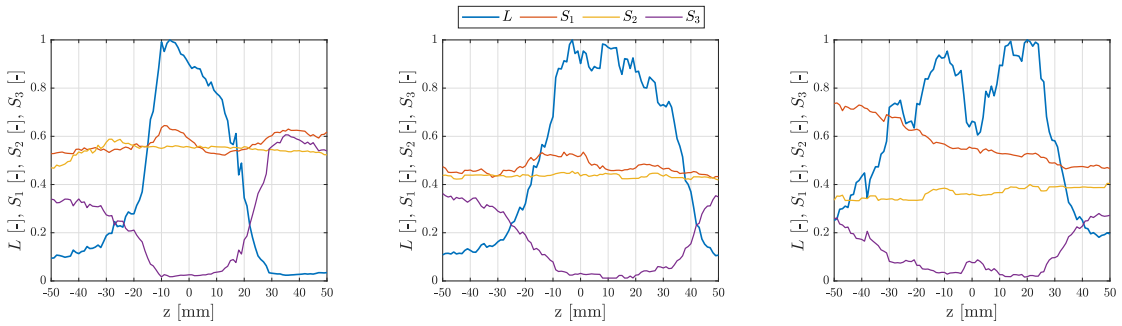
(a) **(b)** **(c)** **(d)**

Figure 4.6: Approximate likelihood L **(a)** and similarities S_1 **(b)**, S_2 **(c)**, and S_3 **(d)** for the box in Figure 4.1c, located at a distance of 1.2 m from the camera.

We will now consider the set of pose particles with varying z-positions, while the x- and y-position are fixed. Again, the pose particles are spaced 1 mm apart, but now for a range of -50 mm to 50 mm, so the set consists of 101 pose particles. The approximate likelihood L and similarities S_1, S_2, and S_3 are computed for the three images in Figure 4.1 and the results are shown in Figure 4.7.

The conclusion from these figures is that we can estimate the z-position accurately when the box is close to the camera, but when the box is further away from the camera, the accuracy of the estimation decreases. In Figure 4.7a, a relatively narrow peak around the ground truth position ($z = 0$) can be observed, which is not the case in Figure 4.7c. The reason for this is again that the representation of the pose particles in the image changes only marginally when they are far away from the camera, resulting in high approximate likelihood values for a large range of z-positions. As for Figures 4.4, 4.5, and 4.6, the pose particles with the lowest values for S_3 have the largest values for the approximate likelihood. In general, S_1 and S_2 are insensitive to position changes, except S_1 in Figure 4.7c. In this case, the histograms computed by the blue dots on the inside of the contour seem to correspond more to the reference histogram when the pose particles are closer to the camera.

The accuracy of the estimation of the z-position is ± 10 mm when the box is located at a distance of



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

Figure 4.7: Approximate likelihood L and similarities S_1, S_2, and S_3 for the set of pose particles with varying z-position. Labels correspond to the images of Figure 4.1, so the distance from the camera to the box is 0.55 m **(a)**, 0.8 m **(b)**, and 1.2 m **(c)**.

0.55 m in front of the camera, ±15 mm when the distance is 0.8 m, and ±20 mm for a distance of 1.2 m.

4.1.3 Orientation estimation

In previous section, we discussed the estimation of the position of the box by making use of the approximate likelihood function. We will apply a similar method to estimate the orientation of the box in the three test images of Figure 4.1. The pose particles are first rotated about the x-, y-, and z-axis of frame B and the approximate likelihood and similarities are computed. Thereafter, the approximate likelihood π-*ball* is introduced, which shows the values for the approximate likelihood for rotations about many other axes.

For the rotation about the x-, y-, and z-axis, we create sets of 401 pose particles. The pose particles are rotated from -200 to 200 degrees individually about each axis with respect to the ground truth orientation located at 0 degrees. This is done with steps of 1 degree. The results for the approximate likelihood and similarities can be seen in Figure 4.8, 4.9, and 4.10. In all figures we can see three peaks, which are located near 0, -180, and 180 degrees. When a pose particle is rotated -180 or 180 degrees it will have the exact same representation in the image as the ground truth pose particle at 0 degrees. It is thus logical that we obtain three identical peaks for the approximate likelihood.

The accuracy of the estimation for Figure 4.8, 4.9, and 4.10 is ±5 degrees or less about all axes, except for the rotation about the x-axis in Figure 4.10, since there the accuracy is ±15 degrees. This lower accuracy is caused by a combination of the box being far away from the camera and the specific axis about which the pose particles are rotated. It is namely that case that the approximate likelihood for the in-plane rotation of the pose particles is more discriminating than the approximate likelihood for the out-of-plane rotation. Formally, an in-plane rotation is a rotation about the vector perpendicular to the image plane and an out-of-plane rotation is a rotation about any other vector, however we consider rotations about vectors close to the perpendicular vector also as in-plane rotations. If we take Figure 4.8 as an example, the rotation of the pose particles about the x- and y-axis is an out-of-plane rotation, while the rotation about the z-axis is an in-plane rotation. For Figure 4.10, the rotation about the x-axis is an out-of-plane rotation and the rotation about the y- and z-axis are in-plane rotations. For out-of-plane rotations the representation of the pose particles in the image changes less than for in-plane rotations. This leads to less discriminating values for S_3 and thus less discriminating values for L, as S_3 again has a larger influence on the approximate likelihood than the similarities S_1 and S_2. The approximate likelihood for out-of-plane rotations is not only less discriminating, but also peaks can be seen for orientations that are significantly different from the ground truth orientation, as is the case in the left image of Figure 4.10. These peaks are still lower than the peaks near 0, -180, and 180 degrees, so they do not cause any problems.

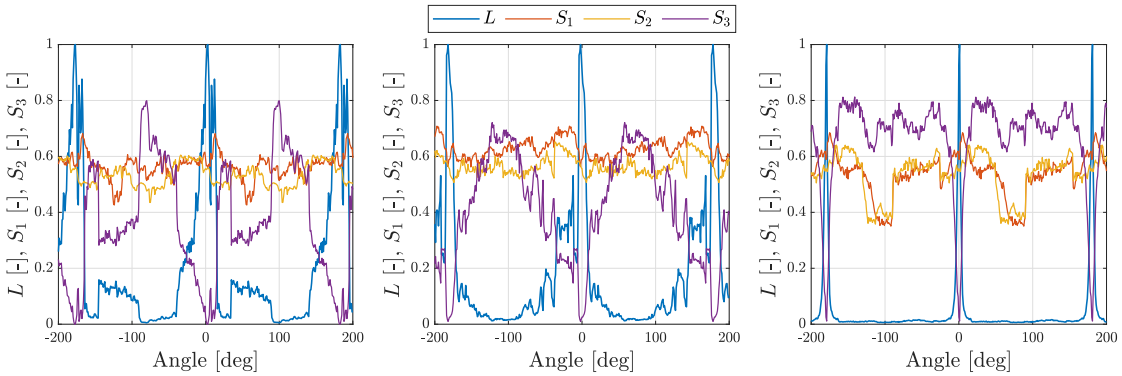


Figure 4.8: Approximate likelihood L and similarities S_1, S_2, and S_3 for pose particles rotated with respect to the ground truth orientation of the box in Figure 4.1a, located at a distance of 0.55 m from the camera. From left to right: rotation about the x-, y-, and z-axis, respectively.

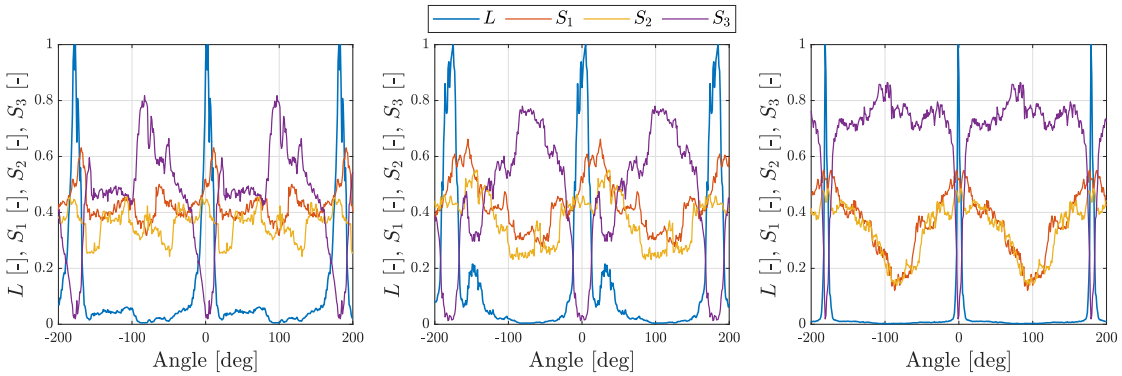


Figure 4.9: Approximate likelihood L and similarities S_1, S_2, and S_3 for pose particles rotated with respect to the ground truth orientation of the box in Figure 4.1b, located at a distance of 0.8 m from the camera. From left to right: rotation about the x-, y-, and z-axis, respectively.

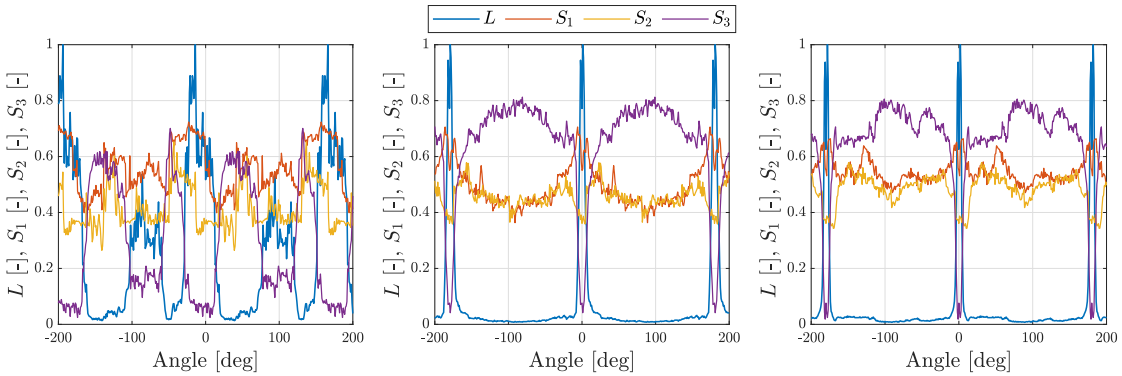
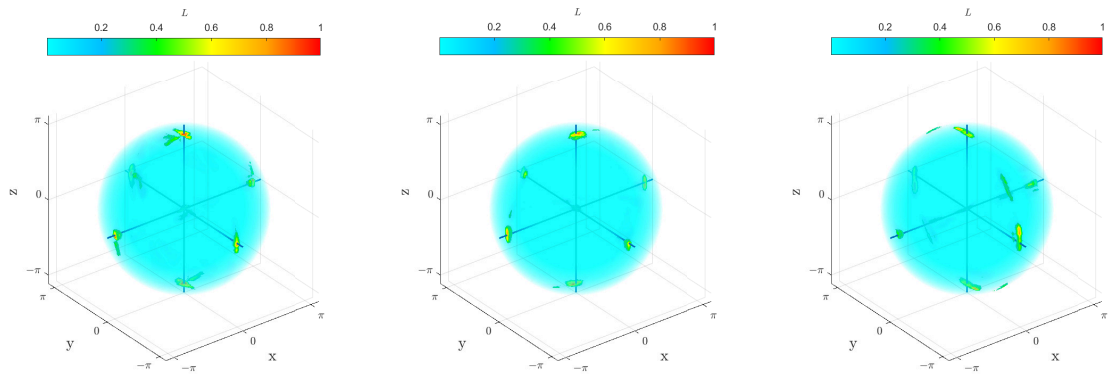


Figure 4.10: Approximate likelihood L and similarities S_1, S_2, and S_3 for pose particles rotated with respect to the ground truth orientation of the box in Figure 4.1c, located at a distance of 1.2 m from the camera. From left to right: rotation about the x-, y-, and z-axis, respectively.

Because we are not only interested in rotation about the x-, y-, and z-axis, the approximate likelihood π-ball is created. The π-ball shows the values of the approximate likelihood for many rotations of the pose particles in SO(3). In Figure 4.11, the π-balls for the images in Figure 4.1 are shown, where the black axes indicate the x-, y-, and z-axis of frame B of the box and the color indicates the value for the approximate likelihood L.



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

Figure 4.11: Approximate likelihood π-balls showing the values of L for many rotations in SO(3). Labels correspond to the images of Figure 4.1, so the distance from the camera to the box is 0.55 m **(a)**, 0.8 m **(b)**, and 1.2 m **(c)**.

The results should be interpreted as follows: the rotation axis about which the pose particles are rotated is a vector starting at the center of the π-ball and ending at an arbitrary point in or on the surface of the ball. The length of this vector determines the magnitude of rotation. Every value of the approximate likelihood on the surface of the π-ball is thus the assigned value to a pose particle that is rotated -180 or 180 degrees about the axis connecting that specific point and the center of the ball, which explains the name π-ball.

The areas with high values for the approximate likelihood can be found close to the center and on the surface of the π-ball near the x-, y-, and z-axis. In other words, pose particles that are rotated a couple of degrees with respect to the ground truth orientation and pose particles that are rotated -180 and 180 degrees about the x-, y-, and z-axis have the highest values for the approximate likelihood. This observation corresponds to the results shown in Figure 4.8, 4.9, and 4.10. From the π-balls can also be concluded that pose particles rotated about other axes mostly have low values for the approximate likelihood. The only exception can be seen in Figure 4.11c, which is the π-ball for the box that is far away from the camera. Here, pose particles rotated -180 and 180 degrees about the $-y, z$- and y, z-axis also have a relatively high value for L. This may happen when a couple of edges of the particles overlap with the edges of the box, while the poses are completely different. When looking closely at the x-axis of the π-ball in Figure 4.11c, we can see small areas with higher values for the approximate likelihood halfway the axis. These areas can be explained by looking at the rotation about the x-axis in Figure 4.10, where also a small increase in L can be seen around -90 and 90 degrees.

4.1.4 Conclusion approximate likelihood computation

The approximate likelihood function is able to accurately estimate the position and orientation of a real box at different distances from the camera, apart from the symmetries at -180 and 180 degrees. The accuracy of the estimation of the x- and y-position is approximately 3 and 4 mm for the box close and far away from the camera. The accuracy of the z-position estimation is 10 mm, 15 mm, and 20 mm for Figure 4.1a, 4.1b, and 4.1c, respectively. The further away the box is from the camera, the less the representation of the pose particles in the image changes, resulting in multiple pose particles with the same value for the approximate likelihood and a slightly worse accuracy. The rotation can be estimated with an accuracy of approximately 5 degrees for all three test images, except for the left image in Figure 4.10, where the accuracy is 15 degrees.

As shown in Appendix A, the approximate likelihood function is also able to estimate the position of a box accurately in an image with motion blur. Estimating the orientation is a bit harder,

but the accuracy is still fairly high. If possible, one should prevent the occurrence of motion blur, for example by increasing the shutter speed of the camera and having a better illumination of the scene. In this research, we want to test the object tracking algorithm in challenging circumstances, so in the images of the recordings always some motion blur occurs.

As the accuracy is comparable for real and synthetic images, the approximate likelihood function is considered to be sufficiently accurate and will thus be used for the state estimation of the box with the particle filters in Section 4.3.

4.2 Obtaining a pose measurement

In this section, it is explained how the measurement $\mathbf{z}_t$ is obtained, which is used in the unscented Kalman filter to create a proposal distribution that is closer to the area with high likelihood, as explained in Section 2.1.4. Such a measurement gives the 3D pose parameters of the box at a given time instant. Section 4.2.1 first discusses the methods employed in [6] to obtain a measurement. Then, in Section 4.2.2, a new method is proposed that makes use of object detection and the approximate likelihood function to compute $\mathbf{z}_t$.

4.2.1 Measurement in previous work

In [6], two different methods are used to obtain the measurement. The first method computes the approximate likelihood of all predicted particles and subsequently determines the weighted mean to obtain a measurement of the pose parameters of the box. The second method adds Gaussian noise to the ground truth pose parameters to simulate a measurement that could be obtained from an image-processing algorithm. Developing such an algorithm was beyond the scope of the work of Jongeneel [6].

Measurement from approximate likelihood computation

During the filtering process, the sigma points of the particles are propagated through the motion model, as discussed in Section 2.1.4. This results in the predicted set of particles, which is used to obtain the measurement. The pose of each particle is denoted by $\mathbf{H}^{(i)} = \big(\mathbf{R}^{(i)}, \mathbf{o}^{(i)}\big) \in \mathcal{P}$ and is evaluated by the approximate likelihood function according to

$$L^{(i)} = \mathcal{L}\big(\mathbf{H}^{(i)}\big), \tag{4.1}$$

where $\mathcal{L} : \mathcal{P} \to \mathbb{R}$ is the approximate likelihood function and $L^{(i)}$ the approximate likelihood assigned to particle i. Here, the weight of the particles is chosen to be proportional to the value for the approximate likelihood, such that a weighted mean can be computed. The procedure discussed in Section 2.2.4 is followed to compute the weighted mean. The starting point of the procedure is choosing an initial value for the mean. We take the pose parameters of the particle with the highest value for L as the initial value, which is denoted by $\tilde{\mathbf{Z}}_t \in \mathcal{P}$. The pose parameters of the other particles are mapped to the tangent space of $\mathcal{P}$ at $\tilde{\mathbf{Z}}_t$ with

$$\boldsymbol{\mathcal{H}}^{(i)} = \mathrm{Log}\Big(\tilde{\mathbf{Z}}_t^{-1}\mathbf{H}^{(i)}\Big) \tag{4.2}$$

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

$$\boldsymbol{\mathcal{Z}}_t = \sum_{i=1}^{N} \tilde{L}^{(i)}\boldsymbol{\mathcal{H}}^{(i)}. \tag{4.3}$$

In (4.3), $\tilde{L}^{(i)}$ is the normalized weight of particle i and is determined with

$$\tilde{L}^{(i)} = \frac{L^{(i)}}{\sum_{j=1}^{N} L^{(j)}}. \tag{4.4}$$

The measurement $\mathbf{Z}_t$ can now be obtained with

$$\mathbf{Z}_t = \tilde{\mathbf{Z}}_t \text{Exp}(\boldsymbol{\mathcal{Z}}_t), \tag{4.5}$$

when $||\boldsymbol{\mathcal{Z}}_t||$ has converged to a predefined threshold.

Measurement from ground truth data

In absence of an image-processing algorithm, the measurement can be simulated by adding Gaussian noise to the ground truth pose parameters, denoted by $\mathbf{GT}_t = (\mathbf{R}_t, \mathbf{o}_t) \in \mathcal{P}$. The measurement can thus be obtained with

$$\mathbf{Z}_t = \mathbf{GT}_t \text{Exp}\big(\phi_t^{\mathbf{R}}, \phi_t^{\mathbf{o}}\big). \tag{4.6}$$

The values for $\phi_t^{\mathbf{R}}$ and $\phi_t^{\mathbf{o}}$ are sampled from normal distributions given by

$$\begin{aligned} \phi_t^{\mathbf{R}} &\sim \mathcal{N}\big(\mathbf{0}, \boldsymbol{P}^{\mathbf{R}}\big), \quad \boldsymbol{P}^{\mathbf{R}} \in \mathbb{R}^{3\times 3}, \\ \phi_t^{\mathbf{o}} &\sim \mathcal{N}\big(\mathbf{0}, \boldsymbol{P}^{\mathbf{o}}\big), \quad \boldsymbol{P}^{\mathbf{o}} \in \mathbb{R}^{3\times 3}, \end{aligned} \tag{4.7}$$

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

4.2.2 Measurement from newly proposed method

As discussed in Section 4.2.1, no image-processing algorithm is used in [6] to obtain the measurement $\mathbf{z}_t$, instead Gaussian noise is added to the ground truth pose parameters to simulate the measurement. In this section, we propose a new method with which the measurement of the pose parameters of the box can be obtained from image data. The main components of this method are object detection to localize and classify the box in the image and a local Monte Carlo search on the pose parameters for the highest value of the approximate likelihood to estimate the pose of the box in 3D space. These components are discussed below.

Object detection

By making use of the approximate likelihood function, the 3D pose of the box can be estimated, as demonstrated in Section 4.1. Since the location of the box in the image was known due to observation by eye, we were able to create sets of pose particles close to the ground truth pose of the box. While localization of objects in an image is a straightforward task for humans, computers do not have this ability. For a computer, an image is just a set of pixels with corresponding pixel values, so it does not associate specific shapes or colors with an object. Sampling pose particles with different positions and orientations in the complete image to estimate the pose of the box is an inefficient and computationally heavy procedure, therefore an object detector is used to localize and classify the box in the image. When we have a rough estimate of the position of the box in the image, we can efficiently sample pose particles close to the true box and obtain an estimate of its pose.

Several methods exist to detect an object in an image, which can be subdivided into two categories: traditional computer vision techniques and deep learning techniques [77]. Examples of traditional computer vision techniques are edge detection, corner detection, or the computation of color histograms. Each class of objects is defined by a combination of different features, so when a large number of specific features is extracted from an image, an object can be classified. The downside of such techniques is that human intervention is required to decide which features describe different classes of objects best. One can imagine that doing this task for many different classes takes a long time, but also the performance of the classification may decrease when multiple classes are defined by similar features. Deep learning techniques, on the other hand, typically make use of convolutional neural networks (CNN) to automatically extract the most descriptive features of different classes of objects from an annotated dataset of images. The labour does not only become less intensive, also the performance of neural networks is better than that of the traditional techniques. This is the reason why we will use deep learning techniques to detect the box in the image.

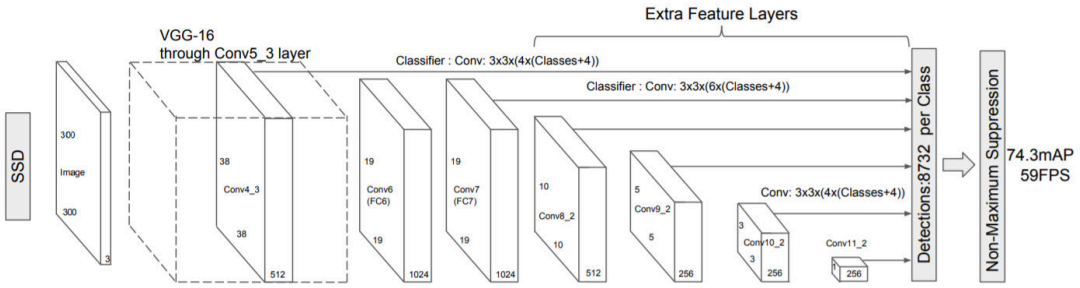


Figure 4.12: Architecture of the Single Shot MultiBox detector (SSD). Image taken from [78].

In this research, we use the Single Shot MultiBox Detector (SSD) [78], which is a deep neural network that is able to detect multiple objects in an image with a single pass (shot) through the network. The SSD is a one-stage detector, so the localization and classification of objects is performed by a single network, while two-stage detectors first propose regions where the object might be located and afterwards apply a convolutional-based classifier to each proposal [79]. This makes the SSD much faster than state-of-the-art object detectors, such as Faster R-CNN [80]. Values of up to 59 FPS are obtained with the SSD, whereas Faster R-CNN operates at 7 FPS [78], so the SSD can even be used for real-time object detection. The accuracy of detection, often indicated by the mean average precision (mAP), is above 70% for the SSD and is comparable to the accuracy of the two-stage object detectors, which are known for their high accuracy.

In Figure 4.12, the architecture of the SSD can be seen as proposed in [78]. The SSD consists of a *base network* and additional convolutional feature layers that construct the *detection network*. The base network, in [78] chosen to be the VGG-16 neural network, acts as a feature extractor and provides feature maps to the detection network. The detection network then performs the classification and localization. Since the focus of this section is on introducing and applying the SSD to detect boxes in an image, the reader is referred to [78] for a detailed explanation about the method that the SSD uses to detect objects.

Instead of the VGG-16 network, we will use the MobileNetV2 network, as it is much smaller and computationally much less intensive [81]. The MobileNetV2 is pre-trained on the COCO dataset [82], which contains more than 300 thousand images and 80 object categories, so it is not needed to train the complete network from scratch. As carton boxes are not included in the COCO dataset, the SSD will not be able to assign the class 'box' to the boxes, therefore some finetuning of the MobileNetV2 network is required. Finetuning parameters of a pre-trained model to make the model perform slightly different tasks is called *transfer learning* [83]. In our case, we finetune the last layers of the model such that the detector is able to localize and classify the boxes.

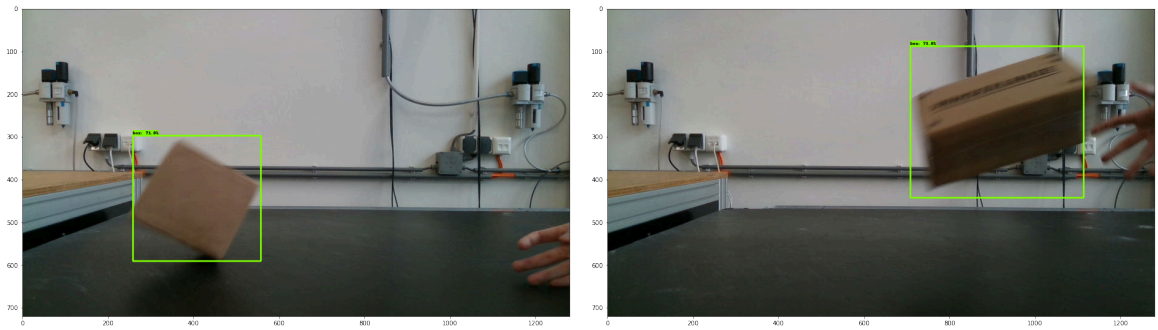


Figure 4.13: Output of the SSD for two test images. In both images the bounding box, the class, and the detection score can be seen.

To train the object detector, we make use of Google Colab. Colab is often used in the machine learning community and allows to make use of GPUs to speed up the code. Before the training can be started, first images of the boxes have to be collected and annotated. In Appendix B, the process behind the annotation of images is discussed. The number of images needed to make it possible for the SSD to detect an object that is not in the used dataset purely depends on the object. When it is a similar object to the other objects in the database, less images are required. Usually, more images leads to a better detection performance. We use a total set of 200 annotated images, from which 150 are assigned to the training set and 50 to the validation set. These images contain boxes at different distances from the camera, boxes with different orientations, and boxes with motion blur, as they should be as representative as possible for the images of the boxes that collide with the conveyor belt.

By making use of a GPU, the training of the object detector takes approximately 45 minutes. Once the detector has been trained, it can be applied on test images. The output is a bounding box around the object, the assigned class, and a detection score. In Figure 4.13, the output of the object detector is shown for two test images with different boxes. Even though the boxes have different poses and some motion blur, the SSD is able to localize and classify the box in the image. Some bounding boxes do not fit very tightly around the box, but this is not problematic. Possible solutions could be to use more training images or to use images without motion blur.

Pose estimation

The bounding boxes provided by the object detector are now used for the estimation of the 3D pose of the carton boxes. The pose estimator discussed in this section is inspired by the method employed in [84], where also bounding boxes and a likelihood function are used to estimate the pose of objects in a 2D image. To obtain the measurement $\mathbf{z}_t$ of the pose parameters of the box, the pose estimator is applied on every frame of a video. Below, the complete procedure is explained in detail.

As the bounding boxes fit fairly tightly around the boxes in most cases, it can be assumed that the pixel coordinates u and v of the centers of the bounding boxes are close to the pixel coordinates of the projected centers of mass of the boxes onto the images. By making use of this information, the components of ${}^{A}\mathbf{p}$, so the 3D position of the center of mass of the box with respect to camera sensor frame A, can be determined with

$$\begin{aligned} u &= {}^{A}\mathbf{p}_z^{-1}\big(\alpha_x \cdot {}^{A}\mathbf{p}_x + u_0 \cdot {}^{A}\mathbf{p}_z\big), \\ v &= {}^{A}\mathbf{p}_z^{-1}\big(\alpha_y \cdot {}^{A}\mathbf{p}_y + v_0 \cdot {}^{A}\mathbf{p}_z\big), \end{aligned} \tag{4.8}$$

which is a rewritten form of (2.93). However, this set of two equations has infinitely many solutions, since there are three unknowns. To solve this problem, pose particles are projected onto the image plane with varying values for ${}^{A}\mathbf{p}_z$, such that ${}^{A}\mathbf{p}_x$ and ${}^{A}\mathbf{p}_y$ in (4.8) can be computed. Recall that ${}^{A}\mathbf{p}_z$ indicates the distance in the z-direction from an object to the camera, so by varying ${}^{A}\mathbf{p}_z$, the pose particles appear with different sizes in the image, while the pixel coordinates of their projected centers of mass do not change. The values of ${}^{A}\mathbf{p}_z$ are chosen to be in the range of 0.6 m to 1.2 m, with steps of 0.05 m. After the pose particles are projected onto the image plane, the approximate likelihood is computed and the coordinates ${}^{A}\mathbf{p}_x$, ${}^{A}\mathbf{p}_y$, and ${}^{A}\mathbf{p}_z$ are saved for the pose particle with the highest value for the approximate likelihood.

Sometimes it is the case that the box is not exactly at the center of the bounding box, which may lead to a less accurate estimation of the position. To cope with this problem, the same computations as above are performed for a range of values around the pixel coordinates u and v of the center of the bounding box. From calculations can be concluded that the difference between the center of the bounding box and the center of the carton box is not larger than 20 pixels in the horizontal and vertical direction. A region of 40×40 pixels around the center of the bounding box is therefore considered to be more than large enough.

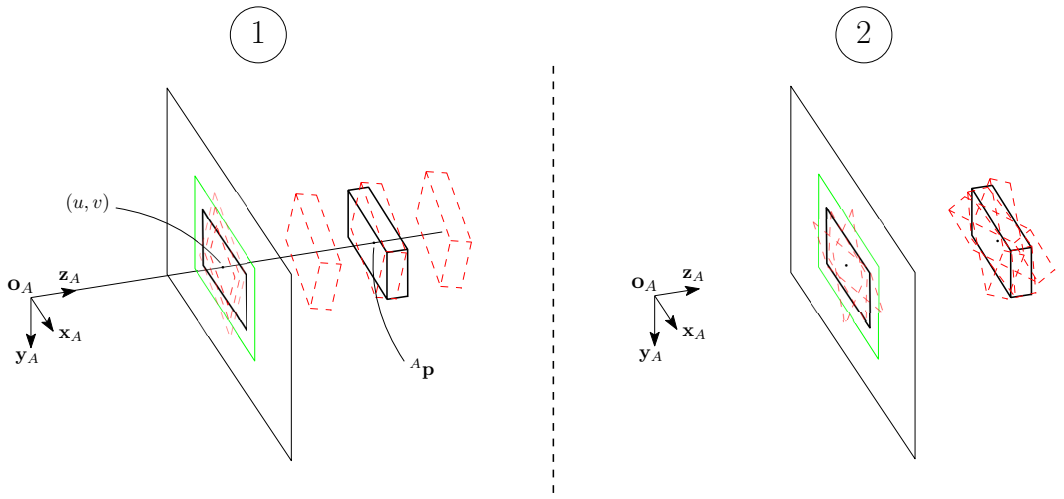


Figure 4.14: Visualization of the procedure to estimate the 3D pose of the box from a 2D image. In part 1 (left), the position of the box is estimated. In part 2 (right), the orientation of the box is estimated. The green rectangle represents the bounding box, the red wireframes represent the pose particles, and the black wireframe is the true box.

The next step of the procedure is to estimate the orientation of the box. The pose particle with the highest value for the approximate likelihood from the position estimation stage is rotated about multiple axes in space with different angles (same method as for the π-ball), which results in a new set of pose particles. Similar to the estimation of the position, the rotation matrix ${}^A\mathbf{R}_B$ is saved for the pose particle with the highest value for the approximate likelihood. We have now obtained an estimate of the 3D pose of the box from a 2D image. In Figure 4.14, a visualization of the complete procedure is shown.

For the first frame of a recording it is necessary to provide a rough guess of the initial orientation of the box before the position and orientation can be estimated. This is done to prevent problems with the symmetry of the box. In practice, the rough guess can be obtained from forward kinematics of a robotic arm, however we just choose to have a rough guess that differs ± 30 degrees from the ground truth orientation. The obtained estimation of the orientation in the previous frame is used as the initial orientation in the next frame, except for the first frame. As the boxes do not rotate much between frames, it is sufficient to rotate the pose particle 30 degrees with steps of 1 degree about different axes in space during the orientation estimation stage, even when impacts occur.

The number of evaluated pose particles is 5733 for the position estimation and 3751 for the orientation estimation. For all the frames of a recording these numbers are the same. From the region of 40×40 pixels around the center of the bounding box, 441 pixels are used instead of all 1600 to keep the computational time low. The range for ${}^A\mathbf{p}_z$ varies from 0.6 m to 1.2 m with steps of 0.05 m, so 13 pose particles per pixel. This leads to a total number of 5733 pose particles for the estimation of the position. As mentioned before, the pose particles are rotated 30 degrees with steps of 1 degree about different axes in space, which means 31 pose particles per axis. The total number of axes is 121, leading to 3751 pose particles for the estimation of the orientation.

By applying the pose estimator on each frame of a video, the pose parameters of the box are obtained, and thus also the measurement $\mathbf{Z}_t$. Mathematically, we can write this as

$$\mathbf{Z}_t = {}^A\mathbf{H}_B(t) = \left({}^A\mathbf{R}_B(t), {}^A\mathbf{o}_B(t)\right) \in \mathcal{P}. \tag{4.9}$$

4.2.3 Conclusion measurement

In this section, the methods employed in [6] to obtain the measurement are discussed and a new method is proposed that makes use of object detection and the approximate likelihood function. The measurement that is obtained by adding Gaussian noise to the ground truth pose parameters is not used during the performance evaluation of the GUPF, because it is replaced by the newly proposed method. In the remainder of this report, the measurement resulting from only the approximate likelihood computation, as used in [6], is denoted by $\mathbf{L}_t$ and the measurement resulting from the method proposed in Section 4.2.2 is denoted by $\mathbf{Z}_t$.

4.3 Performance evaluation of the GUPF

To evaluate the performance of the GUPF, the algorithm is applied on three videos of different boxes that collide with the conveyor belt. As discussed in Section 3.2, if the GUPF is able to estimate the state of the three boxes accurately in all frames of a recording, it is assumed that it works for all uniformly filled carton boxes with similar properties. Because the trajectories of the boxes are different in each video, the algorithm is tested on different scenarios. The box may, for instance, be close or far away from the camera and it may tumble a lot or not rotate much.

In this section, the results for the videos of Box 3 and Box 5 are discussed. The results for the video of Box 4 can be found in Appendix C, as the motion of Box 4 is similar to the motion of Box 5. Four different object tracking algorithms are introduced, namely the PF-CV, PF-NS, GUPF-NS-L, and the GUPF-NS-Z. The first part of these abbreviations represents the type of particle filter, which is either an ordinary Particle Filter or the Geometric Unscented Particle Filter. The second part indicates whether a Constant Velocity motion model is used or a Nonsmooth motion model that takes into account gravity, friction, and impact dynamics. Finally, the L and Z refer to the type of measurement that is incorporated in the unscented Kalman filter to create a proposal distribution for each particle, as explained in Section 2.1.4. Recall that measurement $\mathbf{L}_t$ is obtained by computing the approximate likelihood of all predicted particles and subsequently determining the weighted mean. Measurement $\mathbf{Z}_t$, on the other hand, is obtained from a newly proposed method, where an object detector and pose estimator are used. The ordinary particle filter does not make use of a measurement, since there is no unscented Kalman filter, therefore the names are just PF-CV and PF-NS.

Before evaluating the performance of the particle filters, Section 4.3.1 first mentions the settings of the algorithms and discusses general subjects, such as the constant velocity motion model of the ordinary particle filter, the equations used to compute the position and orientation error, and the naming of the videos. Then, in Section 4.3.2 and 4.3.3, the results for the video of Box 5 and Box 3 are shown, respectively. Lastly, conclusions are drawn in Section 4.3.4.

4.3.1 Settings and general subjects

Constant velocity motion model

Until now, only the nonsmooth motion model is discussed in Section 3.4, however state-of-the-art particle filters [48, 50, 85] usually have a constant velocity motion model where no preference is given to any direction of motion. The equations of motion for the constant velocity motion model are given by

$$\mathbf{H}(t_e) = \mathbf{H}(t_a)\mathrm{Exp}\Big(\Delta t \mathbf{v}^\wedge(t_a) + \sqrt{\Delta t}\mathbf{a}^\wedge\Big), \tag{4.10}$$

$$\mathbf{v}^\wedge(t_e) = \frac{1}{\Delta t}\mathrm{Log}\Big(\mathbf{H}(t_a)^{-1}\mathbf{H}(t_e)\Big), \tag{4.11}$$

where $\mathbf{a} \in \mathbb{R}^6$ is a zero-mean white noise vector, t_a and t_e are the beginning and end time a time step, and $\Delta t = 1/30$, since the frame rate of the RealSense D415 camera is chosen to be 30 fps. In

(4.10), $\mathbf{H}$ and $\mathbf{v}$ are short-hand notations for ${}^{F}\mathbf{H}_{B}$ and ${}^{B}\mathbf{v}_{F,B}$, respectively. In Section 4.3.2 and 4.3.3, the tracking results obtained with the state-of-the-art particle filter with a constant velocity motion model are compared to particle filters with a nonsmooth motion model.

Error computation and performance measures

All tracking results from the particle filters are compared to the ground truth trajectory, obtained with the OptiTrack motion capture system. Instead of expressing the results with respect to the frame of the base of the robot F, it is better to express them with respect to camera sensor frame A. The reason for this is the fact that the z-axis of frame A is not exactly parallel to the conveyor belt. Transforming the results from F to A is simply done with ${}^{A}\mathbf{H}_{B} = {}^{A}\mathbf{H}_{F}{}^{F}\mathbf{H}_{B}$.

The difference between the estimated center of mass of the box and the true center of mass of the box is referred to as the position error and is defined as

$$e_{\mathbf{o}} = {}^{A}\mathbf{o}_{B} - {}^{A}\mathbf{o}_{B}^{GT}, \tag{4.12}$$

where ${}^{A}\mathbf{o}_{B}$ is the estimated position of the center of mass of the box, obtained from any object tracking algorithm. Next to the position error, there is also an orientation error, which is defined as

$$e_{\mathbf{R}} = \mathrm{Log}\Big(\left(\mathbf{R}^{GT}\right)^{-1}\mathbf{R}\Big) \in \mathfrak{so}(3), \tag{4.13}$$

where $\mathbf{R}^{GT} = {}^{A}\mathbf{R}_{B}^{GT}$ and the rotation matrix obtained from any object tracking algorithm is $\mathbf{R} = {}^{A}\mathbf{R}_{B}$. The time dependencies are omitted in both (4.12) and (4.13). When plotting the results, the norm of the position and orientation error are taken, denoted by respectively $\|e_{\mathbf{o}}\|$ and $\|e_{\mathbf{R}}\|$. Taking the norm of $e_{\mathbf{R}}$ leads to the angular error in radians, which is converted into degrees afterwards.

The RMS values of the norm of the position and orientation errors and the relative error reduction are taken as the performance measures of the various algorithms, in correspondence with [6].

Parameters and coefficients

There are several parameters that need to be defined before the particle filters can be applied on the videos. In Section 3.4, the friction coefficient μ and the coefficients of restitution e_N and e_T are introduced. We do not know the exact values for these coefficients, as no parameter identification has been carried out for the three boxes of Vanderlande. Poort has performed a parameter identification for two smaller carton boxes [7], so it is assumed that the obtained values for μ, e_N, and e_T can be used as an initial guess for the coefficients of our boxes. By running the object tracking algorithms multiple times and for multiple videos, the coefficients are tuned. Since the motion of the particles after an impact is heavily influenced by the coefficients, we adjust the values until the particles stay close to the true box in the image. The final values for the coefficients are taken as $\mu = 0.5$, $e_N = 0.15$, and $e_T = 0$. From the results in the work of Poort [7], it is concluded that the influence of e_T is significantly smaller than that of the other coefficients and [67] states that e_T is zero is most practical cases, therefore we choose to use $e_T = 0$.

As discussed in Section 3.4.5, a time-stepping algorithm is employed to obtain the state $\mathbf{x}_{t+1}$, given the state $\mathbf{x}_t$ at time t. The time step is here also taken as $\Delta t = 1/30$, due to the frame rate of 30 fps of the RealSense D415 camera, so the motion of the particles is simulated between two subsequent frames. Because the impacts only take a very short amount of time, Δt is discretized into ten smaller steps, which means that the motion model now runs at a frequency of 300 Hz. Discretizing into even more steps affects the computational time in a negative way and does not lead to a noticeable improvement of the tracking results.

Another important parameter is the number of particles. The more particles, the more accurate the weighted mean will be. However, using a lot of particles also results in excessive computational times. We will use 100 particles for each filter to be able to make a fair comparison.

Initial state and covariance

The particle filter is initialized by sampling a set of particles from an initial uncertainty distribution $p(\mathbf{x}_0)$ for the state. The state mean $\bar{\mathbf{x}}_0$ and covariance $\boldsymbol{P}_0^{\mathbf{x}}$ of this initial distribution are given by

$$\bar{\mathbf{x}}_0 = ({}^F\mathbf{R}_B, {}^F\mathbf{o}_B, {}^B\boldsymbol{v}_{F,B}, {}^B\boldsymbol{\omega}_{F,B}) \in \mathcal{S} \tag{4.14}$$

and

$$\boldsymbol{P}_0^{\mathbf{x}} = \begin{bmatrix} \boldsymbol{P}^{\mathbf{R}} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{P}^{\mathbf{o}} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \boldsymbol{P}^{\boldsymbol{v}} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \boldsymbol{P}^{\boldsymbol{\omega}} \end{bmatrix} \in \mathfrak{s} \otimes \mathfrak{s}, \tag{4.15}$$

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. In the work of Jongeneel [6], the initial pose and velocity of the object are chosen arbitrarily, but for real-life videos the initial state variables have to be determined.

While the true position and orientation of the box in the first frame of a video can easily be obtained from the motion capture system, the linear and angular velocity have to be approximated. This is due to the fact that OptiTrack's motion capture only estimates the pose of the box and not the velocity. A central differencing scheme is used to approximate the linear and angular velocity at time t_k, so

$${}^F\dot{\mathbf{o}}_B(t_k) \approx \frac{1}{2\Delta t}\big({}^F\mathbf{o}_B(t_{k+1}) - {}^F\mathbf{o}_B(t_{k-1})\big), \tag{4.16}$$

$${}^B\boldsymbol{\omega}_{F,B}^{\wedge}(t_k) \approx \frac{1}{2\Delta t}\Big(\mathrm{Log}\big({}^F\mathbf{R}_B^T(t_k){}^F\mathbf{R}_B(t_{k+1})\big) - \mathrm{Log}\big({}^F\mathbf{R}_B^T(t_k){}^F\mathbf{R}_B(t_{k-1})\big)\Big), \tag{4.17}$$

$${}^B\boldsymbol{v}_{F,B}(t_k) \approx {}^F\mathbf{R}_B^T(t_k){}^F\dot{\mathbf{o}}_B(t_k), \tag{4.18}$$

where $\Delta t = 1/360$, which is the time between two frames captured with the motion capture system.

The values for the initial state covariances are based on the accuracy of the pose estimation with the motion capture system and the approximation with the central differencing scheme. From judging by eye, the following initial state covariances are used for all algorithms:

$$\boldsymbol{P}^{\mathbf{R}} = 1 \cdot 10^{-5}\mathrm{diag}([1\ \ 1\ \ 1])\ \ \mathrm{rad}^2, \qquad \boldsymbol{P}^{\mathbf{o}} = 1 \cdot 10^{-5}\mathrm{diag}([1\ \ 1\ \ 1])\ \ \mathrm{m}^2,$$
$$\boldsymbol{P}^{\boldsymbol{v}} = 1 \cdot 10^{-4}\mathrm{diag}([1\ \ 1\ \ 1])\ \ \mathrm{m}^2/\mathrm{s}^2, \qquad \boldsymbol{P}^{\boldsymbol{\omega}} = 1 \cdot 10^{-3}\mathrm{diag}([1\ \ 1\ \ 1])\ \ \mathrm{rad}^2/\mathrm{s}^2.$$

As the pose of the box is estimated with sub-millimeter accuracy with the motion capture system and the approximation of the linear and angular velocity is less accurate, lower values for $\boldsymbol{P}^{\mathbf{R}}$ and $\boldsymbol{P}^{\mathbf{o}}$ are chosen than for $\boldsymbol{P}^{\boldsymbol{v}}$ and $\boldsymbol{P}^{\boldsymbol{\omega}}$. When $\boldsymbol{P}^{\boldsymbol{v}}$ and $\boldsymbol{P}^{\boldsymbol{\omega}}$ are compared, it can be seen that $\boldsymbol{P}^{\boldsymbol{v}}$ has lower values than $\boldsymbol{P}^{\boldsymbol{\omega}}$. The reason for this is that the approximation of ${}^B\boldsymbol{v}_{F,B}(t_k)$ stays about the same for neighbouring data points, whereas the approximation of ${}^B\boldsymbol{\omega}_{F,B}^{\wedge}(t_k)$ changes more significantly.

Covariance computation for a given signal

In this section, we will explain how to determine the covariance matrices corresponding to the position and orientation for a given signal, which are both computed in a different way. It is assumed that the position and orientation errors are known for each frame of a recording. The first step of the computation of the covariance matrix corresponding to the position is determining the mean error of the x-, y-, and z-position. These mean errors are combined in vector $\bar{\mathbf{x}}$. Then, the covariance matrix Q can be computed with

$$Q = \frac{1}{N}\sum_{i=1}^{N}(\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{x}_i - \bar{\mathbf{x}})^T, \tag{4.19}$$

where N is the number of data points (frames) and $\mathbf{x}_i$ is the vector containing the errors of the x-, y-, and z-position for data point i. Since the errors of the x-, y-, and z-position are uncorrelated, the

off-diagonal elements are set to 0.

To compute the covariance matrix corresponding to the orientation, we will follow the procedure discussed in Section 2.2.4. After selecting a rotation matrix close to the true mean and going through all steps of the procedure, the mean is obtained which is used in (2.64) to compute the covariance matrix. The off-diagonal elements are again set to 0.

Naming of videos

The GUPF is applied on three videos of different boxes to test the algorithm on different scenarios. Each video will be indicated with a number followed by the box that is shown in the videos. The following naming convention will be employed in this report:

- Video 1 (Box 5)
- Video 2 (Box 3)
- Video 3 (Box 4)

In Section 4.3.2 and 4.3.3, the tracking results after applying the GUPF on the videos of Box 5 and Box 3 are discussed. The tracking results for the video of Box 4 are given in Appendix C. Box 5 in Video 1 has a similar motion to Box 4 in Video 3, while Box 3 in Video 2 tumbles considerably more. Therefore, it is decided to discuss Video 1 and Video 2 in the main text and Video 3 in the appendix.

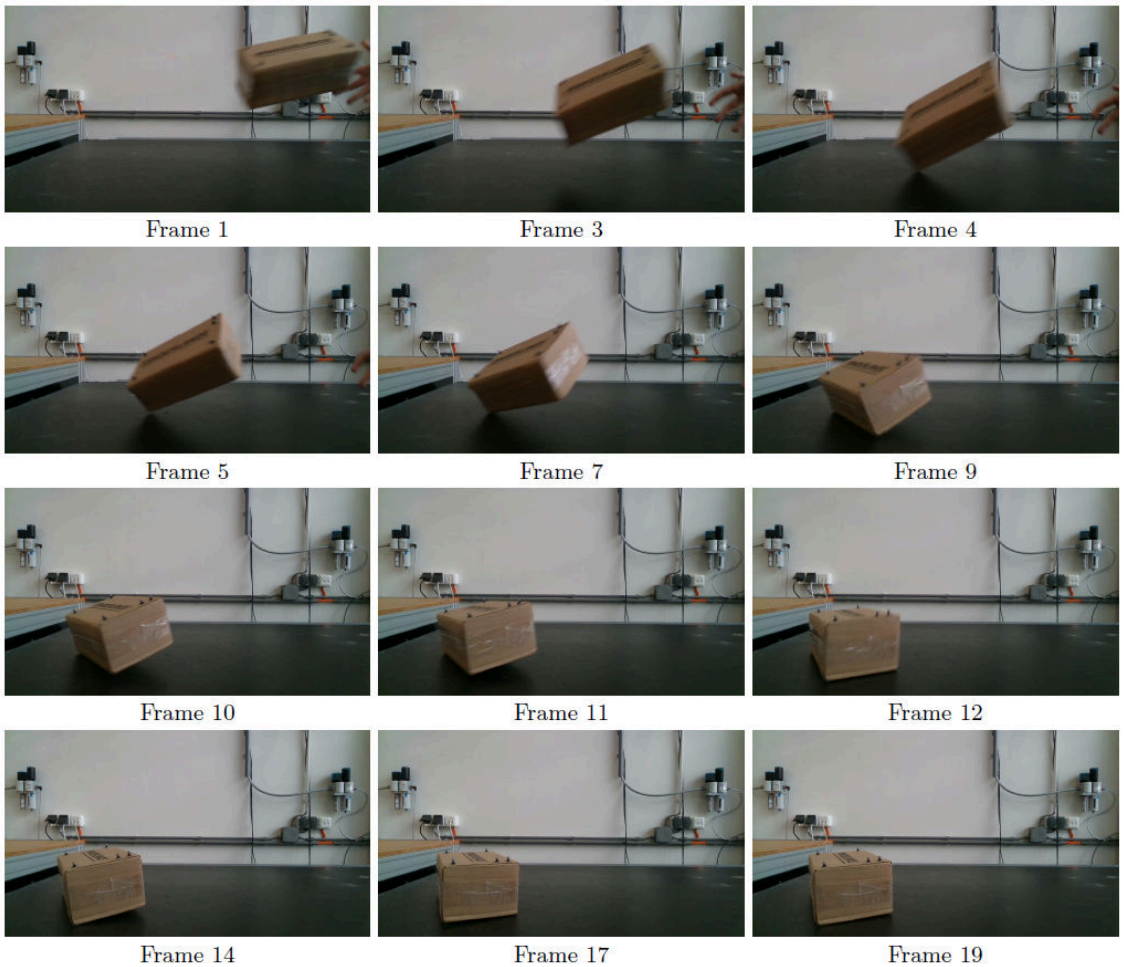


Figure 4.15: Most important frames of Video 1 of Box 5. The video has a length of 19 frames and impacts with the conveyor belt occur at frame 5, 7, 9, and 12.

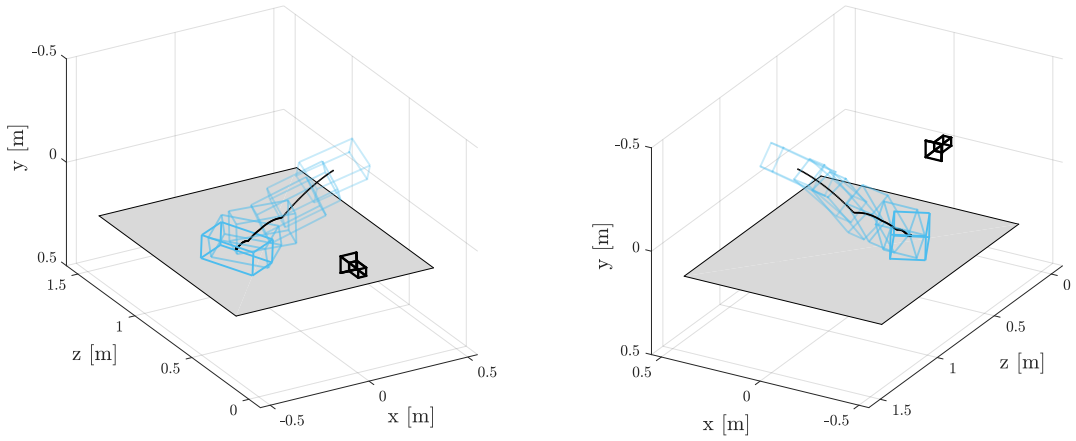


Figure 4.16: 3D trajectory of Box 5 in Video 1, where the black line represents the ground truth trajectory ${}^{A}\mathbf{o}_{B}^{GT}(t)$ and the blue wireframes are ground truth poses of the box at several time instants.

4.3.2 Results Video 1 (Box 5)

Recorded video and 3D trajectory

We will first discuss Video 1 of Box 5 that collides with the conveyor belt. The length of this video is 19 frames and it is captured with a frame rate of 30 fps, so the average time between two frames is 0.033 s. In Figure 4.15, the most important frames of this video are shown. The impacts occur at frames 5, 7, 9, and 12, and are therefore included in Figure 4.15.

To show the motion of the box in 3D space more clearly, Figure 4.16 is created. In this figure, the black line represents the ground truth trajectory ${}^{A}\mathbf{o}_{B}^{GT}(t)$ and for several time instants the ground truth pose of the box is depicted.

Measurement results

As mentioned in Section 4.2.3, the two measurements that we will consider during the performance evaluation of the GUPF are $\mathbf{L}_t$ and $\mathbf{Z}_t$. Measurement $\mathbf{L}_t$ is used as a benchmark to see whether implementing the new measurement $\mathbf{Z}_t$ leads to improvement of the tracking performance. $\mathbf{L}_t$ is the result of computing the approximate likelihood for the predicted set of particles and taking the weighted mean to obtain the pose parameters. $\mathbf{Z}_t$ is also obtained by making use of the approximate likelihood function, however first an object detector is used to determine the location of the box in the image. Figure 4.17 shows the norm of the position and orientation error for the measurements $\mathbf{L}_t$ and $\mathbf{Z}_t$.

The RMS value of the norm of the position error is 0.026 m for $\mathbf{L}_t$ and 0.084 m for $\mathbf{Z}_t$, and the RMS value of the norm of the orientation error is 10.5 deg for $\mathbf{L}_t$ and 11.6 deg for $\mathbf{Z}_t$. Even though the RMS values for $\mathbf{L}_t$ are lower than for $\mathbf{Z}_t$, a disadvantage of $\mathbf{L}_t$ can be seen in the right plot of Figure 4.17. After a couple of impacts have occurred, the norm of the orientation error starts growing, while the graph of $\mathbf{Z}_t$ decreases. This is due to the fact that the measurement $\mathbf{L}_t$ heavily depends on the predicted set of particles and thus on the motion model. When the set of particles is not close to the true box anymore, the estimated pose will differ a lot from the ground truth pose and the measurement will be less accurate. For measurement $\mathbf{Z}_t$, the bounding boxes provided by the object detector are used to localize the box in the image, so there is no dependency on the motion model and the created pose particles will always be close to the true box.

The measurement noise covariance $\boldsymbol{R} \in \mathfrak{p} \otimes \mathfrak{p}$ for $\mathbf{L}_t$ is set by hand, based on the accuracy of the approximate likelihood function. For $\mathbf{Z}_t$, the measurement noise covariance can be computed,

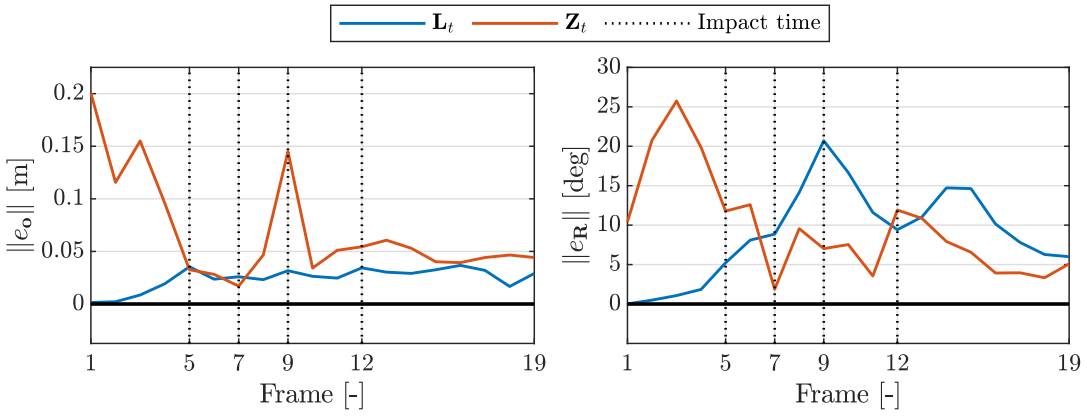


Figure 4.17: Norm of the position and orientation error of the measurements $\mathbf{L}_t$ and $\mathbf{Z}_t$ for Video 1 of Box 5.

assuming that it is Gaussian distributed noise. In Section 4.3.1, an explanation is given about the computation of the covariance for a given signal. The reason why the measurement noise covariance for $\mathbf{Z}_t$ can be computed is that this measurement is obtained beforehand with a separate algorithm. This is not the case for $\mathbf{L}_t$, which is obtained while the object tracking algorithm is running. Note that $\boldsymbol{R}$ is the measurement noise covariance used in the generalization of the unscented particle filter to Lie groups, in contrast to $\boldsymbol{P}_t^n$ in the Euclidean case, as discussed in Section 2.1.4. The measurement noise covariance for $\mathbf{L}_t$ and $\mathbf{Z}_t$ are denoted by $\boldsymbol{R}_L$ and $\boldsymbol{R}_Z$, respectively, and their elements are set as

$$\boldsymbol{R}_L = 1 \cdot 10^{-3}\text{diag}([10 \;\; 10 \;\; 10 \;\; 1 \;\; 1 \;\; 1])$$

and

$$\boldsymbol{R}_Z = 1 \cdot 10^{-5}\text{diag}([2510 \;\; 3230 \;\; 1490 \;\; 31 \;\; 160 \;\; 7]),$$

where the first three elements correspond to the orientation and the last three to the position.

Comparing the particle filters

In this section, particle filters with different motion models and different filtering techniques are compared to each other to see the influence of taking into account complex dynamics in the motion model and to see the effect of incorporating a measurement of the pose parameters of the box.

Before comparing the particle filters, the process noise covariance $\boldsymbol{Q} \in \mathfrak{s} \otimes \mathfrak{s}$ is discussed, which can be considered as the trust in the motion model. $\boldsymbol{Q}$ is again different from $\boldsymbol{P}_t^m$ used in the Euclidean case. In our case, the process noise covariance for the GUPF-NS-L and GUPF-NS-Z is chosen to be

$$\boldsymbol{Q} = 1 \cdot 10^{-5}\text{diag}([10 \;\; 10 \;\; 10 \;\; 1 \;\; 1 \;\; 1 \;\; 1 \;\; 1 \;\; 1 \;\; 1 \;\; 1 \;\; 1]),$$

where the first three elements correspond to the orientation, the fourth to sixth element to the position, the seventh to ninth to the linear velocity, and the tenth to twelfth to the angular velocity. The values for the orientation are set higher than the values for the position, because the orientation of the box is slightly harder to estimate. After an impact, the orientation changes more than the position, so it is beneficial for the accuracy of the estimation to have particles with slightly different orientations, which is regulated with the elements of $\boldsymbol{Q}$. Adjusting the values for the angular velocity does not lead to improved tracking results.

Firstly, the PF-CV, PF-NS, and GUPF-NS-L are compared. In Figure 4.18, the x-, y-, and z-components of ${}^A\mathbf{o}_B$ for the different particle filters are shown, besides the ground truth trajectory and the impact times. The ordinary particle filter with constant velocity motion model, PF-CV, is

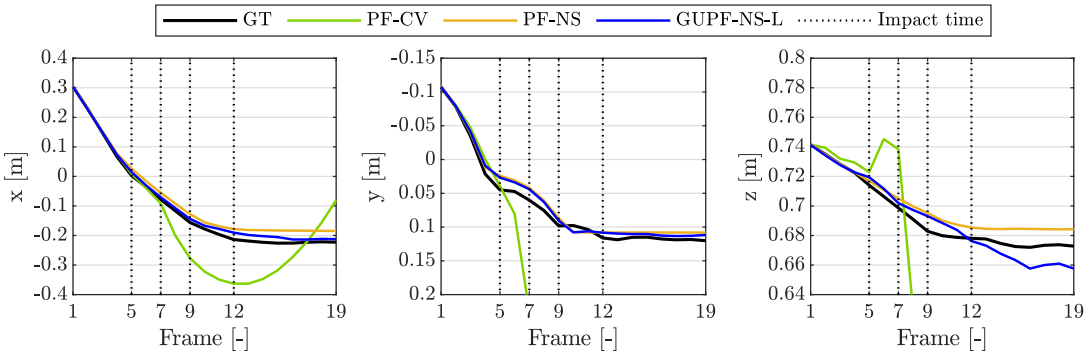


Figure 4.18: Results of the position estimation with the PF-CV, PF-NS, and GUPF-NS-L for Video 1 of Box 5. The results for the x-, y-, and z-position can be seen in the left, middle, and right plot, respectively. The ground truth trajectory and impact times are also shown in the plots.

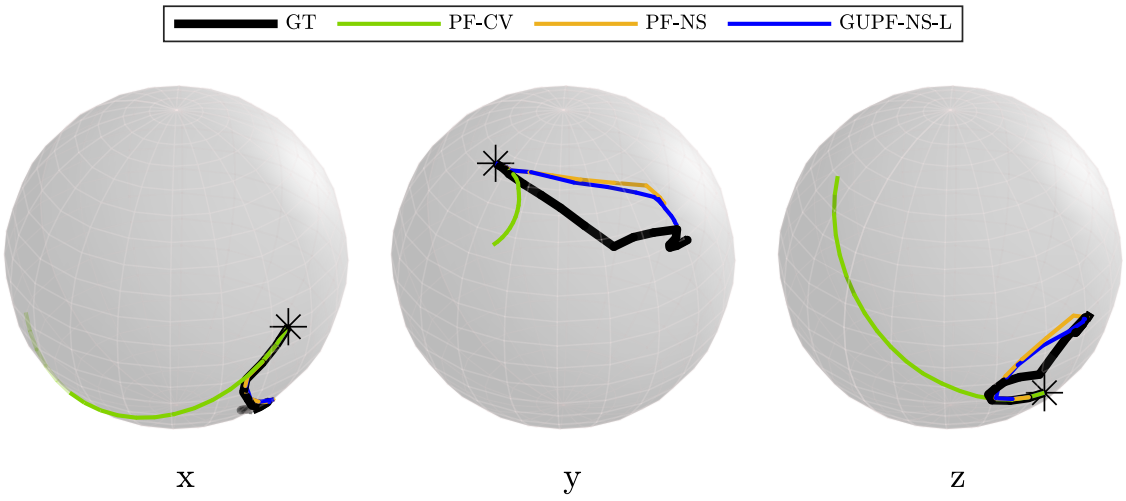


Figure 4.19: Results of the orientation estimation with the PF-CV, PF-NS, and GUPF-NS-L for Video 1 of Box 5. The graphs are the result of plotting ${}^{A}\mathbf{R}_{B}(t)v_j$ on the unit sphere with $\{v_j\}_{j=1}^{3}$, as given by (4.20). From left to right: v_1, v_2, and v_3. The asterisk indicates the initial frame.

able to estimate the position of the box fairly accurately before impacts occur, however after the first impact, it clearly loses track of the box. On the other hand, the output of the particle filter with the nonsmooth motion model, PF-NS, stays close to the ground truth trajectory. The performance of the GUPF-NS-L is marginally better than the performance of the PF-NS for the estimation of the position of the box, since measurement $\mathbf{L}_t$ moves the particles closer to the area with high likelihood.

To visualize the estimation of the orientation by the different algorithms, the unit spheres in Figure 4.19 are created. The graphs are the result of plotting ${}^{A}\mathbf{R}_{B}(t)v_j$, where v_j, with $j = 1, 2, 3$, is given by

$$v_1 = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}^T, \quad v_2 = \begin{bmatrix} 0 & 1 & 0 \end{bmatrix}^T, \quad v_3 = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T. \tag{4.20}$$

This visualization thus shows the trajectory of the tip of the unit vectors of coordinate frame B. Just as for the position estimation, the results for the PF-CV diverge from the ground truth orientation of the box after a couple of frames. The PF-NS and GUPF-NS-L are again almost equal when it comes to the performance of the orientation estimation, with the GUPF-NS-L being slightly better.

In Figure 4.20, the norm of the position and orientation error are shown for the PF-CV, PF-NS,

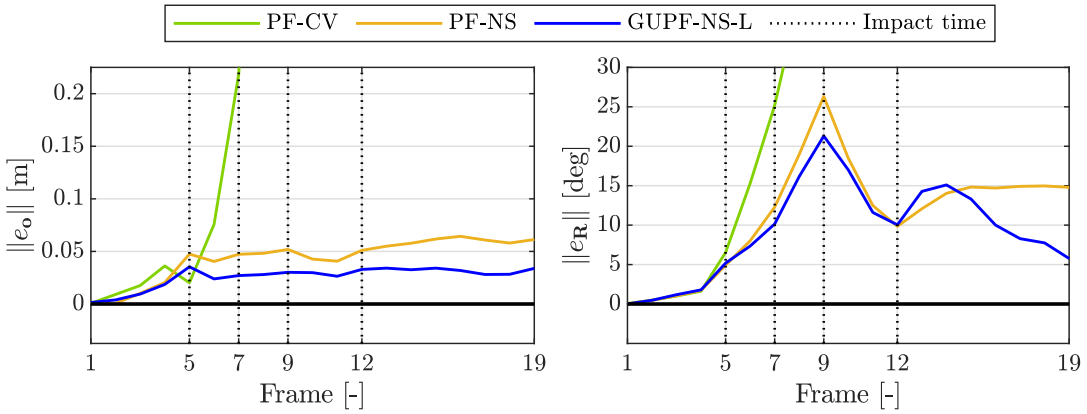


Figure 4.20: Norm of the position and orientation error for the PF-CV, PF-NS, and GUPF-NS-L for Video 1 of Box 5.

Table 4.1: 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-CV	1.237	-	70.7	-
PF-NS	0.048	96.2	13.2	81.3
GUPF-NS-L	0.028	97.8	11.0	84.4

and GUPF-NS-L. Both the position and orientation error of the PF-CV increase quickly after the first impact at frame 5, which is not the case for the PF-NS and GUPF-NS-L. The orientation error for the GUPF-NS-L decreases after frame 14, where the error of the PF-NS stays constant. This is due to the measurement $\mathbf{L}_t$ that moves the particles closer to the box in the image. As can be seen in Figure 4.15, the box almost lies still after frame 14, so the particles in the PF-NS will hardly rotate anymore. For the GUPF-NS-L this is not the case, because $\mathbf{L}_t$ is used in each frame to steer the particles in the right direction.

The RMS values of the norm of the position and orientation error can be found in Table 4.1. Next to the RMS values, also the reduction with respect to the state-of-the-art particle filter, PF-CV, is given. By taking into account the complex dynamics in the motion model, the position error and orientation error are reduced by 96.2% and 81.3%, respectively. If also the measurement $\mathbf{L}_t$ is used, the errors are even reduced by 97.8% and 84.4%.

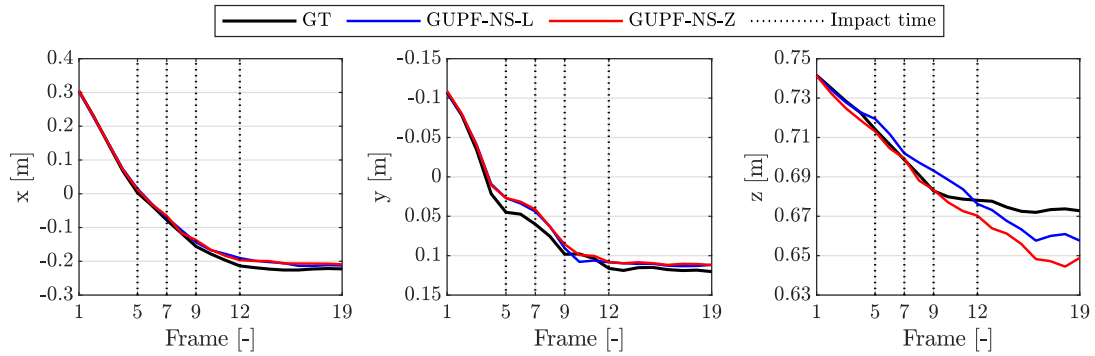


Figure 4.21: Results of the position estimation with the GUPF-NS-L and GUPF-NS-Z for Video 1 of Box 5. The results for the x-, y-, and z-position can be seen in the left, middle, and right plot, respectively. The ground truth trajectory and impact times are also shown in the plots.

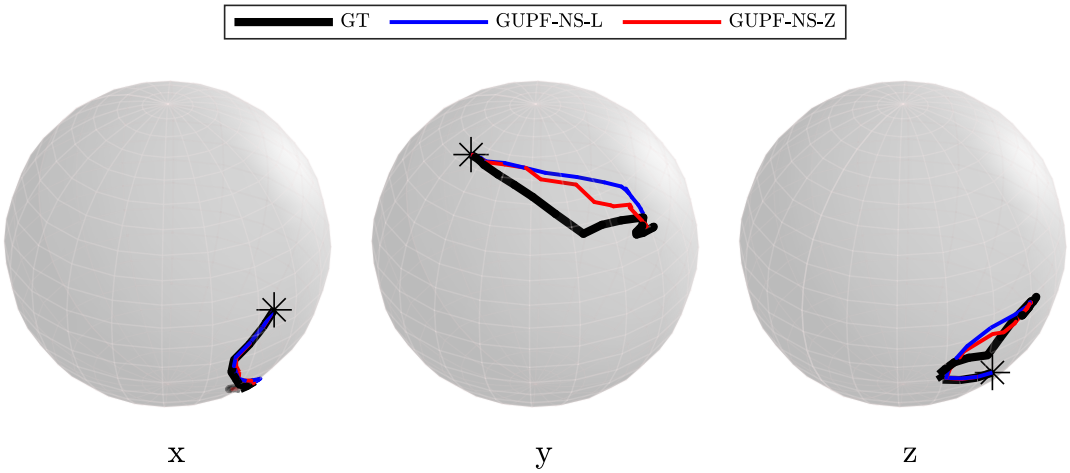


Figure 4.22: Results of the orientation estimation with the GUPF-NS-L and GUPF-NS-Z for Video 1 of Box 5. The graphs are the result of plotting ${}^{A}\mathbf{R}_{B}(t)v_j$ on the unit sphere with $\{v_j\}_{j=1}^{3}$, as given by (4.20). From left to right: v_1, v_2, and v_3. The asterisk indicates the initial frame.

We will now compare the GUPF-NS-L and GUPF-NS-Z, to see the influence of the measurement $\mathbf{Z}_t$, obtained from the newly proposed method. In Figure 4.21, again the results of the estimation of the position are shown, but only for the GUPF-NS-L and GUPF-NS-Z. Both filters are able to estimate the position well, only a small difference between estimation and ground truth is visible for the z-position. The accuracy of the estimation of the z-position decreases after frame 9 for the GUPF-NS-Z, which can be explained by looking at Figure 4.17, where a peak can be seen at frame 9 for the norm of the position error. The pose estimation algorithm has difficulty with estimating the position from frame 7 to frame 9, due to the out-of-plane rotation of the box. Because of the peak in the error, particles are moved a little further away from the true box, leading to a less accurate estimation.

The results for the orientation estimation are shown in Figure 4.22. The output of the GUPF-NS-Z is in this case closer to the ground truth than the output of the GUPF-NS-L. This observation can again be explained easily by looking at Figure 4.17, as the norm of the orientation error of $\mathbf{Z}_t$ is for many frames lower than the error of $\mathbf{L}_t$.

To draw the final conclusions, the norm of the position and orientation error are plotted and a similar table to Table 4.1 is given. Figure 4.23 shows the norm of the position and orientation error

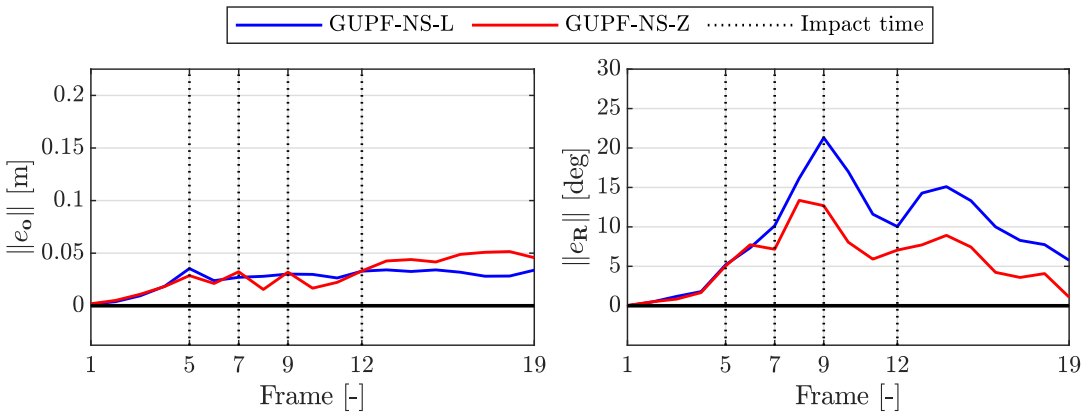


Figure 4.23: Norm of the position and orientation error for the GUPF-NS-L and GUPF-NS-Z for Video 1 of Box 5.

Table 4.2: 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 [%]
GUPF-NS-L	0.028	-	11.0	-
GUPF-NS-Z	0.033	-20.7	6.8	38.4
$\mathbf{Z}_t$	0.084	-	11.6	-

for the GUPF-NS-L and GUPF-NS-Z and Table 4.2 shows the RMS values and the error reduction of $\|e_{\mathbf{o}}\|$ and $\|e_{\mathbf{R}}\|$ with respect to the GUPF-NS-L. The RMS value of the norm of the position error is 5 mm smaller for the GUPF-NS-L, however the orientation is clearly estimated more accurately with the GUPF-NS-Z. A reduction of 38.4% is achieved for the orientation error with respect to the GUPF-NS-L. The measurement $\mathbf{Z}_t$ is also included in Table 4.2, to show that, in this case, the GUPF-NS-Z leads to more accurate tracking results than the pose estimation algorithm discussed in Section 4.2.2.

For the video of Box 5, the GUPF-NS-Z shows the best tracking results. In Figure 4.24, the same frames as in Figure 4.15 are depicted, but also the output of the GUPF-NS-Z is projected onto the image. This is done to visualize the difference between the most accurate particle filter and the true box.

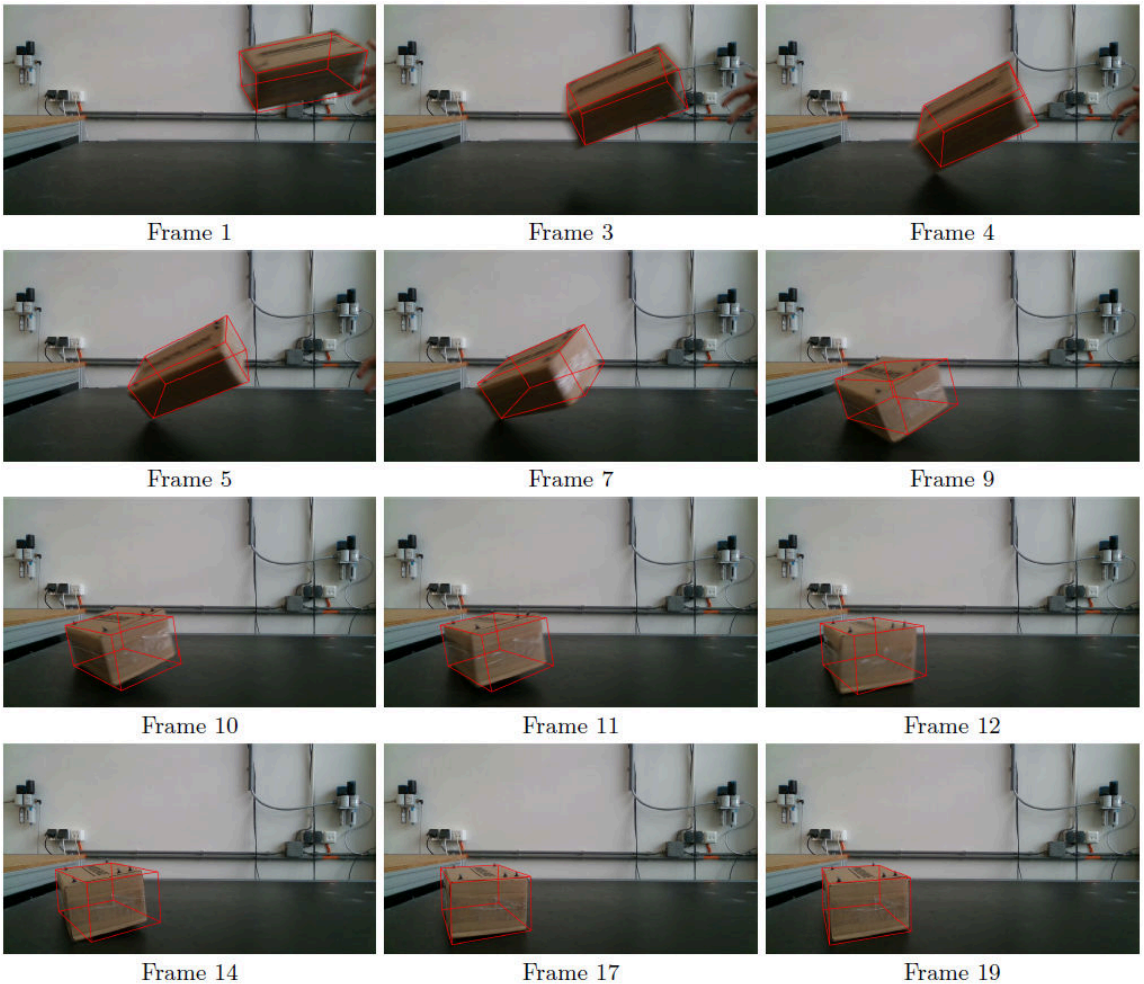


Figure 4.24: Output of the GUPF-NS-Z for Video 1 of Box 5 projected onto the image to visualize the difference between the estimation and the true box.

4.3.3 Results Video 2 (Box 3)

In this section, the results for Video 2 of Box 3 are discussed. Box 3 is smaller and tumbles more than Box 5, so it will be more challenging for the particle filters to accurately estimate the state of the box. This section will have the same structure as Section 4.3.2.

Recorded video and 3D trajectory

The video of Box 3, shown in Figure 4.25, has a length of 30 frames and the impacts occur at frame 3, 5, 8, and 21. In comparison to Video 1 of Box 5, the orientation of the box changes relatively much after an impact. Due to this quick change of orientation, motion blur occurs, as can be seen in frame 4. The trajectory of Box 3 in 3D space is depicted in Figure 4.26.

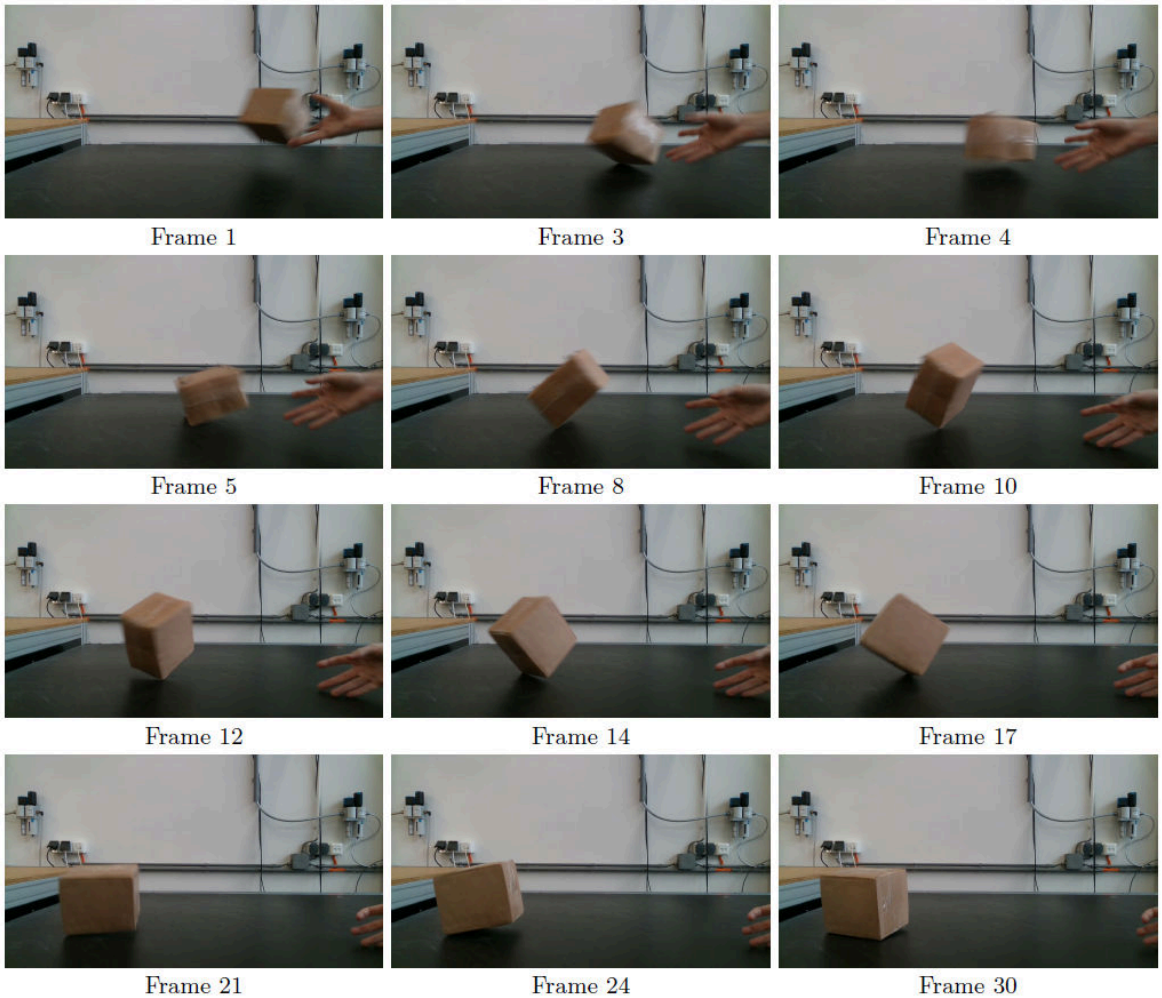


Figure 4.25: Most important frames of Video 2 of Box 3. The video has a length of 30 frames and impacts with the conveyor belt occur at frame 3, 5, 8, and 21.

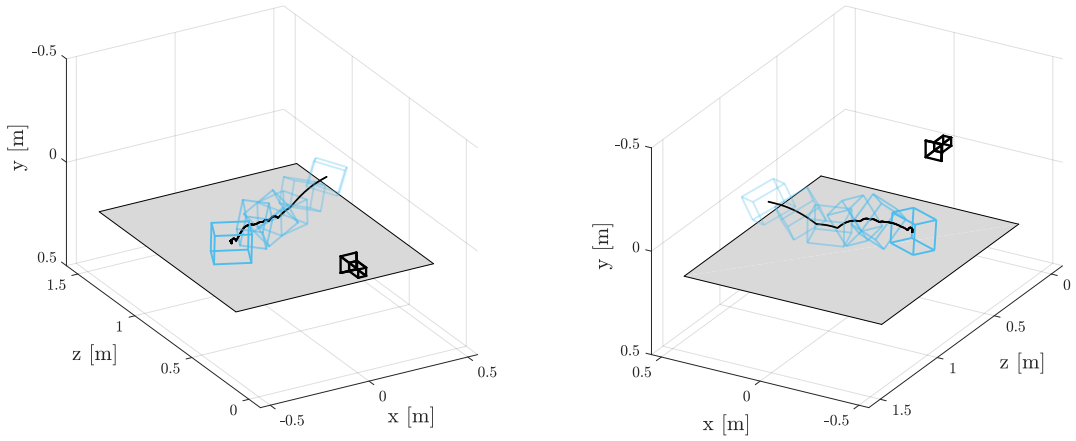


Figure 4.26: 3D trajectory of Box 3 in Video 2, where the black line represents the ground truth trajectory ${}^{A}\mathbf{o}_{B}^{GT}(t)$ and the blue wireframes are ground truth poses of the box at several time instants.

Measurement results

In Figure 4.27, the norm of the position and orientation error for the measurements $\mathbf{L}_t$ and $\mathbf{Z}_t$ can be seen. The RMS value of the norm of the position error is 0.037 m for $\mathbf{L}_t$ and 0.066 m for $\mathbf{Z}_t$, and the RMS value of the norm of the orientation error is 37.8 deg for $\mathbf{L}_t$ and 18.3 deg for $\mathbf{Z}_t$.

As discussed in Section 4.3.2, the motion model has a large influence on the measurement $\mathbf{L}_t$. In this case, the position of the particles is close to the true box for all frames, however the orientation starts to differ after frame 10. Once the particles start rotating in the wrong direction, the correct orientation is not recovered and the error keeps on growing. If the motion model does not describe the motion accurately enough at a specific frame, the situation of Figure 4.27 may occur. Because the measurement $\mathbf{Z}_t$ is obtained without using the motion model, such a problem will not arise and even if the orientation differs much from the ground truth orientation for a certain frame, the pose estimator is able to correct.

Around frame 5, a large increase in the norm of the orientation error can be seen for $\mathbf{Z}_t$. This peak is the result of motion blur in frame 4, which makes it hard to estimate the orientation of the box with the approximate likelihood function. This shows the importance of preventing motion blur as much as possible. Almost no motion blur occurs in the frames of the video of Box 5, therefore the

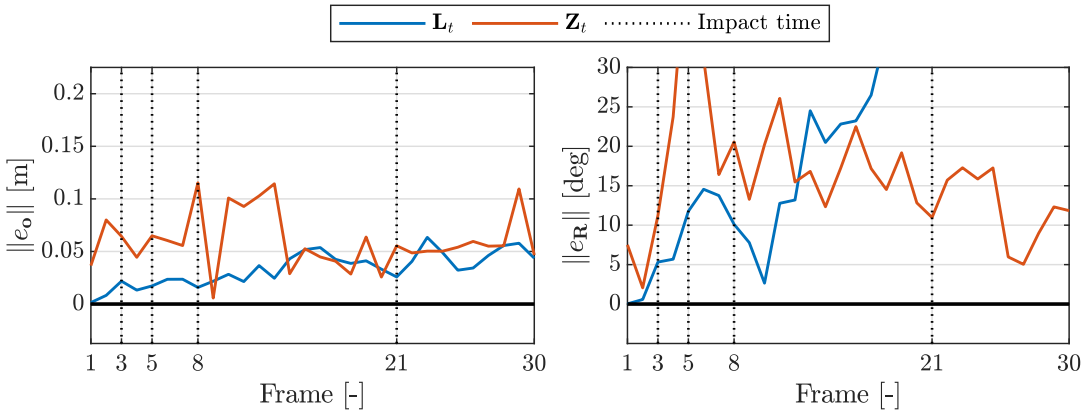


Figure 4.27: Norm of the position and orientation error of the measurements $\mathbf{L}_t$ and $\mathbf{Z}_t$ for Video 2 of Box 3.

orientation error is much lower in Figure 4.17. The measurement noise covariance for $\mathbf{L}_t$ is set as

$$\boldsymbol{R}_L = 1 \cdot 10^{-2}\text{diag}([10 \;\; 10 \;\; 10 \;\; 1 \;\; 1 \;\; 1])$$

and the computed measurement noise covariance for $\mathbf{Z}_t$ is given by

$$\boldsymbol{R}_Z = 1 \cdot 10^{-5}\text{diag}([4910 \;\; 6050 \;\; 5980 \;\; 11 \;\; 93 \;\; 4]),$$

where the order of magnitude for $\boldsymbol{R}_L$ is now 10^{-2}, instead of 10^{-3}. This value has been changed to make sure that the output of the GUPF-NS-L depends less on the computed measurement $\mathbf{L}_t$, which slightly improves the results when the particles start moving away from the area with high likelihood.

Comparing the particle filters

Similar to Section 4.3.2, first the PF-CV, PF-NS, and GUPF-NS-L are compared and afterwards the GUPF-NS-L and GUPF-NS-Z. The process noise covariance for the GUPF-NS-L and GUPF-NS-Z is given by

$$\boldsymbol{Q} = 1 \cdot 10^{-5}\text{diag}([100 \;\; 100 \;\; 100 \;\; 1 \;\; 1 \;\; 1 \;\; 1 \;\; 1 \;\; 1 \;\; 1 \;\; 1 \;\; 1]),$$

where the elements corresponding to the orientation are now chosen to be 10 times larger, because this box tumbles more than Box 5. Again, the values for the angular velocity are not adjusted, as this has no significant influence on the tracking performance.

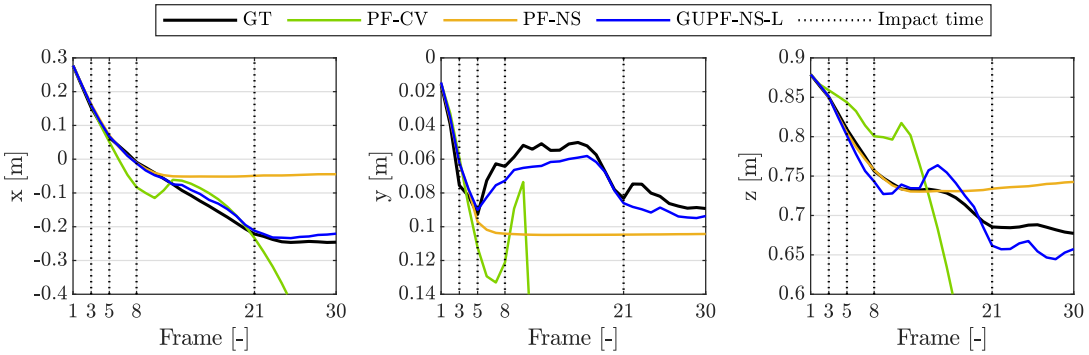


Figure 4.28: Results of the position estimation with the PF-CV, PF-NS, and GUPF-NS-L for Video 2 of Box 3. The results for the x-, y-, and z-position can be seen in the left, middle, and right plot, respectively. The ground truth trajectory and impact times are also shown in the plots.

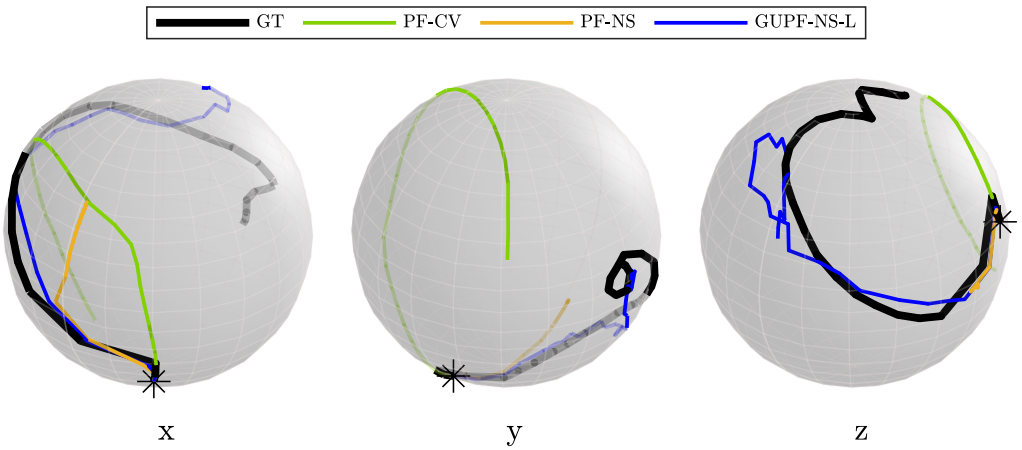


Figure 4.29: Results of the orientation estimation with the PF-CV, PF-NS, and GUPF-NS-L for Video 2 of Box 3. The graphs are the result of plotting ${}^{A}\mathbf{R}_B(t)v_j$ on the unit sphere with $\{v_j\}_{j=1}^{3}$, as given by (4.20). From left to right: v_1, v_2, and v_3. The asterisk indicates the initial frame.

Figure 4.28 shows the results of the estimation of the x-, y-, and z-position of the box. As can be seen, the PF-CV again loses track of the box after the first impact at frame 3. The PF-NS is able to stay closer to the ground truth trajectory than the PF-CV, but after frame 5 the estimation of the y-position is very inaccurate. Due to the nonsmooth motion model, the PF-NS does not completely lose the box after the first impact, however when the box starts tumbling, it is really necessary to make use of a measurement to move the particles closer to the area with high likelihood. When taking into account a measurement, which is done for the GUPF-NS-L, the position can be estimated accurately for all frames.

In Figure 4.29, the results for the orientation estimation can be found. Only for the first frames the PF-CV estimates the orientation well, thereafter the box is lost, just as for the position estimation. The PF-NS stays closer to the ground truth for a couple more frames, but loses track when the box start tumbling. The best results are once again achieved with the GUPF-NS-L, although the estimation of the orientation is also not accurate for half of all frames.

To compactly visualize the results shown in Figure 4.28 and 4.29, the norm of the position and orientation error are depicted in Figure 4.30. Table 4.3 states the RMS values of $\|e_{\mathbf{o}}\|$ and $\|e_{\mathbf{R}}\|$ and the reduction with respect to the PF-CV. As we also concluded from Figure 4.28, the GUPF-NS-L is clearly better at estimating the position than the PF-CV and PF-NS. A reduction of 95.5% is achieved with respect to the PF-CV. None of the particle filters is able to accurately estimate the orientation of the box for all frames of the recording. The poor performance of the GUPF-NS-L can be explained by the inaccurate measurement $\mathbf{L}_t$ in Figure 4.27. This measurement rotates the particles in the wrong direction and the particle filter is not able to recover.

We will now investigate the results by making use of measurement $\mathbf{Z}_t$. The position and orientation estimation plots are given in Figure 4.31 and 4.32. Not a large difference between the GUPF-NS-L and GUPF-NS-Z is observable for the position plots, but from the orientation plots can be derived

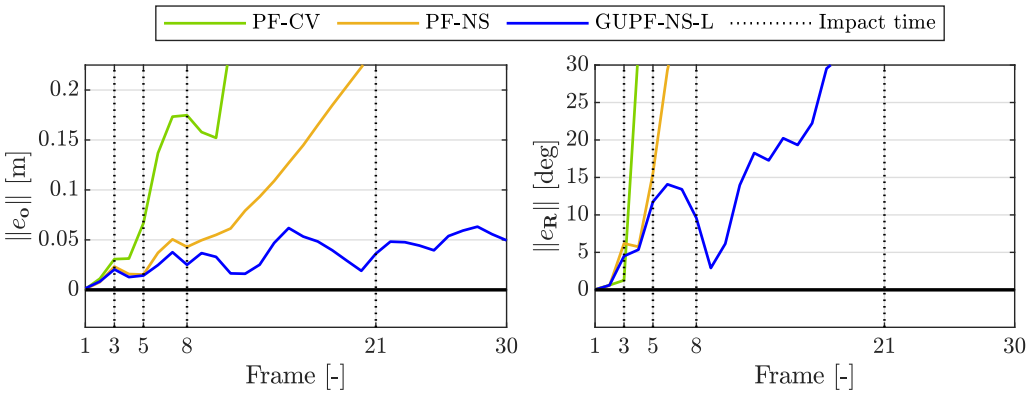


Figure 4.30: Norm of the position and orientation error for the PF-CV, PF-NS, and GUPF-NS-L for Video 2 of Box 3.

Table 4.3: 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-CV	0.875	-	110.1	-
PF-NS	0.179	79.6	117.2	-6.4
GUPF-NS-L	0.039	95.5	38.3	65.2

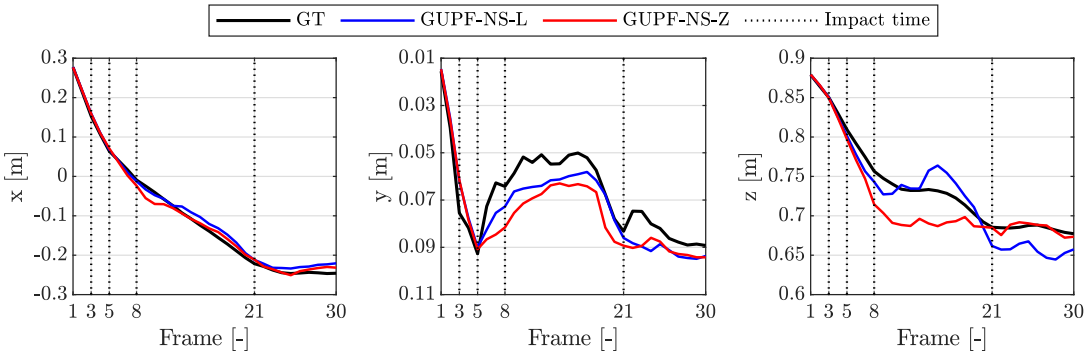


Figure 4.31: Results of the position estimation with the GUPF-NS-L and GUPF-NS-Z for Video 2 of Box 3. The results for the x-, y-, and z-position can be seen in the left, middle, and right plot, respectively. The ground truth trajectory and impact times are also shown in the plots.

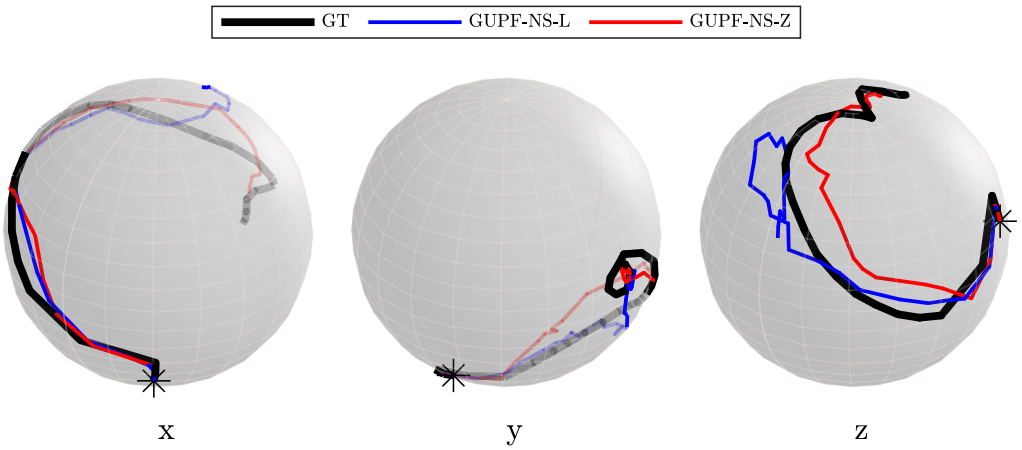


Figure 4.32: Results of the orientation estimation with the GUPF-NS-L and GUPF-NS-Z for Video 2 of Box 3. The graphs are the result of plotting ${}^{A}\mathbf{R}_{B}(t)v_j$ on the unit sphere with $\{v_j\}_{j=1}^{3}$, as given by (4.20). From left to right: v_1, v_2, and v_3. The asterisk indicates the initial frame.

that the GUPF-NS-Z is superior. The graphs for the GUPF-NS-Z end at roughly the same place as the ground truth graphs, which indicates that the algorithm is able to follow the box to the last frame.

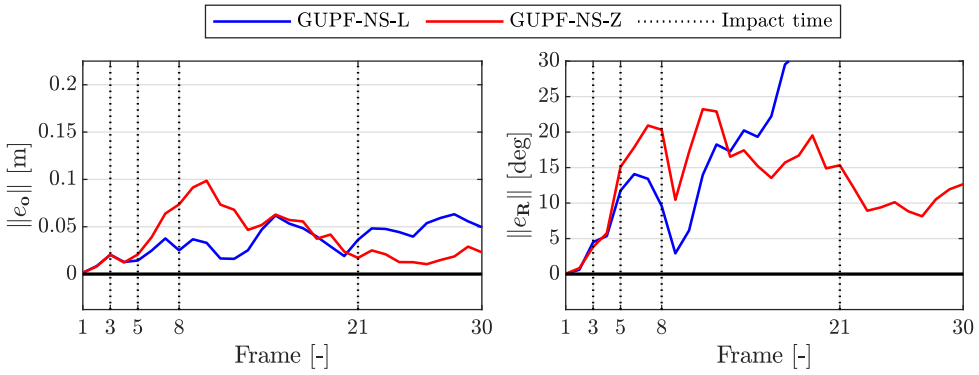


Figure 4.33: Norm of the position and orientation error for the GUPF-NS-L and GUPF-NS-Z for Video 2 of Box 3.

Table 4.4: 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 [%]
GUPF-NS-L	0.039	-	38.3	-
GUPF-NS-Z	0.046	-16.1	14.4	62.3
$\mathbf{Z}_t$	0.066	-	18.3	-

The same conclusions can be drawn from Figure 4.33. Looking at the RMS values in Table 4.4, the GUPF-NS-Z has a slightly worse accuracy than the GUPF-NS-L for the position estimation, yet a far better accuracy for the orientation estimation. It could even be said that using the pose estimation algorithm introduced in Section 4.2.2 is a better choice than using the GUPF-NS-L to estimate the orientation. On the other hand, the GUPF-NS-Z improves upon both the position and orientation estimation of measurement $\mathbf{Z}_t$. What also can be derived from the results of GUPF-NS-Z is that it is not necessary to have a very accurate measurement to estimate the state well, as it can even be done with a noisy measurement. For the GUPF-NS-L it is important that the predicted set of particles stays close to the true box in all frames, otherwise it loses track of the box and is unable to recover.

Because the GUPF-NS-Z is the best performing particle filter, its output is again projected onto the image as shown in Figure 4.34. The largest differences between the true box and the estimation can be seen in frame 10 and 12, which corresponds to the plots in Figure 4.33.

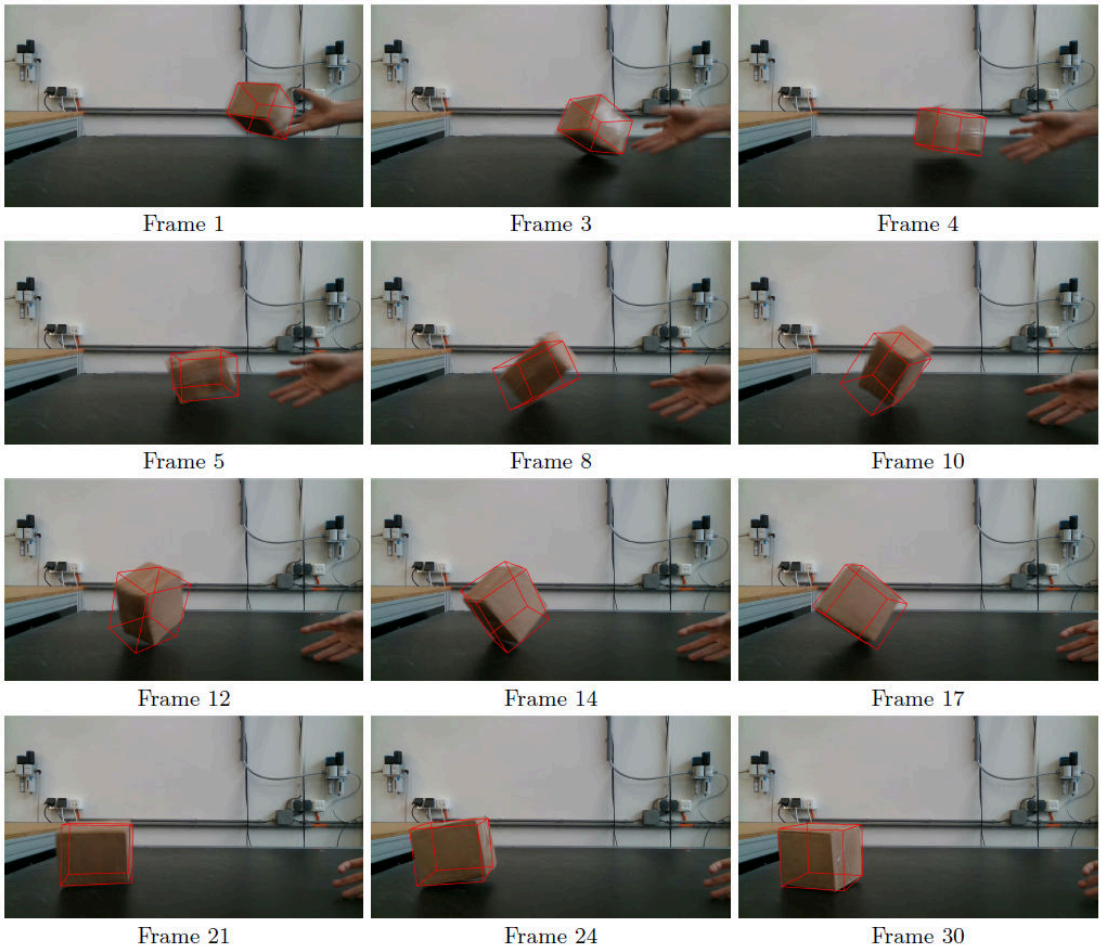


Figure 4.34: Output of the GUPF-NS-Z for Video 2 of Box 3 projected onto the image to visualize the difference between the estimation and the true box.

4.3.4 Conclusion performance evaluation GUPF

To evaluate the performance of the GUPF, three videos of different boxes that collide with a surface are considered. Since the results are similar for each video, we can draw some clear conclusions. The state-of-the-art particle filter with a constant velocity motion model, PF-CV, always loses track of the box after the first impact, which is due to the fact that the motion model does not take into account complex dynamics, such as friction and impacts. When comparing the PF-CV and PF-NS, it can be seen that making use of a nonsmooth motion model improves the tracking results significantly, as the particles stay close to the true box after impacts. The position is estimated fairly accurately with the PF-NS, however the accuracy of the orientation estimation is lower, as can be seen in Figure 4.20, 4.30, and C.6 in Section 4.3.2, 4.3.3, and Appendix C, respectively. Especially for cases where the box tumbles several times, the PF-NS has difficulties estimating the orientation. Using a measurement to help steering the particles closer to the area with high likelihood improves both the estimation of the position and orientation of the box, as can be derived from the tracking results for the GUPF-NS-L and GUPF-NS-Z (Figure 4.23, 4.33, and C.9 in Section 4.3.2, 4.3.3, and Appendix C, respectively). The GUPF-NS-L is slightly better at estimating the position, but the GUPF-NS-Z is considerably better at estimating the orientation. A disadvantage of the GUPF-NS-L is that the measurement depends on the motion model. When the motion model does not describe the movement accurately, the measurement will be inaccurate and the particles are moved in the wrong direction. Figure 4.33 in Section 4.3.3 shows the consequence of an inaccurate measurement (Figure 4.27) on the estimation of the orientation by the GUPF-NS-L. The measurement used in the GUPF-NS-Z is obtained with a separate algorithm, therefore it is not influenced by the motion model.

Every implemented object tracking algorithm has a different computational time. The PF-CV and PF-NS are the fastest algorithms and only need 14 and 30 seconds to estimate the state of the box in each frame of Video 2 of Box 3, respectively. Video 2 is taken as an example as this is the longest of the three recorded videos. On the other hand, the GUPF-NS-L and GUPF-NS-Z both need approximately 26 minutes to estimate the state in each frame, which is significantly longer than the PF-CV and PF-NS need. Because the PF-NS, GUPF-NS-L, and GUPF-NS-Z all make use of the same nonsmooth motion model, it can be concluded that the unscented Kalman filter is the cause of the longer computational times. The measurements $\mathbf{L}_t$ and $\mathbf{Z}_t$ are incorporated in the unscented Kalman filter to create a proposal distribution that is close to the area with high likelihood. It must be mentioned that the code has not been optimized, so it is possible to reduce the computational time considerably.

4.4 Conclusion

In this chapter, the main contributions of this work are presented. The performance of the approximate likelihood computation is evaluated on real images, from which is concluded that the pose of the box can be estimated accurately with the approximate likelihood function. The accuracy of the x- and y-position estimation is approximately 3 mm for boxes at a distance of 0.55 m and 0.8 m from the camera, and 4 mm for a distance of 1.2 m. For the estimation of the z-position, the accuracy is 10 mm, 15 mm, and 20 mm for the distances of 0.55 m, 0.8 m, and 1.2 m from the camera, respectively. The orientation can be estimated with an accuracy of approximately 5 degrees for all distances. Besides this, a new method that makes use of an object detector and the approximate likelihood function is proposed to obtain a measurement of the pose parameters of the box. Finally, the GUPF is applied and validated on real-life videos and outperforms the state-of-the-art particle filter. By making use of the new measurement $\mathbf{Z}_t$, the GUPF becomes more robust than when the measurement is obtained from the predicted set of particles. In the next and final chapter, a conclusion is given and recommendations are proposed for future research.

Chapter 5

Conclusion and recommendations

As stated in Section 1.3, the goal of this research is to validate the developed object tracking algorithm proposed in [6] on real-life videos and to improve the tracking performance. Before the algorithm could be validated and improved, first the videos of boxes colliding with the conveyor belt had to be recorded and the ground truth data had to be obtained. This is done with the RealSense D415 camera and OptiTrack motion capture system, respectively. The next step was to evaluate the performance of the approximate likelihood computation, to make sure that the pose of the box in real images could be estimated accurately by making use of color histograms. Computing the approximate likelihood of particles is not only an important process in the state estimation with a particle filter, but we also use it to obtain a measurement of the pose parameters of the box. An object detector is employed to localize and identify the box in the image and afterwards the approximate likelihood function is applied to obtain the 3D pose of the box. Once the approximate likelihood computation was evaluated and the measurement was obtained, the Geometric Unscented Particle Filter (GUPF) could be applied on the recorded videos to estimate the state of the box in each frame. By comparing the results with the ground truth data, conclusions regarding the accuracy of the state estimation could be drawn. To see the influence of incorporating the nonsmooth motion model and the measurement $\mathbf{Z}_t$, the PF-CV, PF-NS, GUPF-NS-L, and GUPF-NS-Z are compared. In this chapter, the conclusions of this research are stated and recommendations for further improvements are given.

5.1 Conclusion

In Section 1.3, three subgoals were defined that had to be reached to accomplish the main goal of this research. These three subgoals will be discussed in the following sections.

5.1.1 Approximate likelihood computation

The subgoal formulated for the approximate likelihood computation is:

Evaluate the performance of the likelihood computation based on color histograms, as in [6], on real images containing parcels with a uniform color and, if necessary, come up with solutions or another method to increase performance.

In [6], a geometric model with sets of points is projected onto the image to compute the approximate likelihood for a particle. Each visible point falls on a specific pixel and by making use of the information of these pixels, color histograms are computed. High values for the approximate likelihood are assigned to particles that are close to the true pose of the object. For synthetic images containing a cuboid with a distinctive color on each face and a background with a uniform color, the approximate likelihood function is able to estimate the pose very accurately. In our case, the images contain a box with a uniform color and a background with many different colors. Furthermore, motion blur may be visible in the images. It was unknown whether the method used in [6] also produces meaningful results for real images instead of synthetic images. Therefore the performance of the approximate likelihood computation is evaluated.

As the cuboid in the work of Jongeneel [6] has a distinctive color on each face, six reference color histograms were used. Carton boxes usually have a uniform color and we will thus only use a single reference color histogram. Besides this, a couple of other adjustments are done. The dimensions of the geometric model are chosen to be equal to those of the boxes and the distance between points on and around the geometric model are tuned, depending on the size of the box and the motion blur in the image. No changes are made to the scaling and weighting parameters in the distance and likelihood function.

Three test images with boxes at different distances from the camera are considered for the evaluation of the performance of the approximate likelihood computation. Sets of pose particles are created close to the ground truth pose of the box in order to see how accurately the position and orientation can be estimated. From the results can be concluded that the x- and y-position of the box can be estimated with an accuracy of ± 3 mm when it is located at a distance of 0.55 m and 0.8 m from the camera, and an accuracy of ± 4 mm for a distance of 1.2 m. The further away the box is from the camera, the harder it is to accurately estimate the position. This is due to the fact that the box is represented by less pixels, so neighbouring sample points will fall on the same pixel, leading to non-representative color histograms. As is also concluded in [6], estimating the z-position is difficult, especially when the box is far away from the camera. The reason for this is that the representation of the pose particles in the image only changes slightly when they are far away from the camera, resulting in high approximate likelihood values for pose particles in a large range of z-positions. Nevertheless, the z-position can still be estimated with an accuracy of ± 10, ± 15, and ± 20 mm for the box being at 0.55, 0.8, and 1.2 m from the camera, respectively.

For the rotation of the pose particles about the x, y, and z-axis of frame B, the accuracy of the orientation estimation is ± 5 degrees for all test images, with one exception. The approximate likelihood for in-plane rotations of pose particles is more discriminating than for out-of-plane rotations. This is again due to the fact that the representation of the pose particles only changes marginally for out-of-plane rotations. Next to the rotations about the x, y, and z-axis, we also looked at many other rotations in 3D space. These results are visualized with the approximate likelihood π-ball, from which can be concluded that pose particles close to the ground truth orientation and pose particles rotated -180 and 180 degrees about axes near the x, y, and z-axis have the highest value for the approximate likelihood.

The approximate likelihood function is also able to accurately estimate the position of the box in an image with motion blur, however the accuracy of the orientation estimation is slightly worse. This can be derived from the π-ball, which shows high values for the approximate likelihood for rotations about many other axes than the x, y, and z-axis. Since the accuracy of the pose estimation is comparable to the accuracy for synthetic images, the approximate likelihood function is considered to be accurate enough for both images with and without motion blur. It was thus not necessary to come up with another method to increase the performance and the subgoal has been achieved.

5.1.2 Object detection and 3D pose estimation

The subgoal formulated for the object detection and pose estimation is:

Extend the developed object tracking algorithm in [6] with a state-of-the-art object detector and 3D pose estimator, which are used to obtain a measurement of the pose parameters of the parcel.

Jongeneel [6] has used two different methods to obtain the measurement $\mathbf{z}_t$. The first method computes the approximate likelihood for all particles in the predicted set and subsequently the weighted mean is computed to obtain the pose parameters. The other method adds Gaussian noise to the ground truth data to simulate a measurement that can be obtained from an image-processing algorithm. Developing an image-processing algorithm was beyond the scope of [6], however in this work such an algorithm is proposed where use is made of an object detector and the approximate likelihood function to obtain the measurement. Due to the object detector, we are always able to sample pose

particles close to the true box in the image, which is not the case when the motion model is used. If the motion model does not describe the movement accurately enough, the particles move away from the area with high likelihood. The newly proposed method is thus more robust than the employed method in [6].

The Single Shot MultiBox Detector (SSD) is used to localize and identify the box in the image. As the COCO dataset does not contain images of carton boxes, the SSD is not able to classify them in the images, so it is necessary to finetune the MobileNetV2 base network. This is done by training the neural network again after creating an extra dataset of images containing carton boxes with different poses, different lighting conditions, and motion blur. Once the finetuning has been performed, the SSD is able to detect the boxes in every test image. The bounding boxes provided by the object detector are then used for the estimation of the pose.

Obtaining the pose of the box is done in two steps, where the approximate likelihood function plays a crucial role. First, the position is estimated by making use of the centers of the bounding boxes and sampling pose particles at different distances from the camera. The position of the pose particle with the highest value for the approximate likelihood is then saved and used for the estimation of the orientation. Just as for the π-balls, the pose particle is rotated about many axes with different angles. Again, the pose particle with the highest value for the approximate likelihood is saved. We have now determined an estimate of the pose parameters of the box in an image. This procedure is followed in every frame of a recording to obtain the measurement $\mathbf{z}_t$. The orientation of the pose particle from previous frame is used as the initial orientation in the current frame, except for the first frame. During the position and orientation estimation stages, 5733 and 3751 pose particles are evaluated, respectively. For all frames the number of pose particles is the same.

To see if the measurement from the newly proposed method is an improvement over the measurement used in [6], the results are compared. The measurement $\mathbf{z}_t$ obtained with the object detector and approximate likelihood function is denoted by $\mathbf{Z}_t$ and the measurement obtained from the predicted set of particles is denoted by $\mathbf{L}_t$. For all videos, the norm of the position error is lower for $\mathbf{L}_t$, however the norm of the orientation error for $\mathbf{Z}_t$ is lower on average. The main advantage of measurement $\mathbf{Z}_t$ is that it is obtained independently from the motion model. This means that the particles always stay close to the box, which is not the case for $\mathbf{L}_t$. Once the particles start moving away because of an inaccuracy in the motion model, especially the norm of the orientation error of $\mathbf{L}_t$ increases and the overall tracking performance is influenced. It is even possible that the particle filter is not able to recover the correct orientation of the box and the error of the output keeps increasing until the last frame. We can conclude that a more robust method has been developed to obtain the measurement of the pose parameters of the box and that the subgoal has thus been achieved.

5.1.3 Evaluation and improvement of the state estimation

The subgoal formulated for the evaluation and improvement of the state estimation is:

Estimate the state of the parcel in 3D space at any point in time from a real-life video obtained with a single RGB camera and implement modifications in case tracking is unsatisfying in the continuous and/or in the discrete portions.

Three videos of different boxes that collide with a surface are considered during this evaluation in order to test the object tracking algorithms on multiple scenarios. For each video, the ground truth pose of the box and camera are determined with the OptiTrack motion capture system and the tracking results obtained from the particle filters are compared to the ground truth to check the accuracy. By comparing the PF-CV and PF-NS, the conclusion can be drawn that incorporating the nonsmooth motion model developed in [6] improves the tracking results significantly. The reason for this is that the impact dynamics are taken into account, which is not the case for the constant velocity motion model. After an impact occurs, the PF-CV loses track of the box in all videos. Only using a particle filter with a nonsmooth motion model is often not enough to accurately track a colliding

object. Therefore, a measurement of the pose parameters of the box is used in the unscented Kalman filter to move the particles closer to the area with high likelihood. The GUPF-NS-L and GUPF-NS-Z are both particle filters with a nonsmooth motion model and make use of the measurement $\mathbf{L}_t$ and $\mathbf{Z}_t$, respectively. When the PF-NS and GUPF-NS-L are compared, it can clearly be seen that using a measurement, which can even be noisy, improves the performance of the object tracking algorithm. As discussed before, the problem with measurement $\mathbf{L}_t$ is that it is only accurate when the motion model describes the movement accurately and the particles stay close to the box. This is not always the case, especially when the box tumbles much. The tracking results of the GUPF-NS-L and GUPF-NS-Z show an equal accuracy for the estimation of the position of the box, however the orientation estimation is better for the GUPF-NS-Z. An easy explanation for this observation is that the norm of the orientation error of $\mathbf{Z}_t$ is lower than that of $\mathbf{L}_t$ for many frames of a recording. The situation where the norm of the orientation error keeps on growing also does not occur for the GUPF-NS-Z, because $\mathbf{Z}_t$ is obtained independently of the motion model. It can thus be concluded that the GUPF-NS-Z with the measurement from the newly proposed method is an improvement over the GUPF-NS-L from [6] and the subgoal has been achieved.

Since all the formulated subgoals of the research are reached, the main goal of validating the GUPF on real-life videos and implementing modifications to increase tracking performance is also accomplished. In [6], it was already concluded that the GUPF is a significant improvement over the state-of-the-art particle filter in simulation, however it was still necessary to validate the object tracking algorithm on real-life videos. The validation has been done in this work with multiple boxes and also an image-processing algorithm has been developed from which the measurement $\mathbf{Z}_t$ is obtained. The resulting GUPF-NS-Z is a more robust object tracking algorithm than the GUPF-NS-L and shows a better performance in estimating the orientation of the box in the frames of a recording. There are still points of improvement that will increase the tracking accuracy of both the GUPF-NS-L and GUPF-NS-Z. These points will be discussed in next section.

5.2 Recommendations

Just as in previous section, the subjects of the approximate likelihood computation, object detection and 3D pose estimation, and the state estimation are discussed.

5.2.1 Approximate likelihood computation

Even though the pose of the box can be estimated accurately with the approximate likelihood function, some adjustments could be made to the function or the settings of the camera to improve the performance. The most important factor that limits the accuracy is the occurrence of motion blur in the images. By increasing the shutter speed of the camera and having a better illumination of the scene, motion blur can be reduced. If there is no motion blur, the distance of the points to the edges of the geometric model can be reduced, leading to a more peaky distribution of the approximate likelihood. This could not be done for images with motion blur, as a lot of particles close to the ground truth would get the same value for the approximate likelihood.

In this work we used a relatively high resolution, which makes the estimation of the pose easier. When lower resolutions are used and the box is far away from the camera, multiple points may fall on the same pixel, leading to non-representative color histograms. A solution is changing the number of points as a function of the size of the box in the image, as is done in [62]. Once there is no motion blur in the images anymore, the weighting and scaling parameters could be tuned for better results, or even another distance function could be chosen. In [62], shades on the sides of an object are used to get a better estimation of the orientation, which could also be a potential improvement in our case.

5.2.2 Object detection and 3D pose estimation

The object detector is able to localize and classify the boxes in all test images, however sometimes it is the case that the bounding box does not fit tightly around the box. By adding more images of carton boxes to the created dataset, it should be possible to make the bounding boxes fit more tightly. This will improve the performance of the introduced pose estimator. Another thing that will lead to a better estimation of the pose is the improved approximate likelihood function, as discussed in previous section. Reducing motion blur will thus have a positive influence on both the approximate likelihood computation and the measurement of the pose parameters of the box.

Instead of using the approximate likelihood to obtain the measurement, the Augmented Autoencoder (AAE) [86] could be employed. The bounding box provided by the object detector is then used by the AAE to determine the pose of the object. This method is more robust to background clutter than the approximate likelihood function, as it depends on a 3D model of the object, instead of color.

5.2.3 State estimation with the GUPF

There are still many points of improvement to increase the tracking performance of the GUPF. One of these points is that the coefficients of friction and restitution have been chosen manually, but it would be much better to perform a parameter identification for the boxes, as also done in the work of Poort [7]. Another point of improvement is the computational time of the algorithm. The PF-CV and PF-NS only need less than 30 seconds to estimate the state of the box in a video of 30 frames, while the GUPF-NS-L and GUPF-NS-Z both need approximately 26 minutes. By optimizing the code, the computational times of the GUPF-NS-L and GUPF-NS-Z can both be reduced considerably. Another option is to increase the frame rate of the camera, since the motion model simulation will then take a shorter time. If the algorithm is faster, more particles can be used without having an excessive computational time. The more particles, the better the estimation of the state will be.

In real-life applications, it would be necessary to have the GUPF working in real-time. Because the computational time is currently far too high for real-time usage, the algorithm needs to be implemented in `C` or `C++`. Besides that, the object detection and pose estimation algorithm would have to be implemented in the object tracking algorithm, instead of determining the measurement before applying the GUPF.

From the results of the pose estimation algorithm can be derived that its performance is not much worse than the performance of the GUPF. This means that the combination of an object detector with a pose estimator is also able to keep track of the pose of a box that collides with a surface, without using a complex dynamical model. When motion blur is reduced and the approximate likelihood function is improved, an even better performance of the pose estimation algorithm is possible and a lower number of particles could then be used. In general, this method would be a simpler, faster, and possibly more accurate method than the GUPF.

Next to the approximate likelihood function, other modifications could be made to the pose estimator to improve measurement $\mathbf{Z}_t$. After estimating the orientation of the box, an extra step could be taken to refine the estimation of the position. It is also possible to use a depth camera to obtain additional information about the position of the object, which could be used to improve the estimation of the pose. Disadvantages of depth cameras are the limited reach and sensitivity to light and noise, so the accuracy depends a lot on the environment and the application.

Bibliography

[1] A. S. Jalal and V. Singh, "The State-of-the-Art in Visual Object Tracking," *Informatica (Slovenia)*, vol. 36, pp. 227–248, 2012.

[2] T. Acharya and A. K. Ray, *Image Processing: Principles and Applications*. Wiley, 2005.

[3] UDTech, "Computer Vision Solutions in Warehouse Automation," https://udtech.co/blog/computer-vision-solutions-in-warehouse-automation. Accessed: 14-06-2021.

[4] Technavio, "6 Major Types of Industrial Robots Used in the Global Manufacturing 2018," https://blog.technavio.com/blog/major-types-of-industrial-robots. Accessed: 05-07-2021.

[5] Vanderlande, "Smart Item Robotics," https://www.vanderlande.com/systems/picking/smart-item-robotics/. Accessed: 14-06-2021.

[6] M. J. Jongeneel, "Model-Based Visual Object Tracking with Collision Models," Master's thesis, Eindhoven University of Technology, 2020.

[7] L. Poort, "Developing a Numerical Model for Robotic Tossing of Boxes, Enhanced and Validated using Motion Capure Data," Master's thesis, Eindhoven University of Technology, 2020.

[8] V. Lepetit and P. Fua, "Monocular Model-Based 3D Tracking of Rigid Objects: A Survey," *Foundations and Trends in Computer Graphics and Vision*, vol. 1, no. 1, pp. 1–89, 2005.

[9] D. Comaniciu, V. Ramesh, and P. Meer, "Kernel-based object tracking," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 5, pp. 564–577, 2003.

[10] F. Chaumette and S. Hutchinson, "Visual servo control. I. Basic approaches," *IEEE Robotics and Automation Magazine*, vol. 13, no. 4, pp. 82–90, 2006.

[11] W. Bing and L. Xiang, "A Simulation Research on 3D Visual Servoing Robot Tracking and Grasping a Moving Object," in *2008 15th International Conference on Mechatronics and Machine Vision in Practice*, 2008, pp. 362–367.

[12] D. G. Lowe, "Fitting Parameterized Three-Dimensional Models to Images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 13, no. 5, pp. 441–450, 1991.

[13] 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*. Springer Berlin Heidelberg, 2004, pp. 28–39.

[14] 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," *Robotics and Autonomous Systems*, vol. 58, no. 6, pp. 784–795, 2010.

[15] S. Särkkä, *Bayesian Filtering and Smoothing*. Cambridge University Press, 2013.

[16] G. Welch and G. Bishop, "An Introduction to the Kalman Filter," *Proc. Siggraph Course*, vol. 8, 2006.

[17] E. A. Wan and R. Van Der Merwe, "The Unscented Kalman Filter for Nonlinear Estimation," in *Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium*, 2000, pp. 153–158.

[18] 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*. MIT Press, 2000, pp. 563–569.

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

[20] W. A. Hoff, K. Nguyen, and T. Lyon, "Computer Vision-Based Registration Techniques for Augmented Reality," *Proceedings of SPIE - The International Society for Optical Engineering*, vol. 2904, 1996.

[21] H. Kato and M. Billinghurst, "Marker tracking and HMD calibration for a video-based augmented reality conferencing system," in *Proceedings 2nd IEEE and ACM International Workshop on Augmented Reality (IWAR'99)*, 1999, pp. 85–94.

[22] 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, 1999, pp. 262–268.

[23] P. Lanvin, J. C. Noyer, and M. Benjelloun, "An hardware architecture for 3D object tracking and motion estimation," in *2005 IEEE International Conference on Multimedia and Expo*, 2005, pp. 4 pp.–.

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

[25] M. Juřík, V. Šmídl, J. Kuthan, and F. Mach, "Trade-off Between Resolution and Frame Rate for Visual Tracking of Mini-robots on Planar Surfaces," in *2019 International Conference on Manipulation, Automation and Robotics at Small Scales (MARSS)*, 2019, pp. 1–6.

[26] 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, 2002.

[27] A. Doucet and A. Johansen, "A Tutorial on Particle Filtering and Smoothing: Fifteen Years Later," *Handbook of Nonlinear Filtering*, vol. 12, 2009.

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

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

[30] A. Doucet and N. J. Gordon, "Simulation-based optimal filter for maneuvering target tracking," in *Signal and Data Processing of Small Targets 1999*, vol. 3809. SPIE, 1999, pp. 241–255.

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

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

[33] J. D. Hol, "Resampling in particle filters," Linköpings universitet, Department of Electrical Engineering, Tech. Rep., 2004.

[34] 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)*, 2015, pp. 2454–2461.

[35] 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, 2002.

[36] 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. SPIE, 1997, pp. 182–193.

[37] Y. Wu, D. Hu, M. Wu, and X. Hu, "Unscented Kalman Filtering for Additive Noise Case: Augmented versus Nonaugmented," *IEEE Signal Processing Letters*, vol. 12, pp. 357–360, 2005.

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

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

[40] S. Traversaro and A. Saccon, *Multibody Dynamics Notation (Version 3)*. Eindhoven University of Technology, 2021.

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

[42] 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.

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

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

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

[46] G. Dubbelman, "Intrinsic statistical techniques for robust pose estimation," PhD thesis, University of Amsterdam, 2014.

[47] 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, 2014.

[48] 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," *International Journal of Robotics Research*, vol. 31, no. 4, pp. 498–519, 2012.

[49] J. Kwon and F. C. Park, "Visual Tracking via Particle Filtering on the Affine Group," in *International Conference on Information and Automation*, 2008, pp. 997–1002.

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

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

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

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

[54] D. Forsyth and J. Ponce, *Computer Vision: A Modern Approach*. Prentice Hall, 2011.

[55] NaturalPoint, "OptiTrack - Motion Capture Systems," https://optitrack.com/. Accessed: 14-06-2021.

[56] Intel, "RealSense Depth Camera D415," https://www.intelrealsense.com/depth-camera-d415/.

[57] NaturalPoint, "OptiTrack Prime 17W - Technical Specifications," https://optitrack.com/cameras/prime-17w/specs.html. Accessed: 14-06-2021.

[58] MathWorks, "Single Camera Calibrator App," https://www.mathworks.com/help/vision/ug/single-camera-calibrator-app.html. Accessed: 14-06-2021.

[59] Z. Zhang, "A Flexible New Technique for Camera Calibration," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 11, pp. 1330–1334, 2000.

[60] S. S. Beauchemin and R. Bajcsy, "Modelling and Removing Radial and Tangential Distortions in Spherical Lenses," in *Multi-Image Analysis*. Springer Berlin Heidelberg, 2001, pp. 1–21.

[61] Form X, "Sculpture block," https://www.formx.nl/clays/sculpture-block/index.php. Accessed: 14-06-2021.

[62] 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)*, 2008.

[63] L. Zhong and L. Zhang, "A Robust Monocular 3D Object Tracking Method Combining Statistical and Photometric Constraints," *International Journal of Computer Vision*, pp. 1–20, 2018.

[64] 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*. Springer Berlin Heidelberg, 2007, pp. 171–182.

[65] 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, 2000, pp. 142–149.

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

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

[68] R.I. Leine and H. Nijmeijer, *Dynamics and Bifurcations of Non-Smooth Mechanical Systems*, ser. Lecture Notes in Applied and Computational Mechanics. Springer-Verlag Berlin Heidelberg, 2004, vol. 18.

[69] P. D. Panagiotopoulos and C. Glocker, "Analytical Mechanics. Addendum I: Inequality Constraints with Elastic Impacts. The Convex Case," *ZAMM Zeitschrift fur Angewandte Mathematik und Mechanik*, vol. 78, no. 4, pp. 219–229, 1998.

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

[71] 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, 2004, pp. 3232–3237.

[72] C. Glocker, "Formulation of spatial contact situations in rigid multibody systems," *Computer Methods in Applied Mechanics and Engineering*, vol. 177, pp. 199–214, 1999.

[73] M. Anitescu and F. Potra, "A time-stepping method for stiff multibody dynamics with contact and friction," *International Journal for Numerical Methods in Engineering*, vol. 55, pp. 753–784, 2002.

[74] 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*. Springer Netherlands, 2004, pp. 101–125.

[75] B. Brogliato, *Nonsmooth Mechanics*, 3rd ed. Springer International Publishing Switzerland, 2016.

[76] R. Leine and C. Glocker, "A set-valued force law for spatial Coulomb–Contensou friction," *European Journal of Mechanics*, vol. 22, no. 2, pp. 193–216, 2003.

[77] N. O. Mahony, S. Campbell, A. Carvalho, S. Harapanahalli, G. Velasco-Hernández, L. Krpalkova, D. Riordan, and J. Walsh, "Deep Learning vs. Traditional Computer Vision," in *CVC*, 2019.

[78] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "SSD: Single Shot Multibox Detector," in *Computer Vision – ECCV 2016*. Springer International Publishing, 2016, pp. 21–37.

[79] S. Arabi, A. Haghighat, and A. Sharma, "A deep learning based solution for construction equipment detection: from development to deployment," *arXiv e-prints*, 2019.

[80] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 6, pp. 1137–1149, 2017.

[81] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications," *arXiv*, 2017.

[82] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: Common Objects in Context," in *Computer Vision – ECCV 2014*. Springer International Publishing, 2014, pp. 740–755.

[83] H. M. D. Kabir, M. Abdar, S. M. J. Jalali, A. Khosravi, A. Atiya, S. Nahavandi, and D. Srinivasan, "SpinalNet: Deep Neural Network with Gradual Input," *arXiv*, 2020.

[84] X. Deng, A. Mousavian, Y. Xiang, F. Xia, T. Bretl, and D. Fox, "PoseRBPF: A Rao–Blackwellized Particle Filter for 6-D Object Pose Tracking," *IEEE Transactions on Robotics*, pp. 1–15, 2021.

[85] Y. K. Baik, J. Kwon, H. S. Lee, and K. M. Lee, "Geometric particle swarm optimization for robust visual ego-motion estimation via particle filtering," *Image and Vision Computing*, vol. 31, no. 8, pp. 565–579, 2013.

[86] M. Sundermeyer, Z.-C. Márton, M. Durner, and R. Triebel, "Augmented Autoencoders: Implicit 3D Orientation Learning for 6D Object Detection," *International Journal of Computer Vision*, vol. 128, pp. 714–729, 2019.

Appendix A

Performance approximate likelihood under motion blur

In this appendix, the performance of the approximate likelihood function is evaluated for an image of a box with motion blur. This image with the corresponding ground truth pose particle is shown in Figure A.1 and the distance from the box to the camera is approximately 0.75 m. We will follow the exact same procedure as in Section 4.1, so we start with determining the accuracy of the position estimation and afterwards we determine the accuracy of the orientation estimation.

A.1 Position estimation

First, a set of pose particles is created in the xy-plane at a fixed distance z. Recall that this set consists of 441 pose particles, which are spaced 1 mm apart in a range of -10 mm to 10 mm. The approximate likelihood is then computed again with (3.22), without changing the scaling and weighting parameters. The results for the approximate likelihood L are shown in Figure A.2. Figure A.3 also shows the similarities S_1, S_2, and S_3 next to the approximate likelihood.

After comparing Figure A.3 with Figures 4.4, 4.5, and 4.6, the conclusion can be drawn that the approximate likelihood is less discriminating for an image with motion blur than for the images without motion blur, which is as expected. Especially the pose particles that are sampled in the x-direction still have a high value for the approximate likelihood when they are far away from the ground truth position. The values for the approximate likelihood can be explained by looking at the similarities. Similarity S_3 is again the dominating factor in the approximate likelihood computation, as low values for S_3 leads to high values for L. What we can also derive from Figure A.3 is that the values for S_1 and S_2 are in general lower and S_3 is less discriminating than for the figures without motion blur.

Even though the approximate likelihood is less discriminating for an image with motion blur, we can still estimate the x- and y-position with an accuracy of $\pm$ 5 mm, which is not much worse than for images without motion blur.

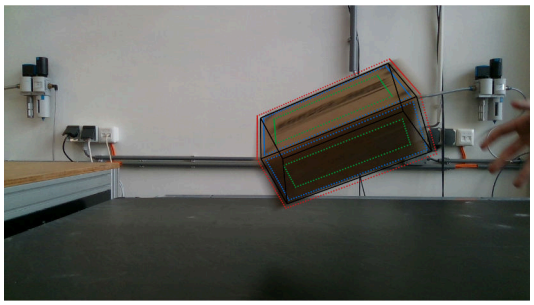


Figure A.1: Test image with motion blur used for the evaluation of the approximate likelihood computation.

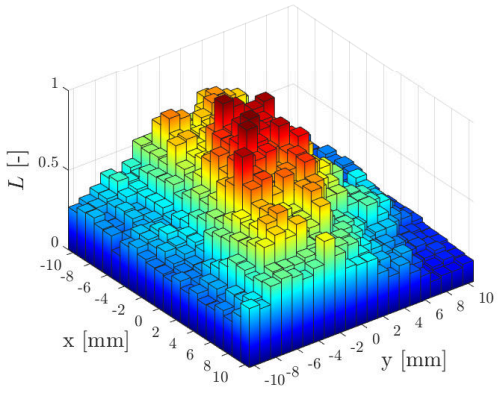
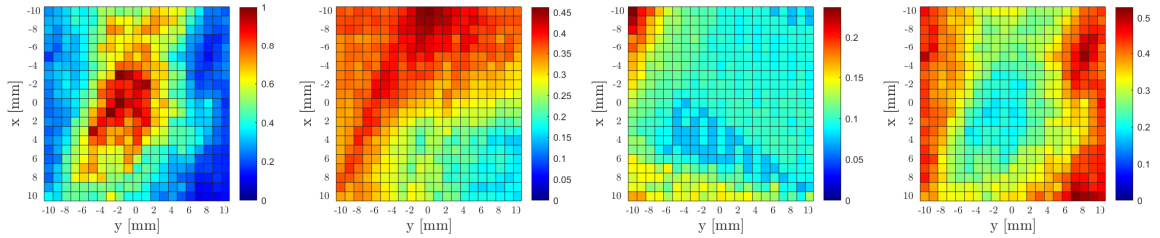


Figure A.2: Approximate likelihood L for the set of pose particles in the xy-plane at a fixed distance z.



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

Figure A.3: Approximate likelihood L **(a)** and similarities S_1 **(b)**, S_2 **(c)**, and S_3 **(d)** for the set of pose particles in the xy-plane at a fixed distance z.

For the estimation of the z-position, a set of 101 pose particles is used. Each pose particle has a fixed x- and y-position, but the z-position is varied with steps of 1 mm in a range of -50 mm to 50 mm. The approximate likelihood L and similarities S_1, S_2, and S_3 for this set of pose particles are given in Figure A.4.

To be able to draw conclusions regarding the accuracy of the estimation of the z-position, we compare Figure A.4 with Figure 4.7b, since the boxes are located at roughly the same distance in front of the camera. Which is also the case for the estimation of the x- and y-position, the approximate likelihood is less discriminating for an image with motion blur. For a distance of -50 mm in front of the ground truth position, L still has a value of 0.4, while this is 0.1 in Figure 4.7b.

When we compare the similarities, S_1 and S_2 are clearly lower in Figure A.4 than in Figure 4.7b, as was also observed for the estimation of the x- and y-position. Although there is no clear minimum for the graph of S_3, still high values of L are assigned to pose particles close to the ground truth position. This shows the importance of having a high value for the weighting parameter κ_3.

The accuracy of the z-position estimation is 15 mm, so comparable to the accuracy without motion blur. From the results in Figure A.3 and Figure A.4 can be concluded that the accuracy of the position estimation of the box with the approximate likelihood function is still high for an image with motion blur. In Section A.2, the performance of the approximate likelihood computation for rotation is discussed.

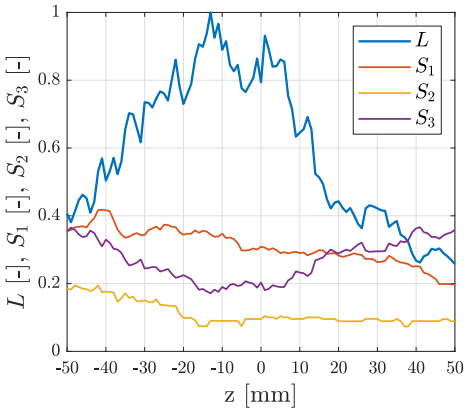


Figure A.4: Approximate likelihood L and similarities S_1, S_2, and S_3 for the set of pose particles with varying z-position.

A.2 Orientation estimation

Similar to Section 4.1.3, we will start with rotating the pose particles about the x-, y-, and z-axis of frame B of the box and computing the values for the approximate likelihood L and the similarities S_1, S_2, and S_3. Afterwards, the π-ball is shown, to check the performance of the approximate likelihood computation for rotations about other axes.

Consider Figure A.5, which has been created by rotating particles with steps of 1 degree in a range of -200 to 200 with respect to the ground truth orientation. It can be seen that the pose particles close to the ground truth orientation and the pose particles rotated -180 and 180 degrees have the highest value for L. The approximate likelihood for the rotation about the x-axis also has peaks near -110 and 70 degrees. The reason for this is that the rotation about the x-axis is an out-of-plane rotation, so the representation of the pose particles in the image almost does not change. It is then possible that low values for S_3, and thus high values for L, are assigned to pose particles with a significantly different orientation than the ground truth orientation. The accuracy of the estimation for rotation about the x-, y-, and z-axis of the box is ± 3 degrees or less, so very accurate.

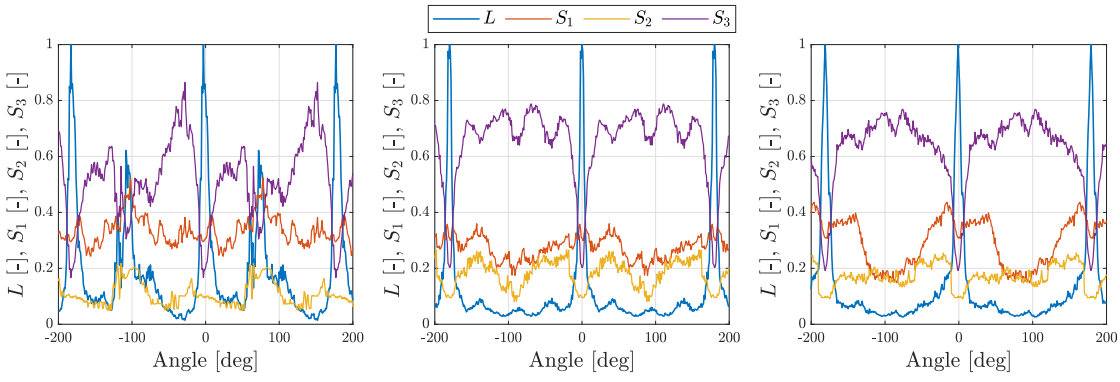


Figure A.5: Approximate likelihood L and similarities S_1, S_2, and S_3 for pose particles rotated with respect to the ground truth orientation of the box in Figure A.1. From left to right: rotation about the x-, y-, and z-axis, respectively.

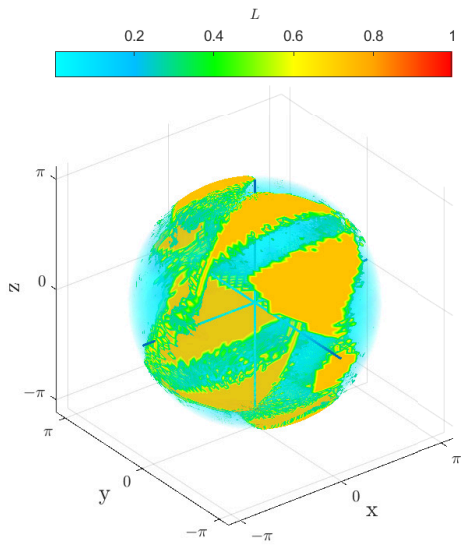


Figure A.6: Approximate likelihood π-ball showing the values of L for many rotations in SO(3) for the box in Figure A.1 with motion blur.

In Figure A.6, the π-ball for the box in Figure A.1 can be found. This π-ball differs a lot from the π-balls in Figure 4.11, as multiple large areas with high values for the approximate likelihood can be seen on the surface of the ball. Rotations of -180 and 180 degrees about the x, y, z, $-y, z$, y, z, and neighbouring axes all lead to high values for L, so the approximate likelihood has difficulties discriminating between different pose particles. Figure A.6 shows the importance of using the π-ball, as a lot of information is missing in Figure A.5.

A.3 Conclusion

The accuracy of the position estimation of the box in an image with motion blur is roughly equal to the accuracy without motion blur. The x- and y-position can be estimated with an accuracy of $\pm$ 5 mm, which is $\pm$ 3 mm for the box in Figure 4.1b. The accuracy of the z-position estimation is 15 mm, so the same as for the box in Figure 4.1b.

In contrast to the position estimation, it is a bit harder to estimate the rotation of the box accurately in an image with motion blur. Due to the blur, the color of the box is smeared over the image, which has an influence on the computed color histograms. High values for the approximate likelihood are then assigned to pose particles that do not exactly correspond to the box in the image. The accuracy is still acceptable, but the optimal solution would be to prevent the occurrence of motion blur in the images.

Appendix B

Annotation of images

As mentioned in Section 4.2.2, an annotated set of images of boxes is needed to make it possible for the SSD to detect carton boxes. This appendix discusses the goal of annotation and how the images are annotated.

B.1 Goal of the annotation

Since the COCO dataset does not contain images of carton boxes, the SSD will not be able to recognize these objects in the test images. To make it possible for the object detector to detect boxes, finetuning of the final layers of the MobileNetV2 base network is necessary. To this end, many images of boxes are collected and a part of the detector is trained again. Only providing images to the neural network is not enough, as it does not understand the content of the image. Therefore, we draw bounding boxes around the carton boxes and assign the label 'box' to them. Once this is done for all the collected images, the training can be performed and the detector should be able to localize and classify the boxes during the inference.

B.2 Bounding boxes and labels

For the annotation of images, the 'LabelImg' tool is used. Bounding boxes can be drawn easily around objects with this tool and subsequently a label can be assigned to the bounding box. The annotation is then saved as an .xml file and has to be provided to the detector next to the original image. In Figure B.1, an example of the annotation procedure is shown. It is important that the bounding box is drawn as tightly as possible, otherwise the detector thinks that parts of the background belong to the carton box, which may affect the detection performance.

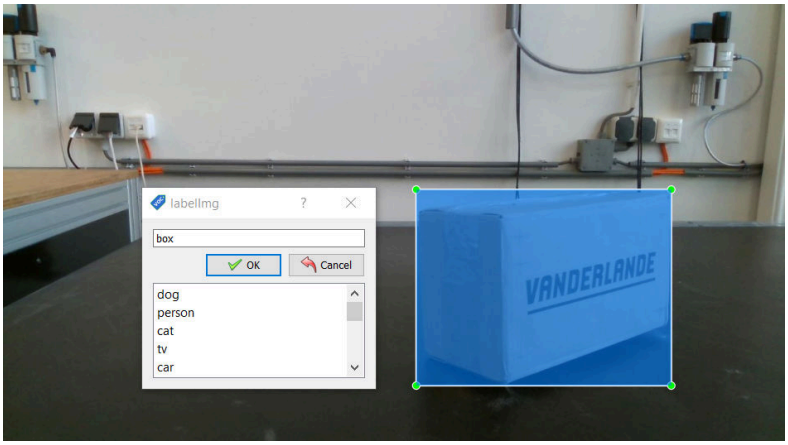


Figure B.1: Visualization of the annotation procedure, which consists of drawing bounding boxes and assigning labels.

Appendix C

Results Video 3 (Box 4)

Video 3, containing Box 4 that collides with the conveyor belt, is not discussed in the main text of this report, since the motion of the box is quite similar to the motion of Box 5. Nevertheless, it is still useful to mention the results for this video, to show that the GUPF is able to accurately estimate the state of Box 4 in all frames, as is also the case for Box 3 and Box 5. This appendix will have the same structure as Section 4.3.2 and 4.3.3.

C.1 Recorded video and 3D trajectory

Video 3 of Box 4 has a length of 23 frames and the most eventful frames are given in Figure C.1. The impacts of the box with the conveyor belt occur at frame 5, 11, and 17. Figure C.2 shows the 3D trajectory of the box.

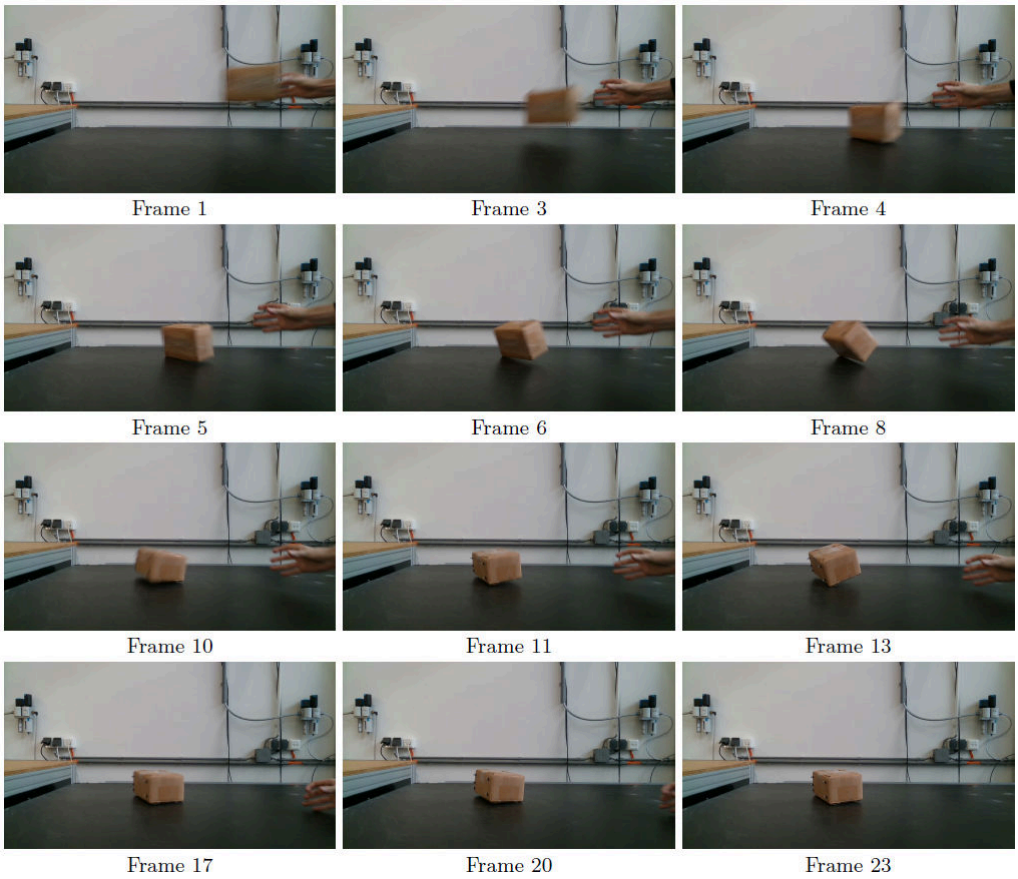


Figure C.1: Most important frames of Video 3 of Box 4. The video has a length of 30 frames and impacts with the conveyor belt occur at frame 5, 11, and 17.

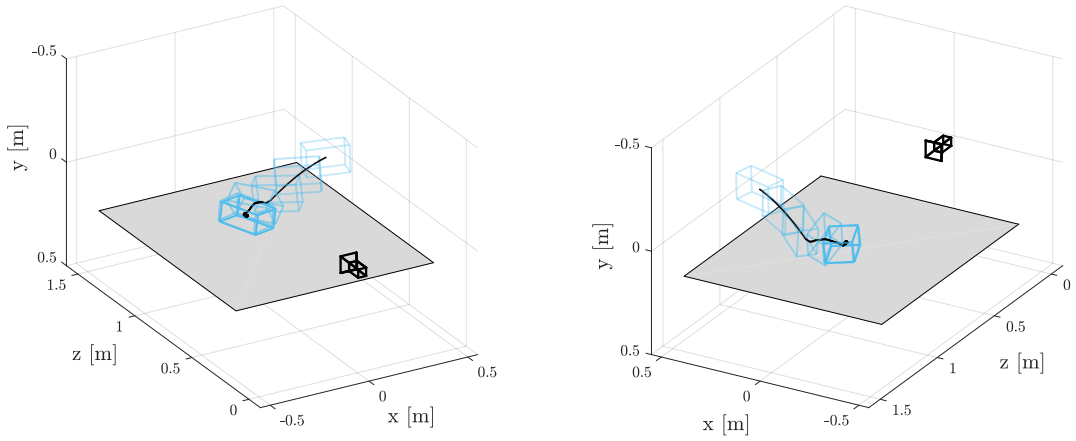


Figure C.2: 3D trajectory of Box 4 in Video 3, where the black line represents the ground truth trajectory ${}^{A}\mathbf{o}_{B}^{GT}(t)$ and the blue wireframes are poses of the box at several time instants.

C.2 Measurement results

As is also done for the video of Box 3 and Box 5, the measurement $\mathbf{L}_t$ and $\mathbf{Z}_t$ are obtained from the methods introduced in Section 4.2. In Figure C.3, the norm of the position and orientation error for the measurements can be found. The RMS value of the norm of the position error is 0.017 m for $\mathbf{L}_t$ and 0.090 m for $\mathbf{Z}_t$, and the RMS value of the norm of the orientation error is 13.6 deg for $\mathbf{L}_t$ and 10.6 deg for $\mathbf{Z}_t$.

In contrast to the results in Figure 4.27, the predicted set of particles stays close to the true box for the entire recording, preventing large peaks in the error plots of $\mathbf{L}_t$. Comparing both measurements, the norm of the position error of $\mathbf{L}_t$ is lower for all frames, while for the norm of the orientation error the opposite holds. For the measurement noise covariance, the two matrices are

$$\boldsymbol{R}_L = 1 \cdot 10^{-2}\text{diag}([10 \;\; 10 \;\; 10 \;\; 1 \;\; 1 \;\; 1])$$

and

$$\boldsymbol{R}_Z = 1 \cdot 10^{-4}\text{diag}([225 \;\; 217 \;\; 163 \;\; 2 \;\; 49 \;\; 1]).$$

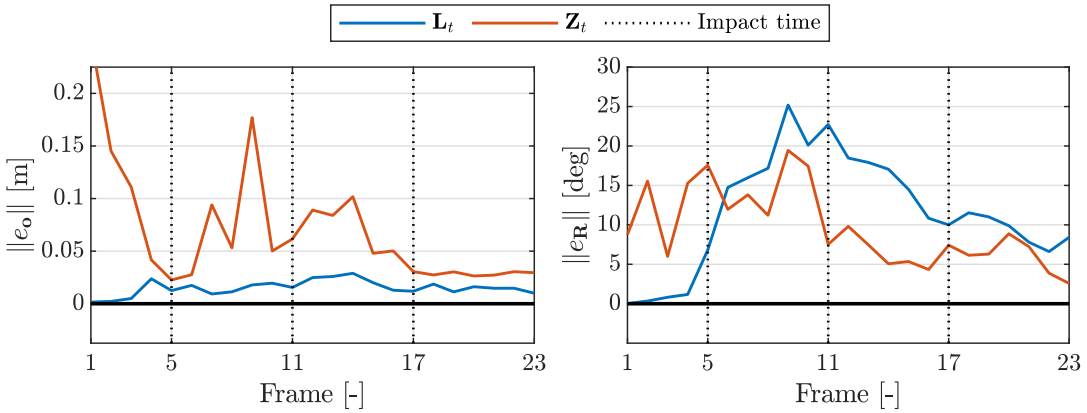


Figure C.3: Norm of the position and orientation error of the measurements $\mathbf{L}_t$ and $\mathbf{Z}_t$ for Video 3 of Box 4.

C.3 Comparing the particle filters

In this section, the position and orientation estimation results are given for the different particle filters, but first the process noise covariance $\boldsymbol{Q}$ is mentioned shortly. Because the motion of Box 4 is similar to that of Box 5, the process noise covariance is also set to

$$\boldsymbol{Q} = 1 \cdot 10^{-5}\text{diag}([10\ \ 10\ \ 10\ \ 1\ \ 1\ \ 1\ \ 1\ \ 1\ \ 1\ \ 1\ \ 1\ \ 1]),$$

which is used for both the GUPF-NS-L and GUPF-NS-Z.

Figure C.4 and C.5 show the estimation of the position and orientation with the PF-CV, PF-NS, and GUPF-NS-L, next to the ground truth trajectory. The graphs look similar to the graphs in Figure 4.18 and 4.19, thus also the same conclusions can be drawn. Until the first impact, the position and orientation results for the PF-CV stay relatively close to the ground truth trajectory, however the constant velocity motion model does not take into account the nonsmooth behaviour, so afterwards the PF-CV loses track of the box. Due to the nonsmooth motion model, the PF-NS and GUPF-NS-L are both able to accurately estimate the position after several impacts, with the GUPF-NS-L being slightly better. This is because of the fact that the measurement $\mathbf{L}_t$ moves the particles closer to the box in the image. For the same reason, the GUPF-NS-L is better at estimating the orientation.

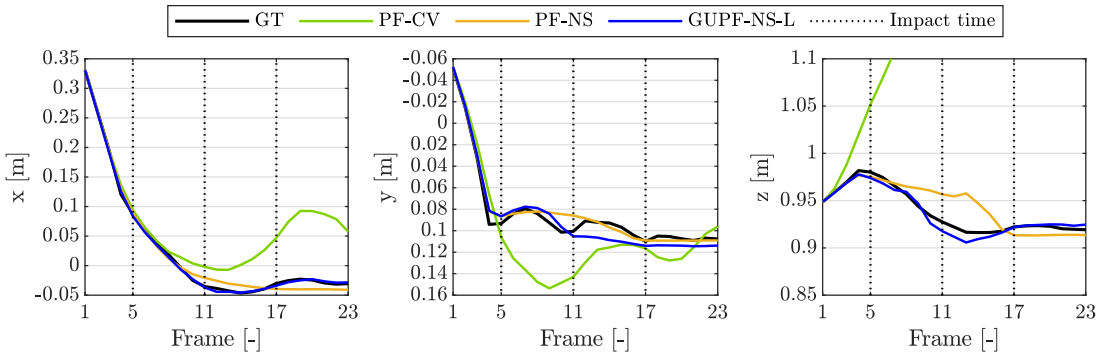


Figure C.4: Results of the position estimation with the PF-CV, PF-NS, and GUPF-NS-L for Video 3 of Box 4. The results for the x-, y-, and z-position can be seen in the left, middle, and right plot, respectively. The ground truth trajectory and impact times are also shown in the plots.

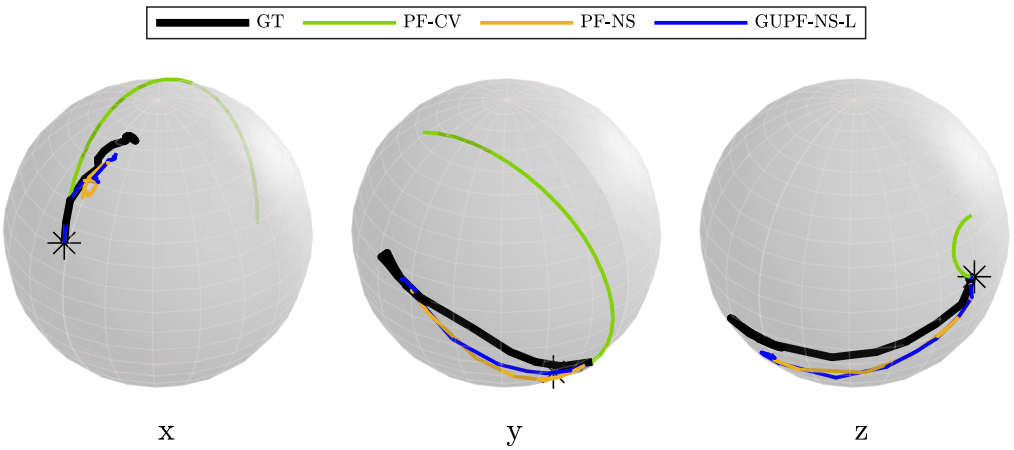


Figure C.5: Results of the orientation estimation with the PF-CV, PF-NS, and GUPF-NS-L for Video 3 of Box 4. The graphs are the result of plotting ${}^{A}\mathbf{R}_{B}(t)v_j$ on the unit sphere with $\{v_j\}_{j=1}^{3}$, as given by (4.20). From left to right: v_1, v_2, and v_3. The asterisk indicates the initial frame.

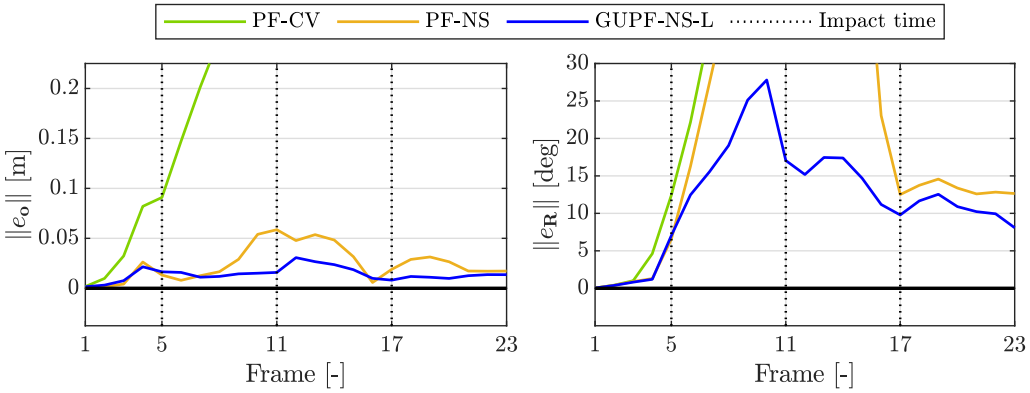


Figure C.6: Norm of the position and orientation error for the PF-CV, PF-NS, and GUPF-NS-L for Video 3 of Box 4.

Table C.1: 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-CV	0.485	-	88.9	-
PF-NS	0.030	93.8	43.6	51.0
GUPF-NS-L	0.016	96.8	13.9	84.3

The conclusions regarding the accuracy of the position and orientation estimation are confirmed by Figure C.6 and Table C.1. A significant reduction of the position error with respect to the PF-CV is achieved by incorporating the nonsmooth motion model in the particle filter. By also making use the measurement $\mathbf{L}_t$, the position and orientation error are reduced even more with 96.8% and 84.3%, respectively.

Now, we compare the GUPF-NS-L to the GUPF-NS-Z. Just as for the videos of the other boxes, both particle filters have a similar accuracy for the position estimation, as can be seen in Figure C.7. The GUPF-NS-Z always has more difficulties with estimating the z-position, which is caused by the measurement $\mathbf{Z}_t$. The position error of $\mathbf{Z}_t$ shown in Figure C.3 is mainly the result of a poor estimation of the z-position of the box by the pose estimator, since it is hard to accurately estimate the z-position with the approximate likelihood function, as discussed in Section 4.1.2. A large error in a specific direction for $\mathbf{Z}_t$ will thus also have an influence on the tracking results of the GUPF-NS-Z in that direction. For the estimation of the orientation, the GUPF-NS-Z again has a better performance than the GUPF-NS-L, as the norm of the orientation error for $\mathbf{Z}_t$ is lower than that for $\mathbf{L}_t$.

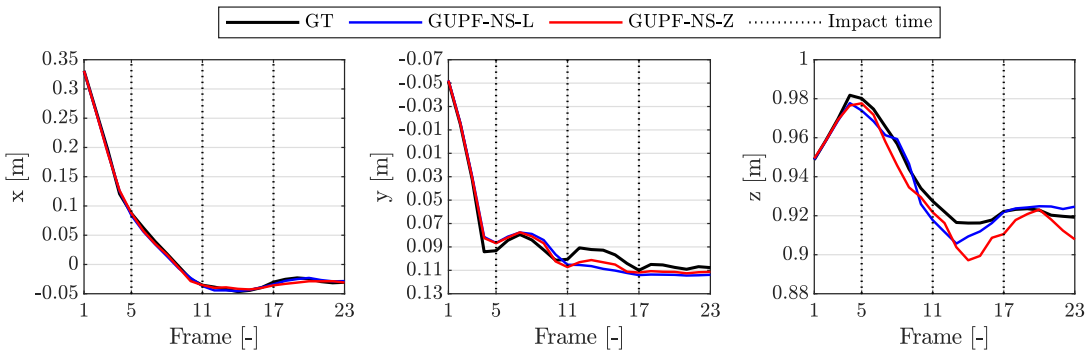


Figure C.7: Results of the position estimation with the GUPF-NS-L and GUPF-NS-Z for Video 3 of Box 4. The results for the x-, y-, and z-position can be seen in the left, middle, and right plot, respectively. The ground truth trajectory and impact times are also shown in the plots.

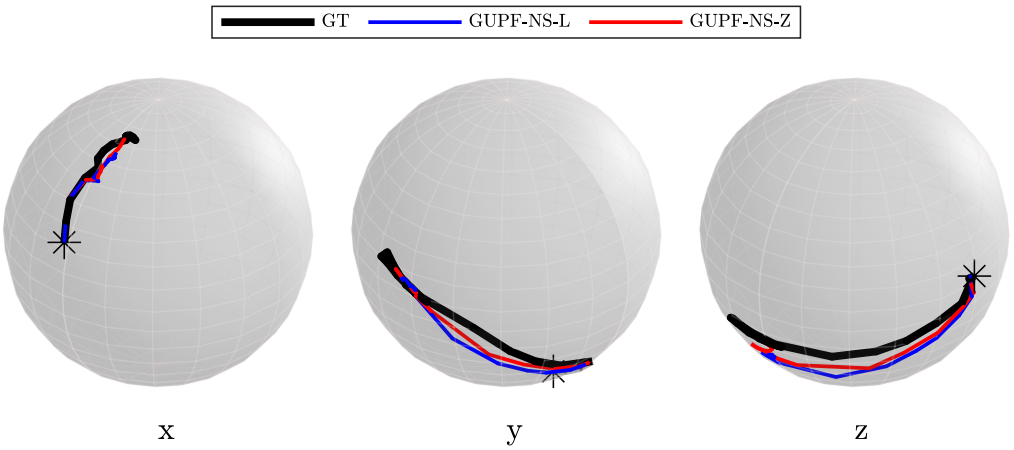


Figure C.8: Results of the orientation estimation with the GUPF-NS-L and GUPF-NS-Z for Video 3 of Box 4. The graphs are the result of plotting ${}^{A}\mathbf{R}_{B}(t)v_j$ on the unit sphere with $\{v_j\}_{j=1}^{3}$, as given by (4.20). From left to right: v_1, v_2, and v_3. The asterisk indicates the initial frame.

From Figure C.9 can be derived that the accuracy of the estimation of the position is really equal for the GUPF-NS-L and GUPF-NS-Z. To decide which one is more accurate, the RMS value mentioned in Table C.2 is used. It seems that the RMS value for $\|e_{\mathbf{o}}\|$ is 1 mm larger for the GUPF-NS-Z, so a very small difference. For $\|e_{\mathbf{R}}\|$, the GUPF-NS-Z reduces the error with 3.8 deg, which is equal to 27.7%. Comparing measurement $\mathbf{Z}_t$ with both particle filters, $\mathbf{Z}_t$ has a much larger position error, but the orientation error is smaller than that of the GUPF-NS-L.

As is also the case for the video of Box 3 and Box 5, the GUPF-NS-Z shows the best tracking results for Box 4. In Figure C.10, the output of the GUPF-NS-Z is projected onto the image. The estimation is very accurate for every frame of the recording, even after the impact at frame 5.

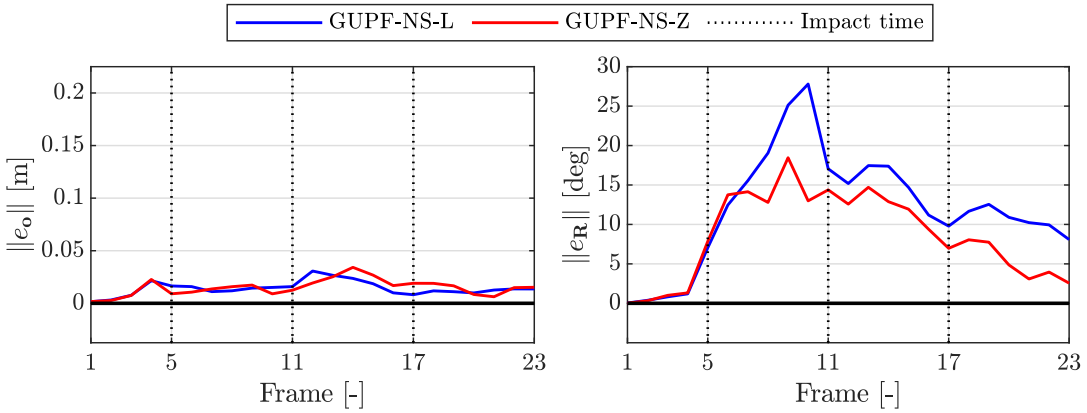


Figure C.9: Norm of the position and orientation error for the GUPF-NS-L and GUPF-NS-Z for Video 3 of Box 4.

Table C.2: 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 [%]
GUPF-NS-L	0.016	-	13.9	-
GUPF-NS-Z	0.017	-7.4	10.1	27.7
$\mathbf{Z}_t$	0.090	-	10.6	-

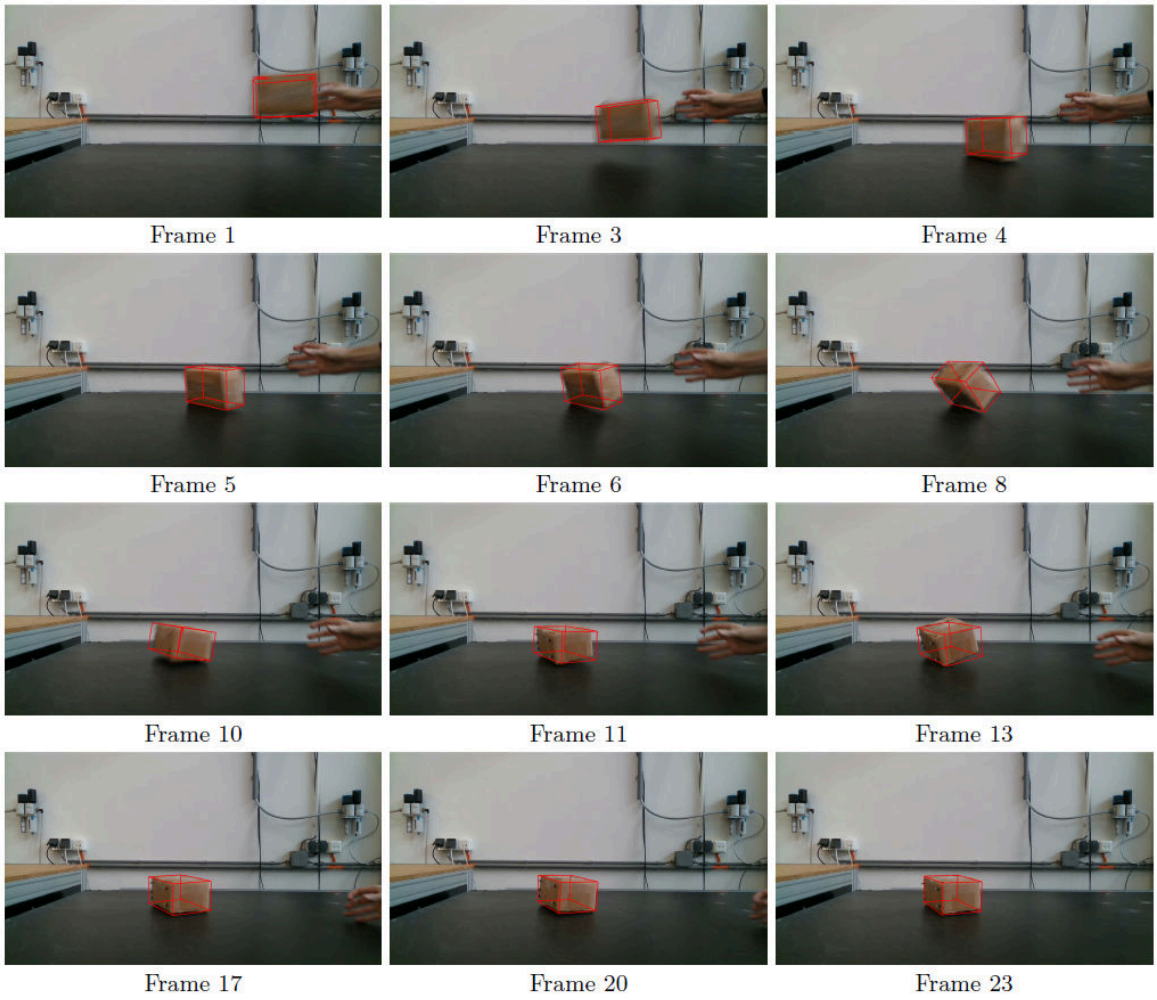


Figure C.10: Output of the GUPF-NS-Z for Video 3 of Box 4 projected onto the image to visualize the difference between the estimation and the true box.

Declaration concerning the TU/e Code of Scientific Conduct for the Master's thesis

I have read the TU/e Code of Scientific Conduct[i].

I hereby declare that my Master's thesis has been carried out in accordance with the rules of the TU/e Code of Scientific Conduct

Date

19-08-2021

Name

Sander Dingemans

ID-number

1002724

Signature

Sander

Submit the signed declaration to the student administration of your department.

[i] See: http://www.tue.nl/en/university/about-the-university/integrity/scientific-integrity/
The Netherlands Code of Conduct for Academic Practice of the VSNU can be found here also.
More information about scientific integrity is published on the websites of TU/e and VSNU