Master's Thesis

Model, Match, Vote and Track:

6-DoF Pose Filtering with Multi-resolution Surfel Maps

Submitted by:

Manus McElhone

on the 22nd of April 2013

Under the supervision of: Jörg Stückler

1st Examiner: Prof. Dr. Sven Behnke

2nd Examiner: Prof. Dr. Armin B. Cremers

Rheinische Friedrich-Wilhelms-Universität Bonn
Institute of Computer Science

Abstract

Estimating and tracking the pose (position and orientation) of the camera relative to an object is an important Computer Vision task with applications ranging from Robotics to Augmented Reality. In a Robotics context, visual servoing tasks require accurate pose estimation and tracking in real-time.

We propose a method for real-time, model-based, 3D object tracking in a particle filter framework which incorporates the latest observation in the proposal distribution. In contrast to many approaches which assume an initial pose constraint, we seek to detect the object in the first frame and estimate its pose. We do not rely on the availability of accurate CAD models, but employ Multi-resolution Surfel Maps as a concise representation of object shape and texture.

Inspired by object detection methods based on point pair features, we propose a colour surfel pair feature as a means of describing the relative variations in texture and shape of two surfels. By finding consistent arrangements of surfel pairs in input scene and object model, we can detect the object and estimate its pose in a Hough voting framework.

We evaluate our approach on a publicly available RGB-D Object Tracking dataset, as well as on additional sequences captured for this thesis, and show high rates of detection and good tracking performance with respect to different speeds of camera motion and occlusions

Acknowledgements

I would like to express my heartfelt thanks to Jörg Stückler, who in supervising this thesis offered motivation and imparted valuable insights. Thanks also, to Professor Sven Behnke for the opportunity of working in his group, access to the excellent facilities in his department and for his helpful suggestions.

On a personal level, thanks to my friends in Bonn, with whom it has been a pleasure to spend this time. Thanks especially to those who have taken a particular interest in this work or have helped with proofreading, namely German, Ishrat, Aljosa and Mohammad.

Deepest thanks also to my parents Charles and Angela, and to my sister Nora and brother John for their love, support, advice and craic! Last, but by no means least, deepest thanks to Arti, for her constant love, loyalty and patience.

Contents

1 Introduction

The problem of tracking is one which spans many different fields of scientific research, from mobile robot localization, to econometrics and target-tracking applications. Put broadly, it is the problem of estimating the state of a system given a set of observations accumulated over time.

In this thesis, we consider the problem of object pose estimation and tracking in a joint framework such that prior knowledge of the initial pose is not required. We present an approach to robustly detect objects in RGB-D images and estimate their pose, which we then track under movement of the camera or object.

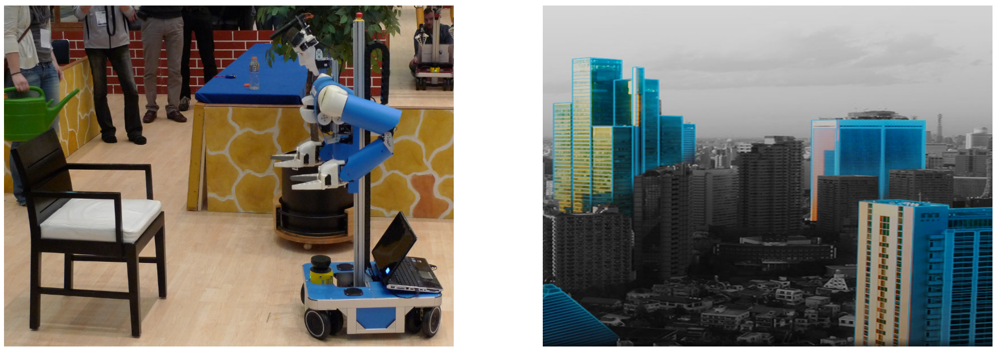


Image: metaio (2013)

Figure 1.1: Object Detection and Tracking are crucial components in Robotics and Augmented Reality systems.

Object detection and tracking are interrelated problems in Robotics and Computer Vision research with a range of applications. In Robotics, an obvious area of application is in mobile manipulation tasks, where *visual servoing* is a common technique. This method is often used to enable a mobile robot to manipulate objects in its environment based on sensory input in the form of images.

Augmented Reality (AR) is another area where visual detection and tracking are of interest. In AR, models of objects or parts of the environment are rendered and aligned in real-time with a view of the physical environment. In both of these applications, accuracy and real-time performance are crucial.

This thesis is structured as follows. In the remainder of this chapter we define the problem of interest and examine the state of the art with respect to object detection, pose estimation and tracking. Finally we briefly present our proposals.

In Chapter 2 we will thoroughly examine the scientific and mathematical background related to 3D visual object pose estimation, and tracking, and expand upon the most relevant related work. In Chapter 3, the main contributions of this thesis will be presented in detail, while in Chapter 4 we assess the performance of our approach. Finally in Chapter 5 we present our conclusions as well as ideas for subsequent work.

1.1 Problem Definition

In this thesis, we aim to estimate and track the time-varying 6 degree of freedom (DoF) pose of a known rigid 3-dimensional object given a time series of 3D image data acquired from an RGB-D camera.

We pose our problem as the estimation and tracking of the pose of the camera with respect to a known object. Hence, in each frame the goal is to determine the most likely camera pose x_t given the observations $z_{1:t}$. Furthermore, the initial pose x_0 is assumed to be unknown, and as such we also aim to detect the object and estimate its pose in an unconstrained way.

1.2 Related Work

As evidenced by the surveys of Kragic and Vincze (2009) and Uchiyama et al. (2012), object detection and tracking are highly active and productive areas of research with a number of mature methods.

In this section we present a review of the state of art with respect to object detection and tracking. We first consider the leading approaches to object detection based on 2D and 3D features, before looking at a number of methods for visual object tracking.

1.2.1 Object Detection and Pose Estimation

Research into solving the interrelated problems of visual object detection and pose estimation has yielded a wide variety of techniques according to the characteristics of the sensor data or the target application.

Most object detection systems are based on describing objects by a set of *features*. Features can be *global* - describing the object as a whole - or *local* - by distinctive parts of the object. Typical features are geometric primitives such as corners, edges, or image patches resulting from an object's texture or shape. Detection is then achieved by establishing correspondences between learned and observed features which can be used to establish the position and orientation of the object in the image.

2D Image Features

In the area of 2D Vision, local invariant 2D features such as SIFT (Lowe, 1999) or SURF (Bay et al., 2008), which describe the local texture surrounding interest points in the image, have been widely and successfully applied for the detection and pose estimation of textured objects. Such local features have generally outperformed global template-matching approaches, especially in robustness to partial occlusions.

In the MOPED (Martinez et al., 2010) framework, a 3D object model comprising SIFT features is learned from images captured from different viewpoints, while efficient detection and pose estimation is achieved in large object databases by leveraging GPU acceleration.

Methods which rely solely on an object's texture, may however prove to be less robust when considering objects with limited texture.

3D Image Features

With the advent of affordable, high-quality depth sensors, a considerable amount of attention has been given to incorporating geometric features for object detection. Such features typically describe the local geometry of a surface point in terms of surface curvature and normals, and have been successfully incorporated to detect objects and estimate their pose in Hough voting or RANSAC frameworks.

In Wahl et al. (2003), Surflet-pair histograms are proposed as a compact representation of 3D objects by the distribution of geometric relations between pairs of surface points. On a similar theme, Rusu et al. (2009) proposed Point Feature Histograms (PFH) as a local geometric feature describing the variations in geometry in the neighbourhood of a point, while Hinterstoisser et al. (2012) construct object templates of 2D and 3D features.

Approaches based on Point-pair features (PPF) (Drost et al., 2010; Papazov and Burschka, 2011) detect and localize 3D objects by finding locally consistent arrangements of point pairs through Hough Voting or RANSAC. Several enhancements to the descriptive power of PPFs have been proposed using visibility context (Kim and Medioni, 2011), contours (Choi et al., 2012) or colour (Choi and Christensen, 2012b).

1.2.2 Real-time Object Tracking

Object tracking is a research area with a rich body of literature, as illustrated by the survey of Lepetit and Fua (2005). While there are a multitude of approaches to the problem, they can be broadly grouped into techniques which consider the temporal constraints between frames (*recursive tracking*), and those which attempt to detect the object in each frame individually (*detection-based tracking*).

Most tracking approaches can be further classified by the means of object representation. *Model-based* methods employ an explicit model of the object shape, which may also describe texture (e.g. CAD, geometric or learned 3D models), whereas *Appearance-based* methods describe the object in terms of 2D or 3D features (e.g. edges, points etc.). In both cases the goal is to find the optimum alignment of the model elements or features with those occurring in the image.

1.2.2.1 Recursive Tracking

The leading approaches to tracking which consider the temporal constraints between frames are mostly based either upon optimization techniques or Bayesian filtering.

Optimization Techniques

Tracking by optimization typically involves iteratively refining the previous pose estimate, often by employing non-linear least squares minimization techniques such as the Newton-Rhapson or Levenberg-Marquadt methods.

The early model-based method of Harris (1993) was based on aligning projected model edges with those observed in the scene, and has inspired a multitude of edge-based trackers. In Drummond and Cipolla (2002) and Comport et al. (2004) real-time model-based tracking is achieved by using Iteratively Re-weighted Least Squares (IRLS) to align model edges in the image. Edges and texture information are combined in the approach of Vacchetti et al. (2004) to facilitate tracking of textured and textureless objects.

In Stückler and Behnke. (2012), Multi-Resolution Surfel Maps (MRSMaps) are proposed as a compact and efficient representation for aggregated RGB-D images. In an efficient SLAM method, RGB-D data is registered incrementally in order to build indoor maps or 3D object models, while real-time tracking is achieved by recursively optimizing the current pose estimate.

Bayesian filtering

Bayesian filtering has been among the most popular choices for tracking for decades. The Kalman filter and its variants, as well as the Particle filter have been widely adopted realizations of the Bayesian framework.

Since the introduction of the Bootstrap filter (Gordon et al., 1993), and especially the Condensation algorithm of Isard and Blake (1998), the application of Monte Carlo sampling to Bayesian estimation has been widely used for visual tracking. Particle filtering techniques are particularly attractive for this purpose due to their flexibility in coping with arbitrary noise characteristics and non-linear motion and observation models. Moreover the multi-hypothesis nature of particle filters has increased robustness by admitting multiple correspondences.

Several works explicitly handle the changing appearance of a moving object, by employing machine learning methods to adapt a classifier over time (Ross et al., 2008; Grabner et al., 2006), while depth information has been exploited to estimate the 3D object position (García et al., 2012).

Appearance-based tracking has also been proposed in Kwon and Park (2010), where tracking of 2D object templates is achieved using a particle filter modelled on the $Aff(2)$ group with auto-regressive state dynamics.

Model-based approaches have also been applied in a particle filter framework. Klein and Murray (2006) exploit the GPU to render visible edges which are tracked using an annealed particle filter. In Choi and Christensen (2011, 2012a) edge-based tracking of textured and textureless objects is achieved while modelling the state evolution on the $SE(3)$ Group with autoregressive dynamics.

1.2.2.2 Detection-based Tracking

In contrast to the techniques described above, *Tracking by Detection* does not consider the temporal constraints between successive frames, but rather aims to estimate the object pose in each frame individually. Tracking by detection is a relatively new concept, arising from the advent of efficient object detection algorithms, advances in computational technology and the exploitation of parallel processing on GPUs to achieve real-time performance.

In Lepetit and Fua (2006) frame-rate detection and pose estimation is achieved by employing fast key-point matching and a random-tree classifier learned using a 3D object model in a training phase. Fast keypoint detection is also a feature of Ozuysal et al. (2010), where random ferns are learned off-line and online in order to perform fast detection and pose estimation.

Skrypnyk and Lowe (2004) proposed detection-based tracking with invariant features for Augmented Reality.

In general, however, taking account of the previous estimate yields a strong prior for determining the object's pose in the current frame, and yields better temporal coherence of the estimated trajectory.

1.2.3 Combined Pose Estimation and Tracking

While the related problems of object detection, pose estimation and tracking have seen a wealth of research, few works have specifically addressed integrated solutions, with most tracking approaches assuming that the initial pose is known.

Prisacariu and Reid (2012) present an approach to real-time 3D object segmentation and tracking by aligning contours between model and scene, while high frame rates are achieved by leveraging parallel processing on GPUs.

In Choi and Christensen (2011) initial pose hypotheses are extracted by matching keypoint features, while tracking proceeds by matching points on the edges of a projected wireframe model with those in the image. Choi and Christensen (2012a) define edge-templates which are matched in the image in order to identify initial pose candidates, these are then refined using an annealed particle filter.

1.2.4 Discussion

While there there has been a wealth of research into the very related problems of object detection, pose estimation and tracking, few integrated approaches have been proposed.

For applications such as Robotics and Augmented Reality, 6-DoF pose estimation and tracking is of particular interest. In this context, model-based methods tend to offer the highest performance in terms of accuracy, however it may also be advantageous to model objects' texture. Most model-based methods assume an object can be described by geometric shapes, or require CAD models which may not be available, or are time-consuming to create.

In terms of detection, leveraging depth information, where available, can result in accurate pose estimates, while texture remains an important cue.

For tracking, optimization methods promise accuracy, while filtering techniques lend robustness and multi-hypothesis tracking, which allows for refinement of initial pose estimates. However most methods assume external initialization.

Few approaches have combined 6-DoF pose estimation and tracking for learned 3D object models. With this in mind, in this thesis we will be concerned with detecting 3D objects in RGB-D images, estimating their position and orientation, and robustly tracking the pose of the camera.

We propose to extend learned object models (based on the MRSMap representation) with pair features which describe local shape and texture. Moreover, we will track the object with a particle filter which incorporates the latest observation in the proposal distribution.

1.3 Voting-based Object Pose Estimation and Improved Proposal Particle Filter Tracking

We propose a method for the estimation and tracking of the 6-DoF pose of known 3D objects. Our method does not rely on the availability of accurate CAD models, instead we employ the MRSMap framework of Stückler and Behnke. (2012) in order to learn 3D object models in multiple resolutions.

We extend this representation with Surfel-pair features which describe the variations in shape and texture between two Surfels. By matching such features between an input scene map and the model, we establish hypotheses for the object pose in an efficient voting framework.

The pose hypotheses serve as initialization for a particle filter tracker in which the state dynamics are modelled on the Euclidean Group, and which incorporates an efficient registration method to yield improved proposal distributions.

2 | Background

In this chapter we explore the theoretical and mathematical background associated with the problem of 3D visual object detection, pose estimation and tracking, and examine the related work in more detail.

This chapter is structured as follows. In Section 2.1 we consider the characteristics of RGB-D sensors and the data they provide, while in Section 2.2 we look at Multi-resolution Surfel Maps for representing and aggregating RGB-D images. In Section 2.3 we examine at rigid body motion and pose representation, while in Section 2.4 we explore the problem of tracking in detail, especially from the point of view of Bayesian filtering, and with particular emphasis on the Particle Filter. Finally, in Section 2.5 we consider 3D object detection and pose estimation by matching point pairs in a voting framework.

2.1 Image Acquisition with RGB-D Sensors

Affordable, light and compact sensors which yield RGB and dense depth images (such as the Microsoft Kinect (Microsoft, 2013) or ASUS Xtion Pro Live (ASUS, 2013)) have gained popularity in recent years and have been applied to a variety of vision-related problems, from human pose estimation to Simultaneous Localisation and Mapping (SLAM).

In this section we examine the characteristics of such sensors, including their strengths and weaknesses. We first provide a brief overview of the two main types of RGB-D sensors, before looking at the particular characteristics of they data they provide.

2.1.1 Overview of RGB-D Sensors

RGB-D sensors generally either use a structured light or time of flight (ToF) technique in order to produce a depth map of the scene.

The Microsoft Kinect and ASUS Xtion Pro Live RGB-D cameras both feature sensing technology which employs the structured light depth estimation technique. This technique involves projecting a known pattern of infrared (IR) dots into the scene, the infrared light is reflected by objects in the field of view and captured by a CMOS sensor. The captured signal is then compared to a calibrated reference map in order to produce a depth map of the scene.

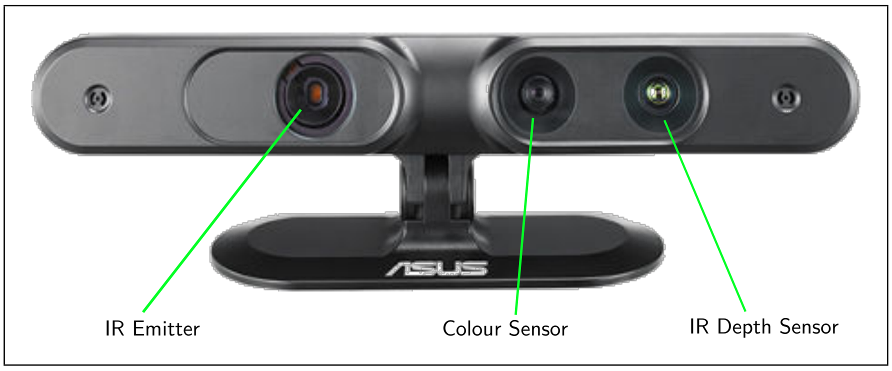


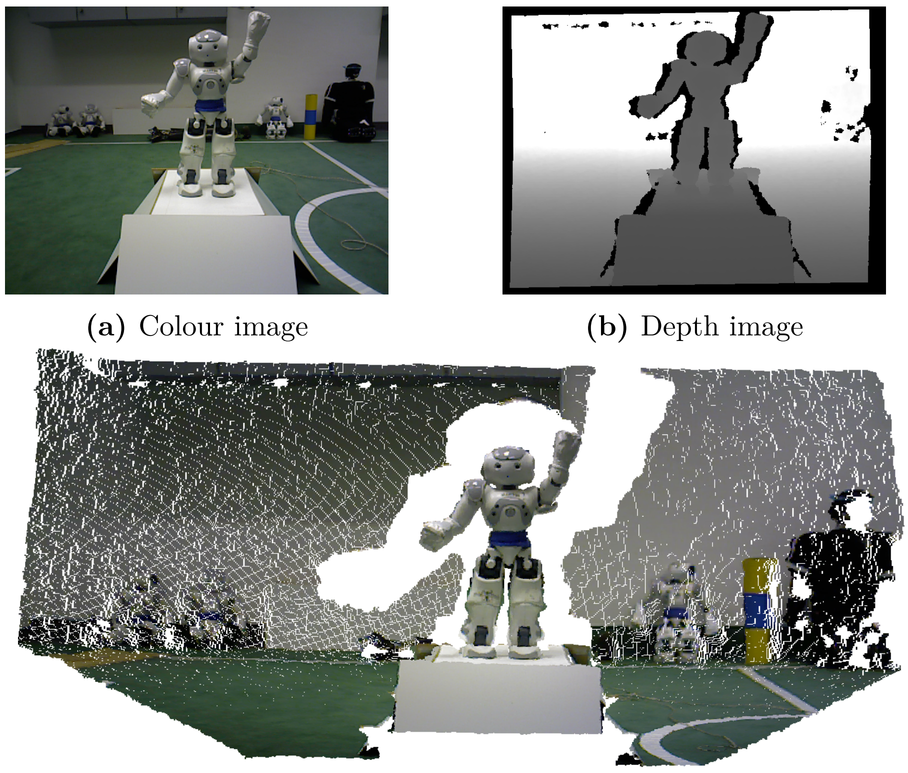
Figure 2.1: The ASUS Xtion Pro Live RGB-D Camera. Image adapted from ASUS (2013)

In the case of ToF cameras, the depth is computed by projecting light into the scene and measuring the time of arrival of reflected light captured by a suitable sensor.

The lower relative cost of structured light sensors, as well as attractive features such as high resolution and relatively high frame rates has seen their wide adoption at the expense of the traditionally more costly ToF sensors. With this in mind, in the next section we consider the particular characteristics of structured light depth sensors.

2.1.2 Characteristics of RGB-D Sensor Data

The Asus Xtion Pro Live is capable of producing RGB images at resolutions up to 1280x1024, and depth images up to 640x480 at 30Hz. The depth sensor


(a) Colour image **(b)** Depth image

(c) Point cloud

Figure 2.2: Top: Example RGB and Depth images captured using an Asus Xtion Pro Live. Bottom: Corresponding pointcloud generated according to eq. 2.1

has an operating range of approximately 0.8m - 3.5m with a field of view of 58° horizontal and 45° vertical.

One important characteristic of this type of depth sensor is that the accuracy of depth measurements decreases with distance from the sensor. This is a typical phenomenon of stereo vision cameras. In fact, the error in depth determination is related quadratically to the distance of the object from the sensor (ROS, 2012).

The RGB and Depth images are registered, allowing the determination of the depth and colour for each pixel. However, due to the offset of the RGB and depth sensors, there will be parts of an object for which there are colour measurements but not depth, resulting in “holes" in the depth map. The offset of the IR emitter and the depth sensor means that some portions of the scene may be illuminated but not observed due to occlusion, and vice versa.

The particular textural properties of objects in the scene can be another source of errors. Reflective, and transparent materials can also pose problems for this type of sensor since the specular properties of such surfaces mean that light is not reflected back to the sensor.

2.1.3 RGB-D Data Representations

RGB-D sensors typically yield an 8-bit RGB color and a depth image, which have a pixel-wise correspondence. The colour point cloud is a convenient 3-dimensional image format which combines both RGB and depth data, with each point having an XYZ coordinate and RGB color value.

Given an aligned RGB image I and depth image D of width w and height h, the xyz point coordinates for a pixel with coordinates u,v are given by:

$$\begin{aligned} c_x &= w/2 \\ c_y &= h/2 \\ z &= D(u,v) \\ y &= u - c_y \cdot z \cdot f^{-1} \\ x &= v - c_x \cdot z \cdot f^{-1} \end{aligned} \tag{2.1}$$

according the pinhole camera model, where f is the focal length of the colour camera.

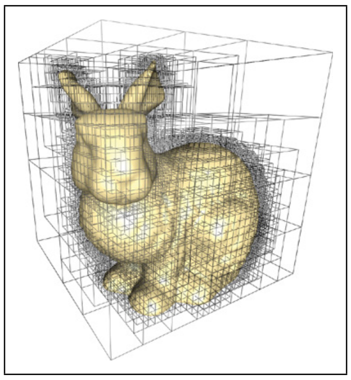
2.2 Multi-resolution Object Modelling with Surfel Maps

On of the key challenges brought about by dense depth sensors is how to efficiently process and store the data they provide. One approach is to aggregate points into a more compact format to allow for efficient storage and computations. In this section we consider Multi-Resolution Surfel Maps (MRSMaps) as just such a compact representation.

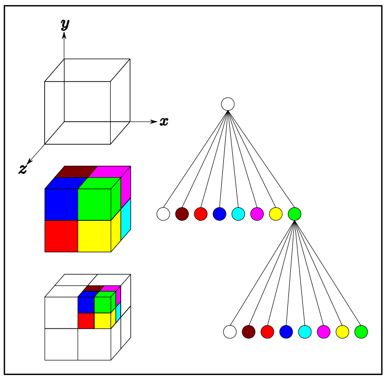
For this work, we build upon the Multi-Resolution Surfel Map (MRSMap) framework for RGB-D image aggregration of Stückler and Behnke. (2012) (Code: Stückler (2013)). MRSMaps allow for the efficient aggregation of RGB-D image data in order to build consistent models of objects and scenes with a multi-resolution representation of shape and texture.

2.2.1 Octrees: Representing spatial data

The octree is a hierarchical data structure which is ideal for the multi-resolution representation of 3D spatial data. Octrees describe a volume of 3-dimensional space (a voxel) by a structure of nodes, where each internal tree node divides its volume equally among exactly 8 children.



(a) Octree decomposition of the space occupied by a 3D rabbit model. Image: Lefebvre et al. (2005)



(b) Subdivision of a unit cube into octants

Figure 2.3: Octrees make for efficient use of space by different levels of granularity.

Octrees are very memory efficient since different regions of space can be represented by different maximum resolutions. This property is particularly useful when we are only interested in the detail of specific regions of the space (i.e. those regions occupied by objects), or when the properties of the data are such that the level of accuracy is not constant (as is the case with RGB-D images from range sensors such as those presented in Section 2.1).

The tree structure makes for efficient range queries and lookups of individual voxels and it is possible to perform constant time lookups to the neighbours of a given voxel.

2.2.2 Surfels

Surfels (**Surf**ace **el**ements) are a representation of the shape and textural properties of a surface patch. In Stückler and Behnke. (2012), surfels are defined by statistics which describe the distribution of colour and shape in the region.

Surfels model the spatial and textural properties of a volume by maintaining the joint distribution of 3D point coordinates and colour in a 6D distribution which models, not only the distribution of spatial and colour data, but also the spatial distribution of colour. This distribution is approximated by the sample mean and covariance of points contributing to a surfel, while the the use of the sufficient statistics

$$\mathcal{S}(\mathcal{P}) = \sum_{p \in \mathcal{P}} p \quad \text{and} \quad \mathcal{S}^2(\mathcal{P}) = \sum_{p \in \mathcal{P}} pp^T \tag{2.2}$$

allows for the efficient aggregration
of a set of points $\mathcal{P}$, where $p = [x, y, z, L, \alpha, \beta]$ (described below).

Colour is described in the Luminance-Alpha-Beta colour space ($L\alpha\beta$), which has the advantage of providing a separation of the Luminance value L, from chrominance values α and β, representing the colour in terms of the red-green and blue-yellow spectrums, which are largely invariant to illumination.

The $L\alpha\beta$ space representation of a colour can be computed simply and efficiently from the RGB representation as follows:

$$
\begin{aligned}
L &= \frac{1}{2}(\max\{R,G,B\}+\min\{R,G,B\}) \in [0,1] \\
\alpha &= R-\frac{1}{2}G-\frac{1}{2}B \in [-1,1] \\
\beta &= \frac{\sqrt{3}}{2}(G-B) \in \left[-\frac{\sqrt{3}}{2},\frac{\sqrt{3}}{2}\right]
\end{aligned}
\tag{2.3}
$$

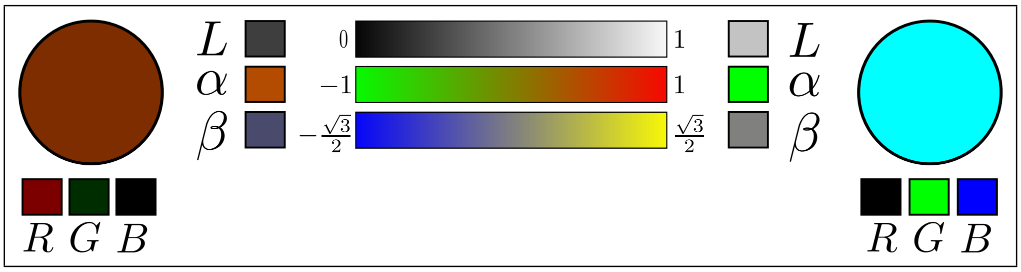


Figure 2.4: RGB and $L\alpha\beta$ values of two colours

Furthermore, to allow to aid in data association, a Shape-Texture descriptor is computed for each Surfel, describing the shape and texture in its local context. The descriptor is computed by pairing a Surfel with its neighbours and constructing histograms of spatial and colour statistics.

2.2.3 Map Representation

The MRSMap representation is based on an octree structure, with each octree node comprising up to 6 Surfels corresponding to the orthogonal viewing directions of a unit cube. Hence MRSMaps are capable of representing a full-view model of an object's surface.

Image content is efficiently distributed in the map by aggregating the values of points in regions of coherent depth in the image which map to the same octree node, and updating the sufficient statistics of the affected surfels. This technique radically reduces the number of octree node lookups and Surfel updates compared to inserting points individually.

Furthermore, MRSMaps explicitly handle the particular characteristics of depth data from RGB-D sensors by limiting the maximum resolution for octree nodes which are further from the sensor, and by detecting virtual borders caused by missing depth information as the result of occlusion.

2.2.4 Registration

A pair of MRSMaps are registered by determining the maximum likelihood transformation between source and target. The registration method is related to the Iterative Closest Point Algorithm (ICP) for aligning two sets of points. Hence the registration iterates association and estimation steps in order to find the optimal alignment.

2.2.4.1 Data Association

First, the data association between surfels in the maps is determined on multiple resolutions by starting at the finest resolution, and using the current camera pose estimate.

A data association between surfels is established if it is the best fit association determined by a combination of the Euclidean distance, and the difference in the Shape-Texture descriptors.

This association process is made efficient by associating nodes first on finer resolutions, and not associating nodes with a successfully associated child. Moreover as the registration process is iterative, efficiency gains can be made by finding the best association among the neighbours of a previous association.

2.2.4.2 Observation Likelihood

Registration of a source map m_s constructed from an RGB-D image z to a target map m_m involves finding the pose which maximizes the observation likelihood $p(z|x, m_m)$. This likelihood is determined by the matching likelihood between source and target, given the pose x, and the set of surfel associations $\mathcal{A}$:

$$p(m_s|x, m_m) = \prod_{(i,j)\in\mathcal{A}} p(s_{s,i}|x, s_{m,j}) \tag{2.4}$$

where $s_{s,i} = (\mu_{s,i}, \Sigma_{s,i})$, $s_{m,j} = (\mu_{m,j}, \Sigma_{m,j})$ are associated surfels. The observation likelihood of a surfel match is given by:

$$\begin{aligned} p\left(s_{s,i} \middle| x, s_{m,j}\right) &= \mathcal{N}\left(d\left(i,j;x\right);0,\Sigma_{i,j}\left(x\right)\right) \\ d\left(i,j;x\right) &= \mu_{m,j} - T\left(x\right)\mu_{s,i} \\ \Sigma i,j\left(x\right) &= \Sigma_{m,j} + R\left(x\right)\Sigma_{s,i}R\left(x\right)^T \end{aligned} \tag{2.5}$$

The aim of registration is to minimize the error in the estimated pose, hence optimizing the log likelihood:

$$L\left(x\right) = \sum_{(i,j)\in\mathcal{A}} \log\left(\left|\Sigma_{i,j}(x)\right|\right) + d\left(i,j;x\right)^T \Sigma_{i,j}^{-1}(x) d\left(i,j;x\right) \tag{2.6}$$

In this context registration is a non-linear least-squares optimization problem. In Appendix A.1 we consider a number of approaches to problems of this form.

2.2.4.3 MRSMap Registration with the Levenberg Marquadt Method

In Stückler and Behnke (2013) the registration is carried out by first performing Levenberg Marquadt optimization in order to efficiently resolve larger misalignments, before refinement with Newton's method.

Given a set of surfel associations $\mathcal{A}$, we aim to find the pose $x^\star$ which maximizes the likelihood $p\left(m_s \mid x, m_m\right)$ of the latest observation (Eq. 2.4). Hemce the procedure is to iteratively adjust the parameters of x to find:

$$x^\star = \mathrm{argmin}_x \, \mathbf{e}^T\left(x\right)\mathbf{W}\,\mathbf{e}\left(x\right) \tag{2.7}$$

where, $\mathbf{e}\left(x\right)$ is a vector of residuals, and $\mathbf{W}$ is a weighting matrix.

$\mathbf{e}\left(x\right)$ is given by the residuals of associated surfels, $\mathbf{e}\left(x\right) = \left[d\left(a;x\right)\right]_{a\in\mathcal{A}}$, while the weighting matrix is block diagonal, with the following form:

$$\mathbf{W} = \mathrm{diag}\left(\left\{w_a \Sigma_a^{-1}\left(x\right)\right\}_{a\in\mathcal{A}}\right) = \begin{bmatrix} \mathbf{W}_{a_1} & 0 & \cdots & 0 \\ 0 & \mathbf{W}_{a_2} & \vdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \mathbf{W}_{a_m} \end{bmatrix} \tag{2.8}$$

where $w_a = \tau - d_{feat.}(a)$ is a measure of the similarity in the shape-texture features of the associated surfels.

In each iteration the parameters are updated according to:

$$x' = x + \left(\mathbf{J}^T \mathbf{W} \mathbf{J} + \lambda \mathbf{I}\right)^{-1} \mathbf{J}^T \mathbf{W} \mathbf{e}(x) \tag{2.9}$$

where the Jacobian $\mathbf{J} = \{\mathbf{J}_a(x)\}_{a \in \mathcal{A}}$ is given by stacking the individual Jacobians per association, $\mathbf{J}_a = \frac{\partial T}{\partial x}(x)\, \mu_{s,a}$. In this way, eq. 2.9 is composed of sums over surfel associations:

$$\begin{aligned} \mathbf{J}^T \mathbf{W} \mathbf{J} &= \sum_{a \in \mathcal{A}} \mathbf{J}_a^T \mathbf{W}_a \mathbf{J}_a \\ \mathbf{J}^T \mathbf{W} \mathbf{e}(x) &= \sum_{a \in \mathcal{A}} \mathbf{J}_a^T \mathbf{W}_a \, d(a; x) \end{aligned} \tag{2.10}$$

Eq. 2.9 is evaluated iteratively and reassociation takes place if the updates converge below a threshold. The optimization is terminated when the parameters of the pose no longer change, or a theshold on the number of iterations is reached.

2.2.5 Tracking with MRSMaps

Real-time tracking of the camera pose with respect to a learned object model is also achieved in the MRSMap framework. In each frame, the pose is optimized by aggregating the latest RGB-D image in a scene map and registering it towards the object model.

2.3 Pose Representation

As stated in the introduction, we pose our problem as that of finding the most likely pose of the camera with respect to an object model in a world reference frame which we refer to as W. The movement of the camera relative to W follows a rigid body trajectory.

In this section, we briefly introduce the fundamentals of Rigid Body Motions, their properties and representations. As the space of Rigid Body Motions constitutes a Lie Group, we explore the theory of Lie Groups, Lie Algebras and in particular the Special Orthogonal and Euclidean Groups in Appendix A.2

We present here the main concepts required for understanding the problems of pose estimation and tracking from a geometric point of view, a much more in-depth look at the topic can be found in Ma et al. (2004).

2.3.1 Rigid Body Motion

For a rigid body $X \subset \mathbb{R}^3$, a rigid body motion is a mapping $g : \mathbb{R}^3 \to \mathbb{R}^3$ which satisfies:

1. Preservation of distances: $\|g(p) - g(q)\| = \|p - q\| \ \forall p, q \in \mathbb{R}^3$
2. Preservation of vector cross-product: $g_* (u \times v) = g_* (u) \times g_* (v)$ $\forall$ vectors $u, v \in \mathbb{R}^3$

These properties imply the additional properties of preservation of angles, and preservation of orientation (handedness).

Hence a rigid body motion can be described by the motion of a coordinate frame attached to a fixed point on the body. The rigid body configuration is then described by the translation and rotation of this coordinate frame relative to some world frame.

2.3.2 Properties of Rigid Body Motion

We have now introduced a representation, and a minimal parameterization for rigid body motions. With these formalisms in hand, we now return to the problem at hand, tracking the movement of a camera or other rigid object with respect to some fixed coordinate frame.

As briefly introduced at the start of this section, we can represent the configuration of the camera with respect to a world frame W, with origin o_w by choosing a fixed point o_c and attaching a right-handed coordinate system C.

If we assume W to have axes given by the right-handed orthonormal basis vectors e_1, e_2, e_3, and the displacement between W and C describes a rigid body motion g, then the pose of configuration of the camera consists of two components:

1. $\mathbf{t}$: the vector from o_w to o_c. This is the translation of the camera with respect to the model frame.
2. R: the relative orientation of the axes of C with respect to W, given by the rotation matrix $R = \begin{bmatrix} \mathbf{u} & \mathbf{v} & \mathbf{w} \end{bmatrix}$

   where $\mathbf{u} = g * (\mathbf{e}_1), \mathbf{v} = g * (\mathbf{e}_2), \mathbf{w} = g * (\mathbf{e}_3)$

This rotation and translation allow us to express the coordinates of a point relative to C. Fig. 2.5 illustrates how coordinate systems W and C are related by a rigid body motion $g_{wc} = (R_{wc}, \mathbf{t}_{wc})$. g_{wc} transforms coordinates expressed relative to C to coordinates relative to W. For a point p with coordinates $\mathbf{v}_c$ relative to C, we can determine its coordinates $\mathbf{v}_w$ with respect to W by:

$$\mathbf{v}_w = g_{wc} * (\mathbf{v}_c) = R_{wc}\mathbf{v}_c + \mathbf{t}_{wc} \tag{2.11}$$

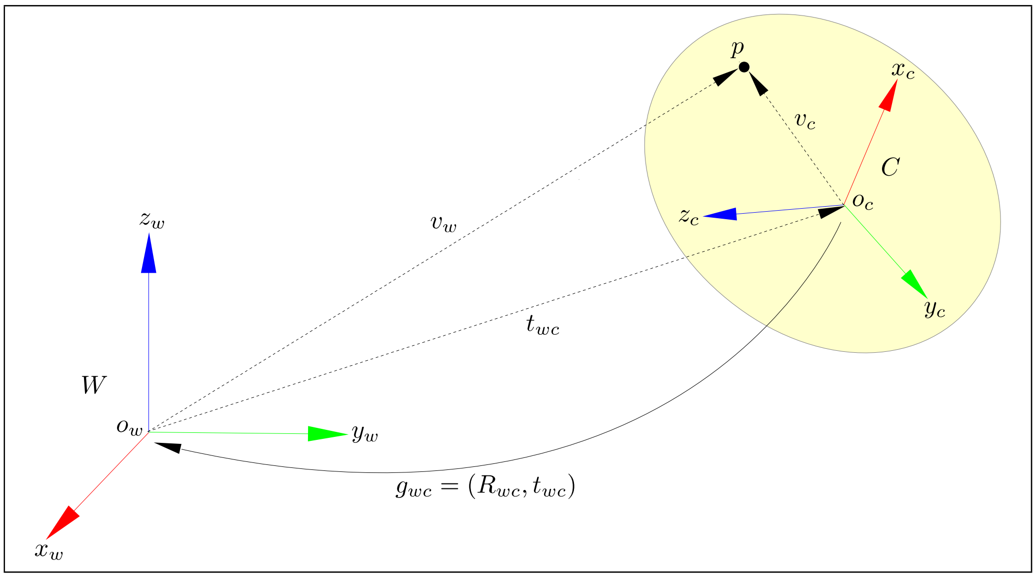


Figure 2.5: Coordinate systems W and C related by a rigid-body motion g

The rotation and translation operations can be combined by constructing a 4×4 homogeneous transformation matrix T_{wc} of the form:

$$T_{wc} = \begin{bmatrix} R_{wc} & \mathbf{t}_{wc} \\ \mathbf{0} & 1 \end{bmatrix} \tag{2.12}$$

We can then represent the coordinate transform of eq. 2.11 by:

$$v_w = T_{wc} v_c \tag{2.13}$$

We consider now the trajectory of the camera moving in a world frame, $g(t) : \mathbb{R} \to SE(3)$, the configuration of the camera at time t represents the relative displacement between the world frame W and the camera frame C, and is given by $g(t) = (R(t), \mathbf{t}(t))$.

The relative displacement between two configurations of the camera at times t_1, t_2 is given by $g(t_2, t_1) = g(t_2)^{-1} g(t_1)$, and consecutive transformations can be composed according to: $g(t_3, t_1) = g(t_3, t_2)\, g(t_2, t_1)$

2.3.3 Alternative representations for Rotations

In the previous section we have seen how a rotation can be defined by a 3×3 rotation matrix $R \in SO(3)$, which can in turn be parameterized by a 3 coordinate vector $\omega \in \mathfrak{so}(3)$.

However, there are a number of other parameterizations available. For example one can define any rotation by the axis, given by a unit vector $\hat{\mathbf{e}}$ and angle of rotation, given by θ. These two components can be combined by choosing a vector $\mathbf{e}$ such that $\hat{\mathbf{e}} = \dfrac{\mathbf{e}}{\|\mathbf{e}\|}, \theta = \|\mathbf{e}\|$. In this case $\mathbf{e} \in \mathfrak{so}(3)$ and the associated rotation matrix R is, as we have seen, given by: $R = \mathrm{e}^{\mathbf{e}} = \mathrm{e}^{\hat{\mathbf{e}}\theta}$.

Euler angles represent rotations by three composed rotations around a single axis, and are described by 3 parameters - the angles of rotation. However the different conventions of moving vs. fixed axes, combined with the different ordering schemes, and the existence of singularities known as Gimbal lock make this representation somewhat disadvantageous.

2.3.3.1 Quaternions

Quaternions in general are numbers of the form $q = q_0 + q_1 i + q_2 j + q_3 k$ where i, j, k satisfy $i^2 = j^2 = k^2 = ijk = -1$, and can be represented as $q = (s, \mathbf{v})$ by a scalar part $s = q_0$, and a vector part $v = [q_1, q_2, q_3]$.

The quaternion conjugate is given by $\bar{q} = (s, -\mathbf{v})$, and its inverse by $q^{-1} = \frac{\bar{q}}{\|q\|^2}$, where $\|q\|^2 = q\bar{q} = q_0^2 + q_1^2 + q_2^2 + q_3^2$.

Of particular interest is the representation of rotations by unit quaternions (those with $\|q\| = 1$). Unit quaternions are a means of encoding an axis-angle representation by four coordinates, consisting of a scalar part, and a vector part $q = (s, \mathbf{v})$.

For a rotation given by the axis $\hat{\mathbf{e}} = [e_x, e_y, e_z]^T$ and angle θ, the corresponding quaternion is given by:

$$q = (s, \mathbf{v}) = \left(\cos\left(\frac{\theta}{2}\right), \left[e_x \sin\left(\frac{\theta}{2}\right), e_y \sin\left(\frac{\theta}{2}\right), e_z \sin\left(\frac{\theta}{2}\right)\right]\right) \tag{2.14}$$

Quaternions can be composed by quaternion multiplication, which is defined as:

$$q_1 q_2 = (s_1, \mathbf{v}_1) \cdot (s_2, \mathbf{v}_2) = (s_1 s_2 - \mathbf{v}_1 \cdot \mathbf{v}_2, s_1 \mathbf{v}_2 + s_2 \mathbf{v}_1 + \mathbf{v}_1 \times \mathbf{v}_2) \tag{2.15}$$

As with the composition of rotation matrices, quaternion composition is associative but not commutative. Rotation of a vector is defined by: $T_q(v) = qv\bar{q}$.

Quaternions are widely applied due their compact format, simple operations, their relation to angle-axis representation, and their freedom from gimbal lock (vis-à-vis Euler angles).

2.4 Stochastic Filtering: A Framework for Sequential State Estimation

In this section we will explore the theoretical and mathematical background of the tracking problem and a general statistical framework in which it can be solved.

We provide a formal context for tracking in the form of a general stochastic process which we want to estimate over time. We then explore a stochastic filtering framework for solving the estimation problem, with particular focus on Bayesian filtering methods. We look in detail at the key concepts, and refer the reader to Chen (2003) for a more comprehensive insight.

The problem of tracking, or sequential state estimation is broadly that of inferring the state of a time-varying process given a series of noisy or partial observations of the states. Such processes are often considered to be stochastic processes, in which the state and observations are modelled as series of $\mathcal{X}$- and $\mathcal{Z}$-valued random variables $\mathbf{X}$ and $\mathbf{Z}$ respectively, where $\mathcal{X}$ denotes the state space, and $\mathcal{Z}$ the observation space.

In the discrete time domain, we denote such a process by $\mathbf{X}_{1:n} = \{\mathbf{X}_t : t \in \{1,\ldots,n\}\}$ and the observations by $\mathbf{Z}_{1:n} = \{\mathbf{Z}_t : t \in \{1,\ldots,n\}\}$. Likewise we denote the sets of true states and measured observations as $\mathbf{x}_{1:n}$ and $\mathbf{z}_{1:n}$ respectively.

In such a stochastic system, we can relate the transition between successive states, and relate states to observations in a dynamic state space representation. The discrete-time transition between states $\mathbf{x}_t$ may be described as in eq. 2.16 (where f is the possibly non-linear state transition function, and $\mathbf{w}_t$ the process noise). Observations $\mathbf{z}_t$ of the state are described by eq. 2.17 (where h is the possibly non-linear measurement function and $\mathbf{v}_t$ the measurement noise).

$$\mathbf{x}_t = f(\mathbf{x}_{t-1}) + \mathbf{w}_t \tag{2.16}$$

$$\mathbf{z}_t = h(\mathbf{x}_t) + \mathbf{v}_t \tag{2.17}$$

In this way we can characterize the system with probability density functions (*pdfs*). The pdf $p(\mathbf{x}_t \mid \mathbf{x}_{t-1})$, is the state transition density, and describes the probability of the state at time t being $\mathbf{x}_t$, given the previous state

$\mathbf{x}_{t-1}$. Similarly $p(\mathbf{z}_t \mid \mathbf{x}_{t-1})$ describes the probability of observing $\mathbf{z}_t$ given the current state $\mathbf{x}_t$, and is known as the measurement likelihood.

These formalisms are the basis of *stochastic filtering.* In filtering we wish to obtain optimal estimates of the state of the system over time, given all the measurements observed up until that time. I.e. we wish to find the conditional posterior density over the states given the observed data: $p(\mathbf{x}_t \mid \mathbf{z}_{1:t})$.

The most popular method for computing or estimating the posterior density are based upon a core Bayesian filtering framework. In the next section we introduce the concepts of Bayesian statistics and derive the Bayes filter formulae.

2.4.1 Recursive Bayesian Estimation

The well-known Bayes rule (Eq. 2.18), allows us to formulate a posterior probability $p(X \mid Y)$ in terms of the *prior* probability $p(X)$, the *likelihood* $p(Y \mid X)$ and the *evidence* $p(Y)$.

$$p(X \mid Y) = \frac{p(Y \mid X)p(X)}{p(Y)} \tag{2.18}$$

This rule is the basis of the *recursive Bayes filter.* The Bayes filter provides a solution to the stochastic estimation problem under the assumption that the system follows a first-order Markov process (Markov assumption): $p(\mathbf{x}_t \mid \mathbf{x}_{0:t-1}) = p(\mathbf{x}_t \mid \mathbf{x}_{t-1})$.

We can introduce our quantities of interest into the Bayes formula, and by introducing $\mathbf{z}_{1:t-1}$, we can represent the posterior density in terms of the observation likelihood $p(\mathbf{z}_t \mid \mathbf{x}_t)$ and the prediction given all data observed until the previous time step $p(\mathbf{x}_t \mid \mathbf{z}_{1:t-1})$.

$$
\begin{aligned}
p(\mathbf{x}_t \mid \mathbf{z}_{1:t}) &= \frac{p(\mathbf{z}_{1:t} \mid \mathbf{x}_t)\,p(\mathbf{x}_t)}{p(\mathbf{z}_{1:t})} \\
&= \frac{p(\mathbf{z}_t, \mathbf{z}_{1:t-1} \mid \mathbf{x}_t)\,p(\mathbf{x}_t)}{p(\mathbf{z}_t, \mathbf{z}_{1:t-1})} \\
&= \frac{p(\mathbf{z}_t \mid \mathbf{z}_{1:t-1}, \mathbf{x}_t)\,p(\mathbf{z}_{1:t-1} \mid \mathbf{x}_t)\,p(\mathbf{x}_t)}{p(\mathbf{z}_t \mid \mathbf{z}_{1:t-1})\,p(\mathbf{z}_{1:t-1})} \\
&= \frac{p(\mathbf{z}_t \mid \mathbf{z}_{1:t-1}, \mathbf{x}_t)\,p(\mathbf{x}_t \mid \mathbf{z}_{1:t-1})\,p(\mathbf{z}_{1:t-1})\,p(\mathbf{x}_t)}{p(\mathbf{z}_t \mid \mathbf{z}_{1:t-1})\,p(\mathbf{z}_{1:t-1})\,p(\mathbf{x}_t)} \\
&= \frac{p(\mathbf{z}_t \mid \mathbf{x}_t)\,p(\mathbf{x}_t \mid \mathbf{z}_{1:t-1})}{p(\mathbf{z}_t \mid \mathbf{z}_{1:t-1})}
\end{aligned}
\tag{2.19}
$$

By introducing a marginalising integral into the prior and evidence terms we obtain the recursive formulation of the Bayes filter in terms of the filtered output of the previous time step.

$$
\begin{aligned}
p(\mathbf{x}_t \mid \mathbf{z}_{1:t}) &= \frac{p(\mathbf{z}_t \mid \mathbf{x}_t) \int p(\mathbf{x}_t \mid \mathbf{x}_{t-1})\,p(\mathbf{x}_{t-1} \mid \mathbf{z}_{t-1})\,\mathrm{d}\mathbf{x}_{t-1}}{\int p(\mathbf{z}_t \mid \mathbf{x}_t)\,p(\mathbf{x}_t \mid \mathbf{z}_{1:t-1})\,\mathrm{d}\mathbf{x}_t} \\
&= \frac{p(\mathbf{z}_t \mid \mathbf{x}_t) \int p(\mathbf{x}_t \mid \mathbf{x}_{t-1})\,p(\mathbf{x}_{t-1} \mid \mathbf{z}_{t-1})\,\mathrm{d}\mathbf{x}_{t-1}}{\int p(\mathbf{z}_t \mid \mathbf{x}_t) \int p(\mathbf{x}_t \mid \mathbf{x}_{t-1})\,p(\mathbf{x}_{t-1} \mid \mathbf{z}_{t-1})\,\mathrm{d}\mathbf{x}_{t-1}\mathrm{d}\mathbf{x}_t} \\
&\propto \underbrace{p(\mathbf{z}_t \mid \mathbf{x}_t)}_{\text{filtering}} \underbrace{\int p(\mathbf{x}_t \mid \mathbf{x}_{t-1})\,p(\mathbf{x}_{t-1} \mid \mathbf{z}_{t-1})\,\mathrm{d}\mathbf{x}_{t-1}}_{\text{prediction}}
\end{aligned}
\tag{2.20}
$$

Hence we have a framework for computing the posterior density $p(\mathbf{x}_t \mid \mathbf{z}_{1:t})$ in a recursive manner in terms of the prediction density computed from the filtering output of the previous step $p(\mathbf{x}_{t-1} \mid \mathbf{z}_{t-1})$, the state transition density $p(\mathbf{x}_t \mid \mathbf{x}_{t-1})$, and by filtering with the likelihood $p(\mathbf{z}_t \mid \mathbf{x}_t)$ of the latest observation.

Despite its recursive formulation, the Bayes filter does represent a practical algorithm for computing the posterior density, since its computation is intractible in all but the most trivial of cases.

2.4.1.1 Bayes Filters in Practice

Kalman Filter

The Kalman filter is a practical implementation of the recursive Bayes filter in the case that state transition and measurement functions are constrained to be linear functions of the state, and when the process and measurement noise is normally-distributed with known mean and covariance. Under these constraints the Kalman filter estimates the state $\mathbf{x}_t$ which maximizes $p(\mathbf{x}_t \mid \mathbf{z}_{1:t})$ – the Maximum A Posteriori (MAP) estimate, and has been shown to be optimal for Linear Gaussian applications.

Most dynamical systems, however, do not satisfy the Linear Gaussian constraints. Often both the state transition and measurement functions feature heavy non-linearities. Observations from visual sensors possess many sources of non-linearity – a particularly obvious example being occlusion. Although variations on the Kalman filter have been proposed which deal with these non-linear functions, large non-linearities can cause the filter to diverge.

Particle Filter

The particle filter is a non-parametric realisation of sequential Bayesian estimation based on Monte Carlo sampling techniques, and is flexible enough to handle non-linear state transition and measurement functions and arbitrary noise densities. In the next section we will explore the particle filter and its derivation in more detail.

2.4.2 Particle Filtering Techniques

The particle filter is the result of the application of Monte Carlo sampling techniques to the recursive Bayesian estimation framework.

In contrast to the Kalman filter, in which the posterior density is represented in a parametric way by the mean and covariance of a normal distribution, the Particle filter approximates the posterior by a number of independent and identically distributed samples of the underlying distribution.

We denote the set of particles at time t by:

$\mathcal{X}_t = \{x_t^{(0)}, x_t^{(1)}, \ldots, x_t^{(N)}\}$

We concentrate on the so-called Sampling Importance Resampling (SIR) particle filter in which particles are drawn from a proposal distribution $q(\mathbf{x})$, weighted via the importance sampling approach, and finally resampled proportional to the particle weights.

2.4.2.1 Monte Carlo Sampling

Given a random variable $\mathbf{x}$ with pdf $p(\mathbf{x})$, and a function $f(\mathbf{x})$ whose value we wish to estimate, The expected value of $f(\mathbf{x})$ is given by:

$$\mathrm{E}[f(\mathbf{x})] = \int_{\mathbf{x}} f(\mathbf{x})\, p(\mathbf{x})\, \mathrm{d}\mathbf{x} \tag{2.21}$$

Monte Carlo techniques approximate this quantity by considering a number of i.i.d. random variables $\{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(N)}\}$ drawn from $p(\mathbf{x})$. Given this sequence of random samples of $p(\mathbf{x})$, the Monte Carlo estimate of eq. 2.21 is given by:

$$\mathrm{E}_{MC(N)}[f(\mathbf{x})] = \frac{1}{N}\sum_{i=1}^{N} f\left(\mathbf{x}^{(i)}\right) \tag{2.22}$$

2.4.2.2 Importance Sampling

Importance Sampling is a technique which aims to concentrate samples in the "important" areas of the target distribution $p(\mathbf{x})$. Since it is often not possible to draw samples directly from this distribution, the approach of Importance Sampling is to consider an alternative, related distribution $q(\mathbf{x})$, known as the proposal, from which it is possible to draw samples, under the assumption that $p(\mathbf{x}) > 0 \rightarrow q(\mathbf{x}) > 0$.

Considering again eq. 2.21, we can rewrite the integral using $q(\mathbf{x})$ and a weight $W(\mathbf{x}) = \dfrac{p(\mathbf{x})}{q(\mathbf{x})}$ which accounts for the difference between $p(\mathbf{x})$ and $q(\mathbf{x})$.

$$\mathrm{E}[f(\mathbf{x})] = \int_{\mathbf{x}} f(\mathbf{x})\, W(\mathbf{x})\, q(\mathbf{x})\, \mathrm{d}x \tag{2.23}$$

The Importance sampling technique estimates this value using a sequence of Monte Carlo samples drawn from the proposal distribution $q(\mathbf{x})$.

$$\mathrm{E}_{IS(N)}[f(\mathbf{x})] = \frac{1}{\sum_{i=1}^{N} W^{(i)}}\sum_{i=1}^{N} W^{(i)} f\left(\mathbf{x}^{(i)}\right) \tag{2.24}$$

where $W^{(i)} = \frac{p(\mathbf{x}^{(i)})}{q(\mathbf{x}^{(i)})}$.

2.4.2.3 Sequential Importance Sampling

With the aim of achieving a sequential importance sampling method, we can rewrite the posterior density $p(\mathbf{x}_{0:t} \mid \mathbf{z}_{1:t})$ in a factorized form:

$$\begin{aligned} p(\mathbf{x}_{0:t} \mid \mathbf{z}_{1:t}) &= p(\mathbf{x}_t \mid \mathbf{x}_{0:t-1}, \mathbf{z}_{1:t}) p(\mathbf{x}_{0:t-1} \mid \mathbf{z}_{1:t-1}) \\ &= \frac{p(\mathbf{z}_t \mid \mathbf{x}_t) p(\mathbf{x}_t \mid \mathbf{x}_{t-1})}{p(\mathbf{z}_t \mid \mathbf{z}_{1:t-1})} p(\mathbf{x}_{0:t-1} \mid \mathbf{z}_{1:t-1}) \end{aligned} \tag{2.25}$$

The proposal can be factorized in a similar way:

$$q(\mathbf{x}_{0:t} \mid \mathbf{z}_{1:t}) = q(\mathbf{x}_t \mid \mathbf{x}_{0:t-1}, \mathbf{z}_{1:t}) q(\mathbf{x}_{0:t-1} \mid \mathbf{z}_{1:t-1}) \tag{2.26}$$

In this way we can derive a recursive form for the importance weights:

$$\begin{aligned} W_t^{(i)} &= \frac{p(\mathbf{x}_{0:t}^{(i)} \mid \mathbf{z}_{1:t})}{q(\mathbf{x}_{0:t}^{(i)} \mid \mathbf{z}_{1:t})} \\ &\propto \frac{p(\mathbf{z}_t \mid \mathbf{x}_t^{(i)}) p(\mathbf{x}_t^{(i)} \mid \mathbf{x}_{t-1}^{(i)}) p(\mathbf{x}_{0:t-1}^{(i)} \mid \mathbf{z}_{1:t-1})}{q(\mathbf{x}_t^{(i)} \mid \mathbf{x}_{0:t-1}^{(i)}, \mathbf{z}_{1:t}) q(\mathbf{x}_{0:t-1}^{(i)} \mid \mathbf{z}_{1:t-1})} \\ &\propto \frac{p(\mathbf{z}_t \mid \mathbf{x}_t^{(i)}) p(\mathbf{x}_t^{(i)} \mid \mathbf{x}_{t-1}^{(i)})}{q(\mathbf{x}_t^{(i)} \mid \mathbf{x}_{0:t-1}^{(i)}, \mathbf{z}_{1:t})} W_{t-1}^{(i)} \end{aligned} \tag{2.27}$$

If we assume that the current state is independent of all previous observations, given the latest observation, i.e. $q(\mathbf{x}_t^{(i)} \mid \mathbf{x}_{0:t-1}^{(i)}, \mathbf{z}_t) = q(\mathbf{x}_t^{(i)} \mid \mathbf{x}_{0:t-1}^{(i)}, \mathbf{z}_{1:t})$ then we can rewrite eq. 2.27 for estimating the posterior for only the latest state, hence the weights become:

$$W_t^i = \frac{p(\mathbf{z}_t \mid \mathbf{x}_t^{(i)}) p(\mathbf{x}_t^{(i)} \mid \mathbf{x}_{t-1}^{(i)})}{q(\mathbf{x}_t^{(i)} \mid \mathbf{x}_{0:t-1}^{(i)}, \mathbf{z}_t)} W_{t-1}^{(i)} \tag{2.28}$$

2.4.2.4 The Sampling Importance Resampling (SIR) Particle Filter

The Sequential Importance Sampling filter provides a means of estimating the posterior, and provided the proposal is sufficiently close to the target distribution, the estimate converges to the true value as the number of particles goes to infinity.

After successive iterations of the SIS filter, often we have many particles with very low weight, and only a few with large weight. Hence the meaningful region of the target distribution is covered by only few particles, which can introduce large errors into the estimate. This phenomenon is known as *particle depletion*.

In order to avoid particle depletion, we can discard samples of very low weight, and duplicate particles of high weight in order to focus particles in the important regions of the target distribution. This process is known as resampling, and involves drawing, with replacement, samples from the particle set with probability of drawing a particle proportional to the weight of each particle.

I.e. we draw new particles from the discrete approximation of the density $p(\mathbf{x}_t \mid \mathbf{z}_{1:t})$ defined by the weighted particle set:

$$p(\mathbf{x}_t \mid \mathbf{z}_{1:t}) \approx \sum_{i=1}^{n} W_t^{(i)} \delta\left(\mathbf{x}_t - \mathbf{x}_t^{(i)}\right)$$

2.4.2.5 Choice of Proposal Distribution

The choice of proposal distribution is an important consideration in particle filtering techniques. The optimal proposal distribution is given by $q = p(\mathbf{x}_t \mid \mathbf{z}_{1:t})$, however usually there is no closed form solution for this density, and it not possible to draw samples directly from it.

A common choice for the proposal distribution is to use the state transition density $p(\mathbf{x}_t \mid \mathbf{x}_{t-1})$, from which it is often possible to draw samples. Under this condition, we obtain the Bootstrap filter (Gordon et al., 1993), and the computation of the importance weights in eq. 2.28 simplifies to:

$$W_t^i = p(\mathbf{z}_t \mid \mathbf{x}_t^{(i)}) W_{t-1}^{(i)} \tag{2.29}$$

If we resample in every step then eq. 2.27 becomes $W_t^i = p(\mathbf{z}_t \mid \mathbf{x}_t^{(i)})$.

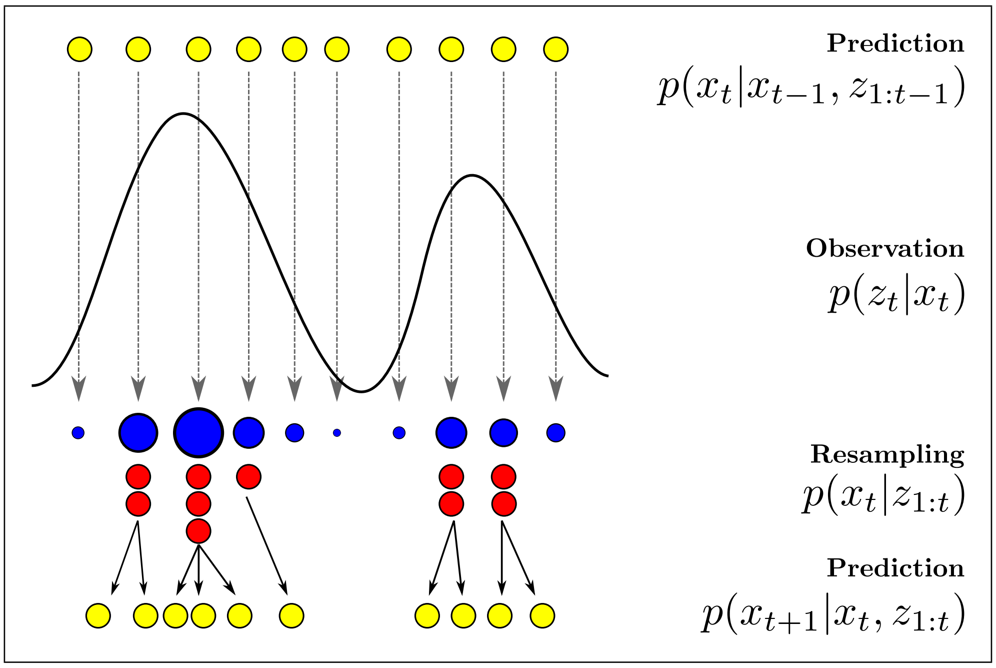


Figure 2.6: Overview of the steps in a Sampling Importance Resampling Particle Filter

A general algorithm for such a SIR Particle Filter is outlined in algorithm 1.

2.4.2.6 Particle Filtering on Lie Groups

The performance of a Particle Filter depends to a large degree on the validity of the state transition model $p(x_t \mid x_{t-1})$, since a good model of the state dynamics yields a valid proposal distribution. A number of different models have been suggested for modelling the state dynamics, which vary in their suitability for different applications.

A particularly simple model is the random walk. In a first order random walk model velocity is included in the state and is subjected to perturbations modelled as a discrete-time diffusion process.

Monte Carlo Filtering on Matrix Lie Groups

When the state dynamics of an application evolve on a Lie Group, it is necessary that the model should conform to its underlying structure. Chiuso

Algorithm 1 Particle Filter algorithm

```
1: Initialization: Draw $n$ samples $\mathbf{x}_0^{(i)} \sim p(\mathbf{x}_0)$, set weights $W_t^{(i)} = \frac{1}{n}$
2: for time steps $t = 0$ to $n$ do
3:     $\mathcal{X}_t \leftarrow \emptyset$
4:     for $i = 1$ to $m$ do
5:         Sampling: $\mathbf{x}_t^{(i)} \sim p\left(\mathbf{x}_t \mid \mathbf{x}_{0:t-1}^{(i)}\right)$
6:         Importance weighting: $W_t^{(i)} \leftarrow p\left(\mathbf{z}_t \mid \mathbf{x}_t^{(i)}\right) W_{t-1}^{(i)}$
7:     end for
8:     for $j = 1$ to $m$ do
9:         Resampling: draw $\mathbf{x}_t^{(j)} \propto p\left(\mathbf{x}_t^{(j)} = \mathbf{x}_t^{(i)}\right) = W_t^{(i)}$
10:        $\mathcal{X}_t \leftarrow \mathcal{X}_t \cup \mathbf{x}_t^{(j)}$
11:    end for
12: end for
```

and Soatto (2000) present a first-order random walk dynamics model for the stochastic estimation of state dynamics evolving on Matrix Lie Groups. That is stochastic differential systems of the form:

$$\begin{aligned} \mathrm{d}X &= \mathrm{e}^{v} X \,\mathrm{d}t \\ \mathrm{d}v &= \mathrm{d}W \end{aligned} \tag{2.30}$$

where X is a member of a Matrix Lie Group G, v an element of the corresponding Lie Algebra g, and W is a diffusion process on g.

A more informed model of the state dynamics may be obtained by employing an *autoregressive* state dynamics model, which predicts the state evolution based on previous estimates of the state.

Discrete-time State Equations on $SE(3)$

In Kwon et al. (2007), a framework is presented for particle filter applications whose state transition evolves on the Euclidean Group $SE(3)$.

The discrete-time state transition equations are obtained via the first-order Euler discretization of the continuous-time equations on $SE(3)$:

$$
\begin{aligned}
\mathbf{X}_t &= \mathbf{X}_{t-1} \cdot \exp\left(\mathbf{A}\left(\mathbf{X}, t\right) \mathit{\Delta} t + \mathrm{d}\mathbf{W}_t \sqrt{\mathit{\Delta} t}\right) \\
\mathrm{d}\mathbf{W}_t &= \sum_{i=1}^{6} \epsilon_{t,i} \mathbf{E}_i \\
\epsilon_t &= [\epsilon_{t,1}, \ldots, \epsilon_{t,6}] \sim \mathcal{N}\left(\mathbf{0}^{6\times 1}, \mathbf{\Sigma}_w\right)
\end{aligned}
\tag{2.31}
$$

where $\mathbf{X}_t \in SE(3)$ is the state at time t, $A : SE(3) \to \mathfrak{se}(3)$ is a possible non-linear map, $\mathrm{d}\mathbf{W}_t$ is a Wiener process noise on $\mathfrak{se}(3)$ with covariance $\mathbf{\Sigma}_w$, and where $E_{1:6}$ are the basis elements of $\mathfrak{se}(3)$:

$$
\begin{aligned}
\mathbf{E}_1 &= \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \mathbf{E}_2 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \mathbf{E}_3 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \\
\mathbf{E}_4 &= \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \mathbf{E}_5 = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \mathbf{E}_6 = \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}
\end{aligned}
\tag{2.32}
$$

Autoregressive State Dynamics on $SE(3)$

Autoregressive (AR) state dynamics models have been seen to often be a more informed choice over random walk models since they incorporate past estimates of the state.

AR models on Lie Groups are considered in Kwon and Park (2010), where visual tracking is performed on the 2D Affine Group, while for the Euclidean Group, Choi and Christensen (2011, 2012a) use a particle filter with AR dynamics for tracking of textured and textureless objects.

For the AR state dynamics the state transition equation in eq. 2.31 then becomes:

$$
\begin{aligned}
\mathbf{X}_t &= \mathbf{X}_{t-1} \cdot \exp\left(\mathbf{A}_{t-1} \mathit{\Delta} t + \mathrm{d}\mathbf{W}_t \sqrt{\mathit{\Delta} t}\right) \\
\mathbf{A}_{t-1} &= \lambda_{ar} \frac{1}{\mathit{\Delta} t} \log\left(\mathbf{X}_{t-2}^{-1} \mathbf{X}_{t-1}\right)
\end{aligned}
\tag{2.33}
$$

where λ_{ar} is the AR-process parameter.

2.4.2.7 Improved Proposal Particle Filtering

In some applications which are computationally intensive, or when the state space has a high number of dimensions, it is often desired to keep the number of samples relatively low in order to retain good performance. To this end, it is advantageous to consider a proposal distribution which better reflects the target when drawing samples, since fewer particles are required to cover the regions of high likelihood in the target distribution.

It is not always possible to track with a low number of samples when taking the transition density $p(\mathbf{x}_t \mid \mathbf{x}_{t-1})$ as the proposal distribution, since there is often a significant mismatch between the proposal and target - with the result that many particles receive low weights and hence a large degree of redundancy is introduced.

According to Doucet et al. (2000) the optimal proposal distribution with respect to the variance of the importance weights is given by:

$$q\left(\mathbf{x}_t \mid \mathbf{x}_{0:t-1}^{(i)}, \mathbf{z}_{1:t}\right) = p\left(\mathbf{x}_t \mid \mathbf{x}_{t-1}^{(i)}, \mathbf{z}_t\right) \tag{2.34}$$

In general eq. 2.34 does not have a closed form and as such, is difficult to sample. However, since the aim is to choose a proposal distribution which is as close as possible to the target, then under certain assumptions, we can often find better approximations to eq. 2.34 than the state transition model by incorporating the latest observation.

Integrating the Latest Observation

Several approaches have been suggested which are based on integrating information from the latest sensor data into the proposal distribution, which is otherwise discarded when sampling from the transition distribution.

In the Auxiliary Particle Filter of Pitt and Shephard (1999), the particles which are to receive high weights are determined by simulation in each time step in order to construct a more informed proposal distribution. This technique is related to the integration a secondary filter in order to compute an improved proposal distribution. Kalman filters and variants have been suggested as an efficient way to compute the proposal (see e.g. Li et al. (2003); Rui and Chen (2001)).

By integrating a secondary particle filter, Shen et al. (2005), propose a second filtering- and sampling-step in each iteration in order to sample particles from areas of high likelihood.

2.4.2.8 Improved Proposals in SLAM

In this section we present the application of improved proposal sampling in two approaches to the Simultaneous Localisation and Mapping (SLAM) problem.

FastSLAM 2.0

In Montemerlo et al. (2003), improved proposal particle filtering is applied to the landmark-based SLAM problem, yielding accurate maps with only a small number of particles. The improved proposal distribution used for sampling is constructed for each particle and incorporates the particle previous state $x_{t-1}^{[i]}$, latest odometry and sensor measurements u_t, z_t, and a data association variable c_t, and is given by:

$$x_t^{[i]} \sim p\left(x_t \middle| x_{1:t-1}^{[i]}, u_{1:t}, z_{1:t}, c_{1:t}\right) \tag{2.35}$$

In order to derive the sampling distribution, we rewrite in terms of the known distributions and motion and sensor models (from Montemerlo et al. (2003); Thrun et al. (2005)):

$$\begin{aligned}
&p(x_t|x_{1:t-1}^{[i]}, u_{1:t}, z_{1:t}, c_{1:t}) \\
&\overset{Bayes}{=} \quad \frac{p(z_t|x_t, x_{1:t-1}^{[i]}, u_{1:t}, z_{1:t-1}, c_{1:t})\, p(x_t|x_{1:t-1}^{[i]}, u_{1:t}, z_{1:t-1}, c_{1:t})}{p(z_t|x_{1:t-1}^{[i]}, u_{1:t}, z_{1:t-1}, c_{1:t})} \\
&= \quad \eta^{[i]} p(z_t|x_t, x_{1:t-1}^{[i]}, u_{1:t}, z_{1:t-1}, c_{1:t})\, p(x_t|x_{1:t-1}^{[i]}, u_{1:t}, z_{1:t-1}, c_{1:t}) \\
&\overset{Markov}{=} \quad \eta^{[i]} p(z_t|x_t, x_{1:t-1}^{[i]}, u_{1:t}, z_{1:t-1}, c_{1:t})\, p(x_t|x_{t-1}^{[i]}, u_t) \\
&= \quad \eta^{[i]} \int p(z_t|m_{c_t}, x_t, x_{1:t-1}^{[i]}, u_{1:t}, z_{1:t-1}, c_{1:t}) \\
&\qquad p(m_{c_t}|x_t, x_{1:t-1}^{[i]}, u_{1:t}, z_{1:t-1}, c_{1:t}) \mathrm{d}m_{c_t}\; p(x_t|x_{t-1}^{[i]}, u_t) \\
&\overset{Markov}{=} \quad \eta^{[i]} \int \underbrace{p(z_t|m_{c_t}, x_t, c_t)}_{\sim \mathcal{N}(z_t; h(m_{c_t}, x_t), Q_t)}\; \underbrace{p(m_{c_t}|x_{1:t-1}^{[i]}, z_{1:t-1}, c_{1:t})}_{\sim \mathcal{N}(m_{c_t}; \mu_{c_t,t-1}^{[i]}, \Sigma_{c_t,t-1}^{[i]})} \mathrm{d}m_{c_t} \\
&\qquad \underbrace{p(x_t|x_{t-1}^{[i]}, u_t)}_{\sim \mathcal{N}(x_t; g(x_{-1}^{[i]}, u_t), R_t)}
\end{aligned}$$

where m_{c_t} is an associated landmark defined by mean and covariance $\mu_{c_t}^{[i]}, \Sigma_{c_t}^{[i]}$.

Hence, the sampling distribution is given by the product of two factors, the state transition distribution, which is normally distributed with mean $g(x_{-1}^{[i]}, u_t)$ and covariance R_t, and the probability of the measurement given by a convolution of Gaussians.

In general a closed form solution of this equation is not available, however the authors construct a Gaussian approximation by linearizing the measurement function:

$$h\left(m_{c_t}, x_t\right) \approx \hat{z}_t^{[i]} + H_m \left(m_{c_t} - \mu_{c_t,t-1}^{[i]}\right) + H_x \left(x_t - \hat{x}_t^{[i]}\right) \tag{2.36}$$

where m_{c_t} is the observed landmark, $\hat{z}_t^{[i]} = h(\mu_{c_t t-1}^{[i]}, \hat{x}_t^{[i]})$ is the expected measurement given the associated landmark estimate $\mu_{c_t,t-1}^{[i]}$ and propagated particle pose $\hat{x}_t^{[i]} = f(x_{t-1}^{[i]}, u_t)$. H_m and H_x are the Jacobian matrices of first derivatives of h w.r.t. m_c and x_t respectively.

The Gaussian proposal distribution is then given by the EKF-style measurement update:

$$\begin{aligned} \Sigma_{x_t}^{[i]} &= \left[H_x^T Q_t^{[i]-1} H_x + R_t^{-1}\right]^{-1} \\ \mu_{x_t}^{[i]} &= \Sigma_{x_t}^{[i]} H_x^T Q_t^{[i]-1} \left(z_t - \hat{z}_t^{[i]}\right) + \hat{x}_t^{[i]} \end{aligned} \tag{2.37}$$

where R_t is the covariance matrix associated with the transition noise, and $Q_t^{[k]} = Q_t + H_m \Sigma_{c_t,t-1}^{[k]} H_m^T$, where Q_t is the covariance associated with the measurement noise, and $\Sigma_{c_t,t-1}^{[k]}$ is the estimated landmark uncertainty.

When sampling the proposal in eq. 2.35, special attention needs to be given to the importance weights, which are given by (from Montemerlo et al. (2003);

Thrun et al. (2005)):

$$
\begin{aligned}
w_t^{[i]} &= \frac{p(x_t^{[i]}|u_{1:t}, z_{1:t}, c_{1:t})}{p(x_t^{[i]}|x_{1:t-1}^{[i]}, u_{1:t}, z_{1:t}, c_{1:t})p(x_{1:t-1}^{[i]}|u_{1:t-1}, z_{1:t-1}, c_{1:t-1})} \\
&= \frac{\cancel{p(x_t^{[i]}|x_{1:t-1}^{[i]}, u_{1:t}, z_{1:t}, c_{1:t})}p(x_{1:t-1}^{[i]}|u_{1:t}, z_{1:t}, c_{1:t})}{\cancel{p(x_t^{[i]}|x_{1:t-1}^{[i]}, u_{1:t}, z_{1:t}, c_{1:t})}p(x_{1:t-1}^{[i]}|u_{1:t-1}, z_{1:t-1}, c_{1:t-1})} \\
&= \frac{p(x_{1:t-1}^{[i]}|u_{1:t}, z_{1:t}, c_{1:t})}{p(x_{1:t-1}^{[i]}|u_{1:t-1}, z_{1:t-1}, c_{1:t-1})} \\
&\overset{Bayes}{=} \eta \frac{p(z_t|x_{1:t-1}^{[i]}, u_{1:t}, z_{1:t-1}, c_{1:t})p(x_{1:t-1}^{[i]}|u_{1:t}, z_{1:t-1}, c_{1:t})}{p(x_{1:t-1}^{[i]}|u_{1:t-1}, z_{1:t-1}, c_{1:t-1})} \\
&\overset{Markov}{=} \eta \frac{p(z_t|x_{1:t-1}^{[i]}, u_{1:t}, z_{1:t-1}, c_{1:t})\cancel{p(x_{1:t-1}^{[i]}|u_{1:t-1}, z_{1:t-1}, c_{1:t-1})}}{\cancel{p(x_{1:t-1}^{[i]}|u_{1:t-1}, z_{1:t-1}, c_{1:t-1})}} \\
&= \eta\, p(z_t|x_{1:t-1}^{[i]}, u_{1:t}, z_{1:t-1}, c_{1:t})
\end{aligned}
$$

Hence, by bringing the current state x_t into the above equation:

$$
\begin{aligned}
w_t^{[i]} &= \eta \int p(z_t|x_t, x_{1:t-1}^{[i]}, u_{1:t}, z_{1:t-1}, c_{1:t}) \\
&\quad p(x_t|x_{t-1}^{[i]}, u_{1:t}, z_{1:t-1}, c_{1:t})\mathrm{d}x_t \\
&\overset{Markov}{=} \eta \int p(z_t|x_t, x_{1:t-1}^{[i]}, u_{1:t}, z_{1:t-1}, c_{1:t})p(x_t|x_{t-1}^{[i]}, u_{1:t})\mathrm{d}x_t \\
&= \eta \int \int p(z_t|m_{c_t}, x_t, x_{1:t-1}^{[i]}, u_{1:t}, z_{1:t-1}, c_{1:t}) \\
&\quad p(m_{c_t}|x_t, x_{1:t-1}^{[i]}, u_{1:t}, z_{1:t-1}, c_{1:t})\mathrm{d}m_{c_t}\; p(x_t|x_{t-1}^{[i]}, u_{1:t})\mathrm{d}x_t \\
&\overset{Markov}{=} \eta^{[i]} \int \underbrace{p(x_t|x_{t-1}^{[i]}, u_{1:t})}_{\sim\mathcal{N}(x_t; g(\hat{x}_{-1}^{[i]}, u_t), R_t)} \int \underbrace{p(z_t|m_{c_t}, x_t, c_t)}_{\sim\mathcal{N}(z_t; h(m_{c_t}, x_t), Q_t)} \\
&\quad \underbrace{p(m_{c_t}|x_{1:t-1}^{[i]}, u_{1:t-1}, z_{1:t-1}, c_{1:t})}_{\sim\mathcal{N}(m_{c_t}; \mu_{c_t,t-1}^{[i]}, \Sigma_{c_t,t-1}^{[i]})}\mathrm{d}m_{c_t}\mathrm{d}x_t
\end{aligned}
$$

Again by linearization of the non-linear measurement function h, this can be approximated by a Gaussian over measurements z_t with the following parameters:

$$L_t^{[i]} = H_x^T Q_t H_x + H_m \Sigma_{c_t,t-1}^{[i]} H_m^T + R_t \tag{2.38}$$

Finally the particle weights are computed by:

$$w_t^{([i])} = |2\pi L_t^{[i]}|^{-\frac{1}{2}} \exp\left\{ -\frac{1}{2}(z_t - \hat{z}_t) L_t^{[i]-1} (z_t - \hat{z}_t)) \right\} \tag{2.39}$$

Grid-based SLAM

In Grisetti et al. (2007), also integrate the latest observation to derive an improved proposal distribution. The particles are first propagated by the motion model, then, for each particle, a scan-matching procedure is carried out to locate the mode. Around this maximum the posterior is locally approximated by a Gaussian.

In this case, the Gaussian proposal is computed not using an EKF-style measurement update, but by unscented transform around the maximum given by the scan-matcher, $\hat{x}_t^{(i)} = \text{argmax}\, p(x_t|m_{t-1}^{[i]}, z_t, x_t^{(i)\prime})$, where $x_t^{(i)\prime}$ is the propagated particle pose.

The proposal distribution employed is given by:

$$p(x_t|m_{t-1}^{[i]}, x_{t-1}^{[i]}), z_t, u_{t-1}) \overset{Bayes}{=} \frac{p(z_t|m_{t-1}^{[i]}, x_t)\, p(x_t|x_{t-1}^{[i]}, u_{t-1})}{p(z_t|m_{t-1}^{[i]}, x_{t-1}^{[i]}, u_{t-1})} \tag{2.40}$$

where $m_{t-1}^{[i]}$ is the map of the i-th particle from the previous timestep.

again the importance weights require special attention, in this case they are given by:

$$\begin{aligned} w_t^{[i]} &= w_{t-1}^{(i)}\, p(z_t|m_{t-1}^{[i]}, x_{t-1}^{[i]}, u_t) \\ &= w_{t-1}^{(i)} \int p(z_t|x')\, p(x'|x_{t-1}^{[i]}, u_t) \mathrm{d}x' \end{aligned}$$

The integral is then approximated by a sum computed from the unscented transform of samples within around $\hat{x}_t^{(i)}$

2.5 Voting-based Object Detection and Pose Estimation

The Generalized Hough Transform is a standard technique for fitting the parameters of a model to observed data in a voting framework. Approaches to object detection and pose estimation based on vote accumulation techniques similar to the Generalized Hough Transform have been the subject of a number of recent works.

In Drost et al. (2010) an object model is defined by a number of point-pair features which describe the relative location and geometry of two oriented points. Features computed from the input image are then matched to the model and votes cast in a reduced pose parameter space.

2.5.1 Alignment using Point-pair Features

A point pair is an ordered relation $\{(m_1,\mathbf{n}_1),(m_2,\mathbf{n}_2)\}$ relating two oriented points m_1,m_2, with normals $\mathbf{n}_1,\mathbf{n}_2$.

Point pair features describe the geometrical relationship of a point pair by:

$$F(m_1,m_2) = [\|\mathbf{d}\|_2, \angle(\mathbf{n}_1,\mathbf{d}), \angle(\mathbf{n}_2,\mathbf{d}), \angle(\mathbf{n}_1,\mathbf{n}_2)] \in \mathbb{R}^4 \tag{2.41}$$

where $\mathbf{d} = m_2 - m_1$, $\|\cdot\|_2$ denotes the Euclidean norm, and $\angle(\mathbf{a},\mathbf{b}) \in [0,\pi]$ denotes the angle between the vectors $\mathbf{a},\mathbf{b}$. These features can be efficiently matched between scene and model by quantizing the feature space and using the feature descriptor as a hashing key, in order to access model features with similar features. Fig. 2.7 shows how F describes two oriented points m_1,m_2.

2.5.1.1 Intermediate Coordinate System

Aligning matching point pairs from scene and model is achieved by aligning matching scene and model reference points and their normals in an intermediate coordinate system, and rotating around the normals to align the referred points.

To illustrate this, consider a model point-pair (m_r,m_i). A coordinate frame is defined with its origin at m_r, and the x-axis aligned to with the normal

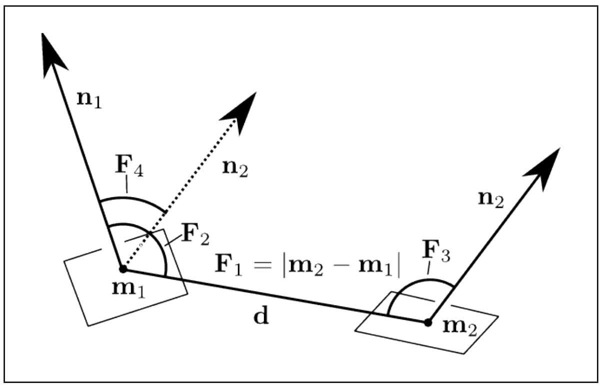


Figure 2.7: Point-pair feature F describing two oriented points. F_1 is the length of the vector from m_1 to m_2, F_2 is the angle between the two normals $\mathbf{n}_1, \mathbf{n}_2$, and F_3 and F_4 are the angles between each normal and the vector $\mathbf{d}$. (Drost et al., 2010)

n_r^m. The transformation from model coordinate to this intermediate frame is denoted $T_{m \to g}$.

For a matching scene point-pair (s_r, s_i), with a reference frame defined by $T_{s \to g}$, the transformation $T_{m \to s}$ which aligns the point-pairs is given by composing three transformations. First, m_r is moved to the origin and its normal rotated to the x-axis by $T_{m \to g}$, similarly for s_r by $T_{s \to g}$. Finally the rotation by α around the x-axis resolves the misalignment of m_i, s_i. Hence, $T_{m \to s}$ is given by:

$$s_i = T_{m \to s} m_i = T_{s \to g}^{-1} R_x(\alpha) T_{m \to g} m_i \tag{2.42}$$

This point-pair alignment scheme is illustrated in fig. 2.8, and defines local coordinates for parameterizing the rigid transformation describing the displacement of scene and model. For a scene point s_r the search space is a reduced to a pair, (m_r, α) which will align model and scene.

2.5.1.2 Voting and Pose Extraction

For a fixed scene reference point s_r, the optimal alignment is determined in a voting scheme for the local coordinates (m_r, α).

Matching point pairs cast votes in a 2-dimensional Hough space spanned by the model reference points, and bins for the angle α. To cast votes, for each

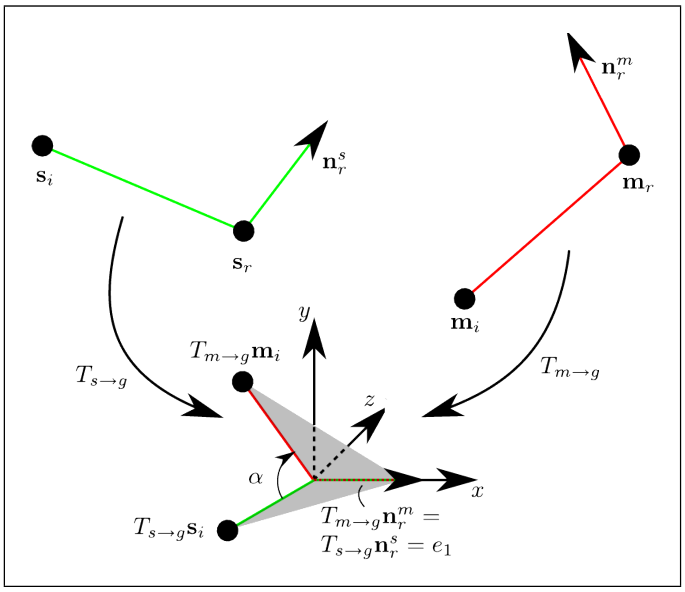


Figure 2.8: The alignment of scene s and model m, point-pair features in an intermediate coordinate frame. (Drost et al., 2010)

pair (s_r, s_i), the matching pairs from the model are found using the feature descriptors as a hashing key.

For each match (m_r, m_i), the angle alpha is determined which aligns (s_r, s_i) and (m_r, m_i) and a vote cast for (m_r, α). After all pairs (s_r, s_i) have been processed, the peaks in the voting space are determined and pose hypotheses determined according to eq. 2.42, before repeating for the next reference point s_r.

The authors suggest an efficient way of determining the angle α, by precomputing for each model point pair (m_r, m_i), the angle α_m which aligns m_i with the half-plane defined by the x- and positive y-axes. Likewise, α_s is computed for each scene pair (s_r, s_i). In the voting loop, the full angle α is given by $\alpha = \alpha_m - \alpha_s$ since $T_{m\to s}$ is given by:

$$\begin{aligned} s_i &= T_{m\to s} m_i \\ &= T_{s\to g}^{-1} R_x(\alpha) T_{m\to g} m_i \\ &= T_{s\to g}^{-1} R_x(-\alpha_s) R_x(\alpha_m) T_{m\to g} m_i \end{aligned} \tag{2.43}$$

In a final step, the hypothesized poses are clustered in an agglomerative clustering scheme which removes isolated poses with low scores and refined by averaging the poses in the best clusters.

2.5.2 Enhancing Point-pair Features

In (Choi et al., 2012), additional feature types were proposed which prove more discriminative at detecting objects which are largely planar, and hence do not exhibit much variety in surface normals. In addition to the point-pair feature already discussed, they implement features which pair points on boundaries, as well as pairing boundary lines, and demonstrate their method in a robotic manipulation task.

Another way to boost the descriptiveness of point-pair features is to consider colour information as in (Choi and Christensen, 2012b). The authors augment the feature in eq. 2.41 with quantized descriptions of the HSV colour values of the points.

3 | Method

In this chapter we present in detail the contributions of this Master thesis. This work builds upon the existing Multi-Resolution Surfel Maps framework of Stückler and Behnke. (2012), in order to perform 3D object detection, pose estimation and tracking. Hence we model our object and input image (scene) using MRSMaps (denoted m_m and m_s respectively throughout this chapter).

This chapter is structured as follows. In Section 3.1 we present our 3D object detection and pose estimation method based on extending MRSMaps with surfel pair features and, by finding consistent arrangements of pairs between scene and model, casting votes for the object pose.

Secondly, in Section 3.2 we present our approach to pose tracking in an Improved Proposal Particle Filter framework. Finally in Section 3.3 we look at an integrated framework for detection and tracking.

3.1 Voting–based Object Detection and Pose Estimation with Surfel Pairs

In this section we provide details of our approach to object detection and pose estimation. Given a known object model m_m and a input scene map m_s we seek to detect the object and estimate the pose of the camera with respect to m_s. We extend the MRSMap representation with colour surfel pair features similar to Drost et al. (2010) and Choi and Christensen (2012b).

Although the point-pair feature of the former performs well for objects with sufficient variation in surface normals, it has been observed that it is less discriminative for objects with planar regions or self-similar structure (Choi et al., 2012; Choi and Christensen, 2012b).

For this reason we include colour information in our surfel pair feature to increase its discriminative power. Our surfel pair feature reflects the relative texture and shape of two surfels s_i and s_j with normals $\mathbf{n}_i$, $\mathbf{n}_j$.

We construct surfel pair relations from existing MRSMaps, and by matching pairs between a scene map m_s and a model map m_m and accumulating votes in a reduced pose parameter Hough space, we recover hypotheses for the object pose, which are refined by in an efficient clustering step.

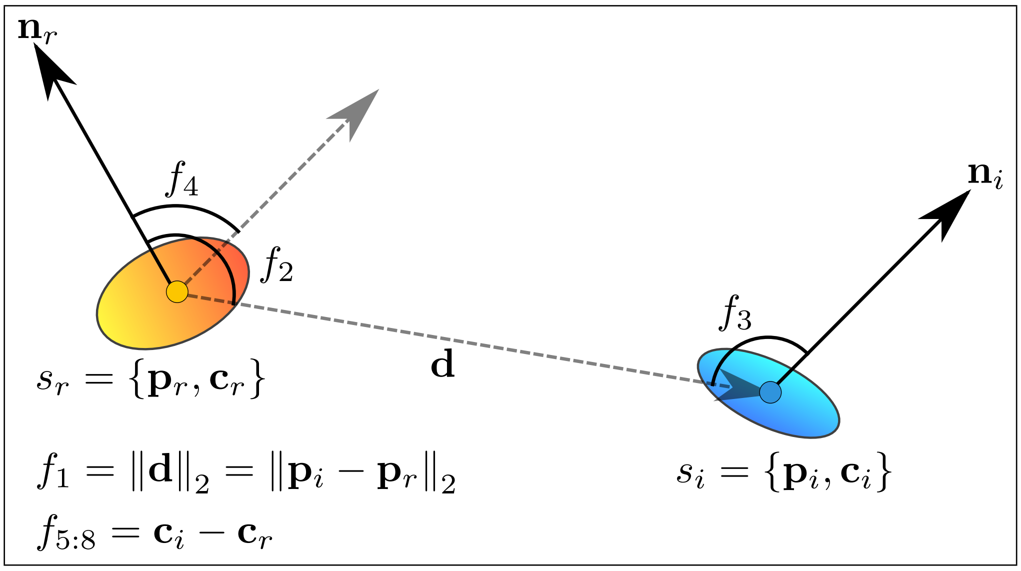

$\mathbf{n}_r$
f_4
f_2
$\mathbf{n}_i$
f_3
$\mathbf{d}$
$s_r = \{\mathbf{p}_r, \mathbf{c}_r\}$
$f_1 = \|\mathbf{d}\|_2 = \|\mathbf{p}_i - \mathbf{p}_r\|_2$
$s_i = \{\mathbf{p}_i, \mathbf{c}_i\}$
$f_{5:8} = \mathbf{c}_i - \mathbf{c}_r$

Figure 3.1: Our colour surfel pair feature encodes the differences in geometry and colour between two surfels. $f_{1:4}$ is defined as in eq 2.41, while the difference between the $L\alpha\beta$ colours of the surfels encodes contrasts in luminance and chrominance.

3.1.1 Surfel Pair Relation

We define a 7-dimensional surfel pair feature extending the point-pair feature of Drost et al. (2010) with 3 extra components which describe the relative contrast in luminance and chrominance between the surfels.

Given a surfel $s = (\mu_s, \Sigma_s)$ with normal $\mathbf{n_s}$, we neglect the covariance and denote the spatial and colour parts of the mean as $\mathbf{p_s} = [p_x, p_y, p_z]^T$ and $\mathbf{c_s} = \left[c_L, c_\alpha, c_\beta\right]^T$, respectively. For ease of notation we represent this triplet as $s = \{\mathbf{p_s}, \mathbf{c_s}, \mathbf{n_s}\}$.

As illustrated in fig. 3.1, each surfel pair relates a *reference* surfel $s_r = \{\mathbf{p}_r, \mathbf{c}_r, \mathbf{n}_r\}$ with a *referred* surfel $s_i = \{\mathbf{p}_i, \mathbf{c}_i, \mathbf{n}_i\}$. Our feature is then defined as follows:

$$F(s_r, s_i) = \left[\|\mathbf{d}\|, \angle(\mathbf{n}_r, \mathbf{d}), \angle(\mathbf{n}_i, \mathbf{d}), \angle(\mathbf{n}_r, \mathbf{n}_i), d_L, d_\alpha, d_\beta\right]^T \in \mathbb{R}^7 \tag{3.1}$$

where d_L, d_α, d_β are given by the difference in the $L\alpha\beta$ colour values of the two surfels:

$$\mathbf{d}_c = \begin{bmatrix} d_L \\ d_\alpha \\ d_\beta \end{bmatrix} = \mathbf{c}_i - \mathbf{c}_r = \begin{bmatrix} \left(c_{L,i} - c_{L,r}\right) \in [-1, 1] \\ \left(c_{\alpha,i} - c_{\alpha,r}\right) \in [-2, 2] \\ \left(c_{\beta,i} - c_{\beta,r}\right) \in \left[-\sqrt{3}, \sqrt{3}\right] \end{bmatrix} \tag{3.2}$$

Hashmap Storage for Efficient Matching

We quantize the feature components, as per eq. 3.3, in order to store similar features in a hash table. For the first four components the values are quantized in steps of d_{dist} and $d_{angle} 2\pi/n_{angle}$ for distance and angles respectively (where n_{angle} is the number of bins for the angle component).

The colour components are first rescaled to the interval $[0, 1]$ (we denote the scaled feature value by $\hat{f}_i$). The normalized feature values are then quantized in steps of $d_{lum} = 1/n_{lum}$, and $d_{chrom} = 1/n_{chrom}$ for luminance and chrominance respectively (where n_{lum} and n_{chrom} are the number of bins).

$$Hashkey\left(\mathbf{f}, d_{dist}, d_{angle}, d_{lum}, d_{chrom}\right) = \begin{bmatrix} \lfloor f_1/d_{dist} \rfloor \\ \lfloor f_2/d_{angle} \rfloor \\ \lfloor f_3/d_{angle} \rfloor \\ \lfloor f_4/d_{angle} \rfloor \\ \lfloor \hat{f}_5/d_{lum} \rfloor \\ \lfloor \hat{f}_6/d_{chrom} \rfloor \\ \lfloor \hat{f}_7/d_{chrom} \rfloor \end{bmatrix} \in \mathbb{Z}^7 \tag{3.3}$$

In a preprocessing step, we compute surfel pair features from the model map m_m. On each resolution r (e.g. $r = 0.5, 0.25, \ldots$) we compute the features according to eq. 3.1, and their hashing index by eq. 3.3 and store lists of features, along with additional information for pose computation in a hash table $\mathcal{H}$.

The procedure for computing surfel pairs and building the feature hash table $\mathcal{H}$ is outlined in algorithm 2, where the input is the model map m_m.

The parameters $d_{dist}, d_{angle}, d_{lum}, d_{chrom}$ are described above, while r_{start} and r_{end} specify the resolutions in the map to use, since too coarse resolutions are not in general meaningful for pose estimation.

The function GetSurfels returns a list of all valid surfels $\{(p_1, n_1, c_1), \ldots, (p_{n_m}, n_{n_m}, c_{n_m})\}_r$ in the model at the specified resolution, while HashInsert inserts the feature into the hashmap at the specified resolution.

Algorithm 2 BuildSurfelPairHashTable

```
    Input: Model $m_m$
    Output: Hash table $\mathcal{H}$
    Params: $d_{dist}, d_{angle}, d_{lum}, d_{chrom}, r_{start}, r_{end}$
    $\mathcal{H} \leftarrow \emptyset$
 1: for $r = r_{start}$ to $r_{end}$ do
 2:     $\mathcal{S} \leftarrow$ GetSurfels($m_m$, $r$ )
 3:     for $m_i \in \mathcal{S}$ do
 4:         for $m_j \in \mathcal{S}$ do
 5:             if $i \neq j$ then
 6:                 $\mathbf{f} = F(m_i, m_j)$                                    ▷ (3.1)
 7:                 $\mathcal{I} \leftarrow$ HashKey($\mathbf{f}, d_{dist}, d_{angle}, d_{lum}, d_{chrom}$)    ▷ (3.3)
 8:                 $\alpha_m \leftarrow$ComputeAngle($m_i, m_j$)                   ▷ (3)
 9:                 HashInsert($\mathcal{H}, r, \mathcal{I}, \{m_i, \alpha_m\}$)
10:             end if
11:         end for
12:     end for
13: end for
```

3.1.2 Surfel Pair Alignment

We consider an arbitrary surfel pair (s_r, s_i) in the scene and assume that it has a corresponding model surfel pair (m_r, m_i). Their alignment is given by the transformation $T_{s \to m} \in SE(3)$ such that $m_i = xT_{s \to m} s_i$, and $m_r = T_{s \to m} s_r$.

The alignment of matching surfel pairs proceeds as in Drost et al. (2010), whereby $T_{s \to m}$ is split into two steps:

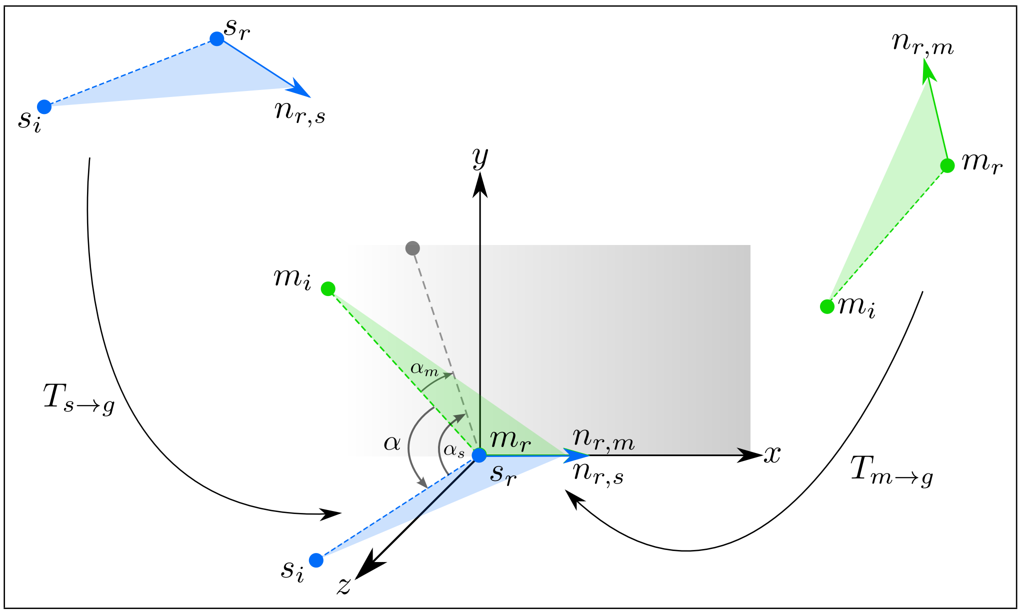


Figure 3.2: Alignment of matching surfel pairs from scene (blue) and model (green). The alignment is accomplished by transforming the pairs into an intermediate coordinate frame, aligning their normals with the x-axis and rotating around the x-axis by α. The planar rotation angle α can be efficiently computed by precomputing the angles α_m, and α_s which align the referred surfels m_i, and s_i with the half-plane defined by the x- and positive y-axes (grey).

1. alignment of scene and model reference surfels and their normals in an intermediate coordinate system
2. alignment of the referred surfels in the intermediate coordinate system by rotation around the aligned normals

Hence, by

$$\begin{aligned} m_r' &= T_{m\to g} m_r \\ s_r' &= T_{s\to g} s_r \end{aligned} \tag{3.4}$$

the model reference surfel m_r, and scene reference surfel s_r are translated to the origin and their normals rotated to align with the x-axis. We denote by $\left(m_r', m_i'\right)$ the image of the model surfel pair in the intermediate coordinate system, and analogously $\left(s_r', s_i'\right)$ for the scene pair.

Since m_i' and s_i' are in general not aligned, we therefore need to resolve their misalignment by rotating m_i' around the aligned normals by α, and hence align the object with the scene. The full transformation $T_{s\to g}$ from the sensor coordinate system to the model coordinate system which aligns object and scene is given by:

$$T_{s\to m} = T_{m\to g}^{-1} R_X(\alpha) T_{s\to g} \tag{3.5}$$

Hence, for s_r, $T_{s\to m}$ is defined only by the matching model reference surfel m_r, and the angle α.

Pre-computation of α

To allow for efficient detection and pose estimation, an optimization was suggested by the authors of the original method which allows for fast computation of α at detection time. To achieve this, α is split into $\alpha = \alpha_s - \alpha_m$ whereby α_m and α_s depend only on the model and scene pair respectively. Hence α_m can be precomputed for each pair in the model, and α_s computed once for each scene pair.

α_m is defined for a model surfel pair (m_r, m_i) as the angle between the vector $m_i' - m_r'$ and the upper-xy plane, with α_s defined analogously. Then the rotation around the x-axis is given by

$$\begin{aligned} R_X(\alpha) &= R_X(-\alpha_m) R_X(\alpha_s) \\ &= R_X(\alpha_s - \alpha_m) \end{aligned} \tag{3.6}$$

Algorithm 3 ComputeAngle

```
   Input: $m_r = (p_r^m, n_r^m, c_r^m), m_i = (p_i^m, n_i^m, c_i^m)$
1: $m_r' \leftarrow T_{m\to g}\, p_r^m$
2: $m_i' \leftarrow T_{m\to g}\, p_i^m$
3: $v \leftarrow m_i' - m_r'$
4: $\alpha \leftarrow \mathrm{atan2}(-v_z, v_y)$
5: **return** $\alpha$
```

3.1.3 Hough Voting Procedure

For a given reference surfel s_r sampled from the scene map, assumed to lie on the object, we seek the optimal object pose, which is parameterized by a model reference surfel and angle α, and which yields the best alignment of model and scene.

To determine the most supported m_r and α we employ a voting scheme similar to the Generalized Hough Transform, where, for each reference surfel in the scene, a voting space is constructed for the parameters of the object pose.

The voting procedure is carried out successively on multiple resolutions, in order be robust to the variation in precision of the RGB-D images with distance of the object from the sensor.

In each resolution the outline of the procedure is as follows:

1. Sample reference surfels from the scene and build surfel pairs,
2. Find matching surfel pair features in model and vote for m_r, α,
3. Find maxima in the voting space and compute the resulting poses.

Voting Accumulator

For s_r fixed, an accumulator array is defined with a row for each model reference surfel and columns given by the number of bins for α, giving a $n_m \times n_{angle}$ voting space. s_r is then paired with other surfels $s_i \in S$ in the scene, the corresponding surfel pair feature $F(s_r, s_i)$ computed according to eq. 3.1 and the hashing key $\mathcal{I}$ computed by eq. 3.3.

Matching and Voting

$\mathcal{I}$ can then be used to retrieve the list of model surfel pairs with similar features from $\mathcal{H}$ in constant time on average. For each matching pair (m_r, m_i), the angle α is computed, and a vote cast in the corresponding bin $(m_r, \hat{\alpha})$ in the voting array (see below).

Enhanced Voting Procedure

Since α typically falls between two bins in the voting array, we adopt a simple fractional binning in order to achieve increased robustness. This procedure is outlined in algorithm 4.

Algorithm 4 CastVote($A, m_r, \alpha, n_{angle}$)

```
1: $\hat{\alpha} = \alpha / n_{angle}$
2: $a_{low} = \lfloor \hat{\alpha} \rfloor$
3: $a_{high} = \lceil \hat{\alpha} \rceil$
4: $A[m_r, a_{low})] \leftarrow A[m_r, a_{low}] + (1 - (|\hat{\alpha} - a_{low}|))$
5: $A\left[m_r, a_{high}\right] \leftarrow A\left[m_r, a_{high}\right] + (1 - (|\hat{\alpha} - a_{high}|))$
```

Additionally, we also store the computed values of α during the voting process, and when computing the resulting poses, compute the median of the values of α which fell into that particular bin, in order to reduce inaccuracy due to discretization.

Pose Extraction

Once all pairs for s_r have been processed, the maximum is found in the voting space and the corresponding pose computed according to eq. 3.5, before sampling the next reference surfel. We also add poses with scores above a certain fraction γ_s of the maximum.

After all sampled reference surfels have been processed a final refinement step is used to identify the most supported poses, as described in the next section.

The complete voting procedure is outlined in algorithm 5. The parameter λ_s is the sampling rate for surfels in the scene, while γ_s is the fraction with which we accept non-maxima in the voting space. The remaining parameters are described in the next section. The function RandomSample returns a uniformly distributed sample from $\mathcal{S}$ without replacement.

3.1.4 Pose Refinement

Typically, the pose hypotheses found by the above procedure will contain a number of false positives. In order to remove isolated poses and consolidate the more likely candidates, we utilize a refinement process.

We first sort the extracted poses by number of votes, and perform an agglomerative clustering with a fixed threshold on translation δ_{trans} and orientation δ_{rot}, until all poses have been processed.

Algorithm 5 PoseVoting

```
    Input: Model Hash table $\mathcal{H}$, Scene map $s_s$
    Output: Found poses $\mathcal{P}'$
    Params: $d_{dist}, d_{angle}, d_{lum}, d_{chrom}, \lambda_s, \gamma_s, r_{start}, r_{end}, \delta_{trans}, \delta_{rot}, n_p$
    $\mathcal{P} = \emptyset$
 1: for $r = r_{start}$ to $r_{end}$ do
 2:     $\mathcal{S} \leftarrow$ GetSurfels($s_s$, $r$)
 3:     $A \leftarrow \mathbf{0}_{n_{m_r} \times n_{angle}}$
 4:     $n_{s_r} \leftarrow 0$
 5:     while $n_{s_r} < \lambda_s \cdot n_s$ do
 6:         $s_r \leftarrow$RandomSample( $\mathcal{S}$ )
 7:         $n_{s_r} \leftarrow n_{s_r} + 1$
 8:         for $s_i \in \mathcal{S}$ do
 9:             if $r \neq i$ then
10:                 $\mathbf{f} = F(s_r, s_i)$                                        ▷ (3.1)
11:                 $\mathcal{I} \leftarrow$ HashKey($\mathbf{f}, d_{dist}, d_{angle}, d_{lum}, d_{chrom}$)   ▷ (3.3)
12:                 $\alpha_s \leftarrow$ComputeAngle($s_r, s_i$)                        ▷ (3)
13:                 $\mathcal{Q} \leftarrow$ HashSearch($\mathcal{H}, r, \mathcal{I}$)
14:                 for $(m_r, \alpha_m) \in \mathcal{Q}$ do
15:                     $\alpha \leftarrow \alpha_s - \alpha_m$
16:                     CastVote($A, m_r, \alpha, n_{angle}$)                       ▷ (4)
17:                 end for
18:             end if
19:         end for
20:         $\mathcal{W} \leftarrow$ FindPeaks($A, \gamma_s$)
21:         for $(m_r, \alpha) \in \mathcal{W}$ do
22:             $T_{s \to m} \leftarrow$ ComputePose($T_{s \to g}, T_{m \to g}, \alpha$)          ▷ (3.5)
23:             $\mathcal{P} \leftarrow \mathcal{P} \cup \{T_{s \to m}\}$
24:         end for
25:     end while
26: end for
27: $\mathcal{P}' \leftarrow$PoseClustering($\mathcal{P}, \delta_{trans}, \delta_{rot}, n_p$)
28: return $\mathcal{P}'$
```

Finally we find the clusters with the highest scores (given by the sum of their poses) and return the mean pose for the top n_p clusters.

3.2 Pose Tracking with Improved Proposal Particle Filter

In this section we present our object tracking approach in detail. We track the 6-DoF pose of the camera with respect to a known object in a particle filter framework with improved proposal sampling. The choice of a particle filter for tracking is a good coupling for our detection approach, which yields multiple pose hypotheses, since the distribution over states is modelled in a non-parametric way. Hence it is possible to consider multiple correspondences, while convergence to a single hypothesis is made possible by integrating successive observations.

By modelling the state transition on the $SE(3)$ group, and employing a simple autoregressive state dynamics model to predict the motion of the camera in each time step, we achieve an informed state prediction.

An improved proposal distribution further improves the quality of samples which are then weighted according to the observation likelihood and pose uncertainty. The tracker is initialized from the detection procedure described in the previous section, which is also used to reinitialize when the object leaves then re-enters the field of view of the camera.

3.2.1 State Transition and Measurement Equations

We pose our problem at the estimation of the full 6-Dof configuration of the camera $x_t = (\mathbf{R}_t, \mathbf{t}_t) \in SE(3)$ at discrete time steps t. For convenience we refer to the pose as x_t, its homogeneous transformation matrix by $\mathbf{T}(x_t)$, with rotation $\mathbf{R}(x_t)$ and translation $\mathbf{t}(x_t)$.

We employ the first-order, discrete-time AR state dynamics described in Section 2.4.2.6 in order to propagate the particles in each time step:

$$\begin{aligned} g(x_{t-1}, \mathrm{d}\mathbf{W}_t) &= \mathbf{T}\left(x_{t-1}\right) \cdot \exp\left(\mathbf{A}_{t-1} \mathit{\Delta} t + \mathrm{d}\mathbf{W}_t \sqrt{\mathit{\Delta} t}\right) \\ \mathbf{A}_{t-1} &= \lambda_{ar} \frac{1}{\mathit{\Delta} t} \log\left(\mathbf{T}\left(x_{t-2}\right)^{-1} \mathbf{T}\left(x_{t-1}\right)\right) \end{aligned} \tag{3.7}$$

where $x_t \in SE(3)$ is the state estimate at time t, $\mathbf{A}_{t-1} \in \mathfrak{se}(3)$ is the velocity estimated in the previous time step (assumed to be part of the state), $\mathrm{d}\mathbf{W}_t$ is the Wiener process noise on $\mathfrak{se}(3)$, and exp and log are the exponential (eq. A.19) and logarithmic (eq. A.20) maps described in Section A.2.3, while

λ_{ar} is the AR process parameter. For $\mathrm{d}\mathbf{W}_t = \mathbf{0}$ this is the deterministic motion model, $g(x_{t-1})$.

Intuitively, the velocity is given estimated from the displacement of pose estimates between time steps. λ_{ar} determines to what extent this is incorporated when propagating the state in the sampling step.

The measurement equation is a non-linear function of the state assumed to be corrupted by Gaussian white noise.

$$z_t = h(x_t) + v_t, \quad v_t \sim \mathcal{N}(0, \Sigma_z) \tag{3.8}$$

3.2.2 Observation Representation

At each time step t the current observation z_t is an RGB-D Image, from which we build a scene map m_{s_t}. In contrast to the full-view object model m_m, the scene map represents the scene as observed from a single viewpoint. Hence, each node $n \in m_{s_t}$ has only a single meaningful surfel.

3.2.3 Improved Proposal Distribution

Although eq. 3.7 provides an improved motion estimate when compared to a random walk, when the likelihood is peaked then many particles are still required to sufficiently cover the areas of high likelihood, and hence a lot of computation time is wasted on particles which are assigned very low weights. This is illustrated in Figure 3.3, where the motion model provides a poor approximation of the target distribution.

Ideally one would like to choose the proposal distribution as $p(x_t \mid x_{t-1}^{(i)}, m_{s_t})$, in order to sample from the meaningful areas of the likelihood function, however in general this does not have a closed form and as such is difficult to sample.

3.2.4 Integrating the Latest Observation

Grisetti et al. (2007); Montemerlo et al. (2003) demonstrated how the latest observation could be integrated into the proposal distribution by an EKF-style measurement update or unscented transform to improve the quality of samples (see Section 2.4.2.7 for details).

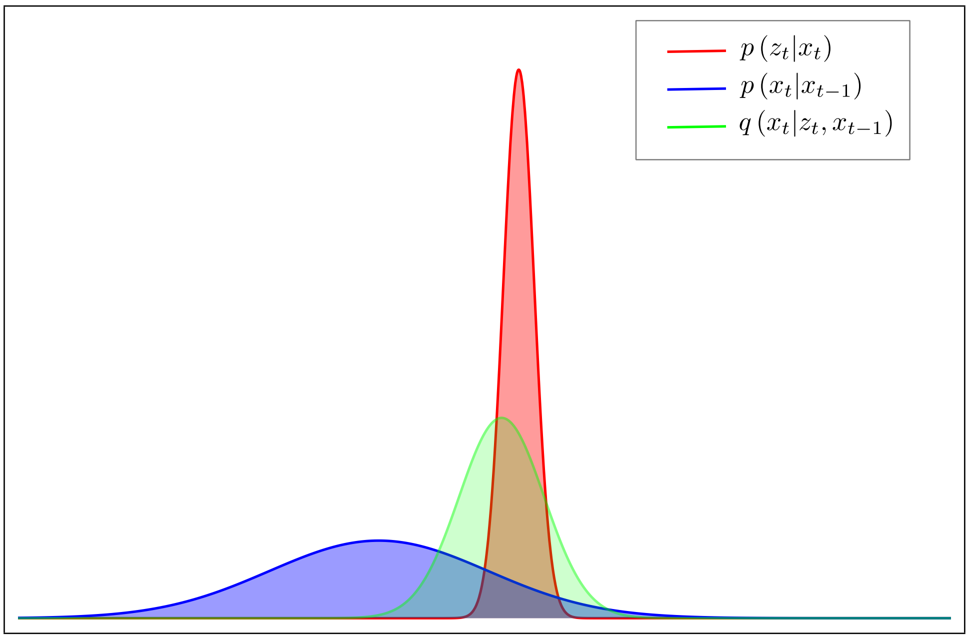


Figure 3.3: Illustration of the unimodal case. The state transition density (blue) often needs to model a high level of uncertainty in order to sample particles from the meaningful area of the observation likelihood function (red). By constructing a proposal distribution which also incorporates the latest observation (green), more efficient sampling can be achieved.

Our method can be seen as similar to those described, our aim is to locally approximate the posterior $p(x_t|m_m, x_{t-1}^{(i)}, m_{s_t})$ for each particle.

$$p(x_t|m_m, x_{t-1}^{(i)}, m_{s_t}) \overset{Bayes}{=} \frac{p(z_t|m_m, x_t)\, p(x_t|x_{t-1}^{(i)})}{p(z_t|m_m, x_{t-1}^{(i)})} \tag{3.9}$$

$$= \frac{p(z_t|m_m, x_t)\, p(x_t|x_{t-1}^{(i)})}{\int p(z_t|m_m, x')\, p(x'|x_{t-1}^{(i)})\, \mathrm{d}x'} \tag{3.10}$$

$$= \eta^{(i)}\, p(z_t|m_m, x_t)\, p(x_t|x_{t-1}^{(i)}) \tag{3.11}$$

Our approach aims to find the maximum of the likelihood function by a registration step. Hence, we register he scene map s_s towards the model map m_m in order to find $\hat{x}^{(i)\prime} = \operatorname{argmax} p(x_t|m_m, m_{s_t}, x_t^{(i)\prime})$ where $x_t^{(i)\prime}$ is the initial guess given by the propagated particle pose. We estimate the uncertainty in the resulting pose by computing an approximation to the observation covariance with respect to the pose.

The computation of the proposal proceeds as follows:

1. State propagation according to $x^{(i)\prime} = g(x_{t-1}^{(i)}, \mathbf{A}_{t-1}^{(i)})$
2. Establishment of associations $\mathcal{A}$ between scene m_s and model m_m
3. Pose optimization via efficient registration
4. Estimation of the resulting pose uncertainty

3.2.4.1 Surfel Association

The propagated pose $x^{(i)\prime}$ is used to establish correspondences between scene and model surfels as described in Section 2.2.4.1 whereby associations are sought on multiple resolutions starting with the finest available.

We iterate through the surfels s_i at the current resolution r, and attempt to establish a correspondence with a model surfel. The query surfel mean is transformed with the current pose estimate $\mathbf{T}(x^{(i)\prime})$ and an efficient volume query is performed in the model map to retrieve potential matches. The size of the search volume is set according to current resolution r. The set of candidate surfels is then:

$$\mathcal{Q} = \left\{ s_j \in m_m \,\middle|\, \left\| d\left(i, j; \mu_t^{(i)}\right) \right\|_2 \leq \gamma \right\} \tag{3.12}$$

where $d(i,j;x) = \mu_{m,j} - \mathbf{T}(x)\mu_{s,i}$ is the difference between surfel means, and $\gamma = 2 \cdot r$.

From the potential matching surfels $s \in \mathcal{Q}$, the best match is selected according to three criteria:

1. Distance: $\|d(i,j;x)\|_2$
2. View Direction: $d_{view}(i,j;x)$
3. Shape Texture Feature Similarity: $d_{feat.}(i,j)$

where $d_{view}(i,j;x) = v_{m,j} \cdot (\mathbf{R}(x)v_{s,i})$ is the difference between the model surfel viewing direction and the rotated scene surfel viewing direction, and $d_{feat.}(i,j)$ is the Euclidean distance between the feature descriptors. A limit of 0.1 is enforced on the feature similarity, and all three components are used to select the best match.

The association on each resolution is parallelized in order to improve performance on multi-core CPUs.

3.2.4.2 Efficient Registration

Given a set of surfel associations $\mathcal{A}$, and an initial pose x', we aim to find the pose $\hat{x}'$ which maximizes the likelihood $p(m_s \mid x, m_m)$ of the latest observation.

The observation likelihood is given, as described in Section 2.2.4.2 by:

$$p\left(m_s \mid x, m_m\right) = \prod_{(i,j)\in\mathcal{A}} p\left(s_{s,i} \mid x, s_{m,j}\right) \tag{3.13}$$

In registering the scene map m_s towards the model map m_m we minimize the log likelihood:

$$L\left(x\right) = \sum_{(i,j)\in\mathcal{A}} \log\left(\left|\Sigma_{i,j}(x)\right|\right) + d(i,j;x)^T \Sigma_{i,j}^{-1}(x) d(i,j;x) \tag{3.14}$$

The registration proceeds using the Levenberg Marquadt (LM) method as described in Section 2.2.4.3 to perform damped Gauss Newton optimization. LM is chosen since it is efficient to compute, requiring only first-order derivatives. Furthermore, this method is known to be highly convergent, and to converge rapidly near the optimum.

The aim is to iteratively adjust the parameters of x to find the pose which maximizes the observation likelihood (hence minimizing the sum of weighted squared errors):

$$\hat{x}' = \text{argmin}_x \, \mathbf{e}^T\left(x\right) \mathbf{W} \, \mathbf{e}\left(x\right) \tag{3.15}$$

where, $\mathbf{e}\left(x\right)$ is a vector of residuals, and $\mathbf{W}$ is a weighting matrix.

3.2.4.3 Covariance Estimation and Importance Sampling

In order to construct a Gaussian proposal distribution, we estimate the uncertainty in the optimized pose, by the observation covariance. A closed-form approximation of the observation covariance has been proposed (Censi, 2007; Stückler and Behnke, 2013):

$$\Sigma(x) \approx \left(\frac{\partial^2 L}{\partial x^2}\right)^{-1} \frac{\partial^2 L}{\partial s \partial x} \Sigma(s) \frac{\partial^2 L}{\partial x^2}^T \left(\frac{\partial^2 L}{\partial x^2}\right)^{-1} \tag{3.16}$$

where x is the pose estimate, s denotes the associated surfels from scene and model maps, and $\Sigma(s)$ is the surfels' covariance.

We compute an efficient estimation of the covariance by approximating the Hessian matrix according the Levenberg Marquadt update $\left(\frac{\partial^2 L}{\partial x^2}\right) \approx J^T W J$, yielding:

$$\Sigma(x) \approx \left(J^T W J\right)^{-1} \frac{\partial^2 L}{\partial s \partial x} \Sigma(s) \frac{\partial^2 L}{\partial x^2}^T \left(J^T W J\right)^{-1} \tag{3.17}$$

SAMPLING FROM THE IMPROVED PROPOSAL

Using the method described above, we can estimate the parameters for a Normal distribution from which poses can be sampled. The sampling distribution is described by the mean $\mu_x = [t_x, t_y . t_z, q_x, q_y, q_z, q_w]^T$, and the covariance of the first 6 parameters, where $\mathbf{t} \in \mathbb{R}^3$ and $\mathbf{q} \in \mathbb{H}$ denote the translation and rotation (as a unit quaternion) respectively.

For a multivariate normal distribution $\mathcal{N}(\mu; \mathbf{\Sigma})$, a common way to draw samples $\mathbf{x} \sim \mathcal{N}(\mu; \mathbf{\Sigma})$ is as follows:

1. Find $\mathbf{A}\mathbf{A}^T = \mathbf{\Sigma}$ (e.g. by Cholesky decomposition).
2. Sample $\mathbf{z} = \{z_1, \ldots, z_n\}, z_i \sim \mathcal{N}(0; 1)$
3. $\mathbf{x}' = [t_x, t_y, t_z, q_x, q_y, q_z]^T = \bar{\mu}_x + \mathbf{A}\,\mathbf{z}$

where $\bar{\mu}_x$ refers to the first 6 parameters of the mean. The real part of the unit quaternion $\mathbf{q}_{x'}$ is then recovered by its normalization constraint, and its sign from the mean. Hence it is trivial to obtain the full rigid body transformation of $\mathbf{x}' = (R, t)$.

3.2.5 Importance Weights

For the importance weights we follow a similar approach to Montemerlo et al. (2003).

Hence for particle k, the maximum likelihood data association is established:

$$\mathcal{A}_t^{(k)} = \text{argmax}_{\mathcal{A}} p(m_{s_t} | \mathcal{A}, x_t^{(k)}) \tag{3.18}$$

and we compute the weight by exponentiating the log likelihood given by:

$$L(x) = \sum_{a \in \mathcal{A}_t^{(k)}} \log(|\Sigma_a(x)|) + d(a;x)^T \Sigma_a^{-1}(x) d(a;x) \tag{3.19}$$

$$\Sigma_a(x) = \Sigma_{m,j} + R(x) \Sigma_{s,i} R(x)^T + J_a \Sigma_x J_a^T \tag{3.20}$$

for associated surfels $a = (i, j)$ where $J_a = \frac{\partial T}{\partial x}(x) \mu_{s,a}$ and Σ_x is the pose covariance.

Hence the weights incorporate the observation likelihood and the uncertainty of the pose. After the importance weights are updated, we resample particles with probability proportional to the weights.

3.2.5.1 Efficient Computation of the Proposal Distribution

In practice, in order to achieve tracking at high frame rates, an optimization is required in order to efficiently compute the proposal distribution. We propose a simple and robust method whereby we do not construct the proposal for each particle. Rather, we identify the modes of the density $p(x_t \mid x_{t-1})$ and generate the proposal in the form of a Gaussian mixture distribution, hence for each particle a sample is drawn from the relevant mixture component.

In order to identify the modes of the transition density we employ a clustering of the particles with a fixed threshold on translation and rotation. For efficient clustering, a kd-tree is constructed from the position estimates of the particles. Particles in a limited volume and with similar orientations are then clustered together until all particles have been assigned.

We further note that when the estimate of the tracker is good, the discrete distribution given by the particles typically has a single mode. However, after initialization or when the uncertainy increases, considering multiple modes can aid robustness.

3.3 Integrating Pose Estimation and Tracking

The pose estimation and tracking methods are integrated in a single framework for tracking without prior knowledge of the initial pose.

Initialization

In the initialization step, n_p pose hypotheses are extracted by constructing a scene map s_s from the first RGB-D image z_1, from which surfel pair features are computed and matched with those precomputed for the model map m_m, and poses estimated in the voting framework described in Section 3.1

Tracking

Particles are then sampled from the extracted pose estimates in order to initialize the particle filter. Once initialized tracking proceeds as described in the previous section. That is, the particles are propagated according the state transition model, clustered into distinct hypotheses, and for each cluster the improved proposal distribution is computed. The improved proposal is then sampled and the sampled particles assigned importance weights.

Once tracking is stable, we can process the observations more efficiently by focusing on a volume of interest close to the last pose estimation. Hence we segment away parts of the scene which are well beyond the spatial distribution of the model (which is extracted from the root node of the map before tracking) under the current estimate.

Reinitialization

Registration failures are handled by falling back to sampling from the state transition density. However, often such failures are an indication of a poor pose estimate. Where multiple clusters are present, misaligned poses receive a low matching likelihood and are resampled, hence the overall quality of the estimate is not affected.

In some circumstances the object may leave the field of view or be highly occluded. In this case it is not possible to establish a sufficient number of surfel associations for robust registration. A number of techniques have been proposed for detecting degeneracy in particle filter methods. A simple scheme would be to detect drifting by a threshold on the displacement between frames.

The Average Likelihood Ratio is a measure of how the average likelihood of the particles is evolving. Hence low values can point to dropping likelihood and degeneracy of the estimate. Similarly, the number of effective particles ($Neff$) statistic, traditionally used for triggering resampling, has been proposed as a measure of the quality of the estimate in Choi and Christensen (2011), for particle filters where resampling is carried out in each step.

When sampling from an improved proposal distribution, particles often get assigned similar weights. In this case the above schemes are not always applicable. Instead we propose a simple scheme for detecting loss of tracking and invoking reinitialization. We monitor the maximum number of associations established for all hypotheses. If this number drops below a threshold it is an indication that the estimate is poor, and hence we attempt to reinitialize.

4 Evaluation

In this chapter we present the experiments and results of our object pose estimation and tracking method. We evaluate the approach against a recently published public dataset as well as on additional sequences captured for this thesis.

4.1 Datasets

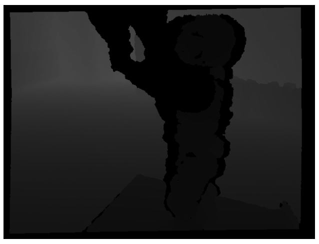
We evaluate our method against the RGB-D Object Tracking Dataset (Stückler and Behnke, 2012).

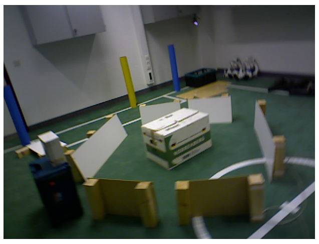
This dataset consists of nine sequences each with around 1100 frames, featuring three objects of different characteristics and challenging aspects such as motion blur and partial occlusions.

We use these datasets to evaluate our object detection approach on individual frames, as well as the combined detection and tracking method on the full sequences.

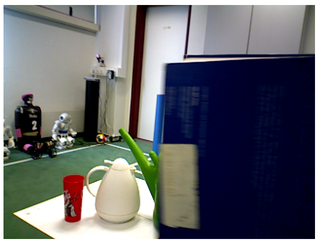
Four additional sequences were captured for this thesis in order to further evaluate performance. These sequences in particular feature smaller objects than those present in the above dataset, in order to assess how adaptable the method is, while added clutter, partial and total occlusions of the object pose a challenge for robust detection and tracking.

Figure 4.1 gives an impression of the sequences, and the challenges they pose. Missing depth results in fewer surfels in the scene map, while the depth precision of RGB-D is lower at larger distances to the object. Fast motions and blur are a challenge to tracking, while robustness against occlusions and clutter is required.


(a) Missing Depth


(b) Distance to object

(c) Fast Motions and Blurring


(d) Occlusions

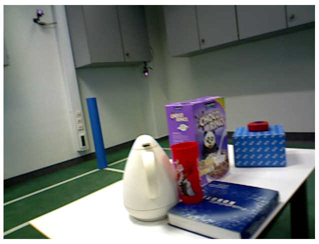

(e) Clutter

Figure 4.1: Challenging situations: Example frames from sequences used in the experiments

4.1.1 Object Model Learning and Ground Truth Capture

Object models are learned using the MRSMap framework. This library provides tools for building object models by fusing views from a training sequence where the camera is moved around the object.

We capture ground truth for the camera pose by attaching reflective markers to the camera and tracking its configuration using a 12 Camera OptiTrack Motion Capture system.

All datasets were recorded using an Asus Xtion Pro Live camera in a resolution of 640x480 and a frame rate of 30Hz,

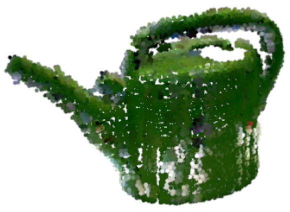
4.2 Testing Environment

All tests were carried out on an Intel Core i7-940 (4 cores + 4 virtualized) with 12GB RAM.

(a) Humanoid

(b) Box

(c) Chair
q


(d) Watering Can

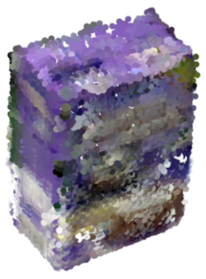

(e) Cereal Box

Figure 4.2: Learned Object Models visualized by samples from the surfels at 2.5cm resolution.

4.3 Overview of Experiments

In this section we give an overview of the experiments carried out for the purpose of evaluating our method.

4.3.1 Detection and Pose Estimation

We assess the performance of the object pose estimation component against individual frames from the available sequences.

Since the aim of detection is to estimate the initial pose for tracking, we evaluate the extent to which the initialization allows the tracker to converge to the correct pose. To this end we plot the evolution of precision and recall over subsets of frames when the tracker is initialized by the detection method.

- The sequence is split into batches of 11 frames.

- In the first frame the detection is run and the recovered poses used to initialize the tracker
- For the remaining frames we update the tracker and extract the particle clusters

In each frame, we record the number of true positives, false positives and false negatives, which are summed for corresponding frame indices $\{1,\ldots,11\}$ over the whole sequence. We count a true positive if the pose of camera is among the hypotheses. We allow a small variation from the true camera pose of 10cm in translation and 15 degrees in rotation. Note, that poses which align the object well within the scene but do not meet this criteria do not count as true detections.

4.3.1.1 Performance Measure

We examine the detection performance in terms of *Precision* and *Recall* (Wikipedia, 2013).

$$\text{Precision} = \frac{tp}{tp + fp} \tag{4.1}$$

$$\text{Recall} = \frac{tp}{tp + fn} \tag{4.2}$$

where tp, fp and fn are the number of true positives, false positives and false negatives respectively.

4.3.1.2 Testing Setup

We evaluate our detection and pose estimation method as described above on all datasets introduced in Section 4.1.

We used the following setup for our experiments:

The minimum resolution was set according to object size, hence for the Humanoid, Box, and Chair sequences, the slightly coarser resolution of 2.5cm was used for performance reasons, while for the Watering Can and Cereal Box sequences, 1.25cm was chosen to provide greater accuracy for the smaller objects.

Parameter	**Value**
Maximum Pose hypotheses	5
Minimum Resolution	1.25cm / 2.5cm
Feature angle (d_{angle})	10 degrees
Feature distance (d_{dist})	5cm
Feature lum. (n_{lum})	3
Feature chrom. (n_{chrom})	3
Peak accept level	0.7

For statistically significant results, all experiments were run 10 times and the results aggregated.

4.3.2 Pose Tracking

We also evaluate the performance of the tracker against the entire sequences in order to assess its robustness.

4.3.2.1 Performance Measure

In order to assess the tracking performance of our approach, we examine the Absolute Trajectory Error (ATE) metric. This metric measures the difference between points on the estimated and ground truth trajectories.

We employ the evaluation tool of Sturm et al. (2012) in order to compute the error with respect to the recorded ground truth trajectory.

ATE is given by the Root Mean Square Error (RMSE) in translation between estimated and true poses:

$$RMSE\left(\mathbf{P}_{1:n}\right) = \left(\frac{1}{n}\sum_{i=1}^{n}\left\|\text{trans}\left(\mathbf{Q}_i^{-1}\mathbf{P}_i\right)\right\|^2\right)^{1/2} \tag{4.3}$$

where $\mathbf{Q}_i$ and $\mathbf{P}_i$ are the i-th true and estimated poses respectively, and trans denotes the translational part of a homogeneous transformation matrix.

Since 4.3 can be affected by outliers, in order to assess overall tracking performance, it is also of interest to consider the median ATE:

$$ATE_{med}\left(\mathbf{P}_{1:n}\right) = \text{median}\left(\left\{\left\|\text{trans}\left(\mathbf{Q}_1^{-1}\mathbf{P}_1\right)\right\|_2, \ldots, \left\|\text{trans}\left(\mathbf{Q}_n^{-1}\mathbf{P}_n\right)\right\|_2\right\}\right) \tag{4.4}$$

4.3.2.2 Testing Setup

We evaluate our tracking method against all datasets introduced in Section 4.1. The detection procedure is run in the first frame of each sequence and the tracker initialized with the returned pose hypotheses. In the event that the track is lost, the tracker can reinitialize itself.

We carried out experiments in two tracking “modes”. For the first set of experiments, we process all frames in each dataset, while in second set of tests, we enable real-time tracking whereby frames are skipped.

We used the following setup for our experiments:

Parameter	**All frames**	**Real-time**
Maximum initial hypotheses	5	5
Minimum Resolution	1.25cm / 2.5cm	1.25cm / 2.5cm
Number of particles	25	25
Maximum LM Iterations	10	20

The minimum resolution was set according to object size, hence for the Humanoid, Box, and Chair sequences, the slightly coarser resolution of 2.5cm was used for performance reasons, while for the Watering Can and Cereal Box sequences, 1.25cm was chosen to provide greater accuracy for the smaller objects.

For statistically significant results, all experiments were run 10 times and the results aggregated.

4.4 Results

In this section we present the results, selected plots and a discussion of the experiments carried out for detection and tracking.

4.4.1 Object Detection and Pose Estimation

4.4.1.1 RGB-D Object Tracking Dataset

The plots of precision and recall against frame indices are seen in Figures 4.3 (humanoid sequences), 4.4 (box sequences) and 4.5 (chair sequences). Average frame times for detection and pose estimation on the humanoid, chair and box sequences are given in Table 4.1.

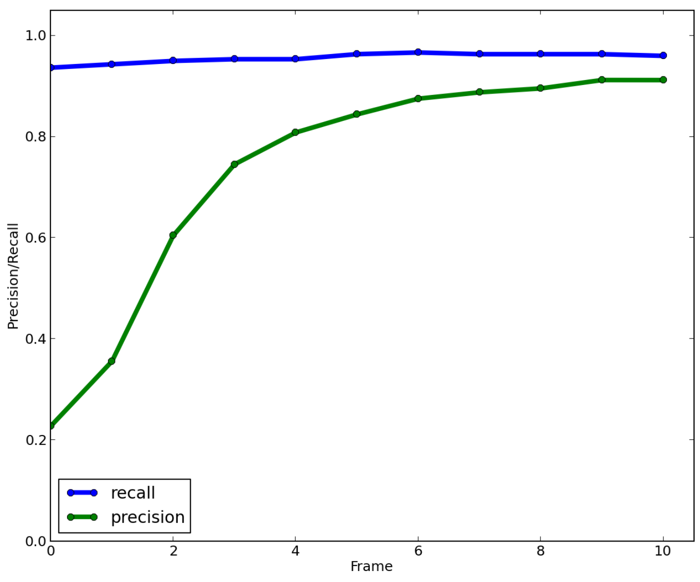


Figure 4.3: Evolution of Precision and Recall over sets of 11 frames for the humanoid sequences

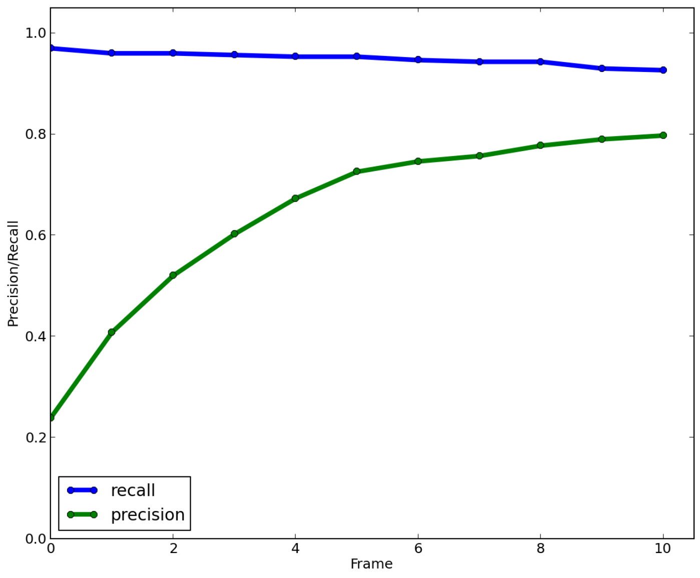


Figure 4.4: Evolution of Precision and Recall over sets of 11 frames for the box sequences

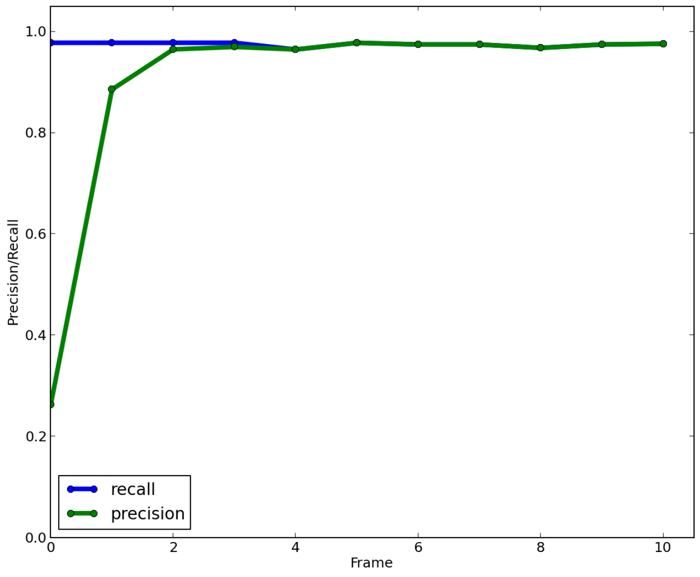


Figure 4.5: Evolution of Precision and Recall over sets of 11 frames for the chair sequences

object	Time
Humanoid	0.750 ± 0.226
Box	3.43 ± 0.764
Chair	1.20 ± 0.416

Table 4.1: Mean frame processing time ± std. deviation (in seconds) for the humanoid, box and chair sequences

4.4.1.2 Additional Sequences

Plots of precision and recall against frame index are shown in Figures 4.6 (watering can), and 4.7 (cereal box). Average frame times for detection and pose estimation for these sequences are given in Table 4.2.

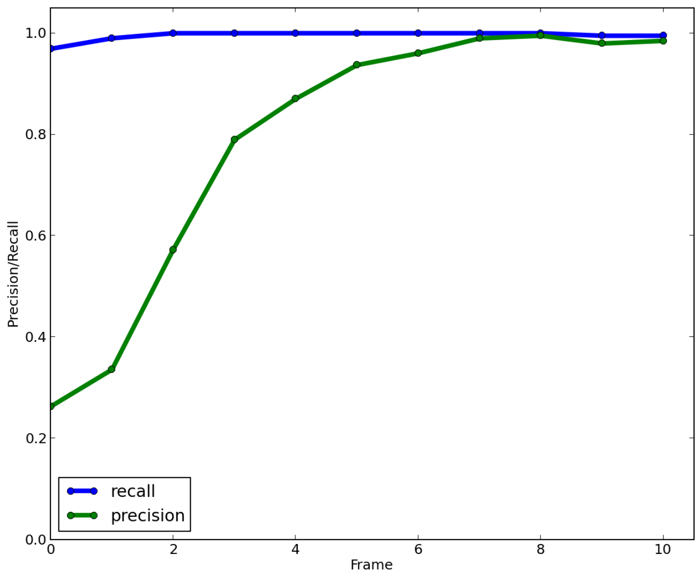


Figure 4.6: Evolution of Precision and Recall over 11 frame sequences for the watering can sequences

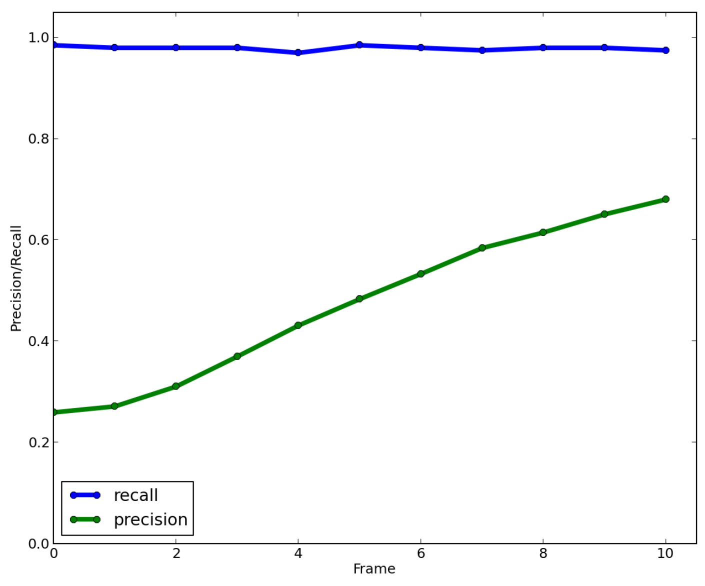


Figure 4.7: Evolution of Precision and Recall over 11 frame sequences for the cereal box sequences

object	**time**
Watering Can	1.05 ± 0.35
Box	0.68 ± 0.22

Table 4.2: Mean frame processing time ± std. deviation (in seconds) for the watering can and cereal box sequences

4.4.1.3 Discussion

As can be seen from the plots, our method achieves high detection rates of ~95-100% on average. However, the high detection rate is coupled with false detections, hence the precision for the detection frame is relatively low.

False detections arise from other parts of the scene, or object itself which are locally similar in shape or texture and result in incorrect associations.

Our choice of the particle filter is clearly justified since, after initialization with the pose hypotheses, we can see that typically the belief converges on the correct pose within just a few frames.

The rate of convergence can be seen to be highly dependent on the object characteristics. The chair sequences (Figure 4.5) show the most rapid rate of convergence, typically within 1 or 2 frames, likely owing to its distinctive shape and texture.

The humanoid (Figure 4.3) and watering can (Figure 4.6) sequences, also see convergence to the correct pose within just a few frames. While the box (Figure 4.4), and cereal box (Figure 4.7) converge slowly (but surely) to the correct hypothesis.

The rate of convergence seems to be highly correlated with the object's degree of symmetry or self-similarity. Competing hypotheses which have a large overlap receive similar weights and hence are hard to differentiate.

The detection time is also related to the object characteristics, despite the finer resolution, the cereal box and watering can (Table 4.2) have fewer surfels and hence can be detected more quickly (typically within 1 second). As can be seen in Table 4.1, the box has the highest detection time, owing to the fact that many surfel pairs lie on planar regions, resulting in many features falling into the same bin, which slows down the voting phase. Meanwhile, the humanoid and chair can be detected in half the time or less, since their more distinctive shape and texture result in fewer features in the same bin.

4.4.2 Object Tracking

In this section we present the results of our tracking experiments. We evaluated against 13 sequences in total, and each experiment was run 10 times and the results aggregated.

4.4.2.1 RGB-D Object Tracking Dataset

We present the quantitative tracking results in Tables 4.3 (humanoid sequences), 4.4 (box sequences), and 4.5 (chair sequences).

sequence	**all frames**		**real-time**	
	ATE (m)	time (ms)	ATE (m)	frames used (%)
slow	0.0226	37.5 ± 4.0	0.0236	59.5
medium	0.0265	38.2 ± 4.6	0.0264	62.9
fast	0.0334	39.4 ± 4.6	0.0334	60.5

Table 4.3: Tracking results for the humanoid sequences. Median Absolute Trajectory Error, mean frame time ± std. deviation and percentage frames used for real-time mode

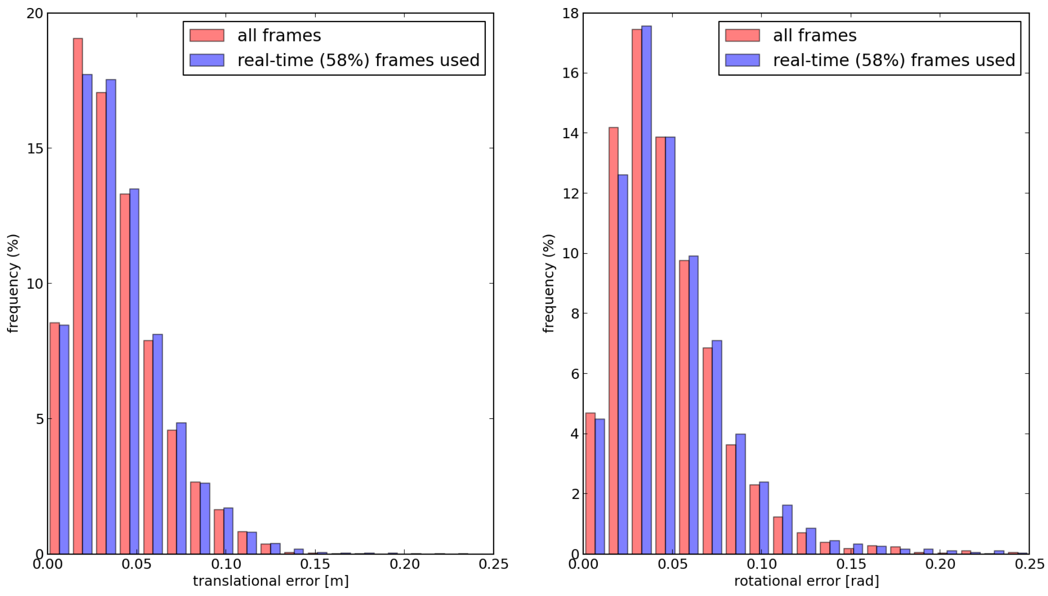


Figure 4.8: Histogram illustrating the distribution of relative translational and rotational errors for the humanoid sequences

sequence	**all frames**		**real-time**	
	ATE (m)	time (ms)	ATE (m)	frames used (%)
slow	0.0234	73.7 ± 9.0	0.0235	40.5
medium	0.0247	68.2 ± 18.3	0.0293	48.3
fast	0.0214	66.3 ± 12.5	0.0223	45.0

Table 4.4: Tracking results for the box sequences. Median Absolute Trajectory Error, mean frame time ± std. deviation and percentage frames used for real-time mode

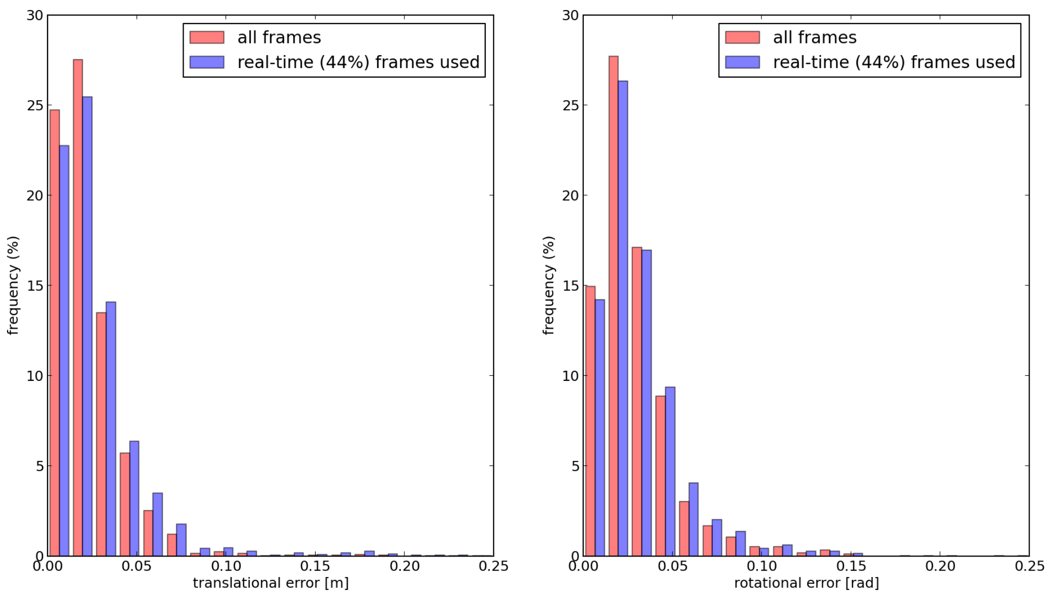


Figure 4.9: Histogram illustrating the distribution of relative translational and rotational errors for the box sequences

sequence	all frames		real-time	
	ATE (m)	time (ms)	ATE (m)	frames used (%)
slow	0.0230	95.8 ± 17.9	0.0293	29.8
medium	0.0158	103.9 ± 13.2	0.0173	29.9
fast	0.0359	93.7 ± 18.4	0.0406	33.4

Table 4.5: Tracking results for the chair sequences. Median Absolute Trajectory Error, mean frame time ± std. deviation and percentage frames used for real-time mode

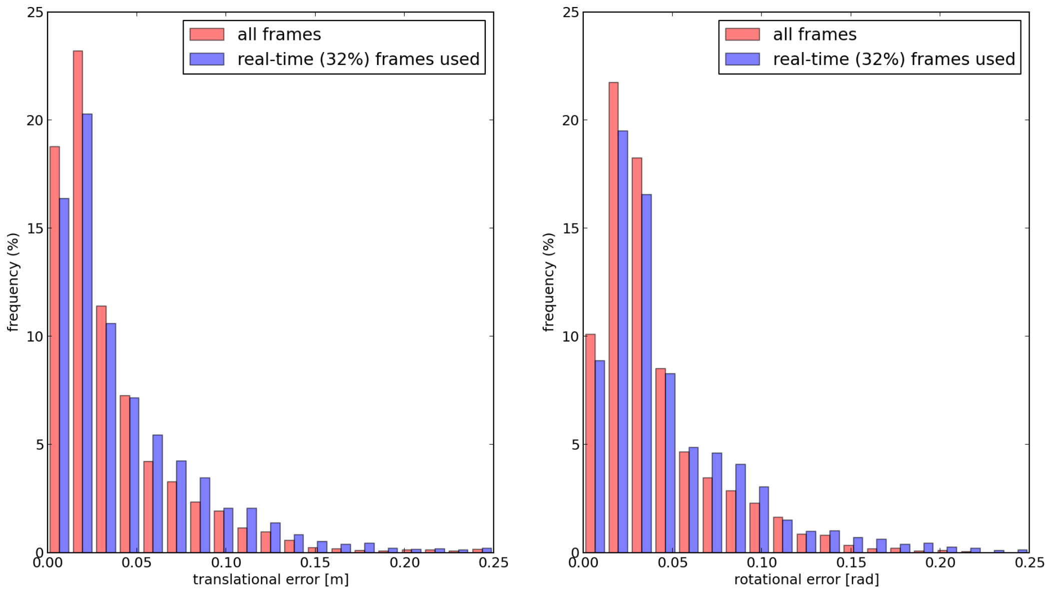


Figure 4.10: Histogram illustrating the distribution of relative translational and rotational errors for the chair sequences

4.4.2.2 Additional Sequences

We present the quantitative tracking results in Tables 4.6 (watering can sequences), and 4.7 (cereal box sequences).

sequence	**all frames**		**real-time**	
	ATE (m)	time (ms)	ATE (m)	frames used (%)
1	0.0247	40.7 ± 7.4	0.0263	56.8
2	0.0221	43.8 ± 9.5	0.0259	59.8

Table 4.6: Tracking results for the watering can sequences. Median Absolute Trajectory Error, mean frame time ± std. deviation and percentage frames used for real-time mode

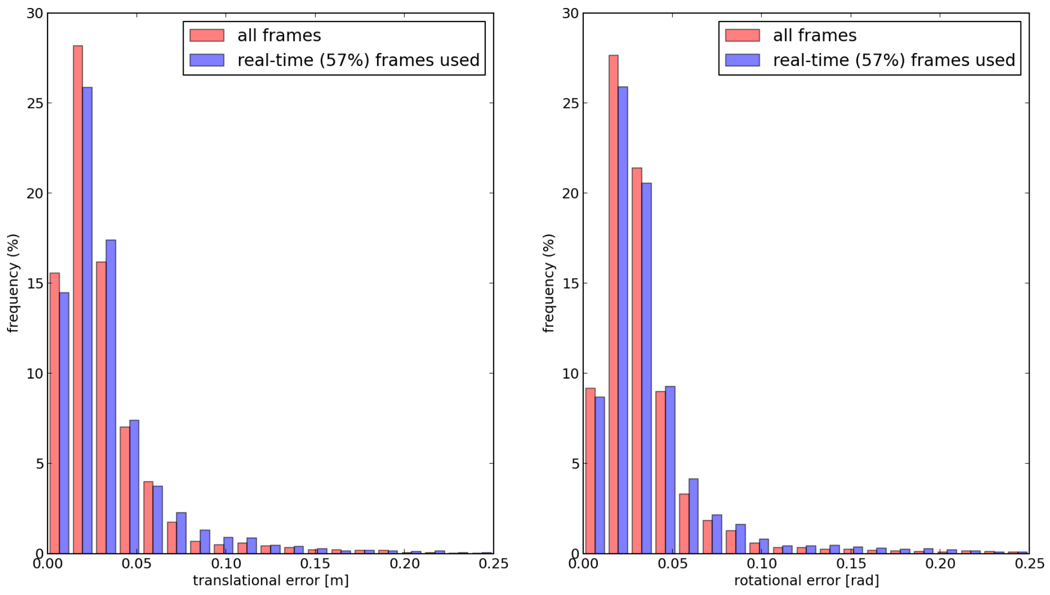


Figure 4.11: Histogram illustrating the distribution of relative translational and rotational errors for the watering can sequences

sequence	**all frames**		**real-time**	
	ATE (m)	time (ms)	ATE (m)	frames used (%)
1	0.0225	36.2 ± 10.6	0.0277	70.9
2	0.0183	36.4 ± 10.0	0.0188	75.2

Table 4.7: Tracking results for the cereal box sequences. Median Absolute Trajectory Error, mean frame time ± std. deviation and percentage frames used for real-time mode

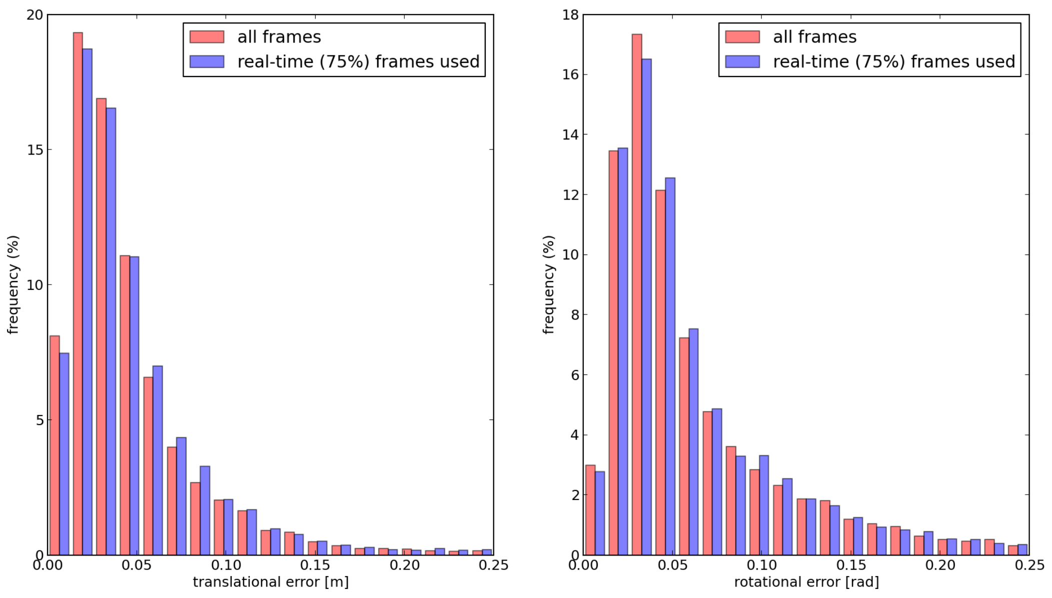


Figure 4.12: Histogram illustrating the distribution of relative translational and rotational errors for the cereal box sequences

4.4.2.3 Discussion

As can be seen from the results and plots, our method robustly tracks the pose of the camera with respect to different objects and under varying conditions.

Overall

The experiments carried out on all frames of the sequences show typical Median Absolute Trajectory Error of approximately 15-35mm, while for the real-time tracking experiments we observed ATEs of ~18-40mm.

RGB-D Object Tracking Dataset

For the first dataset, the best performance was on the box sequences (Table 4.4), where the median ATE was less than 25mm for all sequences. Furthermore, it can be seen in Figure 4.9 that the vast majority of relative pose errors were less than 50mm, and 0.05rad (< 3 degrees) respectively. Despite the fact that less than half of the frames were used for real-time tracking, the performance was not significantly affected, the largest drop (~5mm) being seen in the medium sequence Figure 4.13 shows just how well our tracker estimates the ground truth trajectory on the fast sequence (all frames), with only a small error in y-direction between frames 300-500.

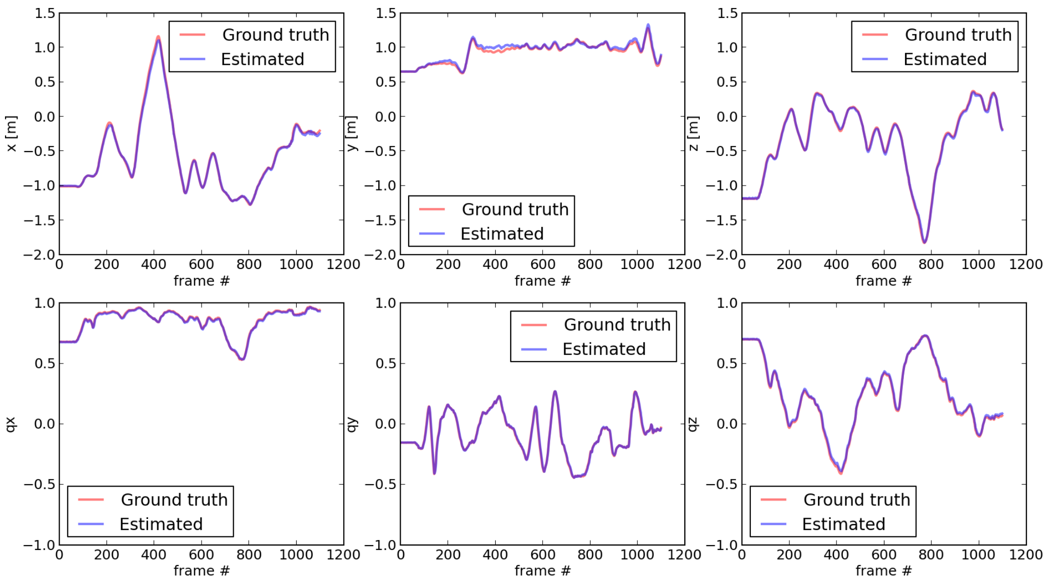


Figure 4.13: Ground truth and estimated pose trajectory for the box fast sequence (all frames)

Experiments on the humanoid sequences (Table 4.3) also showed good performance, with only the fast sequence causing our tracker to drift slightly, however tracking was maintained throughout, while real-time tests saw barely any deterioration in performance (Figure 4.8). In Figure 4.14 we show the model rendered at the tracked pose in three frames from the medium sequence. The track is maintained accurately from different viewpoints and under fast motions of the camera.

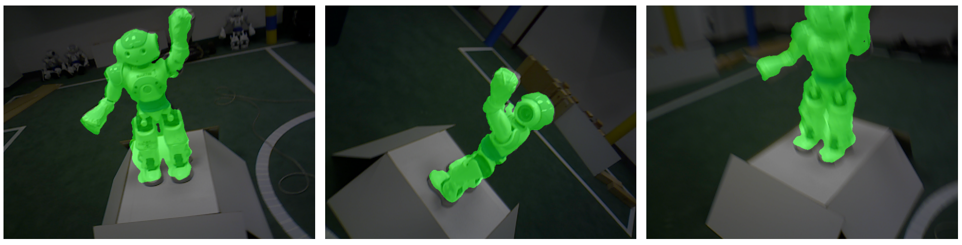


Figure 4.14: Robust tracking: Our tracker handles the varying appearance of the object with ease (left and centre), and copes well with motion blurring (right)

More mixed results were observed in the chair sequences (Table 4.5), where both the lowest and highest median ATE were recorded. Real-time performance in particular suffered on the fast dataset, possibly due to the relatively high percentage of dropped frames, and the fast motions also proved a challenge when tracking all frames. However, as can be seen from the plots (Figure 4.10), accurate tracking was maintained for the vast majority of frames, and any misalignment could be resolved.

Additional Sequences

Tracking performance on the watering can sequences was particularly good (Table 4.6), with median ATE less than 25mm for both sequences on all frames, while real-time tests only saw a small deterioration in performance (~2mm), as can also be seen in Figure 4.11.

In order to test reinitialization, in sequence 2, the object is briefly occluded with a book. Figure 4.15 shows how the track is maintained until the object is highly occluded, before reinitialization when the track is lost.

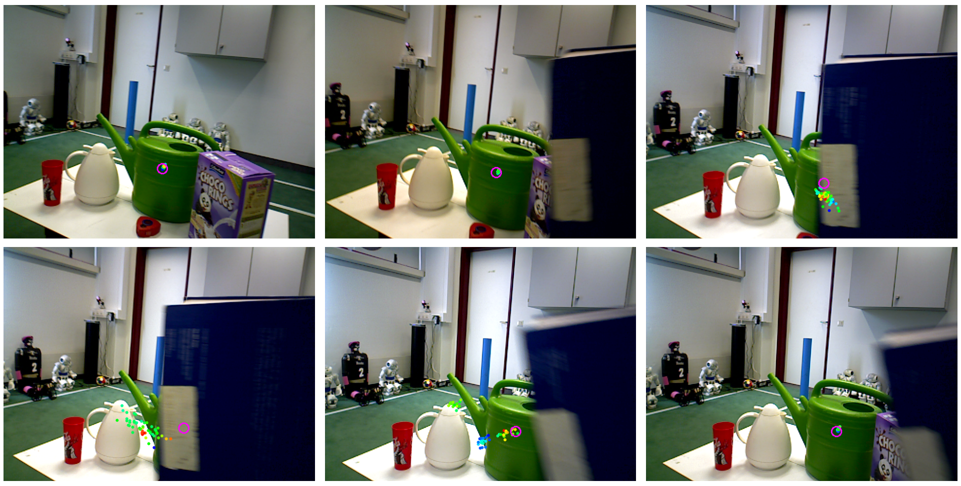


Figure 4.15: Reinitialization: While tracking (top left), the target is covered with a book (top centre). Tracking is maintained with over 50% occlusion (top right). Bottom left: The track is lost with the object almost totally occluded, and reinitialization occurs (bottom centre). Bottom right: the track is recovered from the new hypotheses. [The ground truth is shown as a pink circle, while the particles are shown as dots coloured according to weight (red for high, blue for low)]

The cereal box (4.7), being the smallest object in our tests, presented a minor problem for our tracker.

Performance on the first sequence was steady, however, although the median ATE is low for both sequences, it can be seen in Figure 4.12 that in particular the orientation estimate was much less accurate than on other sequences.

As can be seen in Figure 4.16, the orientation estimate is particularly affected between frames 100-200 and 800-900. Figure 4.17 further illustrates this behaviour.

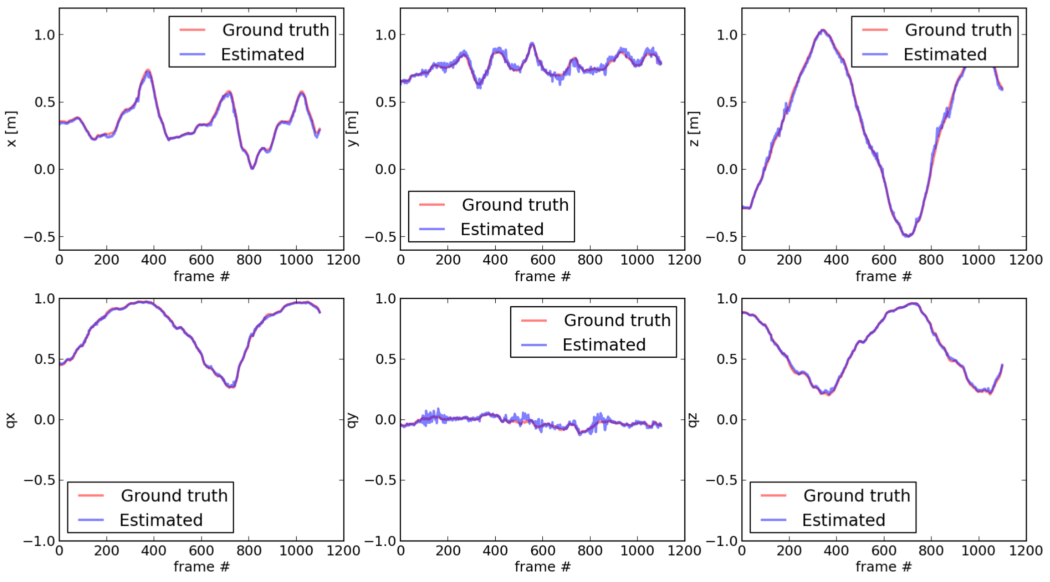


Figure 4.16: Ground truth and estimated pose trajectory for the cereal box sequence 2

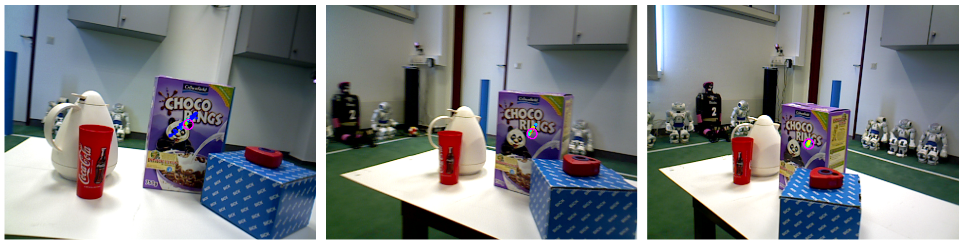


Figure 4.17: Orientation uncertainty: With only one side of the box visible to the camera, the uncertainty in camera orientation grows (left), when more of the object comes into view, the estimate improves (centre). The estimate remains good despite partial occlusion of the object (right).

We noticed this tends to occur when only one side of the box is visible, which seems to introduce error into the pose estimate. However, although the track is somewhat unstable, when more of the object comes into view, accurate tracking is possible again.

4.5 Summary

In this chapter we presented the results of experiments carried out on our detection and tracking methods.

In presenting the performance of our detection and pose estimation method in Section 4.4.1, we showed good performance with high detection rates for all objects and illustrated the ability of our method to converge to the correct pose hypothesis.

In Section 4.4.2 we presented detailed results of our tracking experiments. The results show good performance on all objects, with Median Absolute Trajectory Error typically below 3cm. Our tracker proved robust against fast motions and occlusions and was able to reinitialize when the object left then re-entered the field of view.

We observed a problem with our method when tracking along one side of planar objects, however the robustness of our method allowed tracking to be maintained despite degradation in the quality of the estimate.

5 | Conclusions and Future Work

In this thesis we presented a novel approach for model-based estimation and tracking of the 6-DoF pose of 3D objects in RGB-D image sequences. Unlike many object tracking methods, the tracker does not assume an initial pose constraint but rather estimates the initial alignment in a voting framework.

We employed Multi-Resolution Surfel Maps as a concise model of object shape and texture, and by extending the representation with Colour Surfel pair features, reliably detect the object by finding consistent arrangements of surfel pair features in scene and model, and estimate its pose in a voting framework similar to the Generalized Hough Transform.

Our tracking method employs a particle filter which integrates the latest observation into the proposal distribution in order to sample particles from regions of high likelihood. We compute the proposal distribution using an efficient registration method to optimize the initial pose, enabling particles to be sampled more effectively.

We demonstrated high detection rates and accurate pose estimation of objects with a range of different visual and geometric characteristics. Furthermore, our method has been seen to be capable of robustly tracking the 6-DoF pose of the camera under a wide range of motions and occlusion. We integrated the pose estimation and tracking methods to allow tracking without prior knowledge of the initial pose, and enabling reinitialization when the track is lost.

5.1 Future Work

There are a number of possible directions for future work related to this thesis. One such avenue would be to consider incorporating object boundary information into the detection and tracking pipeline, which could further improve performance on textureless objects and help reduce uncertainty in orientation as observed in the experiments on the cereal box sequences.

Another option for future work would be to explore an implementation which leverages the parallel processing capabilities of GPUs, which could likely give a performance boost given the choice of a particle filter for tracking.

Appendices

A Fundamentals

A.1 Non-Linear Least Squares Optimization

In parameter optimization we seek to minimize the sum of (weighted) squared errors between a set of measured values $\mathbf{y} = \{y_1, \ldots, y_m\}$ and a (non-linear) function $f_i(\mathbf{x})$ parameterized by $\mathbf{x}$. The errors (or residuals) between fitted and measured values are of the form:

$$e_i(\mathbf{x}) = y_i - f_i(\mathbf{x}) \tag{A.1}$$

Since we wish to find the parameters of $\mathbf{x}$ which best fit to the observed data, we aim to minimize the chi-squared (χ^2) criterion, or goodness of fit measure:

$$\begin{aligned}\chi^2(\mathbf{x}) &= \frac{1}{2}\sum_{i=1}^{m}\left(\frac{e_i(\mathbf{x})}{w_i}\right)^2 \\ &= \frac{1}{2}\mathbf{e}(\mathbf{x})^T\,\mathbf{W}\,\mathbf{e}(\mathbf{x})\end{aligned} \tag{A.2}$$

where $\mathbf{W} = \operatorname{diag}(\mathbf{w})$ is a diagonal weighting matrix for the residuals. Hence the goal is to adjust the parameters of $\mathbf{x}$ to find $\mathbf{x}^\star = \operatorname{argmin}_{\mathbf{x}} \chi^2(\mathbf{x})$

For non-linear functions the optimization must be carried out iteratively. In each iteration the parameters are adjusted so as to reduce eq. A.2. Here we give a very brief overview of some popular techniques to carry out the optimization, and refer the reader to Madsen et al. (2004) for a more in-depth treatment.

A.1.1 Gradient Descent

The idea of gradient descent is to adjust the parameters in the direction of the steepest descent of the gradient of the error function:

$$\mathbf{h}_{gd} = \alpha\,\mathbf{J}^T\mathbf{W}\,\mathbf{e}(\mathbf{x}) \tag{A.3}$$

where α is the step size parameter, and $\mathbf{J} = \dfrac{\partial\hat{\mathbf{f}}}{\partial\mathbf{x}}$ is the Jacobian matrix.

Although Steepest Descent methods can be highly convergent, the convergence can be slow close to the minimum

A.1.2 Newton's Method

Newton's method is an iterative method for finding stationary points, i.e. points where the first derivative of $\chi^2(\mathbf{x})$ is zero. The minimization proceeds by iteratively approximating the objective function by a quadratic function around the current value of $\mathbf{x}$:

$$\mathbf{h}_n = \alpha\,\mathbf{H}^{-1}\mathbf{J} \tag{A.4}$$

where $\mathbf{H}$ is the Hessian matrix of second derivatives of $\hat{\mathbf{f}}$.

Newton's method converges much faster than Gradient Descent close to the minimum. However, if the initial guess is too far from the minimum the convergence can be slow.

A.1.3 Gauss Newton Method

the Gauss Newton method is a variation to the above for minimizing a function under the assumption that it is approximately quadratic in its parameters close to the minumum.

In contrast to Newton's method, this technique does not require second derivatives, Instead the Hessian is approximated by $\mathbf{J}^T\mathbf{W}\,\mathbf{J}$.

In each iteration h_{gn} is found satisfying:

$$\left[\mathbf{J}^T \mathbf{W} \mathbf{J}\right] \mathbf{h}_{gn} = \mathbf{J}^T \mathbf{W} \mathbf{e}(\mathbf{x}) \tag{A.5}$$

Convergence is typically faster than Gradient Descent, but slower than Newton's method, and can fail if the initial guess is poor.

A.1.4 The Levenberg-Marquardt Method

The Levenberg-Marquardt method introduces an additional parameter to perform *damped* Gauss-Newton steps:

$$\left[\mathbf{J}^T \mathbf{W} \mathbf{J} + \lambda \mathbf{I}\right] \mathbf{h}_{lm} = \mathbf{J}^T \mathbf{W} \mathbf{e}(\mathbf{x}) \tag{A.6}$$

Large values of λ result in Gradient Descent updates, while small values adjust the parameters according to Gauss-Newton. Far from the minimum, Gradient Descent provides steady convergence, while Gauss-Newton provides rapid convergence close to the optimum.

A.2 Lie Groups, Lie Algebras, and the Special Orthogonal and Euclidean Groups

A.2.1 Lie Groups and Lie Algebras

We introduce briefly here the key concepts of Lie Groups and Lie Algebras, for a more in-depth treatment of Lie Groups and Lie Algebras, we refer the reader to Kirillov (2008).

A Lie Group is a smooth manifold which satisfies the following properties:

- Existence of Identity element: $\exists e \in G : a * e = e * a = a \, \forall a \in G$
- A binary group operation $* : G \times G \to G$ which satisfies the conditions of closure, associativity, and which is a smooth map.

- Existence of Inverses: $\forall a \in G : \exists b : a * b = b * a = e$ with the inverse map denoted by $^{-1} : G \to G$, which is a smooth map.

Examples of Lie Groups include; Euclidean Space R^n (with the group operation addition), the Circle Group S^1 (with the group operation multiplication), and the set of real-valued orthogonal matrices of dimension $O(n)$ (with the group operation of matrix multiplication).

A Lie Algebra consists of a vector space $\mathfrak{g}$ over a field $\mathbb{F}$, and a bilinear multiplication operation $[,] : \mathfrak{g} \times \mathfrak{g} \to \mathfrak{g}$ known as the Lie Bracket, with the following properties:

- Skew-symmetry: $[a,a] = 0,\ \forall a \in \mathfrak{g}$
- The Jacobi identity: $[a,[b,c]] + [b,[c,a]] + [c\,[a,b]] = 0,\ \forall a,b,c \in \mathfrak{g}$

implying anticommutivity $[a,b] = -[b,a],\ \forall a,b \in \mathfrak{g}$

Each Lie Group G has an associated Lie Algebra $\mathfrak{g}$, which is the tangent space of the Lie Group at the identity element. The lie algebra thus describes the local structure of the Lie Group. Members of $\mathfrak{g}$ are mapped to members of the associated group G via the exponential map. For groups over real valued matrices, which are of particular interest in this thesis, $\mathfrak{g}$ consists of the set of matrices A such that for $X \in A : e^{tX} \in G,\ \forall t \in \mathbb{R}$, where e^X is the matrix exponential:

$$e^X = \sum_{k=0}^{\infty} \frac{1}{k!} X^k \tag{A.7}$$

A.2.2 The Special Orthogonal Group SO(3)

The Special Orthogonal Group SO(n) is a subgroup of the Orthogonal Group O(n), and consists of orthogonal $n \times n$ matrices with determinant $+1$, with the group operation of matrix multiplication, and the inverse is given by the transpose operation.

$$SO(n) \doteq \{R \in \mathbb{R}^{n\times n} \mid R^T R = RR^T = I, det(R) = +1\} \tag{A.8}$$

Since we are concerned with motion in 3 dimensions, we consider in particular SO(3). This group is of particular interest since it is the group of 3×3 rotation matrices. A rotation matrix satisfying eq. A.8 has the following form:

$$\mathbf{R} = \left[\begin{array}{ccc} \mathbf{u} & \mathbf{v} & \mathbf{w} \end{array} \right] = \left[\begin{array}{ccc} u_x & v_x & w_x \\ u_y & v_y & w_y \\ u_z & v_z & w_z \end{array} \right] \tag{A.9}$$

where $\mathbf{u} = Re_1, \mathbf{v} = Re_2, \mathbf{w} = Re_3$ are an orthonormal, right-handed triad of unit vectors which represent the orientation relative to the standard basis vectors e_1, e_2, e_3 of $\mathbb{R}^3$. In geometric terms, a rotation turns a rigid body around the origin, along some fixed axis $\hat{e}$, and through some angle θ. Rotations are rigid body transformations, and hence satisfy the properties of preservation of distances and angles.

A.2.2.1 Lie Algebra of SO(3)

The Lie Algebra associated with $SO(3)$ is denoted $\mathfrak{so}(3)$ and is given by the set of all 3×3 skew-symmetric matrices, with the Lie Bracket $[,]$ given by the commutator $[a,b] = a * b - b * a$.

For a vector $\omega \in \mathbb{R}^3$, we can form a skew symmetric matrix by the operator ${}^\wedge : \mathbb{R}^3 \to \mathfrak{so}(3)$:

$$\hat{\omega} = \left[\begin{array}{ccc} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{array} \right] \tag{A.10}$$

The space of skew-symmetric matrices is hence:

$$\mathfrak{so}(3) \doteq \{ \hat{\omega} \in \mathbb{R}^{3 \times 3} \mid \omega \in \mathbb{R}^3 \} \tag{A.11}$$

Conversely, the ${}^\vee$ operator, extracts the components of the vector ω from a skew-symmetric matrix $\hat{\omega}$, ${}^\vee : \mathfrak{so}(3) \to \mathbb{R}^3$.

Exponential Coordinates for Rotation

Hence a member of $\mathfrak{so}(3)$ can be represented by just three parameters, $(\omega_1, \omega_2, \omega_3)$, and it follows that a rotation locally depends on only these three parameters. ω can be interpreted as a rotational velocity vector, representing a rotation around the axis $\dfrac{\omega}{\|\omega\|}$ at $\|\omega\|$ rad/s.

The rotation matrix $R \in SO(3)$ corresponding to $\hat{\omega}$ is obtained by the exponential map, which is given by the Rodrigues formula:

$$R = \mathrm{e}^{\hat{\omega}} = I + \frac{\hat{\omega}}{\|\omega\|} \sin(\|\omega\|) + \frac{\hat{\omega}^2}{\|\omega\|^2} (1 - \cos((\|\omega\|)) \tag{A.12}$$

If $\omega = 0$ then the result is the identity element.

A logarithmic map is also defined which maps elements of SO(3) to their corresponding representation in $\mathfrak{so}(3)$: $\forall R \in SO(3)$, $\exists \omega \in \mathbb{R}^3 : R = \mathrm{e}^{\hat{\omega}}$ where ω is given by:

$$\|\omega\| = \cos^{-1}\left(\frac{\text{trace}(R)-1}{2}\right), \frac{\omega}{\|\omega\|} = \frac{1}{2\sin(\|\omega\|)}\begin{bmatrix} R_{32}-R_{23} \\ R_{13}-R_{31} \\ R_{21}-R_{12} \end{bmatrix} \tag{A.13}$$

The above is not defined if $R = I$, since the identity represents no rotation at all, the axis of rotation can be chosen freely.

A.2.3 The Special Euclidean Group SE(3)

The Special Euclidean group SE(3) is the set of all rigid body motions in three dimensions with the group operator of matrix multiplication. A rigid body motion is a distance-, and orientation-preserving transformation consisting of a rotation and a translation:

$$SE(3) \doteq \{g = (R,t) \mid R \in SO(3), t \in \mathbb{R}^3\} \tag{A.14}$$

Rigid body transformations can be represented in a single matrix form by a 4×4 matrix using homogeneous coordinates, of the form:

$$\mathbf{g} = \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix} \tag{A.15}$$

yielding the following representation of SE(3):

$$SE(3) \doteq \left\{ g = \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix} \middle| R \in SO(3), t \in \mathbb{R}^3 \right\} \subset \mathbb{R}^{4\times4} \tag{A.16}$$

A.2.3.1 Lie Algebra of SE(3)

The Lie Algebra of SE(3) is denoted $\mathfrak{se}(3)$ and consists of the set of all 4×4 *twist* matrices $\hat{\xi}$, giving the Lie Algebra the following form:

$$\mathfrak{se}(3) \doteq \left\{ \hat{\xi} = \begin{bmatrix} \hat{\omega} & v \\ 0 & 0 \end{bmatrix} \middle| \hat{\omega} \in \mathfrak{so}(3), v \in \mathbb{R}^3 \right\} \subset \mathbb{R}^{4\times4} \tag{A.17}$$

Exponential Coordinates for Rigid Body Motions

It is clear from the definition that a twist $\hat{\xi} = \begin{bmatrix} \hat{\omega} & v \\ 0 & 0 \end{bmatrix}$ can be represented by just six coordinates, consisting of a linear part v and a rotational part ω, its *twist coordinates* $\xi \in \mathbb{R}^6 = \begin{bmatrix} v \\ \omega \end{bmatrix}$.

As it was for rotations, it is convenient to define two operators $^\vee$ and $^\wedge$ which map between twist matrices and coordinates:

$$^\vee : \mathfrak{se}(3) \to \mathbb{R}^6, \quad ^\wedge : \mathbb{R}^6 \to \mathfrak{se}(3) \tag{A.18}$$

The exponential map $\mathfrak{se}(3)$ to the SE(3) group is defined, using the Rodrigues formula (eq. A.19):

$$\mathrm{e}^{\hat{\xi}} = \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix}, R = \mathrm{e}^{\hat{\omega}}, t = \frac{\left(I - \mathrm{e}^{\hat{\omega}}\right)\hat{\omega}v + \omega\omega^T v}{\|\omega\|} \tag{A.19}$$

The above is defined for $\omega \neq 0$. If $\omega = 0$ then $\mathrm{e}^{\hat{\xi}} = \begin{bmatrix} I & v \\ 0 & 1 \end{bmatrix}$

A logarithmic map is also defined which maps from SE(3) to the algebra $\mathfrak{se}(3)$: $\forall g = (R,t) \in SE(3), \exists \xi = (v,\omega) : g = \mathrm{e}^{\hat{\xi}}$, where ξ is given by:

$$\omega = \log(R),\ v \text{ satisfies } \frac{\left(I - \mathrm{e}^{\hat{\omega}}\right)\hat{\omega}v + \omega\omega^T v}{\|\omega\|} = t \tag{A.20}$$

Since the above is not defined for $R = I$, in this case $\omega = 0, v = t$.

Bibliography

ASUS. Asus - xtion pro live. `http://www.asus.com/Multimedia/Xtion_PRO_LIVE`, 2013.

Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features (surf). *Computer vision and image understanding*, 110(3): 346–359, 2008.

Andrea Censi. An accurate closed-form estimate of icp's covariance. In *Robotics and Automation, 2007 IEEE International Conference on*, pages 3167–3172. IEEE, 2007.

Zhe Chen. Bayesian filtering: From kalman filters to particle filters, and beyond. *Statistics*, pages 1–69, 2003.

Alessandro Chiuso and Stefano Soatto. Monte carlo filtering on lie groups. In *Decision and Control, 2000. Proceedings of the 39th IEEE Conference on*, volume 1, pages 304–309. IEEE, 2000.

Changhyun Choi and Henrik I Christensen. Robust 3d visual tracking using particle filtering on the se (3) group. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 4384–4390. IEEE, 2011.

Changhyun Choi and Henrik I Christensen. 3d textureless object detection and tracking: An edge-based approach. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 3877–3884. IEEE, 2012a.

Changhyun Choi and H.I. Christensen. 3d pose estimation of daily objects using an rgb-d camera. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 3342 –3349, oct. 2012b. doi: 10.1109/IROS.2012.6386067.

Changhyun Choi, Yuichi Taguchi, Oncel Tuzel, Ming-Yu Liu, and Srikumar Ramalingam. Voting-based pose estimation for robotic assembly using a 3d sensor. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA'12), Minnesota, May 2012*, 2012.

Andrew I Comport, Éric Marchand, and François Chaumette. Robust model-based tracking for robot vision. In *Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, volume 1, pages 692–697. IEEE, 2004.

Arnaud Doucet, Simon Godsill, and Christophe Andrieu. On sequential monte carlo sampling methods for bayesian filtering. *Statistics and computing*, 10(3):197–208, 2000.

Bertram Drost, Markus Ulrich, Nassir Navab, and Slobodan Ilic. Model globally, match locally: Efficient and robust 3d object recognition. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2010.

Tom Drummond and Roberto Cipolla. Real-time visual tracking of complex structures. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(7):932–946, 2002.

Germán Martín García, Dominik Alexander Klein, Jörg Stückler, Simone Frintrop, and Armin B Cremers. Adaptive multi-cue 3d tracking of arbitrary objects. In *Pattern Recognition*, pages 357–366. Springer, 2012.

Neil J Gordon, David J Salmond, and Adrian FM Smith. Novel approach to nonlinear/non-gaussian bayesian state estimation. In *IEE Proceedings F (Radar and Signal Processing)*, volume 140, pages 107–113. IET, 1993.

Helmut Grabner, Michael Grabner, and Horst Bischof. Real-time tracking via on-line boosting. In *Proc. BMVC*, volume 1, pages 47–56, 2006.

G. Grisetti, C. Stachniss, and W. Burgard. Improved techniques for grid mapping with rao-blackwellized particle filters. *Robotics, IEEE Transactions on*, 23(1):34 –46, feb. 2007. ISSN 1552-3098. doi: 10.1109/TRO.2006.889486.

Chris Harris. Tracking with rigid models. In *Active vision*, pages 59–73. MIT Press, 1993.

Stefan Hinterstoisser, Cedric Cagniart, Slobodan Ilic, Peter Sturm, Nassir Navab, Pascal Fua, and Vincent Lepetit. Gradient response maps for real-time detection of textureless objects. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 34(5):876–888, 2012.

Michael Isard and Andrew Blake. CondensationâĂŤconditional density propagation for visual tracking. *International journal of computer vision*, 29 (1):5–28, 1998.

Eunyoung Kim and Gerard Medioni. 3d object recognition in range images using visibility context. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pages 3800–3807. IEEE, 2011.

Alexander A Kirillov. *An introduction to Lie groups and Lie algebras*, volume 113. Cambridge University Press, 2008.

Georg Klein and David Murray. Full-3d edge tracking with a particle filter. In *British Machine Vision Conference*, pages 1119–1128, 2006.

Danica Kragic and Markus Vincze. Vision for robotics. *Foundations and Trends in Robotics*, 1(1):1–78, 2009.

Junghyun Kwon and Frank C. Park. Visual tracking via particle filtering on the affine group. *Int. J. Rob. Res.*, 29(2-3):198–217, February 2010. ISSN 0278-3649. doi: 10.1177/0278364909345167. URL `http://dx.doi.org/10.1177/0278364909345167`.

Junghyun Kwon, Minseok Choi, F. C. Park, and Changmook Chun. Particle filtering on the euclidean group: framework and applications. *Robotica*, 25(6):725–737, November 2007. ISSN 0263-5747. doi: 10.1017/S0263574707003529. URL `http://dx.doi.org/10.1017/S0263574707003529`.

Sylvain Lefebvre, Samuel Hornus, and Fabrice Neyret. Octree textures on the gpu. *GPU gems*, 2:595–613, 2005.

Vincent Lepetit and Pascal Fua. *Monocular-Based 3D Tracking of Rigid Objects*. Now Pub, 2005.

Vincent Lepetit and Pascal Fua. Keypoint recognition using randomized trees. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 28(9):1465–1479, 2006.

Peihua Li, Tianwen Zhang, and Arthur EC Pece. Visual contour tracking based on particle filters. *Image and Vision Computing*, 21(1):111–123, 2003.

David G Lowe. Object recognition from local scale-invariant features. In *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, volume 2, pages 1150–1157. Ieee, 1999.

Yi Ma, Stefano Soatto, Jana Košecká, and S. Shankar Sastry. Representation of a three-dimensional moving scene. In *An Invitation to 3-D Vision*, volume 26 of *Interdisciplinary Applied Mathematics*, pages 15–43. Springer New York, 2004. ISBN 978-1-4419-1846-8. doi: 10.1007/978-0-387-21779-6_2. URL `http://dx.doi.org/10.1007/978-0-387-21779-6_2`.

K. Madsen, H. B. Nielsen, and O. Tingleff. Methods for non-linear least squares problems (2nd ed.), 2004.

Manuel Martinez, Alvaro Collet, and Siddhartha S Srinivasa. Moped: A scalable and low latency object recognition and pose estimation system. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 2043–2049. IEEE, 2010.

metaio. metaio - augmented reality. `http://www.metaio.com/`, 2013.

Microsoft. Kinect for windows sensor components and specifications. `http://msdn.microsoft.com/en-us/library/jj131033.aspx`, 2013.

Michael Montemerlo, Sebastian Thrun, Daphne Koller, Ben Wegbreit, et al. Fastslam 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges. In *International Joint Conference on Artificial Intelligence*, volume 18, pages 1151–1156, 2003.

Mustafa Ozuysal, Michael Calonder, Vincent Lepetit, and Pascal Fua. Fast keypoint recognition using random ferns. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32(3):448–461, 2010.

Chavdar Papazov and Darius Burschka. An efficient ransac for 3d object recognition in noisy and occluded scenes. In *Computer Vision–ACCV 2010*, pages 135–148. Springer, 2011.

Michael K Pitt and Neil Shephard. Filtering via simulation: Auxiliary particle filters. *Journal of the American statistical association*, 94(446):590–599, 1999.

Victor A Prisacariu and Ian D Reid. Pwp3d: Real-time segmentation and tracking of 3d objects. *International journal of computer vision*, 98(3):335–354, 2012.

ROS. Precision and accuracy of the kinect sensor. `http://www.ros.org/wiki/openni_kinect/kinect_accuracy`, 2012.

David A Ross, Jongwoo Lim, Ruei-Sung Lin, and Ming-Hsuan Yang. Incremental learning for robust visual tracking. *International Journal of Computer Vision*, 77(1-3):125–141, 2008.

Yong Rui and Yunqiang Chen. Better proposal distributions: Object tracking using unscented particle filter. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 2, pages II–786. IEEE, 2001.

Radu Bogdan Rusu, Nico Blodow, and Michael Beetz. Fast point feature histograms (fpfh) for 3d registration. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, pages 3212–3217. IEEE, 2009.

Chunhua Shen, Michael J Brooks, and Anton Van Den Hengel. Augmented particle filtering for efficient visual tracking. In *Image Processing, 2005. ICIP 2005. IEEE International Conference on*, volume 3, pages III–856. IEEE, 2005.

Iryna Skrypnyk and David G Lowe. Scene modelling, recognition and tracking with invariant image features. In *Mixed and Augmented Reality, 2004. ISMAR 2004. Third IEEE and ACM International Symposium on*, pages 110–119. IEEE, 2004.

J. Stückler and S. Behnke. The rgb-d object tracking dataset. `http://www.ais.uni-bonn.de/download/objecttracking.html`, 2012.

J. Stückler and S. Behnke. Model learning and real-time tracking using multi-resolution surfel maps. In *In Proc. of the AAAI Conference on Artificial Intelligence (AAAI-12), Toronto, Canada, July 2012*, 2012.

Jörg Stückler. mrsmap - rgb-d mapping using multi-resolution surfel maps. `http://code.google.com/p/mrsmap`, 2013.

Jörg Stückler and Sven Behnke. Multi-resolution surfel maps for efficient dense 3d modeling and tracking. *Journal of Visual Communication and Image Representation*, 2013.

J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers. A benchmark for the evaluation of rgb-d slam systems. In *Proc. of the International Conference on Intelligent Robot Systems (IROS)*, Oct. 2012.

Sebastian Thrun, Wolfram Burgard, Dieter Fox, et al. *Probabilistic robotics*, volume 1. MIT press Cambridge, 2005.

Hideaki Uchiyama, Eric Marchand, et al. Object detection and pose tracking for augmented reality: Recent approaches. In *18th Korea-Japan Joint Workshop on Frontiers of Computer Vision (FCV)*, 2012.

Luca Vacchetti, Vincent Lepetit, and Pascal Fua. Combining edge and texture information for real-time accurate 3d camera tracking. In *Mixed and Augmented Reality, 2004. ISMAR 2004. Third IEEE and ACM International Symposium on*, pages 48–56. IEEE, 2004.

Eric Wahl, Ulrich Hillenbrand, and Gerd Hirzinger. Surflet-pair-relation histograms: a statistical 3d-shape representation for rapid classification. In *3-D Digital Imaging and Modeling, 2003. 3DIM 2003. Proceedings. Fourth International Conference on*, pages 474–481. IEEE, 2003.

Wikipedia. Precision and recall. `http://en.wikipedia.org/wiki/Precision_and_recall`, 2013.

Declaration of Authorship

“I hereby declare that I have prepared this thesis independently and without the use of sources and aids other than those stated. All sources have been cited accordingly.”

Bonn

Manus McElhone