# Implicit Shape Representation for 2D/3D Tracking and Reconstruction

**Yuheng (Carl) Ren**
**Oriel College**


Active Vision Group
Department of Engineering Science
University of Oxford

**Trinity Term 2014**

This thesis is submitted to the Department of Engineering Science, University of Oxford, for the degree of Doctor of Philosophy. This thesis is entirely my own work, and, except where otherwise indicated, describes my own research.

Yuheng (Carl) Ren
Oriel College

Doctor of Philosophy
Trinity Term 2014

# Implicit Shape Representation for 2D/3D Tracking and Reconstruction

**Abstract**

This thesis develops and describes methods for real-time tracking, segmentation and 3-dimensional (3D) model acquisition, in the context of developing games for stroke patients that are rehabilitating at home. Real-time tracking and reconstruction of a stroke patient's feet, hands and the control objects that they are touching can enable not only the graphical visualization of the virtual avatar in the rehabilitation games, but also permits measurement of the patient's performs.

Depth or combined colour and depth imagery from a Kinect sensor is used as input data. The 3D signed distance function (SDF) is used as implicit shape representation, and a series of probabilistic graphical models are developed for the problem of model-based 3D tracking, simultaneous 3D tracking and reconstruction and 3D tracking of multiple objects with identical appearance. The work is based on the assumption that the observed imagery is generated jointly by the pose(s) and the shape(s). The depth of each pixel is randomly and independently sampled from the likelihood of the pose(s) and the shape(s). The pose(s) tracking and 3D shape reconstruction problems are then cast as the maximum likelihood (ML) or maximum a posterior (MAP) estimate of the pose(s) or 3D shape.

This methodology first leads to a novel probabilistic model for tracking rigid 3D objects with only depth data. For a known 3D shape, optimization aims to find the optimal pose that back projects all object region pixels onto the zero level set of the 3D shape, thus effectively maximising the likelihood of the pose. The method is extended to consider colour information for more robust tracking in the presence of outliers and occlusions. Initialised with a coarse 3D model, the extended method is also able to simultaneously reconstruct and track an unknown 3D object in real time. Finally, the concept of 'shape union' is introduced to solve the problem of tracking multiple 3D objects with identical appearance. This is formulated as the minimum value of all SDFs in camera coordinates, which (i) leads to a per-pixel soft membership weight for each object thus providing an elegant solution for the data association in multi-target tracking and (ii) it allows for probabilistic physical constraints that avoid collisions between objects to be naturally enforced.

The thesis also explore the possibility of using implicit shape representation for online shape learning. We use the harmonics of 2D discrete cosine transform (DCT) to represent 2D shapes. High frequency harmonics are decoupled from low ones to represent the coarse information and the details of the 2D shape. A regression model is learnt online to model the relationship between the high and low frequency harmonics using Locally Weighted Projection Regression (LWPR). We have demonstrated that the learned regression model is able to detect occlusion and recover them to the complete shape.

# Acknowledgements

First and foremost I would like to thank my supervisors, Professor David Murray and Professor Ian Reid for their continuous guidance and essential help and support, without which this thesis would not have been possible.

Not only in relation to the EU project REWIRE, I would like to thank my collaborator, Dr. Victor Prisacariu, for his long-term encouragement and assistance. Furthermore, I also acknowledge the many interesting and insightful conversations I had, while working on this thesis, with present and past members of the Active Vision Lab, such as (in alphabetical order) Ben Benfold, Amaury Dame, Alex Flint, Duncan Frost, Olaf Kähler, Alex Segal and Eric Sommerlade.

I would also like to thank China Oxford Scholarship Fund for their generous offering of scholarship, which supported my first-year study in Oxford.

This work was supported in part by EU project REWIRE under the $7^{th}$ Framework Program.

Last but definitely not least, I would like to thank my wife Hong Li and my parents for all the love, support and understanding that they have given me throughout the years.

# Contents

# List of Figures

# List of Tables

# Images with Copyrights

The following copyright images have been used in this thesis with permission.

- Fig. 1.1, 1.2, 1.4: Image reproduced from REWIRE website.
  ©REWIRE project.
- Fig. 1.5: Image from Microsoft MSDN webiste.
  ©Microsoft
- Fig. 2.2: Image reproduced from Drummond and Chipolla [39].
  ©2002 IEEE.
- Fig. 2.3: Image reproduced from Skrypnyk and Lowe [130].
  ©2004 IEEE.
- Fig. 2.5: Images reproduced from Gall *et al.* [46].
  ©2008 IEEE.
- Fig. 2.6: Image reproduced from Zhang *et al.* [148].
  ©1999 IEEE.
- Fig. 2.7: Image reproduced from Nayar *et al.* [91].
  ©1996 IEEE.
- Fig. 2.8: Images from Matusik *et al.* [87].
  ©2000 ACM.
- Fig. 2.9: Image from Kutulakos and Seitz [65]
  ©2000 ACM.
- Fig. 2.11: Image reproduced from website by Xiao [146].
  ©Jianxiong Xiao.

# 1

# Introduction

*This chapter details the motivation behind this thesis, the objectives, major contributions and basic outline of our work.*

## 1.1 Stroke rehabilitation

Stroke – the blockage or burst of a blood vessel in the brain – is a sudden and devastating illness that affects the way the patient's brain and body function. Every year, in high and middle income countries, there are around 16,000,000 first occurrences of stroke [86], and there are approximately 152,000 strokes in the UK [134]. Most people affected are over 65, it occurs with decreasing frequency in those who are younger, including children and even babies.

The mean length of stay in hospital for stroke patients is around 20 days [4]. After discharge from hospital, even those unfortunate enough to have suffered substantial physical impairment are required to rehabilitate for the most part at home. Although patients are given daily exercises to perform, the daily systematic *monitoring* of recovery has been all but impracticable on grounds of cost. Instead, patients are left somewhat to their own devices in the periods between increasingly

less frequent visits by a therapist to the home or by the patient back to the hospital. Without daily measurement of progress, the natural tendency is for patients to neglect their routines, only becoming re-motivated shortly before a visit.

Home connection to the Internet and the development of motion capture technology has for some time — perhaps two decades now — promised to brighten this bleak landscape. However, for most of this time sensor technology has been ill-suited to the task. It is unreasonable to expect patients either to attach multiple motion gauges (for inside-out sensing or egocentric) or to wear reflective markers in front of a calibrated multi-camera mocap rig (for outside-in sensing).

A transformational change has been brought about recently by the development of mass-produced, affordable, but highly engineered sensors designed principally for game-playing in the home. Devices such as the Nintendo Wii remote, the Microsoft Kinect, and the Nintendo balance board [3, 1, 2] have been demonstrated tracking human movement with a latency (pipeline delay) and accuracy wholly commensurate with the clinical demands of rehabilitation (*e.g.* Clark *et al.* [30]). Moreover, the game-playing environment for which the sensors are designed is already one which can aid rehabilitation, by turning the tedium of repetitive motion into something more stimulating, as shown by, for example, Boian *et al.* [16], and Lauterbach *et al.* [70].

These devices have already spawned commercial games aimed at the fitness and well-being markets. However, much less developed are games that are born of clinical research and that aim to guide *recovery* through exercise (*e.g.* Cameirao *et al.* [23]; Burke *et al.* 2009 [21]; Zimmerli *et al.* [150]). Many of these non-commercial games do not follow good game design practice (Schell [122]), are over simplified, and have poor graphics. Moreover, both existing commercial and research offerings lack the facilities to record the patient's motion continuously, and to provide feedback to correct possibly harmful motion habits. Beyond the immediate visual

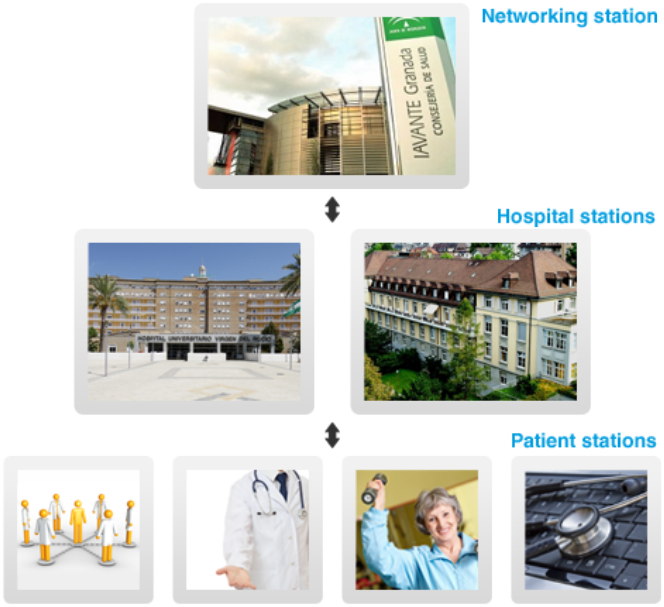



**Figure 1.1:** The overall architecture of the home rehabilitation system developed in the the EU Rewire project.

feedback during the exercise, longer term feedback through an authoritative but friendly graphical agent — an on-screen therapist — is needed to summarize and comment upon progress, and to admonish and encourage as appropriate. This same feedback should also allow a game to self-adapt its complexity to permit a patient always to benefit from playing it, as suggested by Colombo *et al.* [31] and Mainetti *et al.* [83].

## 1.2 The REWIRE project

Developing a system to addresses current shortcomings by combining clinical and information engineering expertise is the objective of the EU FP7 project REWIRE[1], which is October 2014 is undergoing its first clinical trials in both Switzerland and Spain.

[1] http://www.rewire-project.eu


**Figure 1.2:** The Kinect sensor (left) and the setup of patient station of home rehabilitation platform (right).

Fig. 1.1 shows the three main components of system architecture. The *network station* with database server is a central repository for software, clinical expertise, and anonymised patient data for data-mining. The *hospital station* allows clinicians direct access to the performance of their current group of patients, with both recorded data and streamed real-time results available, along with other patient records. The actual performance of each patient is monitored at home by their own *patient station*, comprising a high-end laptop equipped with a GPU as co-processor. Its inputs are from a Microsoft Kinect, a balance board, and in some cases a wearable monitor. The arrangement, using a TV for larger display, set up is shown in Fig. 1.2.

## 1.3 The functional design of patient station

The functional design of the patient station is shown in Fig. 1.3. Moving round the system diagram there are two principal areas of information engineering challenge associated with it. While we write of two engineering challenges, it is worth recalling that the real challenge here rests with a rather unwell patient who must re-learn their damaged motor control skills.

First is that the games to be played, their design, selection and sequencing, must

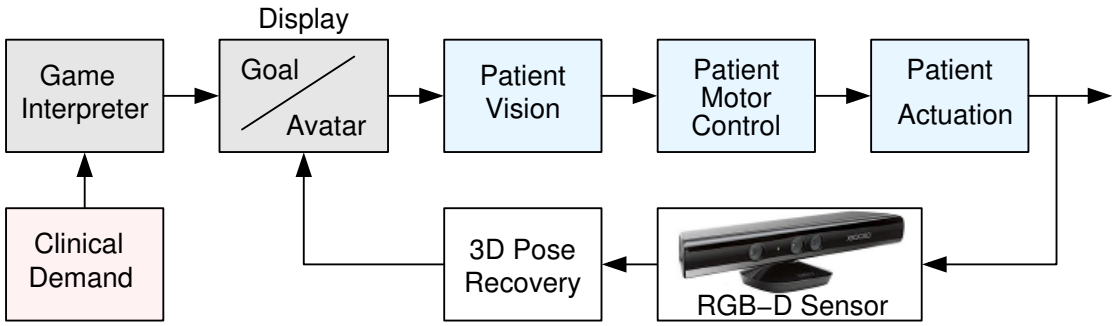
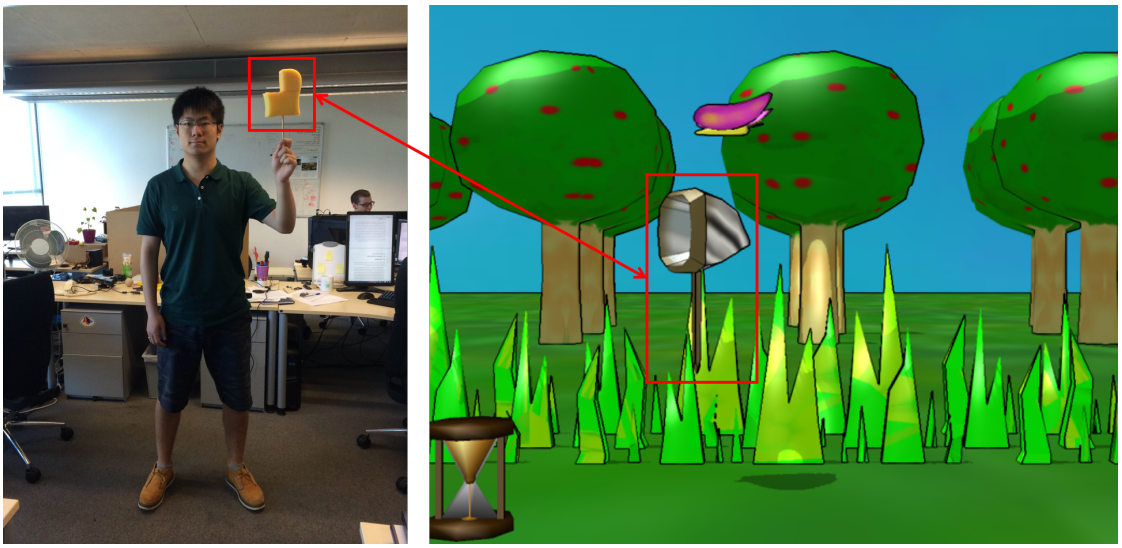



**Figure 1.3:** The functional architectures of the patient station developed in the the EU Rewire project.

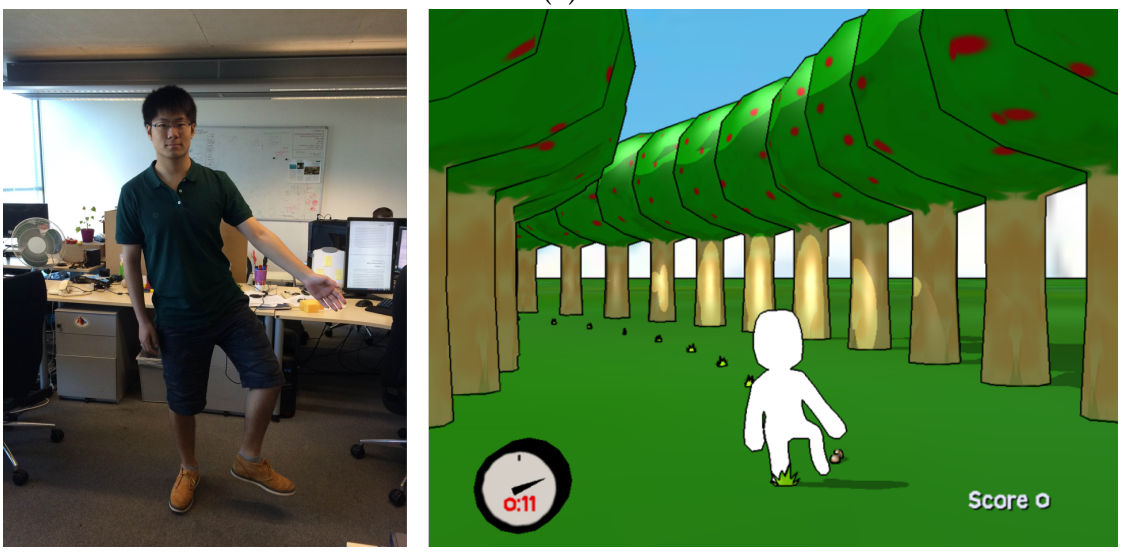
be informed by clinical assessment and determined by existing clinical protocol. The Game Interpreter in Fig. 1.3 is then not a mere game engine, but one wrapped up in a clinical knowledge-based system. This system, the Intelligent Game Engine for Rehabilitation (IGER), has been designed and built by Borghese and collaborators at the University of Milano [18, 101, 17]. On the display, the patient views him- or her-self as an avatar embedded in a virtual game environment, and can interact with it by moving. In most cases, patients prefer to see a cartoon character as avatar rather than their own appearance mapped onto the screen. Unsteady motion or failure to reach a goal is easier to accept if the patient is not represented directly. Via their visual and motor control system, the stimulus on the screen prompts a movement from the patient.

As shown in Fig. 1.3, the second challenge, and the motivation for the work in this thesis, is the segmentation of the patient from the scene and the tracking in 3D of the patient's movement. This allows the changing pose to be fed back to the avatar. Fig. 1.4 shows the output of the game engine during two games that provide exercise for the upper and lower limbs. In (a) the patient uses a control object to move a net in the game to capture moving butterflies. Part (b) is a balancing game, requiring the the patient's feet to be tracked[2]. Patients prefer to be treated rather gently and to be taken somewhat outside reality.

[2]The child-like appearance of the games is the result of careful research by the Milan group [18], rather than one of whimsy.



(a)




(b)

**Figure 1.4:** (a,b) Examples of subject playing games that exercise (the upper limbs and lower limbs, respectively.)

Local monitoring alone is of course insufficient, and as mentioned above the REWIRE system allows the results of exercises to be uploaded to the hospital station for review, and adjustment of the clinical demand to be downloaded. Interestingly, the hospital station also permits virtual interaction between other patients, clinicians, therapists and carers, allowing the sharing of good practice and, importantly, the reduction of patient isolation. Amongst those who consent, patients are allowed to send messages of encouragement and advice to others. Anonymized patient data can be returned to the root database server, and machine learning techniques have been developed to discover common trends in the outcomes rehabilitation treatments.

## 1.4 The Kinect sensor

The research described in this thesis is concerned with the analysis of colour and depth imagery obtained from a Kinect sensor in the patient station. In the patient station, the colour and depth imagery for patient tracking is obtained directly from a Microsoft Kinect sensor. The Kinect sensor is a line of RGB-D input devices by Microsoft for Xbox 360 and Xbox One video game consoles and Windows PCs. The device features an RGB camera, a depth sensor and a multi-array microphone. The internal structure of the Kinect sensor is shown in Fig. 1.5. The colour camera by default output RGB video stream uses 8-bit VGA resolution (640×480 pixels). The depth sensor consists of an infrared laser projector combined with a monochrome CMOS sensor, which captures video data in 3D under any ambient light conditions. The monochrome depth sensing video stream is in VGA resolution (640×480 pixels) with 11-bit depth, which provides 2048 levels of sensitivity. The first-generation Kinect was first introduced in November 2010 and its depth sensor has a practical ranging limit of 1.2∼3.5 m. A version for Windows was released in February,2012, which extended the range to 0.7∼6 m. In our system, we have used the later Kinect

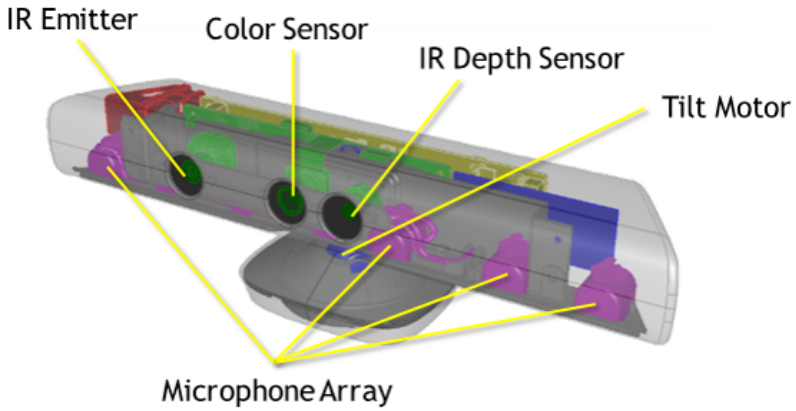



**Figure 1.5:** The internal structure of the Kinect sensor.

for Windows version. The colour camera and the depth sensor of a Kinect is not in the same location, and there is a fixed baseline between the two cameras that requires calibration. Also, the intrinsic parameters of the colour camera and the depth sensor are not provided. Our method for calibrating the Kinect sensor is developed in Chapter 3.

## 1.5 Specific challenges raised in REWIRE's feedback loop

Understanding how patients are moving when in proximity to objects and in a visually cluttered, everyday or uncontrolled environment gives rise to a number of research problems in segmentation, 3D shape reconstruction and 3D model-based pose tracking. In summary, these are:

- **An uncalibrated environment:** A typical setup for 3D reconstruction of an object is a calibrated multi-camera system or a 3D scanner, but neither is suitable for use in a patient's home environment owing to their complexity and cost. Although the Kinect is pre-calibrated by the manufacturer, early tests indicated that the calibration was insufficiently accurate for the task of

tracking small, untextured objects.

- **Full DoF pose estimation:** For the purpose of accurate exercise evaluation, the full 6 degrees of pose freedom of a control object and specific parts of the body need to be recovered at each frame. Naïve tracking in 2D in the image domain, or in 3D using bounding boxes would not be adequate.

- **Occlusion and missing data:** The small objects we wish to track may exhibit drop out in the depth image due to reflection or other triangulation failure of the structured light depth sensor. Furthermore, the object being tracked can be expected to be partially occluded by another moving object. For example, the control object might be occluded by the hand, and one foot may be occluded by the other foot. The tracker must neither end up tracking the wrong object nor fail completely.

- **Appearance changes:** The visual appearance of objects can change over time, due either to changing room lighting or surface conditions or colour. For example, a shoe may be red on one side, blue on the other side.

- **Identical appearance in multi-target tracking:** Neither colour nor depth are wholly reliable discriminators. Similarity in appearance is a fundamental problem when tracking two interacting feet, for example. Independent single object target trackers might easily confuse data association when they move close together.

- **Real-time performance:** The solution to the previous challenges have to be capable of processing the incoming RGB-D data in real-time, in order to be useful. Real-time is taken to be video rate, which is usually defined as 30 Hz or greater with less than 5 ms latency.

## 1.6 Thesis approach and summary

While the REWIRE project provides the motivation for the research presented in this thesis, and provides a exacting limits on practical performance, it seems appropriate in the body of the thesis to address the scientific questions raised in a general rather than specific manner. These questions lie in the areas of segmentation, 3D shape reconstruction and model-buiding, and 3D model-based pose tracking, using as raw input colour and depth imagery

There are two strong threads running through the work. First, throughout the thesis, we use level-set embedding functions to represent 3D and 2D shapes — the surface of a 3D shape and the contour of a 2D shape are implicitly represented by the zero-level set of the embedding function. Second is the adoption throughout of Bayesian methods. It is assumed that each observed RGB-D image generated by the object embedding function and the current object pose such that there is pixel-wise independence. This allows us to model the tracking and reconstruction problem in a probabilistic way and estimate the pose and the shape as a maximum likelihood (ML) or maximum a posteriori (MAP) estimation for the optimal pose or shape (embedding function) from which the current RGB-D image is generated.

**Chapter 2: Review**

Chapter 2 provides a broad overview of previous related methodologies for real-time tracking, 3D reconstruction, and occlusion reasoning, and critical conclusions are drawn. Details of particularly relevant work within the last five years will then be revisited chapter by chapter through the thesis.

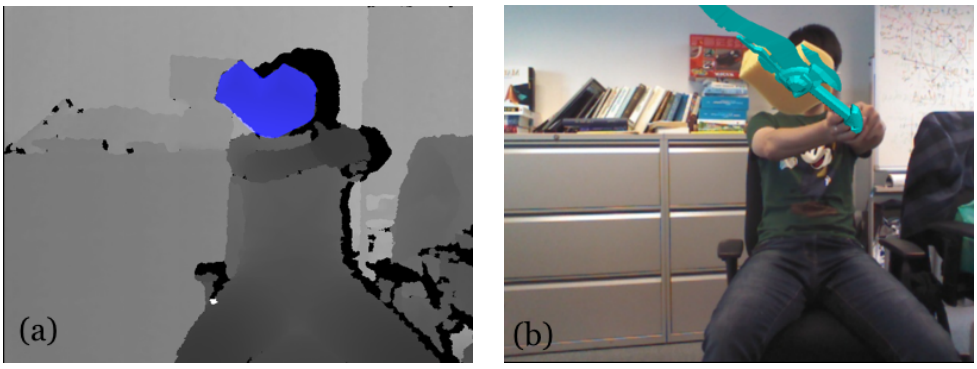



**Figure 1.6:** Example of 3D tracking. (a) The depth image with projection of tracked object overlaid in blue. (b) The recovered pose mapped on to a graphics object. From Chapter 3.

**Chapter 3: 3D pose tracking from depth, and application to calibration, etc**

In Chapter 3, we propose a probabilistic model for tracking a single known object in 3D using only depth data. The pose tracking is cost as a minimisation problem which can be efficiently solved by second-order gradient-based methods. Instead of relying on the gradients of the depth image, which amplifies noise, the minimisation uses the much smoother gradients of the SDF to guide the search for an optimum. Usefully, a number of the gradients required can be precomputed.

Unlike traditional hypothesis – test methods, our work does not 'render' the object with current pose into observation domain, so no computational intensive Z-buffering is required. This makes our method inherently suitable for GPU parallelization. Our framework is region-based, so no point correspondences are required and also it is robust to occlusion and missing data. Both CPU and GPU implementation have achieved real-time performance. An example of 3D tracking with only depth data is shown in Fig. 1.6.

A number of extensions of the same framework are demonstrated. The two most relevant to the REWIRE project are the algorithm's extension from a 6-DoF to an 11-DoF optimization for both pose and intrinsic calibration parameters of the depth camera. The tracking in depth and colour separately is then used to determine the mapping between the colour and depth cameras.

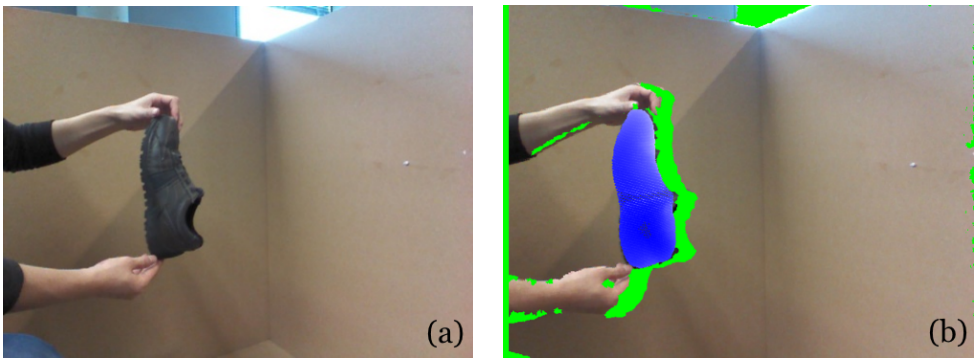



**Figure 1.7:** Example of simultaneous tracking and reconstruction. From Chapter 4.

**Chapter 4: Simultaneous reconstruction and tracking from depth and colour**

In Chapter 4, we extend the graphical model to describe the per-pixel generation of both the depth and colour images of a scene with the aim of permitting simultaneous pose tracking and 3D reconstruction. The tracking and reconstruction problems are casted as the MAP estimation on two simplification of the extended graphical model. As before, the probabilistic model leads to a differentiable cost function that can be efficiently solved by second-order optimization.

The use of colour information makes the tracking method robust to close-to-surface outliers, missing data and occlusion, while still achieving real-time performance on both our CPU and GPU implementation. For reconstruction, we extend the idea of space carving. An inside/outside volumetric model of the object is learnt online while tracking. Initialized with a simple model (*e.g.* a ball or a box), the 3D shape is reconstructed by evolving a 3D level-set embedding function. The reconstruction method can be implemented on GPU in massive parallel fashion, also achieving real-time performance.

An example of simultaneous tracking and reconstruction is shown in Fig. 1.7.

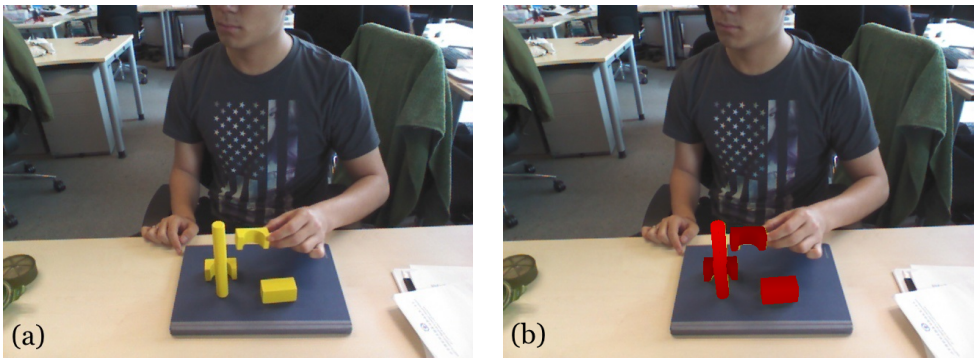



**Figure 1.8:** Example of 3D tracking of multiple objects with identical appearance. (a) The original image showing the yellow objects and (b) now overlaid with projections of the tracked objects in red. From Chapter 5.

### Chapter 5: Tracking multiple objects with identical appearance

Chapter 5 is the last of the trio which exploit the implicit shape representation for efficient tracking and reconstruction. We extend the probabilistic formulation for 3D tracking developed in Chapter 4 to account for the tracking of multiple moving objects with identical appearance. We use an augmented graphical model to handle the data association problem and enforce physical constraint at the same time. Objects are again represented by 3D SDFs, but we propose a convenient fusion the SDFs in the camere frame to handle the two challenging problems in multi-target tracking of (i) data association and (ii) physical constraint.

Unlike most current approaches that use distinctive appearance to distinguish between multiple objects, our probabilistic model automatically compute a 'soft' membership for each pixel in presents of identical appearance, which solves the data association problem efficiently. The physical constraint that multiple objects should not penetrate each other is naturally enforced probabilistically.

The method is computationally efficient, with a linear increase in computational time w.r.t the number of objects. Even the CPU implementation is able to track five objects in real-time. An example of multi-object tracking is shown in Fig. 1.8.

(a) (b)

**Figure 1.9:** Example of 2D occlusion recovery for tracking from Chapter 6.

**Chapter 6: Online learning of 2D shapes for occlusion detection and recovery**

In Chapter 6, we explore the possibility of using an implicit shape representation for online shape learning, and consider the detection and recovery from occlusion for 2D tracking.

2D shapes are represented by the harmonics in the 2D discrete cosine transform (DCT) of their contours. We propose a regression model to map from the high frequency harmonics to low frequency ones. This is learnt online using Locally Weighted Projection Regression (LWPR). After sufficient observations of a set of complete shape as online training, the model is able to detect occlusion and recover them the the complete shape in real-time. The method can be used a flexible component for any region-based or segmentation-based 2D tracker. An example of occlusion recovery prior to 2D tracking is shown in Fig. 1.9.

Though not used in in entirety on-line in REWIRE, a particular useful part has been the DCT mechanism to describe contours in [102].

**Chapter 7: Conclusions, and REWIRE revisited**

In Chapter 5 we present a thesis summary, draw conclusions, and suggest possible avenues for future research. We also provide a short postscript on the current status of the REWIRE project.

**Appendix: Superpixel generation on GPU**

Much of the implementation in the thesis has involved coding on GPUs, and in the Appendix we provide a description of GPU version of the SLIC algorithm for superpixel generation.

## 1.7 Publications

The work in this thesis has been published in a journal and several conferences as follows.

**From Chapter 3:**

[114] C. Y. Ren and I. D. Reid *"A unified energy minimization framework for model fitting in depth"* In Proc 2nd Workshop on Consumer Depth Cameras for Computer Vision in conjunction with ECCV, Fiorenza, Italy, 2012.

[102] M. Pirovano, C. Y. Ren, I. Frosio, P. L. Lanzi, V. A. Prisacariu, D. W. Murray, and N. A. Borghese *"Robust silhouette extraction from Kinect data"* In Proc 17th International Conference on Image Analysis and Processing, Naples, Italy, 2013.

**From Chapter 4:**

[113] C. Y. Ren, V. A. Prisacariu, D. W. Murray and I. D. Reid *"STAR3D: Simultaneous tracking and reconstruction of 3D objects using RGB-D data"* In Proc 14th IEEE International Conference on Computer Vision, Sydney, Australia, 2013.

**From Chapter 5:**

[112] C. Y. Ren, V. A. Prisacariu, O. Kähler, D. W. Murray and I. D. Reid *"3D Tracking of Multiple Objects with Identical Appearance using RGB-D Input"* In Proc International Conference on 3D Vision, Tokyo, Japan, 2014.

**From Chapter 6:**

[116] Y. Ren, V. A. Prisacariu and I. D. Reid *"Regressing local to global shape properties for online segmentation and tracking"* In Proc 22nd British Machine Vision Con-

ference, Dundee, UK, 2011.

[111] C. Y. Ren, V. A. Prisacariu and I D Reid *"Regressing local to global shape properties for online segmentation and tracking"* International Journal of Computer Vision, vol 106, No.3, pp 269–281, 2014.

**From methodology in the entire thesis:**

[38] A. Dame, V. A. Prisacariu, C. Y. Ren and I. D. Reid *"Dense Reconstruction Using 3D Object Shape Priors"* In Proc 26th IEEE Conference on Computer Vision and Pattern Recognition, Portland OR, USA, 2013.

C. Y. Ren, V. A. Prisacariu, O. Kähler, D. W. Murray and I. D. Reid *"Dense Reconstruction and Tracking of Multiple 3D Objects from Depth-Colour Imagery"* Submitted (November 2014) to IEEE Transactions on Pattern Analysis and Machine Intelligence.

**From the Appendix:**

[115] C. Y. Ren and I. D. Reid *"gSLIC: A real-time implementation of SLIC superpixel segmentation"* Technical Report, Department of Engineering Science, Oxford University, 2011

# 2

# Literature Review

## 2.1 Introduction

Our purpose here is to give a broad review on approaches to model-based 3D tracking, 3D model acquisition and multi-object tracking and to provide general discussion of the advantages and disadvantages of existing methods. More recent work — say those within last five years — leading to the state of the art will be discussed in the relevant chapters themselves.

## 2.2 Model-based 3D tracking

Methods for model-based 3D tracking in the literature can be roughly categorized into three sets based on the dimensionality of the image cue that is used, *i.e.*, *edge-based methods*, *point-based methods* and *region based methods* as detailed in the following subsection. Further discussion of real-time model-based 3D tracking approaches in recent years is provided in Chapter. 3.

### 2.2.1 Edge-based methods

Most early work in model-based 3D tracking was edge based. Two early seminal works are Lowe [79] and Harris [48]. These methods are similar in that they both assume known model and an estimate of the current pose is available. The 3D model is then projected in the image to generating a prediction view, and correspondence is sought between the prediction and the image evidence.

Lowe's method first extracts all the edges from the full image, using the Marr and Hildreth [85] edge detector. These edges are then connected to form lines. Correspondence is established between the most reliable lines extracted from the edges and those from the predicted image. Another prediction view is then generated from the new estimation of the pose and further correspondences established between the unmatched lines from the image and the model. The pose update step is a minimisation of a sum of weighted least squares. The multi-step methodology reduces the search space incrementally, using the most likely matches in the image. The disadvantage is that if the first step leads to incorrect matchings, the other step is likely to fail. Related to this work, Koller *et al.* [59] represent the 3D model with line edges, which are defined by length, the coordinates of the middle point and the orientation angle. Pose recovery is done by minimising a function based on the Mahalanobis distance [82] between all model segments and all data segments, using the Levenberg–Marquardt algorithm (Levenberg [73], Marquardt [84]). These approaches tend to be unreliable, as they rely heavily on an accurate edge extraction step.

Harris' alternative method RAPiD, begins by generating the prediction view instead, predicting the position of the higher contrast model edges. By 1D search along the cardinal direction from *control points* on these edges, the algorithm finds the most prominent edge in the image (see Fig. 2.1). The distances between each control point and the closest image edges contribute collectively to a value for the a

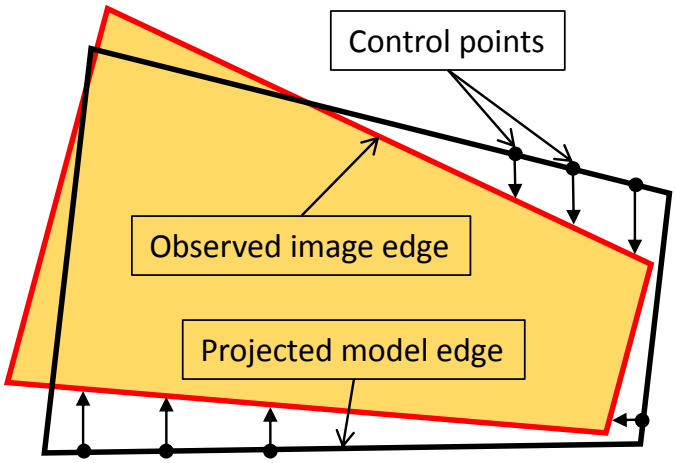



**Figure 2.1:** RAPiD control points and distances from projected to image edges.

pose correction, which is recovered using least-squares. The selection of the closest edge can also be augmented with a culling step, which ignores weak edges. To determine feature visibility, a precomputed table of visible edges is used.

Several improvements have been proposed to RAPiD. For example, Armstrong and Zisserman [8] group control points into robust primitives (lines, conics, etc.), then use RANSAC (see Fischler and Bolles [40]) to filter the edges detected around each control point. These primitives can also be eliminated from the pose update step. A greedy approach is used: the primitives are eliminated one by one and the pose correction is computed from the remaining ones. If the pose correction leads to a smaller projection error, the primitive is eliminated. Another approach is that of Drummond and Cipolla [39] where the influence of the control points in the pose optimization is weighted inverse proportionally to the number of edge strength maxima visible within the search range around the control point. The selection process for the visible edges is also improved by replacing the precomputed indexing table with a 3D rendering stage. Drummond and Cipolla [39] also replace the original least squares approach with a weighted least squares one. Fig. 2.2 shows a set of example results from Drummond and Cipolla [39].

Another variation improvement was introduced by Kollning and Nagel [61]

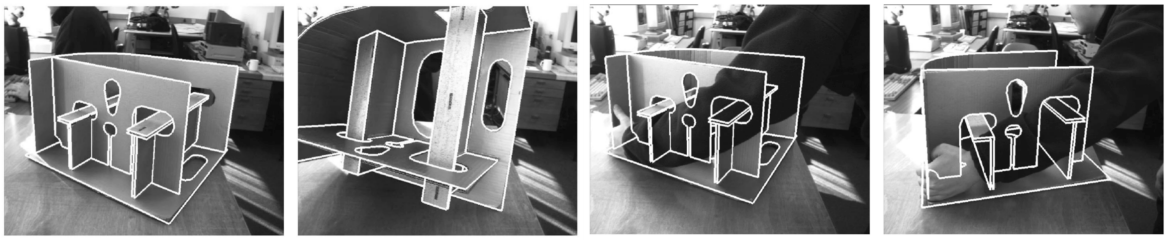


**Figure 2.2:** Example results from Drummond and Cipolla [39].

who proposed removing the edge detection step. Instead, the model projection is fitted the to the image gradients directly. Theoretically this method avoids the loss of information associated with an edge detection step; in practice, it leads to a much smaller basin of convergence.

All the above mentioned methods ignore the inherent multimodality of the image feature-pose mapping *i.e.* multiple object poses lead to similar image features. To solve this problem one solution is to keep track of multiple poses, rather than a single best one. One way of doing this is to use a particle filter. This approach was introduced to computer vision by Isard and Blake [53] and used in the context of 3D tracking by, among others, Vacchetti *et al.* [139] and Klein and Murray [57]. One problem with particle filters is that they require a large number of particles to produce accurate results.

Regardless of the approach used, all edge based methods are susceptible to motion blur, occlusions and cluttered background, as these lead to the corruption of image edges. Furthermore, as many of these methods rely on line segments for matching, they are also often limited to simplistic "well-carpentered" 3D models, so cannot work with arbitrarily shaped objects.

### 2.2.2 Point-based methods

As with edge based methods, point based methods work by matching a set of features from the image with a set of features on the model. The difference is that,

**Figure 2.3:** Example results from Skrypnyk and Lowe [130].

instead of edges, the features they use are interest points. These tend to be more invariant to disturbances (such as changes in scale, viewpoint and illumination), and therefore yield better results. Also, often, they do not require any pose initialization.

Early examples of point-based methods are Uenohara and Kanade [138] and Ravela *et al.* [110]. Rather than keeping a 3D model of the object, these methods use pose-dependent templates images. Patches are selected from both the real and the template image and are matched using a combination of normalized cross correlation and steerable filters (for invariance to translation and rotation). Such methods do show good results, but are not practical since they require a manual selection of the features.

The rise of effective and automatically discoverable descriptors for point features (*e.g.* Lowe [80] or Lepetit *et al.* [72]) has seen real-time methods for point-based model-based tracking become more practical, as shown by, among others Skrypnyk and Lowe [130], Lepetit *et al.* [72] and Ozuysal *et al.* [99]. These articles are exponents of a related methodology, called tracking by detection. The template images of Uenohara and Kanade [138] and Ravela *et al.* [110] are replaced by mapping relating feature points to the pose of the object. At run time, features are extracted from the image and matched to entries in the database.

In Skrypnyk and Lowe [130], the offline phase consists of building a sparse 3D model of the target object, using multiple views and SIFT features (Lowe [80]). Online, at every frame, SIFT features are first extracted. Next they are matched to

the ones from the 3D model using the Best-Bin-First algorithm (Beis and Lowe [11]), resulting in a collection of 2D to 3D correspondences. The final pose is obtained using a combination of RANSAC and Levenberg-Marquardt. The system was too slow for real time performance, running at an average of 4 fps (in 1997). Fig. 2.3 shows example results from Skrypnyk and Lowe [130]. A related method is that of Lepetit *et al.* [72], where a classifier based on randomized trees is used for feature matching. RANSAC is again used pose recovery. Also, unlike Skrypnyk and Lowe [130], both accurate 3D geometry of the object and a few images (which are then used to texture the 3D geometry) are assumed to be known. The requirement for an accurate 3D model is removed in Ozuysal *et al.* [99], where the texture and geometry of the object is computed offline using bundle adjustment, with an ellipsoid as an initial shape approximation.

Assuming a well textured object, point-based methods have been shown to have good accuracy and robustness to occlusions and changes in illumination. Real time performance has been achieved in constrained scenarios. However, issues, such as their inability to cope with motion blur or untextured objects, still exist.

### 2.2.3 Region and template matching-based methods

Region-based methods such as Rosenhahn *et al.* [117, 123] and Dambreville *et al.* [37] seek alignment of the projection of the occluding boundaries of the object with a regional segmentation of the image. This is done by maximizing the discrimination between a foreground and a background region (with known image statistics), with respect to the 3D pose of the known 3D model.

The curve separating the foreground and the background regions can be represented either *explicitly* or *implicitly*. A conventional explicit representation is to use a collection of equally spaced points, denoted by their image coordinates $(x, y)$. Curves can however breakup, merge, move or disappear during the course of their

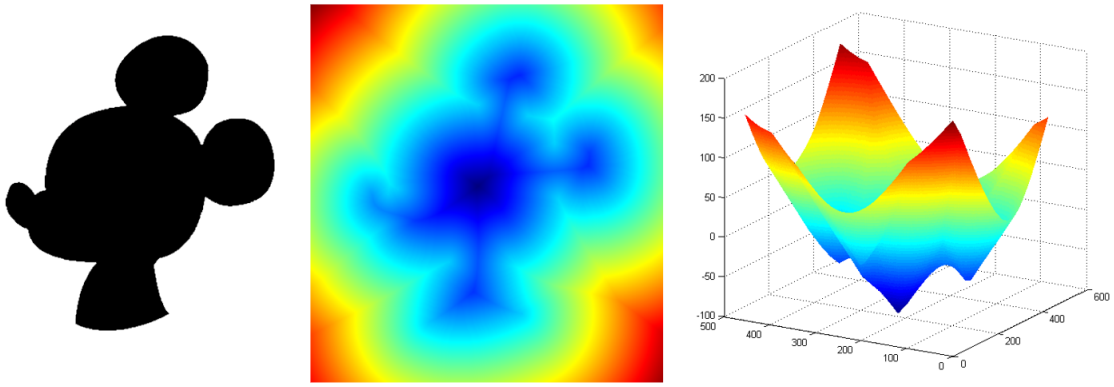



**Figure 2.4:** Example level set function embedding a silhouette. (a) the embedded silhouette. (b,c) 2D and 3D view of the SDF embedding function.

evolution. Using the explicit representation means that complicated methods have to be developed to model such behaviors. This is not required when using implicit representation, for example level sets (Osher and Sethian [98]), as they can handle all these topological changes very easily. A 2D level set function $\Phi$ is a Lipschitz continuous function, implicitly defining a curve as its zero level. That is to say that, for a 2D closed curve $\mathbf{C}$, we can formally write $\mathbf{C} = \{(x, y) \in \mathbb{R}^2 | \Phi(x, y) = 0\}$. This is called the implicit representation of the curve $C$. Often, a subset of these implicit functions, namely *signed distance functions (SDF)* is used (see Fig. 2.4). An SDF is a level set function for which $\Phi(\boldsymbol{x}) = -d(\boldsymbol{x}), \forall \boldsymbol{x} \in \Omega_f$ and $\Phi(\boldsymbol{x}) = d(\boldsymbol{x}), \forall \boldsymbol{x} \in \Omega_b$, where

$$d(\boldsymbol{x}) = \min_{\boldsymbol{x}_c \in \mathbf{C}} |\boldsymbol{x} - \boldsymbol{x}_c| \ . \tag{2.1}$$

Regions were first used by Rosenhahn *et al.* [117], where an infinite dimensional active contour is adapted in a single iterative step, in two stages: first the contour, represented by a zero level-set of a 2D embedding function with a shape term (to encourage similarity between the segmentation and the expected silhouette of the object given the current pose), is evolved to find a segmentation, in the expectation that the contour will then match the projection of the occluding contour of the

**Figure 2.5:** Example results from Gall *et al.* [46].

3D object. Second, each point on the contour is back-projected to a ray and the pose is found that best satisfies the tangency constraints that exist between the 3D object and these rays. This two-stage iteration places only soft constraints on the evolution of the contour *i.e.* the minimization of the energy function is done in an infinite dimensional space, rather than in the space of possible contours. Furthermore the 2D-3D pose matching in the two-stage iteration does not have a large basin of convergence, which will lead to tracking failure when the object has a large displacement. Further extensions to the method are proposed by Gall *et al.* [46], where an image synthesis stage is added, and by Gall *et al.* [20], where the region statistics are augmented with SIFT features and optical flow. An example result is shown in Fig. 2.5.

Level sets are not used by Dambreville *et al.* [37]. Rather, this work is based on an energy function summing two integrals – one over the foreground, one over the background. Here a direct minimization over the pose parameters is proposed, by differentiating these integrals with respect to the pose parameters.

Regions are less disrupted by occlusions, clutter and motion blur than edge-based methods, making them generally more reliable than both edge and point-based ones. Furthermore, these techniques have been shown to work with arbitrarily shaped objects. There are two main failure cases, namely they fail when

the pose-silhouette mapping is ambiguous (*i.e.* when multiple poses project to the same silhouette) and when the foreground and background image statistics are too similar or are corrupted.

## 2.3 3D model acquisition

We categorize the methods in early literature for 3D model acquisition into two sets: *Shape from X methods* and *Structure from Motion-based methods*. Further discussion on recent methods and the state of-the-art in real-time dense 3D model reconstruction is provided in Chapter 4.

### 2.3.1 Shape from X methods

Early works on vision-based 3D reconstruction comprise methods that recover 3D shapes from only visual cues on image. The study of how shape can be inferred from a single cue X is called *shape from X*, and examples of the various cues that can be used include shading, focus, contour and silhouette.

The problem of recovering the shape of a surface from the variation of intensity of image is known as *shape from shading* (Horn and Brooks [51], Zhang *et al.* [148]). The method assumes that the surface under consideration is of a uniform albedo and reflectance, and that the light source directions either known or can be calibrated by the use of a reference object. Under the assumption of distance light source and observer, the variation in the image intensity becomes a pure function of the local surface orientation

$$I(x,y) = R(p(x,y),q(x,y)), \tag{2.2}$$

where $(p,q) = (z_x, z_y)$ are the depth map derivatives and $R(p,q)$ is the *reflectance*

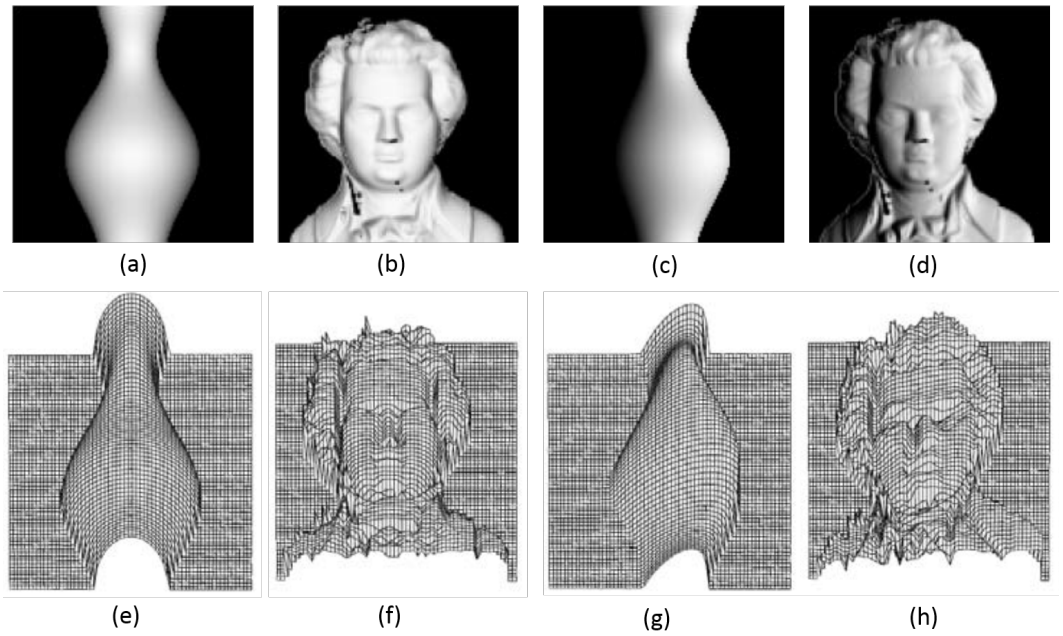



**Figure 2.6:** Synthetic shape from shading from Zhang *et al.* [148]. Shaded images, (a,b) with light from in front and (c,d) with light the front right; (e∼f) corresponding shape from shading reconstructions.

*map*. The surface of the object is recovered either by estimating the depth map derivatives then integrating them, or by directly minimizing the discrepancy in the image formulation in Eqn. 2.2. In practice, however, surfaces other than plaster casts are rarely of single uniform albedo. The shape from shading method therefore needs to be combined with other techniques to achieve an accurate depth map.

Another strong cue for object depth is the degree of blur, which increases as the object's surface moves away from the camera's focusing distance. Unlike many vision-based techniques that estimate 3D surface by using a pin-hole camera model, in *shape from focus* methods, real aperture camera models are used. A scene is modelled as a smooth opaque Lambertian surface, and attached to the surface is a texture $r$. Then the image intensity $I(y)$ at a pixel $y$ is usually modelled as a

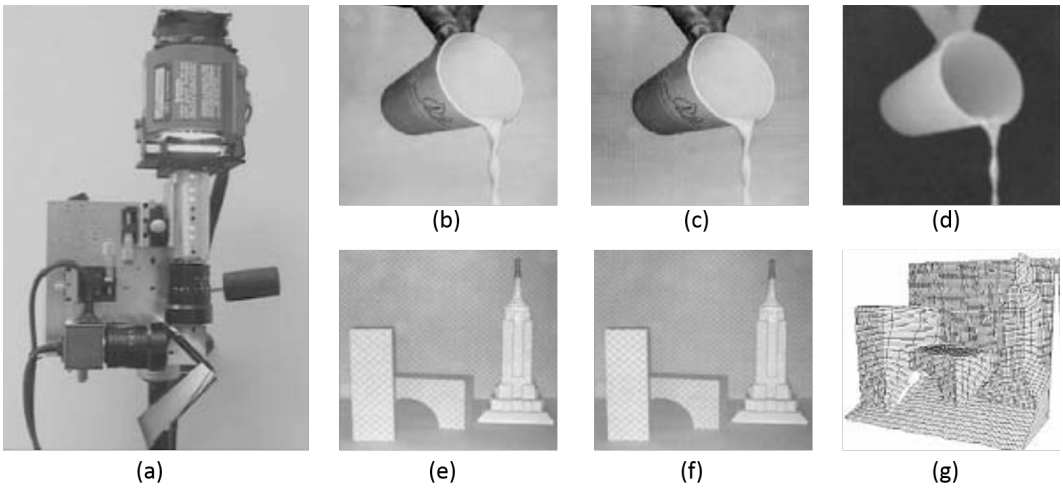



**Figure 2.7:** Real time depth from defocus from Nayar *et al.* [91]. (a) the real-time focus range sensor, which has a prism that splits the image into two CCD sensors and an edged checkerboard pattern illuminated by a Xenon lamp (top); (b,c) input video frames from the two cameras along with (d) the corresponding depth map; (e,f) two frames (you can see the texture if you zoom in) and (g) the corresponding 3D mesh model.

convolution between the Gaussian defocus kernel $h_{u,d}$ and the texture $r$ [24]

$$I(y) = (h_{u,d} * r)(y) \,, \tag{2.3}$$

$$h_{u,d} = \frac{1}{\pi\sigma_{s,u}^2} \exp\{-\frac{(y-x)^\top (y-x)}{2\sigma_{s,u}^2}\} \,, \tag{2.4}$$

where $u$ is the distance between the lens plane and the plane in focus in the scene and $\sigma_{s,u}$ is called the blurring radius which depends on the lens setting.

For example, a real-time depth from focus system was introduced in Nayar *et al.* [91]. Fig. 2.7 shows the system. A real-time focus range sensor which employs two image chips at slightly different depth sharing a common optical path is used as the input device. An active illumination system was used to project a checkerboard pattern from the camera. The system produces high-accuracy real-time depth maps for both static dynamic scenes.

Both shape from shading and shape from focus methods provide only depth measure instead of a whole solution for 3D model acquisition. The shape from

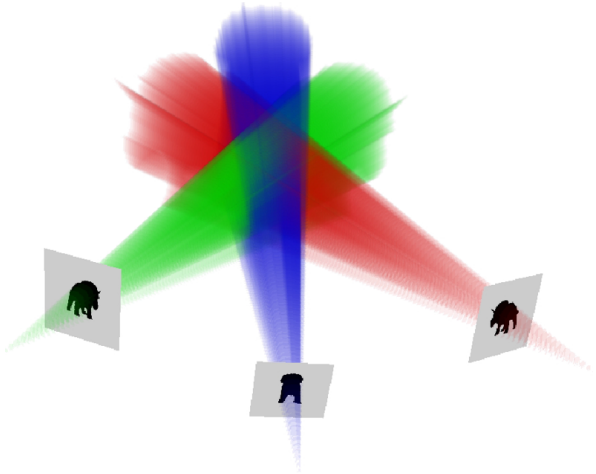


**Figure 2.8:** The intersection of silhouette cones defines an approximate geometric representation of an object call the visual hull. It contains the actual object and it has consistent silhouettes. Images from Matusik *et al.* [87]

silhouette (first introduced in Baumgart [9]) method provides a whole solution for obtaining 3D models from multiple images. The *shape from silhouette* method is based on an approximate geometric representation of the depicted scene known as the visual hull (see Fig. 2.8). A visual hull (Laurentini [69]) is constructed by using the visible silhouette information from a series of reference images to determine conservative shell that progressively enclose the actual 3D shape. In some sense, the visual hull carves away spatial regions where the object 'is not'. A visual hull always contains the object. Moreover, it is an equal or tighter fit than the object's convex hull.

A common method used to convert silhouette contours into visual hulls is volume carving (Curless and Levoy [35], Seitz and Dyer [127]). This method removes unoccupied regions from an explicit volumetric representation. All voxels falling outside of the projected silhouette cone of a given view are eliminated from the volume. The process is repeated for each reference image, resulting in a quantized representation of the virtual hull according to the volumetric grid. In Matusik *et al.* [87], a view-dependent, image-based visual hull system was introduced. Using epipolar geometry, the intersection of 3D rays in computing the virtual hull is

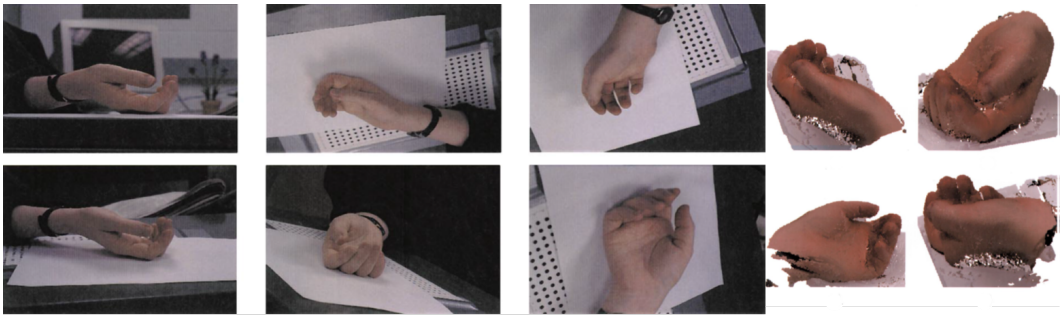



**Figure 2.9:** Results from Space Carving algorithm. Image from Kutulakos and Seitz [65]. (a) Six out of one hundred photographs of a hand sequence. (b) Reconstruction of a hand.

reduced to 2D ray intersection on the epiploar plane. The number of rays being sampled is also limited to the number of pixels of the desired image, resulting in a view-dependent visual hull representation. The system yields real-time performance for 3D shape reconstruction and visualization with a calibrated multi-camera setup.

The limitation of shape from silhouette method is obvious. First, the visual hull is not guaranteed to be the same as the original object, since the concave surface regions can never be distinguished using silhouette information alone. And second, in practice, the visual hull is approximately constructed using only a finite number of views. The number of calibrated views will limit the level of reconstruction accuracy.

Instead of carving empty space, in Cipolla and Blake [29], the authors recover a parameterised surface directly from deforming silhouettes (also know as *shape from contour*). The mathematics of perspective projection and differential geometry is used to analysis deforming silhouettes (apparent contour) of a curved surface. Assuming view motion is known, the local surface curvature along the corresponding contour generator is computed using a spatialtemporal parameterisation of image-curve motion. The approach achieve real-time reconstruction of 3D surface, however, the assumption of known view motion and parameterisable surface is the

fundamental limitation.

Another set of reconstruction methods that are based on the idea of removing spacial regions where the object 'is not' is the Space Carving method (Kutulakos and Seitz [65], Broadhurst [19]). Given a set of input image, the goal of the Space Carving method is to find the *photo-consistent shape* (*i.e.* the shape that reproduces all input images). In the visual hull method, this photo consistency constraint is only exploited in the background region. In Kutulakos and Seitz [65], the authors also explore the photo consistency constraints from non-background pixels using the Lambertian radiance model in the scene. The resulting reconstruction is a subset of visual hull (also called the photo hull), but can contain concavities. In the Space Carving framework, space is also represented by an array of voxels. At each iteration, the algorithm selects a voxel and projects it into all the image where it is visible. It then ask the question: could this pixel be part of the objects? This question is tested by tracking the RGB variance of the corresponding pixels in all images. If the the variance is higher than a threshold, then the voxel is removed. The algorithm repeats until it has exposed the 3D shape of the scene. An example of the Space Carving algorithm is shown in Fig. 2.9.

A crucial part of the Space Carving algorithm is the consistency criteria, which is the mechanism that decides whether a voxel should be kept or discarded. Originally, a single threshold of the variance of RGB value is used, leading to incorrect removal of voxels in presence of image noise. In Broadhurst [19], the authors introduced a probabilistic framework for Space Carving, which does not rely on any global threshold parameter. Instead, the probability of a voxel being visible in each view is first computed then marginalized into a single per-voxel probability of visibility, which is used for the visualization of the model using the $\alpha$-test. This probabilistic framework does not carve holes in the model.

The limitation of Space Carving-based methods is that it requires a accurately

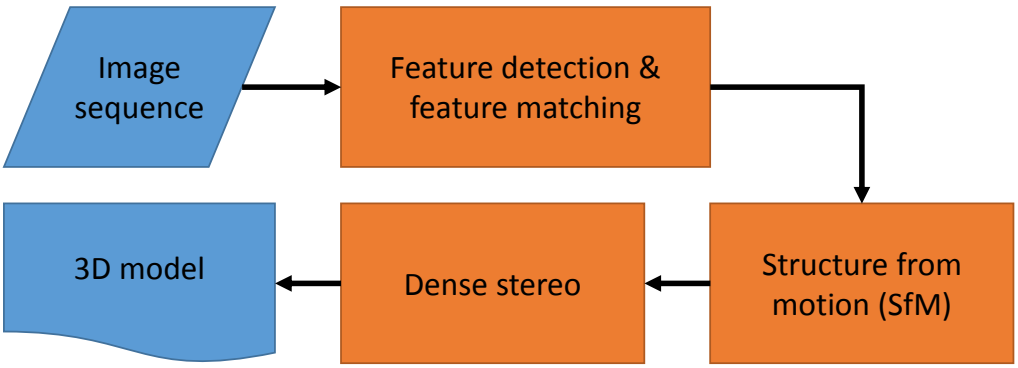



**Figure 2.10:** Structure from motion (SfM)-based model acquisition pipeline

multi-view setup, also, the photo-consistency assumption does not hold for non-Lambertian objects.

### 2.3.2 Structure from motion-based methods

Methods in the Shape from X category either works with a single frame input, or requires a calibrated multi-view setup to reconstruct the 3D shape. The structure from motion (SfM) -based methods allows the reconstruction of 3D shape with a single moving camera. The camera poses and the 3D shape are simultaneously recovered within a single pipeline. Examples of SfM-based method that provides an end-to-end solution to 3D model acquisition includes, among others, Beardsley *et al.* [10], Fizgibbon and Zisserman [42] and Pollefeys *et al.* [103]. Such SfM methods generally follow a similar framework (shown in Fig. 2.10), differences between methods are within each module.

Given a sequence of image as input, a set of feature points is first detected on each frame. Rotation and scale invariant descriptors like SIFT (Lowe [80]) and FAST (Rosten and Drummond [119]) are the most commonly used feature descriptors. Correspondences between the detected features in different frames are then matched based on the descriptor. The second task is to recover the motion of the camera and the structure of the scene. The motion information includes the 6 Dof

pose of the camera, and the intrinsic parameter of the camera. The structure information is represented by the 3D location of the matched features. Given the feature correspondences between multiple views, the geometric constraints (*i.e.* the fundamental matrix or focal tensors) among views can be established using multiple view geometry (Hartley and Zisserman [49]). The projection matrices that represented the motion information then can be recovered. Finally, 3D coordinates of features, *i.e.* structure information, can be computed via triangulation. The reconstruction process usually includes local updates (feature matching and structure reconstruction between local frames), which can lead to inconsistency and accumulated errors in the global result. A global bundle adjustment (Triggs *et al.* [135]) optimization step is then performed to produce a globally consistent result.

The structure created from SfM is very discrete and sparse, not enough for the purpose of visualization or more specifically, the purpose of model-based 3D tracking. Multiple view stereo method is used for obtaining a dense 3D point cloud as the model representation. Generally, dense stereo methods follows two steps: *rectification* and *stereo mapping*. The first exploits the epipolar constraint to prepare the data for the second one by aligning a corresponding pair of epipolar lines alone the same scan line of image thus all corresponding points will have the same $y$-coordinates in two images. This makes the second task, roughly search and match over the whole image faster. Stereo mapping is the task of establishing a dense matching between points of different calibration views, then the matched points are triangulated into model coordinates to produce the final dense point cloud model. There is a vast literature of multi-view stereo methods, however, it is beyond the scope of this thesis, a review of various dense stereo methods can be found in Seitz *et al.* [126].

Depending on the final 3D model representation, the whole reconstruction pipeline may include a 3D modeling step to produce a textured mesh model. Although the SfM-based method provides a complete end-to-end solution for 3D

**Figure 2.11:** Example result of SfM-based reconstruction. Image from Xiao [146]

model acquisition, the limitation is the speed. The feature detection and matching and the SfM step can achieve real-time performance, and they are widely used in Simultaneous Localization And Mapping (SLAM) systems. However, in order to obtain accurate depth map for dense reconstruction, the dense stereo step can only use offline processing. Furthermore, in the SfM step, in order to recover the camera motion, a stationary scene is required. This constraint makes the method not suitable for the purpose of construction moving objects in uncontrolled environment.

## 2.4 Multi-object tracking

Multi-object tracking in 2D video has been well studied over the last decades. Classical approaches to multi-target tracking either assume point-like targets and track their 2D image locations or represent targets as 2D regions of interest (RoI) and track the 2D shapes or 4-DoF poses of bounding boxes. The core problem in multi-object tracking is the data association problem when multiple objects move close together. When objects have distinctive appearances, instantiating several single-body tracking may solve the tracking problem. However, when the targets are "identical", in the sense that the same model is used to describe each target, the task become much more challenging.

In early work, Fortmann *et al.* [43] proposed joint probabilistic data association

filtering (JPDAF) for associating data with targets in a cluttered multi-target environment. Joint posterior association probabilities are computed for multiple targets in Poisson clutter. The method is able to track heavily interfering targets. However, JPDAF fails when the data models are non-linear and non-Gaussian. Sampling-based method or Monte Carlo-based methods were introduced to tackle this. The distribution over the state space is represented as a set of discrete samples, which are easy to implement and can model complex non-linearity. In Schulz *et al.* [125], JPDAF was implemented as a particle filter for the application of tracking people from a mobile robot platform using 2D radar data. Occlusion between objects is explicitly handled using an "occlusion map" that contains, for each position in the surrounding of the robot, the probability that the corresponding position is not visible during data association. In MacCormick and Blake [81], the author introduced "partitioned sampling", a method of using particle filter with multiple objects.This means dividing the state space into several partitions followed by an appropriate weighted resampling operation. The method significantly mitigates the curse of dimensionality, and occlusion is handled using an exclusion principle that prevents a single piece of image data independently contributing to similar hypotheses for different targets. Khan *et al.* [55] use a Markov Chain Monte Carlo (MCMC) based particle filter to incorporate motion priors over target interactions. The coalescence of multiple trackers is handled by instantiating repulsion forces between trackers when their object hypotheses get close.

The well established Bayesian filtering framework has not been used for real-time full DoF tracking of multiple objects in 3D. We believe the reasons are two-fold. First, handling the high dimensionality of state space in multi-object tracking is non-trivial. And second, the simplified model assumptions of points, bounding boxes or region of interest is insufficient for the representation of full DoF poses in 3D. We will further discuss recent work on real-time 3D multi-object tracking in Chapter 5.

# 3

# A Generic Probabilistic Framework for Model Fitting with Depth Data

*This chapter presents a probabilistic model for real-time model-based tracking of 3D objects, which leads to a unified energy minimization framework for generic model fitting problems with a depth cameras. 3D level-set embedding functions are used to represent object models implicitly and a novel 3D chamfer distance based energy function is used for matching. The energy function is minimized by adjusting the back-projection from image domain to object coordinates, which can be parametrised differently according to specific applications. The proposed energy function is fully variational and probabilistic, and thus can be efficiently solved by second-order gradient-based optimization methods. The approach is applied to three tasks, namely real-time 3D tracking, camera calibration and 3D point cloud modelling with primitive shapes. An earlier version of this work was published by Ren and Reid [114]*


**Figure 3.1:** Illustration of the concept of control object tracking. Image (a) shows the original colour image, (b) shows the depth image with tracking result overlaid, (c) visualizes a sword on the colour image with the recovered pose of the control object.

## 3.1 Introduction

The first vision module required for the system for rehabilitation is a method for model-based 3D tracking that can recover the 6-DoF pose of a known "well captured object" (also known as control object) in real-time. The patient is expected to move the control object in front of a RGB-D camera as instructed by the rehabilitation game. Tracking the control object must enable not only the graphical visualization of an interesting virtual object in the rehabilitation game, but also permit accurate measurement of how well the patient performs in the rehabilitation exercise. An example of the control object is shown in Fig. 3.1 (a) and the visualization of the virtual object as a sword overlaid given the actually colour image is shown in Fig. 3.1 (c). In order to solve this 3D tracking problem, we develop a probabilistic model that allows the optimal 6-DoF pose to be found by maximizing the posterior probability of the pose given the depth image input. As we show, the resulting energy function can be applied to other tasks, including camera calibration and point cloud modeling with primitive shapes. The method run effectively in real time on a CPU but using a further GPU implementation it is able to handle larger object and/or very dense depth maps.

Our method is a model-based approach in which the 3D shape of the object is represented implicitly as the zero-level set of a signed distance function (SDF)

computed in the voxelized space around the object. As a natural extension of the 2D Chamfer distance matching method [133], Fitzgibbon [41] showed that SDFs are an effective alternative to the iterative closest point (ICP) algorithm [12] for fast and robust 2D to 2D and 3D to 3D point cloud registration. The most significant benefit is that the SDF-based formulations do not require explicit point to point correspondences, and further, by placing a upper limit on the per-point signed distance, the cost function is made intrinsically robust without explicit segmentation. In practice, as sketched in Fig. 3.2, we discretize 3D locations into a number of voxels surrounding the object. Voxel locations with positive signed distances map to the inside locations of the object and *vice versa*, and the surface is defined by the zero-level. Unlike other hypothesis-and-test or sampling based methods (*e.g.* Ueda [63], Oikonomidis *et al.* [52], Choi and Christensen [28] and Wuthrich *et al.* [145]), we optimize a fully probabilistic and variational energy function without the need to establish any explicit point correspondence between the 3D model and the observation. We take advantage of the gradients of the SDF to guide the search for the optimal pose, so the optimization problem can be efficiently solved by second-order gradient-based methods like Levenberg-Marquardt (LM) [73, 84]. And because a back-projection scheme is used, rather than 'rendering' the object into observation domain, no z-buffering or depth ordering is required for the energy function evaluation, making the method inherently more suitable for the parallelization on a GPU.

The remainder of this chapter is structured as follows: Section 3.2 reviews several closely related works in recent five years, then in Section 3.3, we establish the geometry involved and our notational conventions. The core of the chapter is in Section 3.4 where we develop the graphical model and the mathematical foundations. Section 3.5∼3.8 specialize the approach to three applications in detail and give both qualitative and quantitative results. We further discuss the implementation and performance of our method in 3.9 and draw conclusions in Section 3.10.

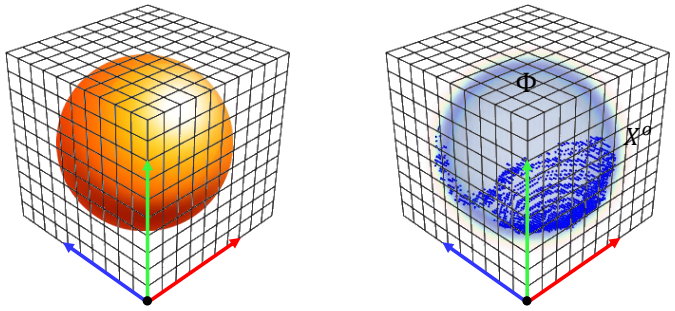



**Figure 3.2:** (Left) An object is defined in a voxelized cube. (Right) Its SDF as embedding function is also defined in object coordinates with the same voxelization. We use $200 \times 200 \times 200$ voxels in this work.

## 3.2 Related work

Early work that uses sparse image features or image regions for 3D tracking has been reviewed in Chapter 2. Here we review some state-of-the-art dense 3D tracking methods using color imagery or RGB-D imagery as input. Recent approaches to real-time tracking of 3D objects can be categorized into two major categories: **gradient-based methods** and **sampling-based methods**.

**Gradient-based methods**, the category our method belongs to, involve the minimization of an objective function, which is characterized by having partial derivatives with respect to the pose parameters. These approaches are able to take the advantage of the well established optimization techniques to efficiently estimate the optimal pose. A common algorithm deployed by gradient-based methods is Iterative Closest Point (ICP) [12]. In the ICP algorithm, point-to-point correspondences between the model and the observed point cloud are first established, then the relative transformation between the model and the point cloud is iteratively refined by minimizing the sum of L2 distances between the corresponding point pairs. In Held *et al.* [50], the authors input RGB-D imagery from the Kinect sensor as input data. Rigid 3D objects (3D puppets) are represented by coloured point clouds and the real-time tracking is accomplished by using the ICP algorithm to

align the observed point cloud and the model point clouds. The system yields robust tracking in presence of occlusion from hands but it heavily relies on an appearance-based pre-segmentation phase to remove the ground surface and the occlusion introduced by the hand.

In the work of Newcombe *et al.* (KinectFusion) [92], the authors simultaneously track and reconstruct a static 3D scene in real-time. The surface of the reconstructed-and-tracked scene is represented by a Truncated Signed Distance Function (TSDF). At each frame, ray-casting from the camera centre to the object frame is used to create a point list from the TSDF. The pose recovery is achieved also by using ICP to align the model point list with the observed depth image. However, a requirement when tracking with KinectFusion is that the entire scene is tracked as a single entity related to the camera. In the context of object tracking, this requires a single object to be segmented from the background, which is a condition that is obviously violated when tracking a moving object with arbitrary motion in a static scene. Our method differs from the ICP-based methods by using a full Signed Distance Function (SDF) to represent the 3D objects and directly use the gradients of the SDF to guide the optimization for pose, without the need to explicitly establish point correspondences.

In Prisacariu and Reid [106], the authors use a region-based method which does not require point correspondences neither. The object model is first projected onto the image domain and a 2D SDF is computed from the silhouette of the projection. The pose of the 3D object is recovered by evolving the 2D SDF in the image domain with respect to the pose to maximize the foreground-background discrepancy of the appearance. Due to the heavy computation of the projection of the 3D object model, the method can only achieves frame-rate performance with GPU implementation. Furthermore, the ambiguity of pose when projecting 3D shapes into 2D image domain is a fundamental limitation of the method.

**Sampling-based methods**, on the other hand define an objective function that describes the discrepancy between the observed visual cues and the expected ones generated by the pose hypothesis. The objective functions are evaluated by "rendering" the object model into the observation domain and computing the differences between the generated and the observed visual cues. The partial derivatives of the objective function are either too expensive or not possible to compute, so this set of methods typically rely on the evaluations of the objective functions at many positions in the hypothesis space. For example, in Oikonomidis *et al.* [52] the author uses Particle Swarm Optimization (PSO) to track an articulated hand, and in Kyriazis and Argyos [67] the authors use the same PSO technique to track the interaction between a hand and an object. Both systems achieve real-time performance by exploiting the power of GPU, however, the level of accuracy that can be achieved by PSO is less thoroughly studied and theoretically justified. Other works in this category mostly use the well-studied particle filter to solve the tracking problem and differ only in the types of visual feature that is used. In Ueda [137], the objective function describes the differences between the rendered and the observed depth map and in Wuthrich *et al.* [145], the authors also model the per-pixel occlusion label at the same time towards more robust tracking in presence of occlusion. In Azad *et al.* [100], 2D image edges and the 2D edges of the rendered model are matched against each other. In Choi and Christensen [27] both 2D edges and 2D image key-points are used, and in their more recent work Choi and Christensen [28], the authors use RGB-D imagery input and added photometric, 3D edges and 3D surface normals into the likelihood function of each particle state. All the above methods can achieve real-time performance with GPU implementations, but due to the computationally expensive nature of the particle filter, they are limited by the number of particles that can be deployed for real-time performance. In contrast, our method relies on highly efficient representation and optimization, which leads to real-time performance on a single CPU.

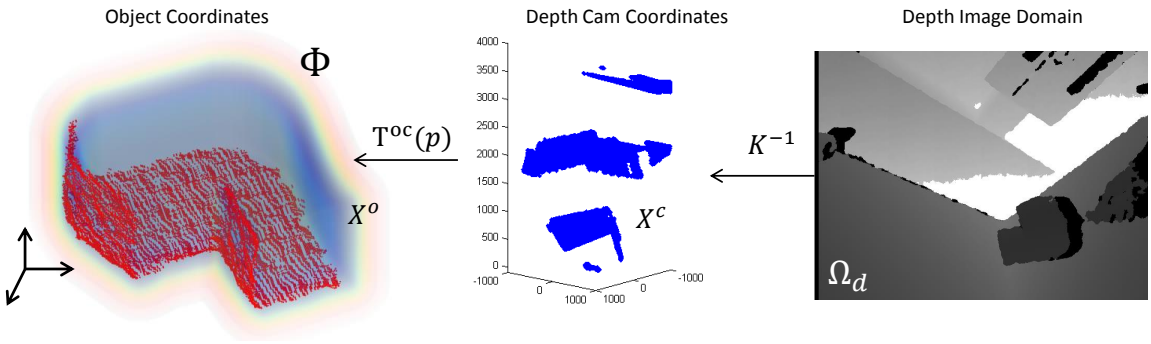



**Figure 3.3:** Illustration of the scene geometry, from right to left: A known object in the depth image, the points in depth camera coordinates and wrapped around the object surface in object coordinates during tracking.

## 3.3 Image and scene geometry

The basic scene geometry is illustrated in Fig. 3.3. Each pixel location $(x, y)$ in the image $\Omega_\mathrm{d}$ of a depth camera holds the depth value $Z(x, y)$ of the associated 3D scene point $\boldsymbol{X}^\mathrm{c} = (X, Y, Z)^\top$ referred to Cartesian coordinates centred on the depth camera's optical centre. The projection is perspective, giving the image position $\boldsymbol{x} = (x, y, 1)^\top$ as

$$\boldsymbol{x} = Z^{-1}\mathtt{K}[\mathtt{I}|\mathbf{0}] \begin{bmatrix} \boldsymbol{X}^\mathrm{c} \\ 1 \end{bmatrix} . \tag{3.1}$$

Knowledge of the intrinsic calibration matrix $\mathtt{K}_{3\times3}$ of the camera allows the recovery of the scene as

$$\boldsymbol{X}^\mathrm{c} = \mathtt{K}^{-1} Z\, \boldsymbol{x} . \tag{3.2}$$

Knowledge of the extrinsic calibration between the depth and colour cameras — in the form $(\mathtt{P}_{3\times3}, \boldsymbol{q}_{3\times1})$ defined later in Section 3.5 — allows the corresponding image point $\boldsymbol{x}' = (x', y', 1)^\top$ to be found in the colour image $\Omega_\mathrm{c}$ as

$$\lambda \boldsymbol{x}' = \mathtt{P} Z \boldsymbol{x} + \boldsymbol{q} , \tag{3.3}$$

where $\lambda$ is a per-point scale. The colour 3-vector can then be copied to the depth image as $\boldsymbol{c}(x, y) \leftarrow \boldsymbol{c}(x', y')$. The pixel at $\boldsymbol{x}$ in the combined image $\Omega$ then provides

$(Z, \boldsymbol{c}^\top)$.

A pixel is the projection of a scene point that is either part of the background or on the surface of a specific object. An object's pose relative to the depth camera is represented by a 6-vector $\boldsymbol{p}=(\boldsymbol{T}^\top, \boldsymbol{r}^\top)^\top$ comprising of three translation parameters $\boldsymbol{T}=(t_x, t_y, t_z)^\top$ and three Modified Rodrigues Parameters (MRP) [128] as rotation $\boldsymbol{r}=(r_1, r_2, r_3)^\top$. The Euclidean transformation from the object coordinates to the camera coordinates $\mathtt{T}$, as a function of the pose $\boldsymbol{p}$ is defined as

$$\mathtt{T}(\boldsymbol{p}) = \begin{bmatrix} \mathtt{R} & \boldsymbol{T} \\ \boldsymbol{0}^\top & 1 \end{bmatrix} . \tag{3.4}$$

Here, $\mathtt{R}$ is the rotation matrix constructed from MRP $\boldsymbol{r}$

$$\mathtt{R} = \frac{1}{(1-\sigma_r)^2} \begin{bmatrix} 4r_1^2 - 4r_2^2 - 4r_3^2 + \omega_r^2 & 8r_1r_2 - 4r_3\omega_r & 8r_1r_3 + 4r_2\omega_r \\ 8r_1r_2 - 4r_3\omega_r & 4r_2^2 - 4r_1^2 - 4r_3^2 + \omega_r^2 & 8r_2r_3 + 4r_1\omega_r \\ 8r_1r_3 + 4r_2\omega_r & 8r_2r_3 + 4r_1\omega_r & 4r_3^2 - 4r_1^2 - 4r_2^2 + \omega_r^2 \end{bmatrix} , \tag{3.5}$$

where

$$\sigma_r = r_1^2 + r_2^2 + r_3^2 , \quad \omega_r = 1 - \sigma_r . \tag{3.6}$$

The Euclidean transformation $\mathtt{T}^{\mathtt{co}}$ relating points in the object and camera frames follows

$$\begin{bmatrix} \boldsymbol{X}^{\mathtt{c}} \\ 1 \end{bmatrix} = \mathtt{T}^{\mathtt{co}} \begin{bmatrix} \boldsymbol{X}^{\mathtt{o}} \\ 1 \end{bmatrix} . \tag{3.7}$$

In our formulation, we instead recover the pose using the back-projection matrix $\mathtt{T}^{\mathtt{oc}}$, which is simply the inverse of $\mathtt{T}^{\mathtt{oc}}$

$$\begin{bmatrix} \boldsymbol{X}^{\mathtt{o}} \\ 1 \end{bmatrix} = \mathtt{T}^{\mathtt{oc}}(\boldsymbol{p}) \begin{bmatrix} \boldsymbol{X}^{\mathtt{c}} \\ 1 \end{bmatrix} . \tag{3.8}$$

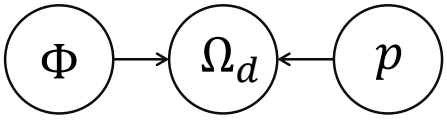



**Figure 3.4:** The graphical model for our model-fitting framework

## 3.4 Graphical model

Fig. 3.4 shows the proposed graphical model [60] from our model-fitting framework with depth data. The current depth image $\Omega_\mathrm{d}$ depends on the current object pose $\boldsymbol{p}$ and the 3D shape $\Phi$. We assume no motion model and estimate the pose of the object in each frame independently by maximizing the posterior probability of the pose given the current depth image and 3D shape, $P(\boldsymbol{p}|\Phi, \Omega_\mathrm{d})$. Following Bayes' rule, the posterior is proportional to the likelihood of the depth image and the prior of the pose

$$P(\boldsymbol{p}|\Phi, \Omega_\mathrm{d}) = \frac{P(\Omega_\mathrm{d}|\Phi, \boldsymbol{p})P(\boldsymbol{p})}{P(\Omega_\mathrm{d}|\Phi)} \,. \tag{3.9}$$

Assuming a single 3D object can have arbitrary 6-DoF pose we take the pose prior term $P(\boldsymbol{p}|\Phi)$ as uniform, and drop it. Assuming pixel-wise independence, the likelihood of depth image $P(\Omega_\mathrm{d}|\Phi, \boldsymbol{p})$ can be decomposed into the product of all per-pixel depth likelihoods

$$P(\Omega_\mathrm{d}|\Phi, \boldsymbol{p}) = \prod_{\boldsymbol{x}_i \in \Omega_\mathrm{d}} P(Z(\boldsymbol{x}_i)|\Phi, \boldsymbol{p}) \,, \tag{3.10}$$

where $Z(\boldsymbol{x}_i)$ is the depth value at 2D depth image location $\boldsymbol{x}_i$. For each 2D depth image location $[x_i, y_i]$ the per-pixel depth $Z_i$ is independently and randomly drawn from the likelihood distribution. The per-pixel likelihood $P(Z(\boldsymbol{x}_i)|\Phi, \boldsymbol{p})$ is

$$P(Z(\boldsymbol{x}_i)|\Phi, \boldsymbol{p}) = \frac{\exp\{\Phi(\boldsymbol{X}_i^\circ)/\sigma\}}{\sigma(\exp\{\Phi(\boldsymbol{X}_i^\circ)/\sigma\} + 1)^2} \,, \tag{3.11}$$

as based on the assumption that a depth value is more likely to be generated if the corresponding pixel can be back-projected near the surface (*i.e.* the zero level-set of

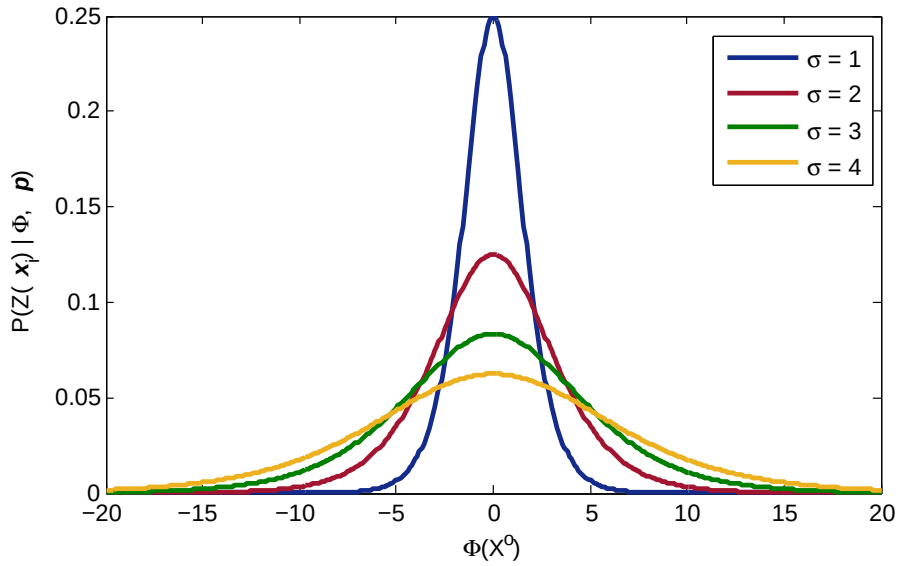



**Figure 3.5:** The form of the per-pixel likelihood function $P(Z(\boldsymbol{x}_i)|\Phi, \boldsymbol{p})$

the SDF) of the 3D shape in object coordinates. The likelihood is a robust function which has large gradients around the zero level of SDF and small gradients when $\Phi(\boldsymbol{X}_i^{\circ})$ is further away from the zero level-set. Here, $\boldsymbol{X}_i^{\circ}$ is the back-projection of $\boldsymbol{x}_i$ given the current pose following Eqn. 3.2 and Eqn. 3.8 and $\sigma$ determines the width of the basin of attraction. This likelihood function is plotted in Fig. 3.5. The energy function is written as a sum of log-likelihoods over all pixels

$$\mathcal{E} = \sum_{\boldsymbol{x}_i \in \Omega_{\mathrm{d}}} \log \left\{ \frac{\exp\{\Phi(\boldsymbol{X}_i^{\circ})/\sigma\}}{\sigma(\exp\{\Phi(\boldsymbol{X}_i^{\circ})/\sigma\} + 1)^2} \right\} . \tag{3.12}$$

As mentioned earlier, the optimization of the energy function of the form in Eqn. 3.12 can be achieved by using sampling methods. In our formulation, sampling in a high dimensional pose space is avoided by using second-order optimization. The energy function is parameterized in terms of a pose change $\boldsymbol{p}^* = (\boldsymbol{T}^{*\top}, \mathtt{R}^{*\top})^{\top}$ from the current pose, and its gradient found as

$$\frac{\partial \mathcal{E}}{\partial \boldsymbol{p}^*} = \sum_{\boldsymbol{x}_i \in \Omega_{\mathrm{d}}} \left\{ \left[ \frac{1}{P_{Z_i}} \frac{\partial P_{Z_i}}{\partial \Phi} \frac{\partial \Phi}{\partial \boldsymbol{X}^{\circ}} \right] \frac{\partial \boldsymbol{X}^{\circ}}{\partial \boldsymbol{p}^*} \right\} , \tag{3.13}$$

where $P_{Z_i} = P(Z(\boldsymbol{x}_i)|\Phi, \boldsymbol{p})$. The term in brackets in Eqn. 3.13, $\frac{\partial \mathcal{E}}{\boldsymbol{X}^\circ}$, are pre-computable for all voxel location

$$\frac{\partial P_{Z_i}}{\partial \Phi} = \frac{\exp\{\Phi/\sigma\}}{\sigma^2 \left(\exp\{\Phi/\sigma\}+1\right)^2} - \frac{2\exp\{2\Phi/\sigma\}}{\sigma^2 \left(\exp\{\Phi/\sigma\}+1\right)^3}\,, \tag{3.14}$$

and

$$\frac{\partial \Phi}{\partial \boldsymbol{X}^\circ} = \begin{bmatrix} \frac{\partial \Phi}{\partial X} & \frac{\partial \Phi}{\partial Y} & \frac{\partial \Phi}{\partial Z} \end{bmatrix}\,. \tag{3.15}$$

The derivatives of the SDF $\begin{pmatrix} \frac{\partial \Phi}{\partial X} & \frac{\partial \Phi}{\partial Y} & \frac{\partial \Phi}{\partial Z} \end{pmatrix}$ are computed trivially, using central finite differences. The incremental 3D transformation of the 3D point $\boldsymbol{X}^\circ$ follows

$$\boldsymbol{X}^\circ(\boldsymbol{p}, \boldsymbol{p}^*) = \mathtt{R}^* \boldsymbol{X}^\circ(\boldsymbol{p}) + \boldsymbol{T}^*\,. \tag{3.16}$$

At each iteration, the partial derivatives of the 3D points $\boldsymbol{X}^\circ = (X^\circ, Y^\circ, Z^\circ)^\top$ with respect to the pose update $\boldsymbol{p}^* = \begin{pmatrix} t_x^*, t_y^*, t_z^*, r_1^*, r_2^*, r_3^* \end{pmatrix}^\top$ are computed at identity (*i.e.* $t_x^* = t_y^* = t_z^* = 0,\ \ r_1^* = r_2^* = r_3^* = 0$)

$$\begin{aligned}
\frac{\partial \boldsymbol{X}^\circ}{\partial t_x} &= \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} & \frac{\partial \boldsymbol{X}^\circ}{\partial t_y} &= \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} & \frac{\partial \boldsymbol{X}^\circ}{\partial t_z} &= \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \\
\frac{\partial \boldsymbol{X}^\circ}{\partial r_1} &= \begin{bmatrix} 0 \\ -4Z^\circ \\ 4Y^\circ \end{bmatrix} & \frac{\partial \boldsymbol{X}^\circ}{\partial r_2} &= \begin{bmatrix} 4Z^\circ \\ 0 \\ -4X^\circ \end{bmatrix} & \frac{\partial \boldsymbol{X}^\circ}{\partial r_3} &= \begin{bmatrix} -4Y^\circ \\ 4X^\circ \\ 0 \end{bmatrix}\,.
\end{aligned} \tag{3.17}$$

Using the derivatives, Levenberg-Marquardt iterations are used to optimize for the pose update $\boldsymbol{p}^*$

$$\boldsymbol{p}^* = \left\{ -\left[\mathtt{J}^\top \mathtt{J} + \lambda \mathrm{diag}\left[\mathtt{J}^\top \mathtt{J}\right]\right]^{-1} \frac{\partial \mathcal{E}}{\partial \boldsymbol{p}^*} \right\}^\top \tag{3.18}$$

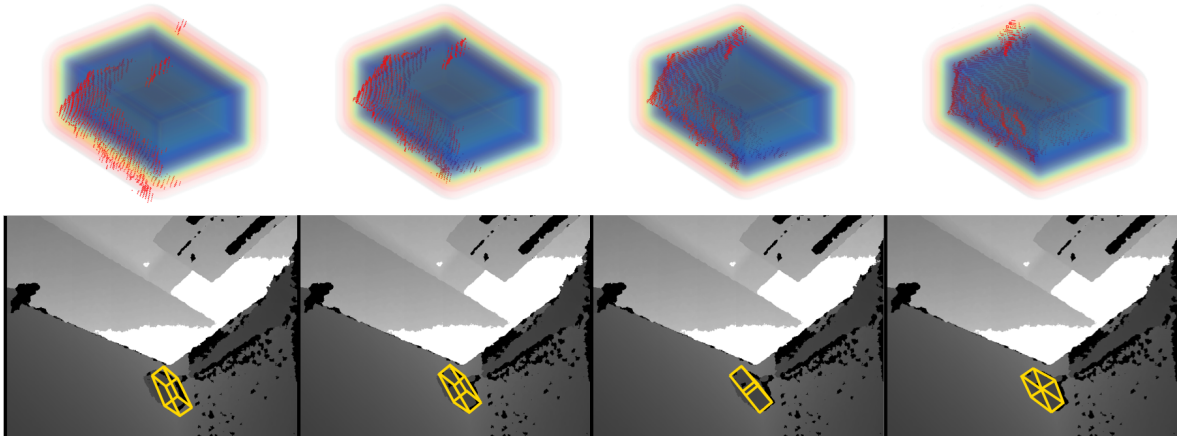


**Figure 3.6:** Typical process of convergence for one frame. The top row shows the back-projected points and the SDF in the object coordinates. The bottom row visualizes the object outline on depth image with corresponding poses. Whole sequence see Video B.1 in Appendix B.

where $\mathtt{J}$ is the Jacobian matrix of the energy function, and $\lambda$ is the non-negative LM damping factor adjusted at each iteration. Taking the solution vector $\boldsymbol{p}^*$ to an element in $\mathbb{SE}(3)$ using Eqn. 3.4, we compose the computed incremental transformation at iteration $n+1$ onto the previously estimated transformation as

$$\tilde{\mathtt{T}}^{n+1} \leftarrow \mathtt{T}^*(\boldsymbol{p}^*)\tilde{\mathtt{T}}^n \text{ , for } n = 1 \text{ to } N \text{ ,} \tag{3.19}$$

and the estimated object pose $\mathtt{T}^{\mathtt{oc}}$ therefore results by composing the final incremental transformation $\tilde{\mathtt{T}}^N$ onto the previous pose:

$$\mathtt{T}^{\mathtt{oc}}_{t+1} \leftarrow \tilde{\mathtt{T}}^N \mathtt{T}^{\mathtt{oc}}_t \text{ .} \tag{3.20}$$

Fig. 3.6 illustrates the output from the tracking process. Intuitively at each iteration, the gradients of the energy functions guide the back-projected points towards the zero-level of the SDF. When the optimization converges, the back-projected points lies on the surface of the object.

**Figure 3.7:** Film strips from a whole tracking sequence. The top row shows the observed depth with object overlay, while we show the tracking result with graphical visualization with a virtual sword on the lower row. Whole sequence see Video B.1 in Appendix B.

## 3.5 Application I: 3D pose tracking

In this section, the performance of our method for 3D tracking is evaluated using qualitative and quantitative measurements. Later sections will expend the application to camera calibration and 3D point cloud modeling.

We first demonstrate the qualitative performance of our method for 3D tracking. Fig. 3.7 shows sample frames from a sequence that we use to track the known control object. The depth image overlaid in blue by the projection of the tracking result is shown in the top row and a virtual sword is visualized on the colour image in the bottom row. As shown the control object is tracked over a large number of frames. Challenging example is shown in Fig. 3.8. In this sequence, we use our method to track the same control object in the presence of heavy occlusion and missing data. The top row shows the original input depth image and the bottom row shows the tracking result. Even when more than half of the object is occluded by hand, the tracker does not lose track. Our energy function is naturally robust to occlusion and missing data because our method back projects the depth point on the image into object coordinate and use the gradient of the level-set embedding function to guide the search for the best pose. In this way, as long as the back-

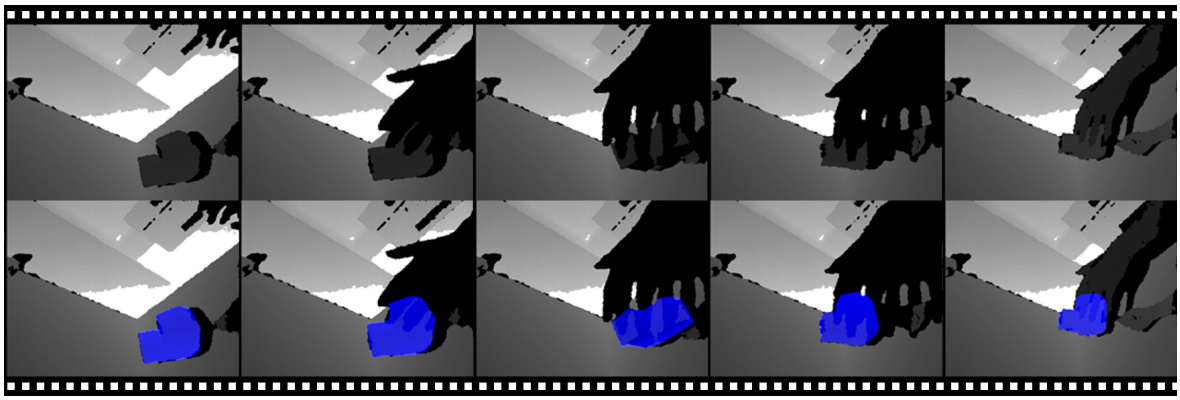


**Figure 3.8:** Film strips showing our algorithm successfully tracking a real rigid object through heavy occlusion. The upper row shows the observed depth sequence, while we show the tracking result on the lower row. Whole sequence see Video B.1 in Appendix B.

projected points encode sufficient information of the location of the object, our method can converge correctly.

To evaluate the performance of 3D rigid object tracker without a general ground truth, we use a known 3D model to generate synthetic depth sequences and run our tracker on the sequences to evaluate the accuracy and robustness of our method for tracking 3D rigid objects. The object for generating the sequence is shown in Fig. 3.9 (a) and sample depth frames are shown in Fig. 3.9 (b). We use a simplified Gaussian noise model

$$Z_o = Z_r + W \quad \text{and} \quad W \sim \frac{1}{\sigma\sqrt{2\pi}} \exp\{-\frac{W^2}{2\sigma^2}\}, \tag{3.21}$$

where $Z_r$ is the depth obtained by projecting the 3D model into image domain, $W$ is the added zero-mean Gaussian noise and $Z_o$ is the generated depth in the final synthetic frame. Synthetic depth sequences with 4 levels of zero-mean Gaussian noise, ranging from $\sigma = 0$ mm to $\sigma = \sqrt{10}$ mm were generated for this experiment.

We show the result in Fig. 3.10. The tracking trajectory plotted in the left column and the errors in translation and rotation are plotted in the right column. The figure shows that even with the highest noise, our tracker can still recover the 6-Dof pose


**Figure 3.9:** (a) Shows the virtual object that we used to generate the synthetic sequence and (b) shows sample frames from our generated synthetic sequence.

without losing track. The tracking errors with low noise ($\sigma = 1$ mm) is less than within 1 mm in translation and $2^\circ$ in rotation.

## 3.6 Application II: Simultaneous tracking and intrinsic calibration

The Kinect sensor doest not provide the calibration parameters for the depth camera, however, for the accurate 3D tracking, the intrinsic matrix of the camera is required. In this section, we show how to use the same energy function to simultaneously track an known object while calibrating the intrinsic parameters of the depth camera.

Assuming zero skew, the intrinsic matrix has the usual form

$$\mathtt{K} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}, \tag{3.22}$$

and to recover the focal lengths and the principal points, the optimization's parameter vector is lengthened to $\boldsymbol{\pi}=(\boldsymbol{p}^{*\top}, \boldsymbol{k}^\top)^\top$ with $\boldsymbol{k}=(f_x, f_y, c_x, c_y)^\top$, and tracking and calibration are carried out simultaneously. The cost and gradient have the same form as Eqns. 3.12 and 3.13, but with the six parameters $\boldsymbol{p}^*$ replaced by the 10 parameters $\boldsymbol{\pi}$. Again Levenberg-Marquardt is deployed as the optimizer.

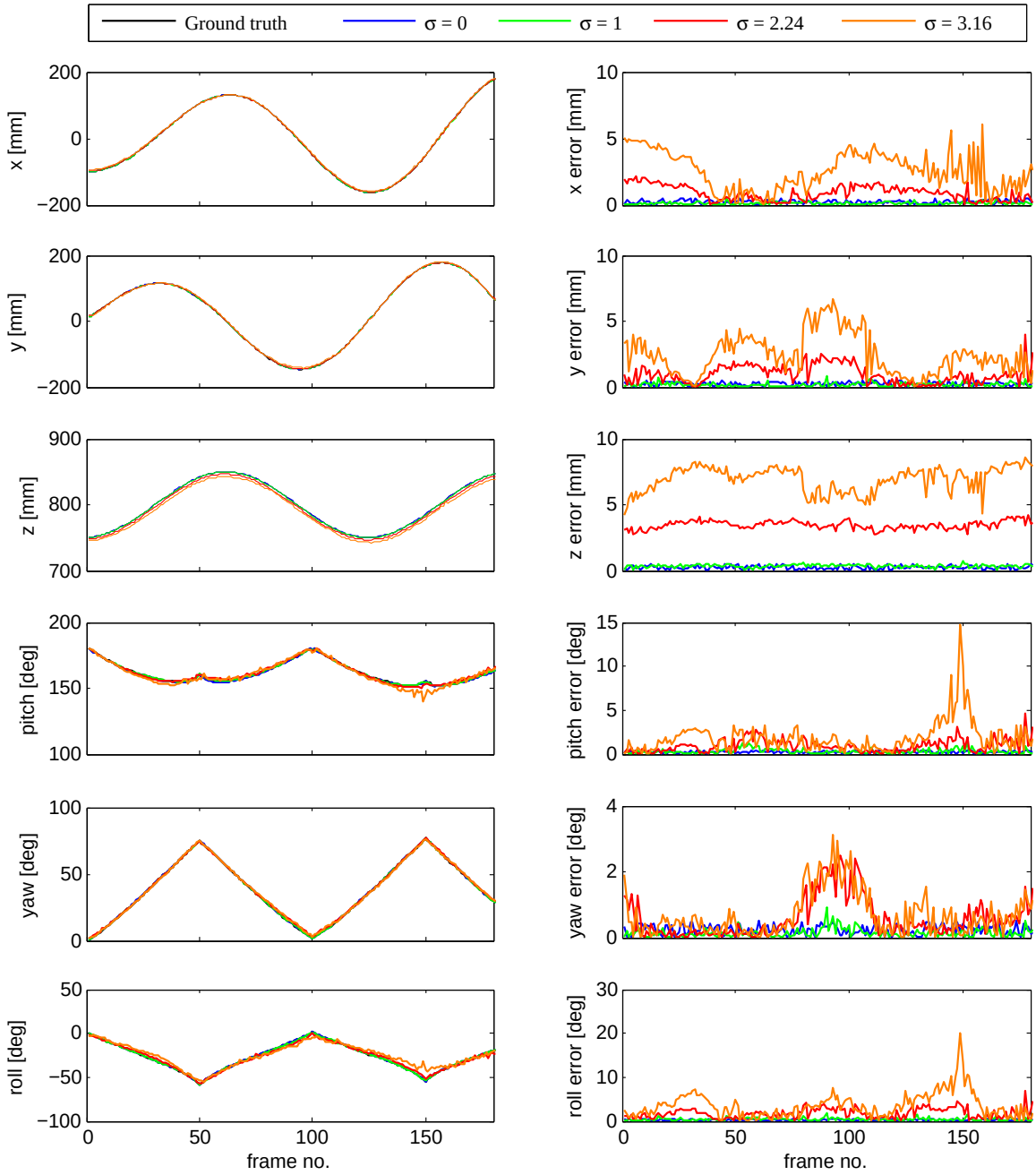



**Figure 3.10:** Quantitative evaluation of the accuracy and robustness of our method for tracking 3D rigid object on synthetic data, with respect to different level of added Gaussian noise. Left column shows the tracking trajectories and the right column shows the pose error.

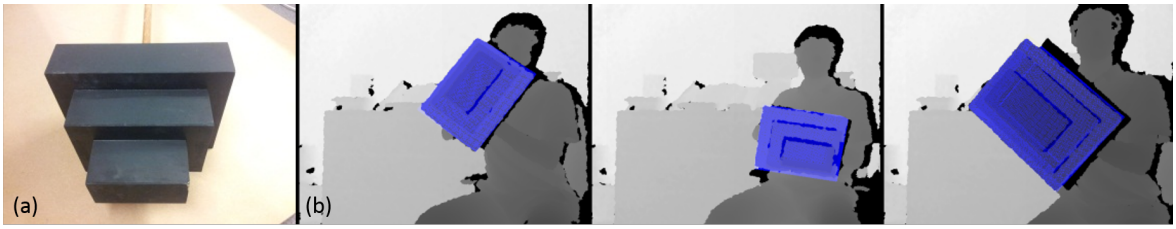



**Figure 3.11:** (a) The calibration object. (b) Film strips showing an example sequence that we used for simultaneous calibration and tracking. Whole sequence see Video B.1 in Appendix B.

Because each video frame $m$ provides an independent optimal calibration given the current pose of the calibration object, the estimates can be combined as

$$\hat{\boldsymbol{k}} = \arg\min_{\boldsymbol{k}} \sum_{m=1}^{M} [\bar{\boldsymbol{k}} - \boldsymbol{k}_m]^\top \boldsymbol{\Sigma}_m^{-1} [\bar{\boldsymbol{k}} - \boldsymbol{k}_m] \tag{3.23}$$

where the covariance $\boldsymbol{\Sigma}_m$ at frame $m$ is approximated by the inverse Hessian. Fig. 3.11(a) shows the calibration object itself and Fig. 3.11(b) shows sample frames of it being moved and traced during simultaneous calibration.

**Test against synthetic data**: To test against ground truth in the face of noise we again use synthetic data. Fig. 3.12 shows the convergence of the overall estimate $\hat{\boldsymbol{k}}$ of the intrinsic parameters as more frames are added, using synthetic data with known ground truth. In low noise ($\sigma$=1 mm), the method recovers the intrinsic parameters accurately ($f_x, f_y$ to $\pm$ 1% and $c_x, c_y$ to $\pm$ 2 pixels) from less than 200 frames ($\approx$7 s of data), while in higher noise, the aspect ratio $(f_x/f_y)$ are still accurate, but the error in principle point increases and the focal length is biased to a low value. In practice, the actual Kinect sensor are in the low noise regime.

**Test against real data**: We have also tested our method on real data. We initialize the tracking with an initial guess of the intrinsics ($f_x = 600, f_y = 600, c_x = 300, c_y = 200$), then track the object (as shown in Fig. 3.11) for 500 frames to estimate the intrinsic matrix. Reprojection error is the most commonly used criterion to evaluate the intrinsic calibration, however, without point correspondences in

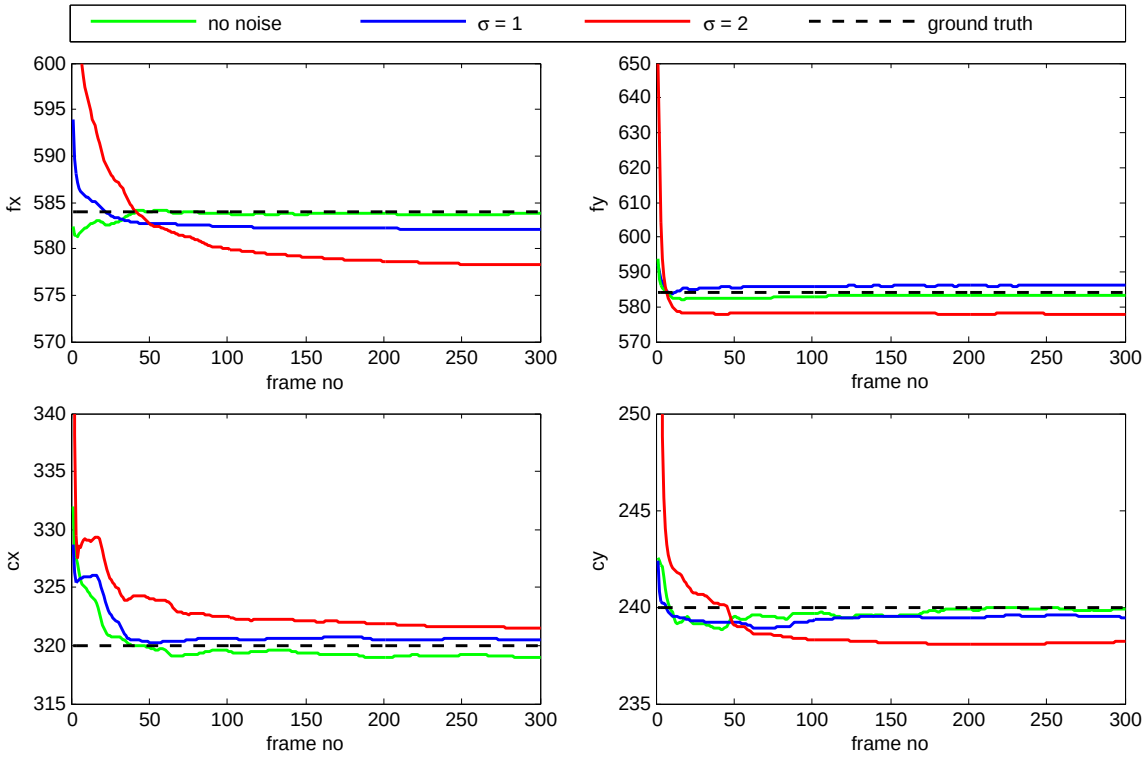



**Figure 3.12:** Quantitative evaluation of our method for calibration on synthetic data, with respect to different level of noise, all intrinsic parameters are in pixels.

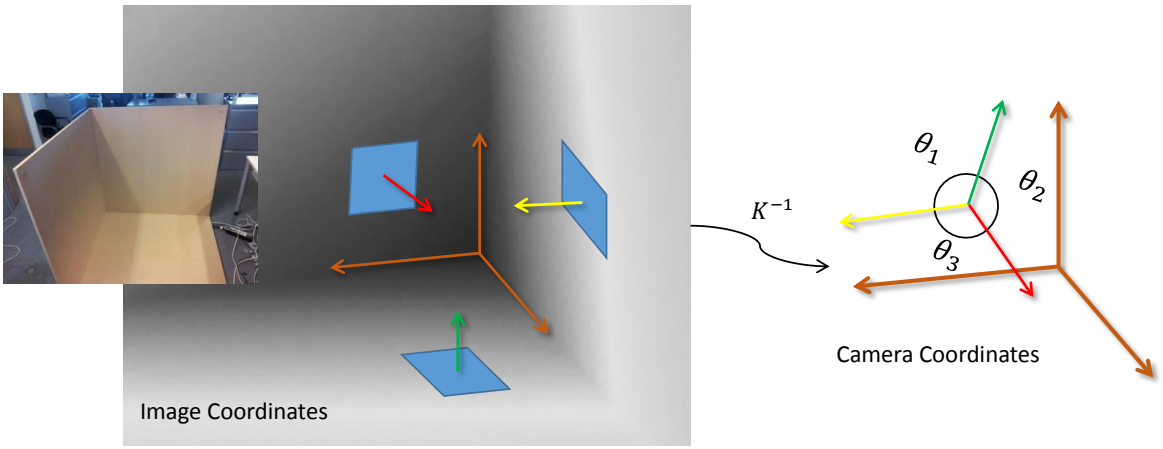



**Figure 3.13:** Illustration of the criteria for the evaluation of intrinsic parameters: given a depth image of three orthogonal planes, we unproject all depth pixels from image coordinate to camera coordinates, then we compute the angle between the norms of the three orthogonal planes as criteria. If the intrinsic matrix is correctly recovered, the angle between the tree norm should all be 90°.

| Method | $\theta_1$ | $\theta_2$ | $\theta_3$ | Overall Err |
|---|---|---|---|---|
| This method | **90.18** | 91.61 | **89.69** | **2.10** |
| Zhang [149] | 90.87 | 92.19 | 87.96 | 5.10 |
| D. Herrera et al. [22] | 91.61 | **89.22** | 90.35 | 2.75 |
| Initial guess | 88.40 | 87.96 | 94.18 | 7.81 |

**Table 3.1:** Quantitative evaluation of the performance of our method for calibration on real data: three orthogonal planes are unprojected into camera coordinates with the recovered intrinsic matrix, the angles (in degree) between the norms of unprojected orthogonal planes are shown in the table as criteria.For a perfectly recovered intrinsic matrix, $\theta_1$=$\theta_2$=$\theta_3$=90 is expected.

the depth image, we can not use this criteria. Thus we use similar criteria as in the Kinect calibration toolbox (Herrera *et al.* [22]), which is based on orthogonal planes, to evaluate the quality of our intrinsic estimation. As illustrated in Fig. 3.13, the calibrated depth camera is used to record the three relative orthogonal planes, then we unproject all depth pixels into the camera coordinates and compute the angle between the normals of the three planes. If the intrinsic matrix `k` is correct, the angles between the three normals should all be 90°.

With this criterion, Table 3.1 compares our calibration result with those from the standard checkerboard calibration method with non-linear refinement Zhang [149] and the state-of-the-art of Herrera *et al.*. [22] (Kinect calibration toolbox). Also shown is the result from the initial guessed intrinsic matrix in the last row. The present method shows the best result for recovering orthogonal plane normals and is also easier to use as Herrera *et al.* [22] requires the user to define the boundary of the calibration plane at all frames. Instead, given a known object model, our method can calibrate the depth sensor with one-click initialization on the first frame. No point correspondences, either by manual clicking or automatic corner detection is required. The disadvantage of our method compared to Herrera *et al.* [22] is that, our method does not calibrate the 6-DoF transformation between the depth camera and the colour camera. A separate method for calibration of the extrinsic parameter of the two cameras is given in Section 3.7. Note that, for

Zhang [149], we attached a checkerboard pattern on a piece of glass then cut off the white area, so that we can do automatic corner detection in depth image. But since sub-pixel accuracy corner detection in the noisy depth image is not possible, the performance of Zhang [149] is limited.

## 3.7 Application III: Extrinsic calibration of the Kinect sensor

The Kinect sensor has both a colour camera and a depth camera. Although the tracking method developed in previous section uses only depth information, later works will use both the depth and the colour camera which are not in the same coordinate system. Being able to map the colour pixels onto the depth image will provide that important colour information. The transfer of values from the colour to the depth camera is achieved by tracking of a "point object" visually in both cameras. Note that although the intrinsic matrix $\mathtt{K}'$ of the colour camera and the rotation and translation $\mathtt{R}'_{3\times3}$ and $\boldsymbol{t}_{3\times1}$ between the cameras appear in the method's explanation, they are not recovered explicitly.

For perspective projection in the RGB camera a scene point $\boldsymbol{X}^{\mathsf{c}}$ in the depth camera coordinates is imaged at

$$\lambda \boldsymbol{x}' = \mathtt{K}' \left[ \mathtt{R}' \boldsymbol{X}^{\mathsf{c}} + \boldsymbol{t} \right] \tag{3.24}$$

where $\lambda$ is a per-point scale factor. Using Eqn. (3.2) we have

$$\lambda \boldsymbol{x}' = \mathtt{K}'\mathtt{R}'\mathtt{K}^{-1} Z \boldsymbol{x} + \mathtt{K}'\boldsymbol{t} = \mathtt{P} Z \boldsymbol{x} + \boldsymbol{q} \, . \tag{3.25}$$

As illustrated in Fig. 3.14, a wand carrying a characteristically coloured sphere

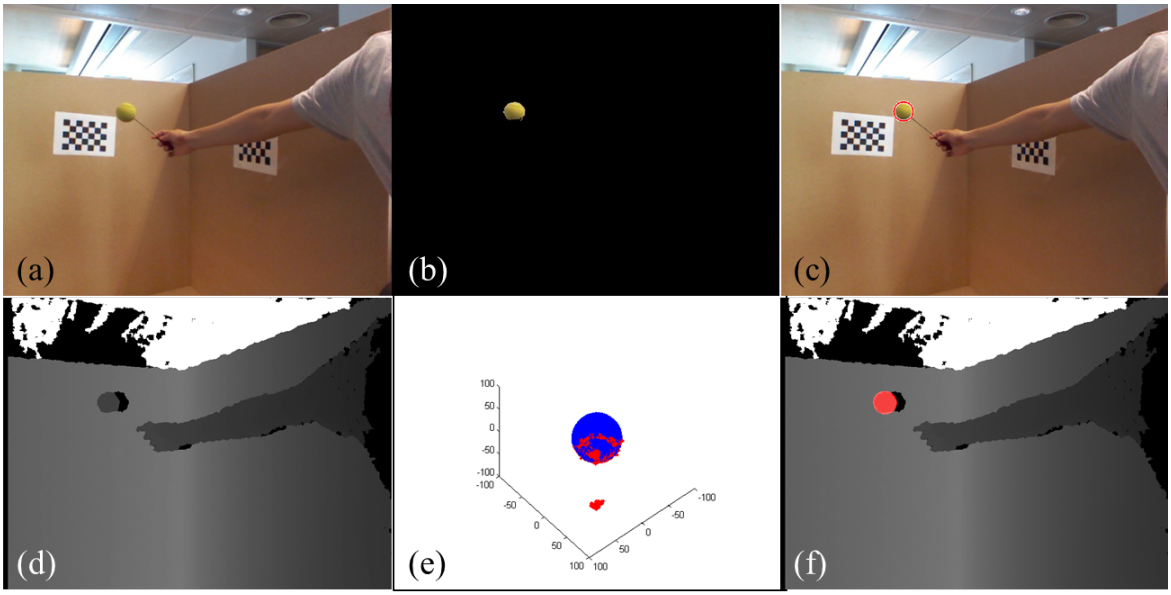



**Figure 3.14:** Extrinsic calibration: (a,b,c) the colour image and spherical wand detected and fitted in it; (d,e,f) the depth image and with object detected and fitted.

is tracked in both cameras to give a set of $\boldsymbol{x} \leftrightarrow \boldsymbol{x}'$ correspondences. For tracking in the colour camera, the ball is detected using a learnt colour model, and a Hough transform applied to the resulting ball/not-ball binary mask to determine the center of the ball. For tracking the sphere in the depth camera, we use the 3D level-set method described in Section 3.5 but instead of tracking with full 6-DoF pose, we only recover the 3-DoF translation in order to recover the sphere's 3D centre .

With $M$ pairs of corresponding points, $\mathtt{P}$ and $\boldsymbol{q}$ are found by minimizing the 2-norm of the reprojection error

$$[\hat{\mathtt{P}}, \hat{\boldsymbol{q}}] = \arg\min_{\mathtt{P},\boldsymbol{q}} \sum_{m=1}^{M} \|\boldsymbol{x}'_m - \hat{\boldsymbol{x}}'_m(\mathtt{P}, \boldsymbol{q}, \boldsymbol{x}_m, Z_m)\|^2 , \tag{3.26}$$

where the $\boldsymbol{x}'$ and $\hat{\boldsymbol{x}}'$ here are just 2-vectors.

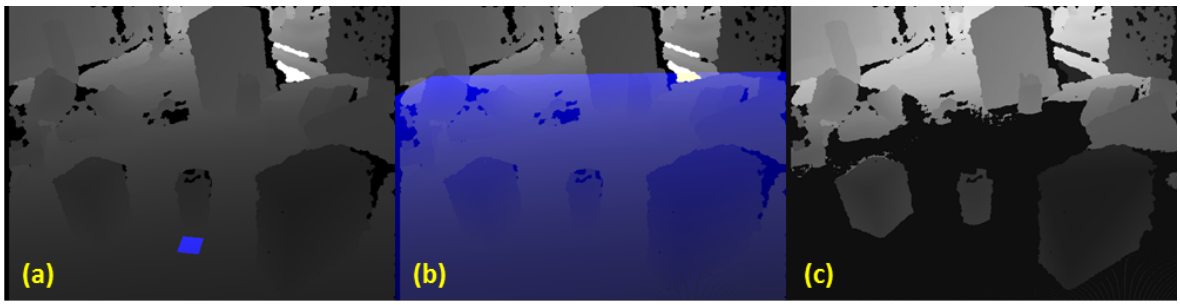



**Figure 3.15:** Illustration of the ground plane removal step: a) fitting a plane patch to any part of the ground plane. b) Extend the plane patch. c) Ground plane is segmented out by removing all depth pixels that are close to the extended plane patch.

## 3.8 Application IV: Point cloud modeling

In this application, the energy function is used to model a point cloud with adaptive primitive shapes (*e.g.* spheres, cylinders and cones) with unknown size. Given the intrinsic parameters and the type of object, the energy function is able to simultaneously recover the pose and the size of the object in the point cloud. The scale of the object model is defined along $x, y, z$ axis in the object frame, the transformation from object coordinates to camera coordinates in Eqn. 3.8 is:

$$\begin{bmatrix} \boldsymbol{X}^{\mathrm{c}} \\ 1 \end{bmatrix} = \mathrm{T}^{\mathrm{co}}(\boldsymbol{p})\,\mathrm{S} \begin{bmatrix} \boldsymbol{X}^{\mathrm{o}} \\ 1 \end{bmatrix} \qquad \mathrm{S} = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{3.27}$$

where $\mathrm{S}$ is the scaling matrix. Then the 3D point $\boldsymbol{X}^{\mathrm{o}}$ is a function of both the 3D pose $\boldsymbol{p}$ and the scaling parameters $\boldsymbol{s} = (s_x, s_y, s_z)^\top$, very similar to the formulation in Section 3.6, the optimization's parameter vector is lengthened to $\boldsymbol{\pi} = (\boldsymbol{s}^\top, \boldsymbol{p}^\top)^\top$. At each frame, the size of object is independently estimated, so we use the same weighted least square as in Section 3.6 to estimate the final scale of the object.

Qualitative results are shown in Figs. 3.15 and 3.16. Before running the point cloud modeling step, we first remove the ground plane from the observed depth

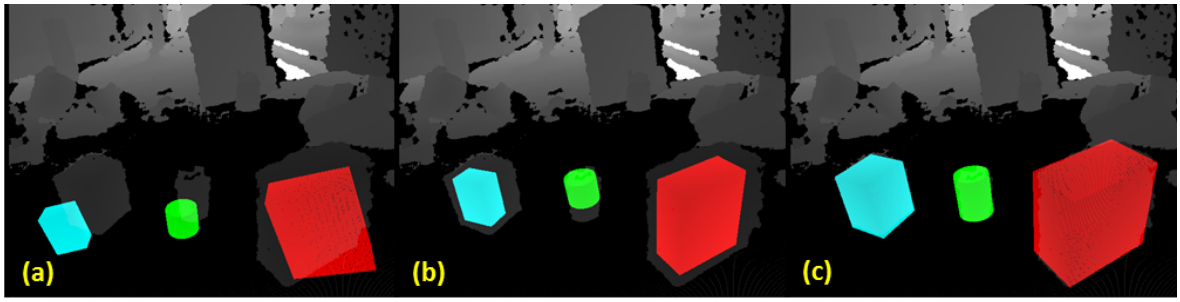



**Figure 3.16:** Illustration of the model adaptation step: a) primitive models are initialized with arbitrary size; b) using our rigid 3D object tracking energy function, models are fitted into point cloud as rigid objects; c) the size and pose of the object in the point cloud is simultaneously tracked and estimated. Whole sequence see Video B.1 in Appendix B.

image. We do this by fitting a plain patch to any part of the ground plane using our 3D rigid object tracker. Then we extend the plane to cover the whole depth image, and compute the discrepancy between the extended plane and the depth image on each pixel. All pixels that have small discrepancies are classified as ground plane and removed. Fig. 3.15 illustrates the ground plane removal process.

After removing the ground plane, we initialize primitive models with arbitrary size, then run the rigid 3D object tracker to align the primitives to the point cloud. The primitive models will of course not fit the point cloud perfectly since the size is arbitrary, but the tracking result is now used as the initialization for the shape adaptation step. In the shape adaptation step we used the energy function parameterized by the joint parameter $\pi$ to simultaneously solve for the size and position of the primitive models. For this application, we only show qualitative results in Fig. 3.16.

## 3.9 Implementation and performance analysis

The whole 3D pose tracking algorithm is listed in Table 3.2, and now we comment on a number of implementation details and analyse the computational requirement of our method. We use 640 $\times$ 480 depth images and 640 $\times$ 480 RGB images from

**Preprocessing:**
- Compute per-voxel partial derivatives $\frac{\partial \mathcal{E}}{\partial \boldsymbol{X}^{\circ}}$ in a derivative volume.

**Runtime:**
- Project all depth pixels $\boldsymbol{x}$ into object coordinates as $\boldsymbol{X}^{\circ}$ with previous pose $\mathtt{T}_t^{\mathsf{oc}}$.
- Evaluate energy function $\mathcal{E}$ using Eqn. 3.12.
- For each back-projected 3D points $\boldsymbol{X}^{\circ}$:
  - Find the partial derivative $\frac{\partial \mathcal{E}}{\partial \boldsymbol{X}^{\circ}}$ in the precomputed derivative volume.
  - Compute the partial derivative $\frac{\partial \boldsymbol{X}^{\circ}}{\partial \boldsymbol{p}}$.
- Compute the Jacobian $\mathtt{J}$ and $\mathtt{J}^{\top}\mathtt{J}$.
- Initialize the LM damping parameter $\lambda = 100$.
- Initialize the overall estimated change in pose $\tilde{\mathtt{T}}^0 = \mathtt{I}$.
- While $\lambda < 10e6$
  - Compute incremental change in pose $\boldsymbol{p}^*$ using Eqn. 3.18.
  - Apply the pose change to the back-projected points $\boldsymbol{X}^{\circ}_{\mathsf{tmp}} \leftarrow \mathtt{T}^*(\boldsymbol{p}^*)\boldsymbol{X}^{\circ}$
  - Evaluate energy function $\mathcal{E}_{\mathsf{tmp}}$ with the updated points $\boldsymbol{X}^{\circ}_{\mathsf{tmp}}$.
  - If $\mathcal{E}_{\mathsf{tmp}} > \mathcal{E}$
    - Accept the pose change: $\tilde{\mathtt{T}}^{n+1} \leftarrow \mathtt{T}^*(\boldsymbol{p}^*)\tilde{\mathtt{T}}^n$, $\boldsymbol{X}^{\circ} \leftarrow \boldsymbol{X}^{\circ}_{\mathsf{tmp}}$, $\mathcal{E} \leftarrow \mathcal{E}_{\mathsf{tmp}}$.
    - Decrease $\lambda$ by a magnitude: $\lambda \leftarrow \lambda/10$.

  else
    - Reject the pose change.
    - Increase $\lambda$ by a magnitude: $\lambda \leftarrow \lambda \times 10$.
- Compose the current pose from previous pose $\mathtt{T}^{\mathsf{oc}}_{t+1} \leftarrow \tilde{\mathtt{T}}^n\mathtt{T}^{\mathsf{oc}}_t$.

**Table 3.2:** Pseudo code for the proposed model-fitting algorithm for one frame, steps with red marker (● or ○) can be implemented in parallel fashion on the GPU.

the Kinect sensor as input, 200 × 200 × 200 voxel cubes are used to hold the object's SDF. Using an Intel Core-i7 3.4 GHz CPU, the proposed algorithm can run at rate in excess of 100 Hz when tracking the control object shown in Fig. 3.7, well in excess of the 30 Hz frame rate requirements. Most of the tasks involved operate per-pixel so when dealing with dense depth map or bigger objects, significant performance gain can be achieved by exploiting the high degree of parallelism on a GPU. We use the NVIDIA CUDA [1] framework to implement the approach on GPU, detailed introduction to GPU computing and the NVIDA CUDA framework is described in Appendix A. In Table 3.2, the steps that can be parallelized on GPU have been marked with red dots • or red circle○ .

When tracking relatively small objects that occupy only several thousand pixels in the image domain, the GPU version is slower than the CPU version. For example the average processing time per frame lies around 20 ms when tracking an object with rough size of 200×200×200 mm with a NVIDA Geforce GTX680 GPU. However, the GPU computational time remains almost constant with the increase in the number of pixels, unlike the CPU version, whose computational time increase linearly with respect to the number of pixel.

## 3.10 Conclusion

In this chapter we have described a novel and quite generic probabilistic framework for 3D model fitting problems involving dense depth data. We use 3D level-set embedding functions to represent the 3D shapes and a fully probabilistic and variational cost function is optimized by adjusting the perspective projection matrix. We have developed four applications of the proposed framework: 3D tracking; intrinsic and extrinsic calibration; and point-cloud modeling. Our algorithm can

[1] http://docs.nvidia.com/cuda/cuda-c-programming-guide

efficiently run on CPU and more importantly, we have also shown that because of its highly parallel nature, our algorithm is amendable to GPU implementation to handle larger amount of data. Our method share the benefits of region-based methods, such as robustness to occlusion and missing data, while still doest not rely on the computational intensive sampling based solvers.

A few difficulties remain. First, since we assume the whole object region is generated solely from the 3D model, the tracker is not robust to close-to-surface occlusion and outlier pixels. The tracking can be lost when the object being tracked is directly held in the user's hand. Second, the model-fitting framework requires a known accurate 3D model to work. Obtaining accurate 3D models for 3D tracking is a bottleneck for the application of our approach and indeed any model-based approaches. In Chapter 4, we resolve these two issues by first showing how we can leverage both the colour and depth information towards more robust tracking then by developing an approach for simultaneous tracking and 3D reconstruction.

# 4

# STAR3D: Simultaneous Tracking and Reconstruction of 3D Objects

*In this chapter, a probabilistic model is developed for simultaneous 3D tracking and reconstruction at video rate, taking as input combined depth and color imagery. 3D shape is represented implicitly as the the zero-level set of a signed distance function computed in the voxelized space around the object. Reconstruction is initialized either from an uninformative prior, or from a generic class shape if something more is known about the object. The process involves two phases, one of tracking during which the shape is fixed and the pose updated by minimizing a robust function of signed distances, the other of reconstruction where camera poses are fixed and the object shape optimised by evolving the 3D level set function. Color appearance models of the surface and background are learned on-line during the process, permitting tracking to continue through heavy occlusion. Qualitative and quantitative experiments are presented to demonstrate the method's performance. The method requires no special set up, other than knowledge of the intrinsic and extrinsic calibrations of the depth and color cameras. An earlier version of this work has been published by Ren et al. [113]*

## 4.1 Introduction

In the stroke rehabilitation system, the tracking of the patient's feet and the control object require accurate 3D models, but without access to a predefined CAD model or similar, the standard acquisition method involves 3D scanning using a precisely calibrated multi-camera or range sensor system. This is costly and difficult to setup, and often slow.

In this chapter we introduce a framework for simultaneous tracking and reconstruction of unknown 3D rigid objects that is simple, fast and effective. The system is initialized with a simple primitive 3D shape (e.g. a sphere or a cube), then the 3D shape of the object being tracked is reconstructed incrementally on-line. Our immediate application interest of course is the video-rate reconstruction of shoes, hand and objects in the context of patients undergoing rehabilitation as discussed in Chapter 1. Of course a large number of application areas can be considered. For example, it can allow users to pick any rigid object from their home, scan it and then use it as a control object to interact with a computer.

The proposed framework comprises of two modules, a tracking module and a reconstruction module.:

- The tracking module is a direct extension of the depth-only tracker from Chapter 3. Instead of using only the depth information, we also use the colour information to model the foreground and background appearance towards more robust tracking in presence of close-to-surface outliers and occlusions.

- A reconstruction module that extend the 2D level-set evolution approach (by Cremers [33] and Bibby and Reid [14]) , to 3D and combines it with the idea of space-carving methods for image-only 3D reconstruction towards a probabilistic 3D reconstruction framework. Prior knowledge of the 3D shape can also be considered through our probabilistic model in the reconstruction pro-

cess.

The formulations of both the tracking and the reconstruction modules lead to fully variational cost functions, which can be very efficiently solved using gradient-based optimization methods.

The rest of the chapter is structured as follows. We first review recent related work in Section 4.2, and follow this by a description of the scene geometry and the basic graphical model in Section 4.3. Then we develop the formulation for single object tracking and reconstruction with RGB-D data in Sections 4.4 and 4.5. We describe the implementation of and experimentation with our method in Section 4.6. We draw conclude in Section 4.7.

## 4.2 Related work

A broader review of the general topic of 3D model acquisition has been given in Chapter 2 and here we review some specific prior work in real-time dense reconstruction.

As we have mentioned in Chapter 2, the shape-from-silhouette based methods provide an end-to-end solution for 3D module acquisition. However, the accuracy of shape-from-silhouette reconstruction is limited by the accuracy of the silhouette extraction and the number of views. Recently, improvements have been made to the method. For example, based on the same idea of carving empty space, in Michel *et al.* [88], the authors extend the Visual Hull method with user interaction. A contacting wand is tracked while moving along the surface of the object being reconstructed. The 3D shape is reconstructed through carving the empty space around the object. Although the system yields real-time performance, the setup is costly and complicated, and thus unsuitable for our purpose of home rehabilitation.

The introduction of customizable, frame-rate RGB-D cameras and powerful GPUs have made dense 3D reconstruction almost a matter of routine. KinectFusion, introduced by Newcombe *et al.* [92] is among the most successful systems. With a single hand-held Kinect device, KinectFusion can incrementally reconstruct the surface of the physical world that the camera sees, in real-time. In Newcombe *et al.* [93], the authors introduce a similar real-time system, DTAM, which uses only color image sequence as input. Both KinectFusion and DTAM use a volumetric representation for the shape, which places a limit on the size of the object that these methods can reconstruct. Chen *et al.* [25] and Niessner [94] have extended the dense, continuous volumetric representation with scalable and lossless sparse representations to allow reconstruction at much larger scale. However, as discussed in the previous section, KinectFusion and DTAM-like depth reconstruction methods rely heavily on a static scene structure to track the camera, thus it can not reconstruct small moving objects. Another related recent work is Weiss *et al.* [144], the authors use a single, fixed, un-calibrated Kinect to scan a human body in a home environment. Accurate 3D human shapes are obtained by combining multiple monocular views of a person moving in front of the sensor. The SCAPE model (Anguelov *et al.* [7]) is used to constrain the alignment of the multiple depth maps from the various views. In Prisacariu *et al.* [107], the authors use monocular 2D image cues to reconstruct 3D shapes. The reconstruction is constrained by a learnt low-dimensional 3D shape space. Both Weiss *et al.* [144] and Prisacariu [107] are capable of reconstructing 3D objects with a single camera. However, they both rely heavily on a learnt shape space to constrain the reconstruction. This forces the reconstructed objects to be within a fixed, prelearnt category, which can not fulfill our purpose of reconstructing any object as the control object.

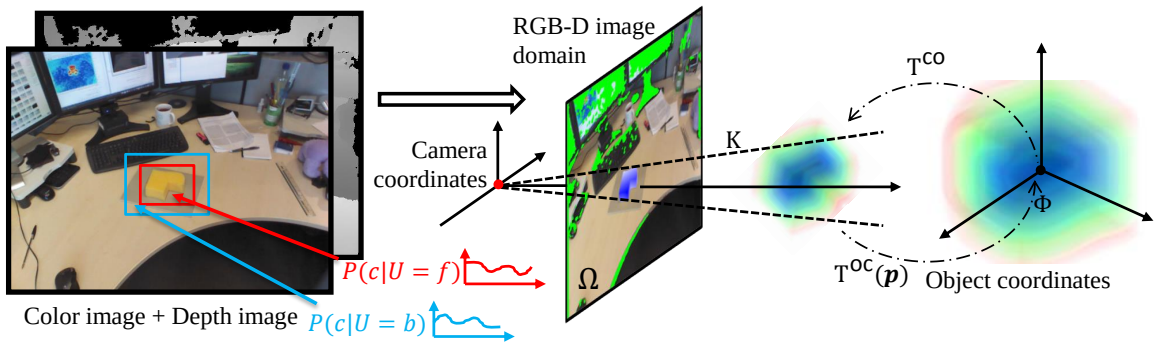



**Figure 4.1:** Representation of the 3D model $\Phi$, the RGB-D image domain $\Omega$, the surface, background appearance models $P(c|U{=}f)$, $P(c|U{=}b)$ and the pose $\mathtt{T}(\boldsymbol{p})$.

## 4.3 Scene geometry and basic graphical model

The scene geometry is shown in Fig. 4.1. Assuming known calibration of both the depth and color camera (*i.e.* known $\mathtt{K}$, $\mathtt{P}$ and $\boldsymbol{q}$, as introduced in Chapter 3), we first reproject the colour image onto the depth image frame and denote the aligned RGB-D image as a bag-of-pixels $\Omega = \left\{\{\boldsymbol{X}^{\mathsf{i}}_1, c_1\}, \{\boldsymbol{X}^{\mathsf{i}}_2, c_2\} \ldots \{\boldsymbol{X}^{\mathsf{i}}_{N_\Omega}, c_{N_\Omega}\}\right\}$ where $\boldsymbol{X}^{\mathsf{i}} = Z\,\boldsymbol{x} = (Zx, Zy, Z)^\top$ is a pixel location in homogeneous coordinates with explicit depth and $c$ is the RGB value of a pixel. The superscript $^{\mathsf{i}}$ denotes the image, to distinguish from the camera coordinates $^{\mathsf{c}}$ and the object coordinates $^{\mathsf{o}}$. The object shape is represented as a bag-of-voxels $\Phi = \left\{\{\boldsymbol{X}^{\mathsf{o}}_1, \phi_1\}, \{\boldsymbol{X}^{\mathsf{o}}_1, \phi_1\} \ldots \{\boldsymbol{X}^{\mathsf{o}}_{N_\Phi}, \phi_{N_\Phi}\}\right\}$ where $\boldsymbol{X}^{\mathsf{o}}$ is a 3D voxel location in object coordinates and $\phi$ is the SDF value of $\Phi$ at location $\boldsymbol{X}^{\mathsf{o}}$. An indicator variable $V$ is introduced for each voxel location, which can take on value $\{\text{in}, \text{out}\}$. $V{=}$in for voxels that belongs to the inside of the object and $V{=}$out otherwise. Theoretically, $V$ can take the value $V{=}$on for voxels that are exactly on the surface of the object, but since the voxel label is only useful for reconstruction and the number of voxels exactly on the surface can be neglected, we ignore the label $V{=}$on.

The object shape $\Phi$ is first transformed into the camera coordinates through the rigid transformation $\mathtt{T}^{\mathsf{co}} \in \mathbb{SE}3$, from which it is projected to generate the RGB-D image. We introduce a co-representation of each RGB-D pixel $\left\{\boldsymbol{X}^{\mathsf{i}}, c, U\right\}$, where

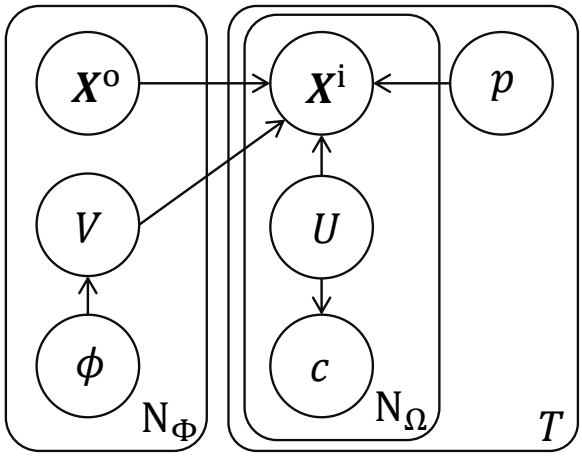



**Figure 4.2:** The basic graphical model for tracking and reconstruction

$U$ is the foreground/background model that takes values $U{=}f$ or $b$ depending on whether the pixel belongs to the foreground or to the background. Two appearance models are used to describe the color statistics of the scene: one for the foreground, which is generated by the *object surface*; and one for the background, which is generated by arbitrary voxels *outside the object*. These are represented by their likelihoods $P(c|U{=}f)$ and $P(c|U{=}b)$ respectively. The two appearance models are represented with normalized RGB color histograms using 16 bins per channel. The histogram can be initialized either from a detection module or from a user-selected bounding box on the RGB image, in which the foreground model is built from the interior of the bounding box and the background from the immediate region outside the bounding box.

The whole graphical model for tracking and reconstruction is shown in Fig. 4.2. The left side of the graphical model is in the 3D object domain, the label of each voxel location $V$ depends on the SDF valude $\phi$. The right side is in the 2D image domain, the 3D pixel location $\boldsymbol{X}^{\mathrm{i}}$ depends on the per-pixel foreground/background model $U$, the pose $\boldsymbol{p}$, the 3D voxel location $\boldsymbol{X}^{\mathrm{o}}$ and the voxel label $V$. The probability of the color $c$ on each pixel on depends on the foreground/background model

$U$. The joint distribution can therefore be expressed as

$$P(\{\boldsymbol{X}^{\mathrm{i}}_{i,t}, c_{i,t}, U\}_{1\dots N_\Omega, 1\dots T}, \boldsymbol{p}_{1..T}, \{\boldsymbol{X}^{\circ}_{j}, V_j, \phi_j\}_{1\dots N_\Phi}) = \tag{4.1}$$
$$\prod_{j=1}^{N_\Phi} P(\boldsymbol{X}^{\circ}_{j}) P(\phi_j) P(V_j|\phi) \prod_{t=1}^{T} \left\{ P(\boldsymbol{p}_t) \prod_{i=1}^{N_\Omega} P(\boldsymbol{X}^{\mathrm{i}}_{i,t}|U, \{\boldsymbol{X}^{\circ}, V\}_{1\dots N_\Phi}, \boldsymbol{p}_t) P(c_{i,t}|U) P(U) \right\} .$$

Full inference leading to a MAP estimate of the shape and poses given the RGBD image sequence

$$\max_{\Phi, \boldsymbol{p}_1 \dots \boldsymbol{p}_T} P(\Phi, \boldsymbol{p}_1 \dots \boldsymbol{p}_T | \Omega_1 \dots \Omega_T) \tag{4.2}$$

appears intractable. Instead, we propose an approximate inference at each new frame by alternating two phases where (i) $\boldsymbol{p}_t$ is optimized while freezing the shape and (ii) the MAP estimate of the shape $\Phi$ is found while freezing the camera poses. This approximation leads to two simplification of the graphical model for tracking and reconstruction, which are shown in Sections 4.4 and 4.5 respectively. In the whole simultaneous tracking and reconstruction framework, we initialize the tracking module (Section 4.4) with a simple initial model (*e.g.* a sphere), and iterate the tracker until it converges to a pose that projects the initial model close to the object region in the RGB-D image. Then the reconstruction module (Section 4.5) will take all the frames and poses up to the current frame as input for the estimation of 3D shape.

## 4.4 Tracking while the shape is frozen

### 4.4.1 Graphical model and inference

For the purpose of tracking, we assume the 3D object shape is known and no motion model. Hence the RGB-D image at time $t$ depends only on the 3D shape and the pose. At this stage, we are interested in describing how the depth and

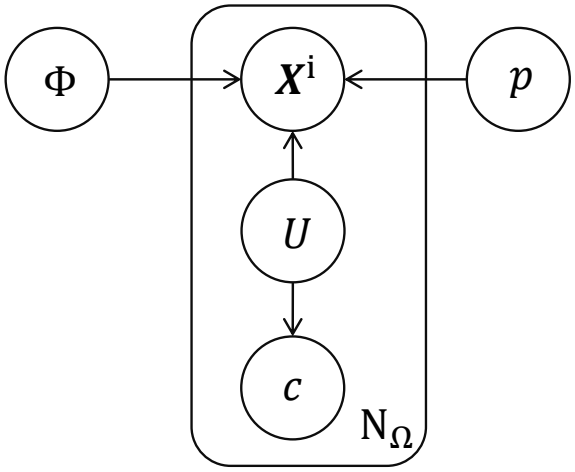



**Figure 4.3:** The graphical model for tracking a single rigid 3D object.

the color of each RGB-D pixel is generated from the pose and the known shape, thus, we simplify the original graphical model by treating the shape $\Phi$ as a single variable and removing the time index $t$. This simplification leads to the graphical model for tracking in Fig. 4.3.

In this model, given the per-pixel foreground/background model $U$, the shape $\Phi$ and the pose $\boldsymbol{p}$, the RGB-D image is sampled as a bag of pixels $\{\boldsymbol{X}^{\mathrm{i}}_j, c_j\}_{1\ldots N_\Omega}$. More specifically, for each pixel location $\boldsymbol{x}$ in the RGB-D image, it is the depth $Z$ that is randomly drawn from the distribution $P(Z(\boldsymbol{x})|\boldsymbol{p}, \Phi, U)$. But for convenience and clarity, we use $\boldsymbol{X}^{\mathrm{i}}$ to denote $Z(\boldsymbol{x})$ in the graphical model and in the following derivation, setting

$$P(Z(\boldsymbol{x})|\boldsymbol{p}, \Phi, U) \equiv P(\boldsymbol{X}^{\mathrm{i}}|\boldsymbol{p}, \Phi, U) \ . \tag{4.3}$$

The joint distribution for a single pixel given by the graphical model in Fig. 4.3 is

$$P(\boldsymbol{X}^{\mathrm{i}}, c, \Phi, U, \boldsymbol{p}, \Omega) = P(\Phi)\, P(\boldsymbol{p})\, P(\boldsymbol{X}^{\mathrm{i}}|U, \Phi, \boldsymbol{p})\, P(c|U)\, P(U) \ . \tag{4.4}$$

Now marginalize over the foreground/background model $U$

$$P(\boldsymbol{X}^{\mathrm{i}}, c, \Phi, \boldsymbol{p}) = P(\Phi)P(\boldsymbol{p}) \sum_{u \in \{f,b\}} P(\boldsymbol{X}^{\mathrm{i}}|U = u, \Phi, \boldsymbol{p})P(c|U = u)P(U = u) \ . \tag{4.5}$$

We seek a ML estimate of the current pose:

$$P(\boldsymbol{X}^{\mathsf{i}}, c|\Phi, \boldsymbol{p}) = \sum_{u\in\{f,b\}} P(\boldsymbol{X}^{\mathsf{i}}|U=u, \Phi, \boldsymbol{p}) P(c|U=u) P(U=u) \ . \tag{4.6}$$

The per-pixel depth likelihoods $P(\boldsymbol{X}^{\mathsf{i}}|U{=}f, \Phi, \boldsymbol{p})$ and $P(\boldsymbol{X}^{\mathsf{i}}|U{=}b, \Phi, \boldsymbol{p})$ are defined using a normalized smoothed delta function and a smoothed and shifted Heaviside function

$$P(\boldsymbol{X}^{\mathsf{i}}|U=f, \Phi, \boldsymbol{p}) = \frac{\delta^{\mathsf{on}}(\Phi(\boldsymbol{X}^{\mathsf{o}}))}{\eta_f} \tag{4.7}$$

$$P(\boldsymbol{X}^{\mathsf{i}}|U=b, \Phi, \boldsymbol{p}) = \frac{H^{\mathsf{out}}(\Phi(\boldsymbol{X}^{\mathsf{o}}))}{\eta_b} \tag{4.8}$$

$$\eta_f = \sum_{i=1}^{N_\Phi} \delta^{\mathsf{on}}(\Phi(\boldsymbol{X}_i^{\mathsf{o}})) \tag{4.9}$$

$$\eta_b = \sum_{i=1}^{N_\Phi} H^{\mathsf{out}}(\Phi(\boldsymbol{X}_i^{\mathsf{o}})) \ , \tag{4.10}$$

where $\boldsymbol{X}^{\mathsf{o}}$ is the back-projection of $\boldsymbol{X}^{\mathsf{i}}$ in object coordinates, following

$$\begin{bmatrix} \boldsymbol{X}^{\mathsf{o}} \\ 1 \end{bmatrix} = \mathtt{T}^{\mathsf{oc}}(\boldsymbol{p}) \begin{bmatrix} \mathtt{K}^{-1}\boldsymbol{X}^{\mathsf{i}} \\ 1 \end{bmatrix} . \tag{4.11}$$

$\delta^{\mathsf{on}}$ is the smoothed Dirac delta function and $H^{\mathsf{out}}$ is a shifted and smoothed Heaviside step function

$$\delta^{\mathsf{on}}(\Phi) = \frac{4\exp\{\Phi/\sigma\}}{(\exp\{\Phi/\sigma\}+1)^2} \tag{4.12}$$

$$H^{\mathsf{out}}(\Phi) = \begin{cases} 1-\delta^{\mathsf{on}}(\Phi) & \text{if } \ \Phi \geq 0 \\ 0 & \text{if } \ \Phi < 0 \end{cases} . \tag{4.13}$$

Here $\sigma$ is a constant parameter that determines the width of the basin of attraction. Larger $\sigma$ gives wider base of convergence to the energy function, while smaller $\sigma$ leads to faster convergence. In our experiments, we empirically use $\sigma = 2$. These

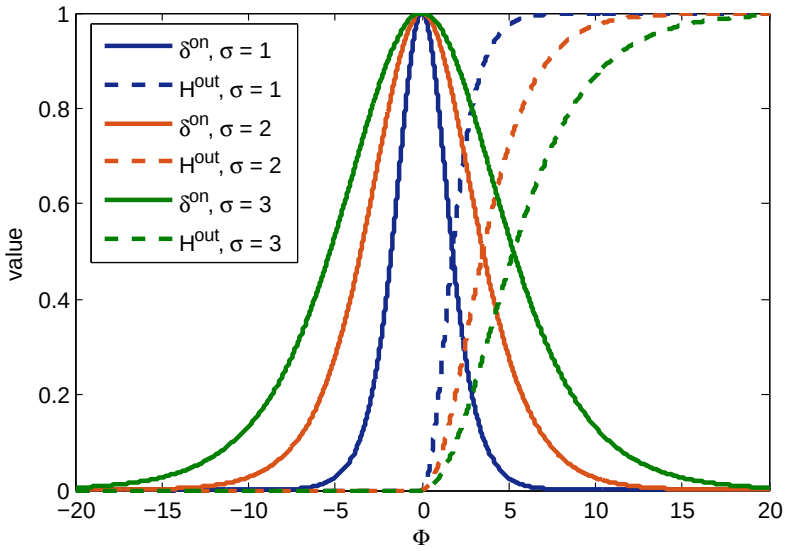



**Figure 4.4:** The form of the smoothed delta function $\delta^{\text{on}}$ and Heaviside function $H^{\text{out}}$

two functions are plotted in Fig. 4.4. The probability of foreground and background models $P(U = f)$ and $P(U = b)$ are uniform distributions:

$$P(U = f) = \frac{\eta_f}{\eta}\,, \qquad P(U = b) = \frac{\eta_b}{\eta}\,, \qquad \eta = \eta_f + \eta_b\,. \tag{4.14}$$

Substituting Eqn. 4.7∼4.14 into Eqn. 4.6, we obtain the likelihood of the pose $\boldsymbol{p}$ for an individual pixel

$$P(\boldsymbol{X}^{\text{i}}, c|\Phi, \boldsymbol{p}) = P^{\text{f}}\delta^{\text{on}}(\Phi(\boldsymbol{X}^{\text{o}})) + P^{\text{b}}H^{\text{out}}(\Phi(\boldsymbol{X}^{\text{o}})), \tag{4.15}$$

where

$$P^{\text{f}} = P(c|U = f)\,, \qquad P^{\text{b}} = P(c|U = b)\,, \tag{4.16}$$

Assuming pixel-wise independence, the likelihood of the pose for the current whole RGB-D image is

$$P(\Omega|\boldsymbol{p}, \Phi) = \prod_{i=1}^{N_\Phi} \left\{ P_i^{\text{f}}\delta^{\text{on}}(\Phi(\boldsymbol{X}_i^{\text{o}})) + P_i^{\text{b}}H^{\text{out}}(\Phi(\boldsymbol{X}_i^{\text{o}})) \right\}\,. \tag{4.17}$$

### 4.4.2 Pose optimization

As in Section. 3, the cost function is written as a sum of log-likelihood over all pixels

$$\mathcal{E} = \sum_{i=1}^{N_\Phi} \log \left\{ P_i^{\mathsf{f}} \delta^{\mathsf{on}}(\Phi(\boldsymbol{X}_i^{\mathsf{o}})) + P_i^{\mathsf{b}} H^{\mathsf{out}}(\Phi(\boldsymbol{X}_i^{\mathsf{o}})) \right\} . \tag{4.18}$$

The energy function is parameterized in terms of the pose change $\boldsymbol{p}^*$, and the gradients are found as

$$\frac{\partial \mathcal{E}}{\partial \boldsymbol{p}^*} = \sum_{i=1}^{N_\Phi} \left\{ \left[ \frac{P_i^{\mathsf{f}} \frac{\partial \delta^{\mathsf{on}}}{\partial \Phi} + P_i^{\mathsf{b}} \frac{\partial H^{\mathsf{out}}}{\partial \Phi}}{P(\boldsymbol{X}_i^{\mathsf{i}}, c_i | \Phi, \boldsymbol{p})} \frac{\partial \Phi}{\partial \boldsymbol{X}_i^{\mathsf{o}}} \right] \frac{\partial \boldsymbol{X}_i^{\mathsf{o}}}{\partial \boldsymbol{p}^*} \right\} , \tag{4.19}$$

where the derivatives of $\delta^{\mathsf{on}}$ and $H^{\mathsf{out}}$ are

$$\frac{\partial \delta^{\mathsf{on}}}{\partial \Phi} = \frac{4 \exp\{\Phi/\sigma\}}{\sigma(\exp\{\Phi/\sigma\}+1)^2} - \frac{8 \exp\{(2\Phi)/\sigma\}}{\sigma(\exp\{\Phi/\sigma\}+1)^3} \tag{4.20}$$

$$\frac{\partial H^{\mathsf{out}}}{\partial \Phi} = \begin{cases} -\frac{\partial \delta^{\mathsf{on}}}{\partial \Phi} & \text{if } \Phi \geq 0 \\ 0 & \text{if } \Phi < 0 \end{cases} . \tag{4.21}$$

The derivatives $\frac{\partial \Phi}{\partial \boldsymbol{X}^{\mathsf{o}}}$ and $\frac{\partial \boldsymbol{X}^{\mathsf{o}}}{\partial \boldsymbol{p}^*}$ follow the same formula in Eqn. 3.13 and Eqn. 3.14 in Section 3. We also use the same Levenberge-Marquardt iterations and local frame pose update in Section 3 for the pose optimization.

### 4.4.3 Online learning of appearance model

The foreground/background appearance model $P(c|U)$ is important for the robustness of the tracking, and we adapt the appearance model online after the tracking is completed on each frame. We use the pixels that have $\Phi(\boldsymbol{X}^{\mathsf{o}}) \leq 3$ (so, points that best fit the surfaces of the object) to compute the foreground appearance model and the pixels in the immediate surrounding region of the objects to compute the

background model. The online update of the appearance model is achieved using a linear opinion pool with learning rates $\{\rho^{\mathsf{f,b}}\}$ :

$$P_t(c|U=u) = (1-\rho^u)P_{t-1}(c|U) + \rho^u P_t(c|U) \ . \tag{4.22}$$

In all our experiments, we set $\rho^{\mathsf{f}} = 0.05$ and $\rho^{\mathsf{b}} = 0.3$. The background appearance mode has a higher learning rate. This is because the object being reconstructed is moving in a uncontrolled environment, and thus the change of appearance in background is much faster that in the foreground.

## 4.5 Reconstruction while poses are frozen

### 4.5.1 Graphical model and inference

For the purpose of reconstruction, we assume the RGB-D image sequence $\Omega_{1\dots T}$ and the set of poses for each frame $\boldsymbol{p}_{1\dots T}$ are known and the goal now is to infer the 3D shape $\Phi$. We still start from the basic graphical model of Fig. 4.2, and since we are now interested in estimating the per-voxel value $\phi$, we simplify the graphical model by treating each RGB-D image as a single variable $\Omega$. The graphical model for our reconstruction method is shown in Fig. 4.5.

The 3D object shape is represented as a set of labeled voxels $\{\boldsymbol{X}^{\circ}, V, \phi\}$, where the label takes values $V=\{\text{in}, \text{out}\}$ depending on the voxel's being inside or outside the object. The number of voxels that are on the surface ($\phi = 0$) is negligible by comparison. The current RGB-D image $\Omega_t$ depends on the current pose $\boldsymbol{p}_t$, the set of voxels and their labels and the labels depend on the SDF value $\phi$. The joint

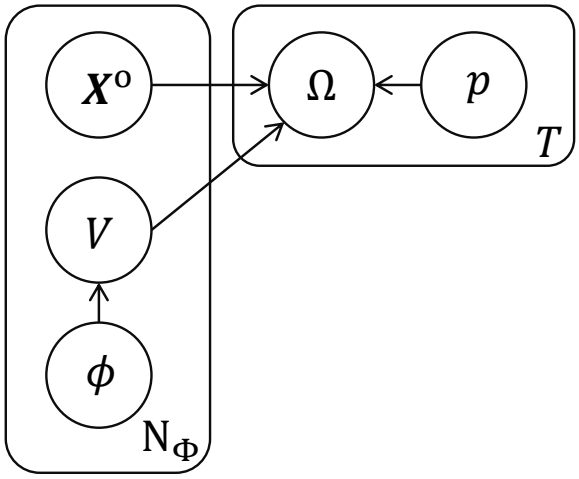



**Figure 4.5:** The graphical model for reconstructing a single object

distribution can therefore be expressed as

$$P(\Omega_{1\dots T}, \boldsymbol{p}_{1\dots T}, \{\boldsymbol{X}^\circ_j, V_j, \phi_j\}_{1\dots N_\Phi}) = \prod_{j=1}^{N_\Phi} P(\boldsymbol{X}^\circ_j) P(\phi_j) P(V_j|\phi_j) \prod_{t=1}^{T} P(\Omega_t|\{\boldsymbol{X}^\circ, V\}_{1\dots N_\Phi}, \boldsymbol{p}_t) P(\boldsymbol{p}_t)\ , \tag{4.23}$$

a form which permits a recursive update of $\{\boldsymbol{X}^\circ, V\}$. Each discretized voxel location $\boldsymbol{X}^\circ_i$ is drawn deterministically once and the term $P(V_i|\phi)$ should properly allow the label $V_i$ of each voxel location $\boldsymbol{X}^\circ_i$ to be drawn at random from the value of the shape $\phi_i$. Given the appearance model $P^{\mathsf{f}}$ and $P^{\mathsf{b}}$ from the tracking module that we described in Section 4.4, we extract a foreground depth-colour image $\hat{\Omega}_t$ from $\Omega_t$ by including only those pixels for which $P^{\mathsf{f}} > P^{\mathsf{b}}$ currently. The objective then is to maximize the posterior probability of $\Phi$ (*i.e.* the value of the SDF) given the first $t$ frames and poses

$$\max_{\Phi} P(\Phi|\boldsymbol{p}_{1\dots t}, \hat{\Omega}_{1\dots t}). \tag{4.24}$$

Assuming independence between each frame and pose and also per-voxel independence, we first marginalize over all possible labels $V_i$ for each pixel on the joint

distribution

$$P(\boldsymbol{p}_{0\ldots t}, \hat{\Omega}_{0\ldots t}, \Phi) \propto \tag{4.25}$$
$$P(\Phi) \prod_{j=1}^{N_\Phi} \sum_{v \in \{\text{in,out}\}} \left\{ P(V_j = v|\phi_j) \prod_{t=1}^{T} P(\hat{\Omega}_t | \{\boldsymbol{X}_j^\circ, V_j = v\}, \boldsymbol{p}_t) P(\boldsymbol{p}_t) \right\} ,$$

where the prior of SDF term is

$$P(\Phi) = \prod_{j}^{N_\Phi} P(\phi_j) = \prod_{j}^{N_\Phi} P(\Phi(\boldsymbol{X}_j^\circ)) . \tag{4.26}$$

The posterior distribution of the shape $\Phi$ follows

$$P(\Phi|\boldsymbol{p}_{0\ldots t}, \hat{\Omega}_{0\ldots t}) = \tag{4.27}$$
$$P(\Phi) \prod_{j=1}^{N_\Phi} \sum_{v \in \{\text{in,out}\}} \left\{ P(V_j = v|\phi_j) \prod_{t=1}^{T} P(\hat{\Omega}_t | \{\boldsymbol{X}_j^\circ, V_j = v\}, \boldsymbol{p}_t) \right\} .$$

The term $P(\hat{\Omega}_t|\{\boldsymbol{X}_j^\circ, V_j = v\}, \boldsymbol{p}_t)$ of Eqn. 4.27 gives, per voxel, the likelihood of generating the measured pixel's depth value. Given a particular voxel $\boldsymbol{X}_j^\circ$ with its labeling $V_j$ and the current pose $\boldsymbol{p}_t$, a pixel position is determined from

$$\hat{\boldsymbol{x}} = \mathtt{KT}^{\mathtt{co}}(\boldsymbol{p}_t)[\boldsymbol{X}_i^\circ \; 1]^\top . \tag{4.28}$$

The closest measured pixel $\boldsymbol{x}$ is determined and back-projected into the scene, allowing a fractional signed range discrepancy $d$ to be determined along the direction of the projected ray as

$$d_j = \frac{[\mathtt{R}(\boldsymbol{p}_t)\boldsymbol{X}_i^\circ + \boldsymbol{T}(\boldsymbol{p}_t)]^\top [\mathtt{K}^{-1} Z \boldsymbol{x}]}{|\mathtt{K}^{-1} Z \boldsymbol{x}|^2} - 1 , \tag{4.29}$$

which is positive if the voxel $\boldsymbol{X}_i^\circ$ lies *behind* the measured surface, and *vice versa*. The resulting likelihoods of the voxel lying inside or out of the object are approximated

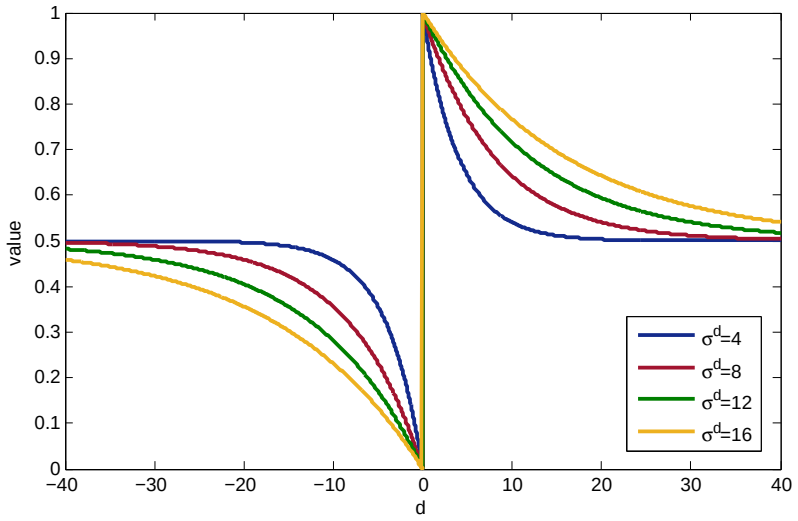



**Figure 4.6:** The form of the likelihood function $L^{\text{in}}$.

using

$$P(\hat{\Omega}_t|\{\boldsymbol{X}_j^\circ, V_j = \text{in}\}, \boldsymbol{p}_t) \propto L^{\text{in}}(j) = 0.5(\text{sgn}(d_j)\text{e}^{-|d_j|/\sigma_d} + 1) \ , \tag{4.30}$$

$$P(\hat{\Omega}_t|\{\boldsymbol{X}_j^\circ, V_j = \text{out}\}, \boldsymbol{p}_t) \propto L^{\text{out}}(j) = 1 - L^{\text{in}}(j) \ , \tag{4.31}$$

forms that makes us equivocal ($L^{\text{in,out}} \approx 0.5$) about voxels that are either far from the surface or exactly on the surface, but give a low probability of being inside if the voxel is immediately in front of the surface, and *vice versa*. $L^{\text{in}}$ is plotted in Fig. 4.6. (This form of likelihood does not increase $L^{\text{out}}$ if the surface is visible behind the voxel. Neglecting that extra information has not been detrimental in our experiments.)

Note that, given any RGB-D image, $L^{\text{in}}$ and $L^{\text{out}}$ can provide information only about voxels between the camera and the object, and nothing about those far behind the surface of the object. So in order to stabilize the estimation of shape, we place a initial guess with respect to a chosen shape (*i.e.* SDF) $\Phi_o$:

$$L_0^{\text{in}}(j) = G(\Phi_o(\boldsymbol{X}_j^\circ)) \ , \tag{4.32}$$

$$L_0^{\text{out}}(j) = 1 - G(\Phi_o(\boldsymbol{X}_j^\circ)) \ . \tag{4.33}$$

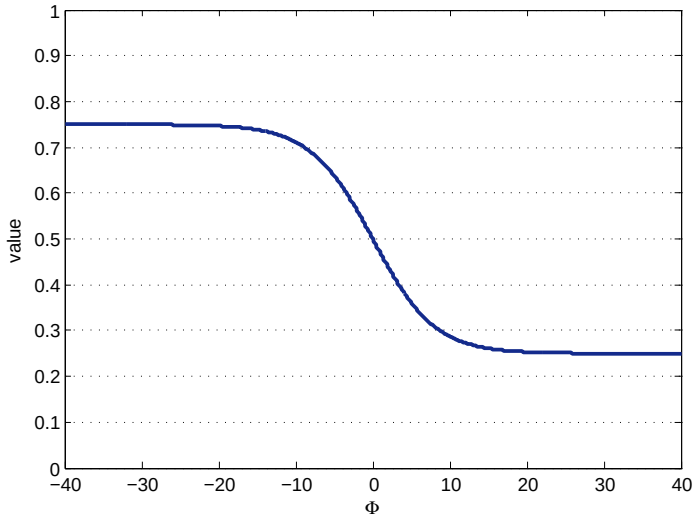



**Figure 4.7:** The form of the initial guess of likelihood $G(\Phi)$.

In all our work we use as initial guess

$$G(\Phi_0) = a(1+e^{\Phi_0/\sigma_G})^{-1} + (1-a)(0.5)\ , \quad a \in [0,1] \tag{4.34}$$

with influence $a$=0.5 and smoothness $\sigma_G$=4, as shown in Fig. 4.7. (An uninformative prior has $a$=0.) The time-cumulative likelihood $\prod_t P(\hat{\Omega}_t|\{\boldsymbol{X}_i^\circ, V_i = v\}, \boldsymbol{p}_t)$ for a particular voxel $j$ is found as

$$\prod_{t=1}^{T} P(\hat{\Omega}_t|\{\boldsymbol{X}_j^\circ, V_j = v\}, \boldsymbol{p}_t) \propto L^{\text{in}}_{1\ldots t}(j) = L^{\text{in}}_0(j) L^{\text{in}}_{1\ldots t-1}(j) L^{\text{in}}_t(j)\ . \tag{4.35}$$

The term $P(V_j = v|\phi_j)$ in Eqn. 4.27 is expressed per voxel by a smoothed Heaviside steps function $H$:

$$P(V = \text{out}|\phi) = H(\phi)\ , \tag{4.36}$$

$$P(V = \text{in}\}|\phi) = 1 - H(\phi)\ , \tag{4.37}$$

$$H = \frac{1}{\exp\{-\phi/\sigma_H\} + 1}\ . \tag{4.38}$$

$\sigma_H$ controls the smoothness of the Heaviside function and as shown in Fig. 4.8.

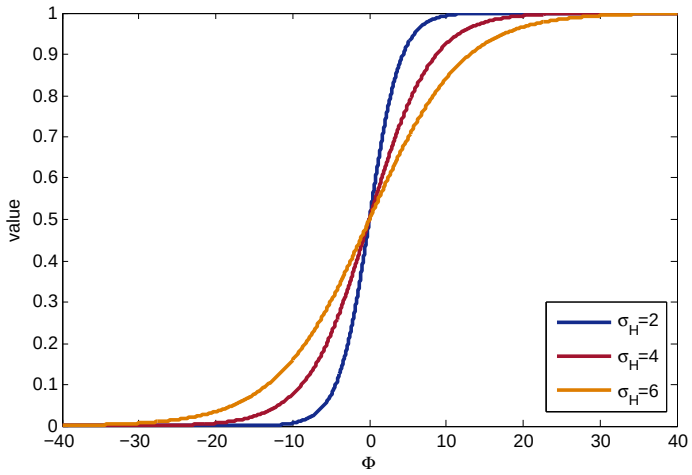



**Figure 4.8:** The form of the smoothed Heaviside function $H$.

Larger $\sigma_H$ results in more stable behavior of the shape optimization , but also leads to greater loss of the details in the 3D shape; and *vice versa* for smaller $\sigma_H$. Empirically, $\sigma_H = 4$ appears a good compromise and we use this in all our experiments.

Following Eqn. 4.26, we now specify a geometric prior on $\Phi$ that rewards a signed distance function with unit gradients (Li *et al.* [75]):

$$P(\Phi) = \prod_{j=1}^{N_\Phi} \frac{1}{\sigma_\Phi \sqrt{2\pi}} \exp\{-\frac{(|\nabla\Phi(\boldsymbol{X}_j^\circ)| - 1)^2}{2\sigma_\Phi}\} \tag{4.39}$$

where $\sigma_\Phi$ specifies the relative weight of the prior. The term $|\nabla\Phi(\boldsymbol{X}_j^\circ)|$ is the magnitude of the gradient at point $\boldsymbol{X}_j^\circ$ and for a SDF, it should equal to one everywhere. Maintaining a good SDF is desirable as the tracking step can then directly use the shape estimation up to the current frame for tracking.

### 4.5.2 Shape optimization

Combining the three terms in Eqn. 4.35∼4.39 and taking logs, we have the log posterior of shape as the cost function

$$\begin{aligned}\mathcal{E}_\Phi = \log P(\Phi|\boldsymbol{p}_{1\ldots t},\hat{\Omega}_{1\ldots t}) \propto \\ \sum_{j=1}^{N_\Phi}\Bigg\{\log\Big\{L^{\text{in}}_{1\ldots t}(j)\,[1-H(\Phi(\boldsymbol{X}^\circ_j))]\,G(\Phi_0(\boldsymbol{X}^\circ_j))+ \\ L^{\text{out}}_{1\ldots t}(j)\,H(\Phi(\boldsymbol{X}^\circ_j))\,[1-G(\Phi_0(\boldsymbol{X}^\circ_j))]\Big\} - \frac{(|\nabla\Phi(\boldsymbol{X}^\circ_j)|-1)^2}{2\sigma_\Phi}\Bigg\}.\end{aligned} \tag{4.40}$$

We now differentiate with respect to $\Phi$:

$$\begin{aligned}\frac{\partial\mathcal{E}_\Phi}{\partial\Phi} = \\ \sum_{j=1}^{N_\Phi}\Bigg\{\frac{\delta(\Phi)\left\{L^{\text{out}}_{1\ldots t}[1-G(\Phi_0)]-L^{\text{in}}_{1\ldots t}G(\Phi_0)\right\}}{L^{\text{in}}_{1\ldots t}[1-H(\Phi)]G(\Phi_0)+L^{\text{out}}_{1\ldots t}H(\Phi)[1-G(\Phi_0)])} - \frac{1}{\sigma_\Phi}\left[\nabla^2\Phi-\text{div}\left(\frac{\nabla\Phi}{|\nabla\Phi|}\right)\right]\Bigg\}.\end{aligned} \tag{4.41}$$

where $\nabla^2$ is the Laplacian operator and $\delta$ is the derivative of the smoothed Heaviside step function, *i.e.* a smoothed Dirac delta function. Interestingly, $\delta(\Phi)$ is a way of expressing the uncertainty on the surface of the object. A small $\sigma_H$ will lead to a tighter basin of $\delta$, which in turn leads to more detailed surface reconstruction. However, less uncertainty on the surface also leads to more tolerance of the violation of the prior constraint $P(\Phi)$, which will result in an unstable SDF. We seek $\frac{\partial\mathcal{E}_\Phi}{\partial\Phi}=0$ by carrying out steepest-ascent using the following gradient flow

$$\frac{\partial\Phi}{\partial t} = \frac{\partial\mathcal{E}_\Phi}{\partial\Phi} \tag{4.42}$$

In practice, we implement this using a simple numerical scheme in a discretized volume. All spatial derivatives are computed using central differences and the Laplacian uses a 3x3x3 spatial kernel. We use $\sigma_\Phi$=3 and a time-step $\Delta t$=1 for all experiments. We follow the requirement for SDF stability (see Li *et al.* [75]) that

$\Delta t / \sigma_\Phi^2 < 0.25$.

## 4.6 Implementation and evaluation

### 4.6.1 Implementation

The tracking and reconstruction algorithms have been implemented both on CPU and on GPU. We use 640×480 colour and depth images as input and 200×200×200 voxel cubes to contain the object's SDF. On an Intel Core i7 3.4 GHz CPU, the tracking algorithm can still run at around 100 Hz frame rate. For real-time performance, the reconstruction algorithm is only suitable for GPU implementation. Each iteration of the shape evolution takes around 1.5∼2 ms on a GTX 680 GPU, and in order to recover a coarse shape for the tracking for the first frame, the shape evolution requires around 100 iterations, which is significantly more than the subsequent frames that only requires around 10∼20 steps.

### 4.6.2 Qualitative evaluation

A variety of evaluations, both qualitative and quantitative have been performed to test the performance of our method. We begin with several real-world tracking-reconstruction sequences, which show that our method is robust to initialization and outliers and can work in unconstrained environments. Next we use generated ground truth data to evaluate the accuracy of both tracking and reconstruction. Finally, we compare both tracking and reconstruction results with KinectFusion [92], arguably the current state-of-the-art.

Figs. 4.9, 4.10 and 4.11 show examples of our method simultaneously tracking and reconstructing different objects: a piece of sponge, a hand with fixed gesture

**Figure 4.9:** Film strip showing our algorithm tracking and reconstructing a piece of sponge. Top row shows the color image and the middle row shows the reconstruction result overlayed with the aligned RGB-D image (black area is the missing depth data). The bottom row shows reconstruction result. Whole sequence see Video B.2 in Appendix B.

**Figure 4.10:** Film strip showing our algorithm tracking and reconstructing a black shoe. Top row shows the color image and the middle row shows the reconstruction result overlayed with the aligned RGB-D image (green area is the missing depth data). The bottom row shows reconstruction result. Whole sequence see Video B.2 in Appendix B.

**Figure 4.11:** Film strip showing our algorithm tracking and reconstructing a hand with fixed hand pose. Top row shows the color image and the middle row shows the reconstruction result overlayed with the aligned RGB-D image (green area is the missing depth data). The bottom row shows reconstruction result. Whole sequence see Video B.2 in Appendix B.

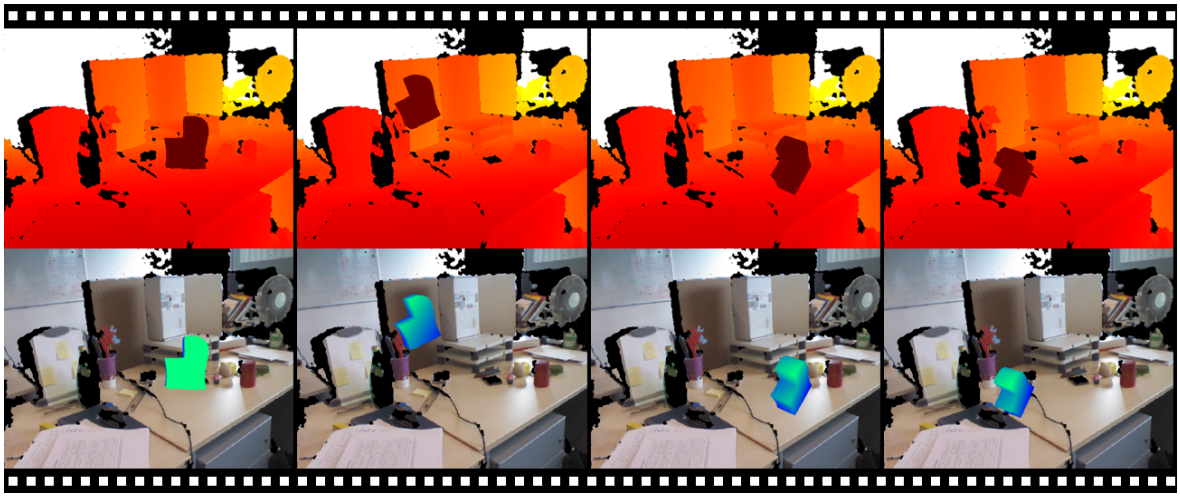


**Figure 4.12:** Sample frames from sequence where a synthetic object has been added to the color (top) and depth image (bottom).

and a shoe. The sponge reconstruction is initialized using a sphere and the other two using a cube. The last row of each figure shows the reconstruction result. All three objects are successfully reconstructed using less than 30 seconds of data. Note that the sequences are filmed in an unconstrained environment and the objects are small, and moving and hence reconstruction using KinectFusion is difficult.

We use pixels that have $P^{\mathsf{f}} > P^{\mathsf{b}}$ for reconstruction. However, since in Fig. 4.10 the object is close to the background (the hand is close to the shoe), that an additional constraint is applied and we use only we only use pixels from the foreground region which are at least 2 pixels away from any background pixel. This makes the reconstruction results slightly smaller than the real object, by a fixed margin. the surface of the reconstructed shoe is also smoother.

### 4.6.3 Quantitative evaluation

Next we evaluate our tracking and reconstruction performance using ground truth data. Again, we use synthetic RGB-D sequences for this evaluation. We use the same method as in Chapter3, but add extra colour information by projecting the

texture of model onto the synthetic image. Some sample frames are shown in Fig. 4.12. The surface of the object has been fully observed across the frames. We present two evaluations. First, we consider a perfectly known model and perform tracking only, measuring the pose accuracy. Second, we initialize with a spherical model and run simultaneous tracking and reconstruction. We evaluate the accuracy of the recovered poses by the difference from the ground truth.

As shown in Fig. 4.13 the maximum error in these "ideal" conditions with a known object is less than $1^\circ$ in rotation and less than 2 mm in translation where the object is at a distance of around 0.8 m distance. When reconstructing and tracking a previously unknown object, the maximum error increases to $6^\circ$ in rotation and 3 mm in translation. Examples showing tracking with reconstructed data, even under significant occlusion by the hand holding the object, are given in Videos B.2 in Appendix B.

The constant parameter $\sigma_d$ in Eqn. 4.31 controls the thickness of the reconstructed model, relative to the volume quantization. We use a volume of $200\times200\times200$ for all experiments in this chapter, with large objects scaled and reconstructed into the same fixed sized volume. To show the sensitivity of reconstruction accuracy to $\sigma_d$ we vary it from 4 to 40 (see Fig. 4.14). We evaluate the accuracy of reconstruction by computing the average alignment error between the reconstructed result and the ground truth. The initial average alignment error is 6 mm. For values of $\sigma < 20$ the error decreases as more frames are observed and converges at around frame 150. When $\sigma_d = 20$, the reconstructed shape is still visually correct, but the average alignment error is becoming unacceptably large. When $\sigma_d = 40$, the reconstructed shape becomes incorrect (*i.e.* too thick) after the first few frames, resulting in tracking failure in all following frames, and the shape is not correctly reconstructed. We used $\sigma_d = 8$ for all our other tests.

In the last experiment, we compare the quality of both our tracking and the re-

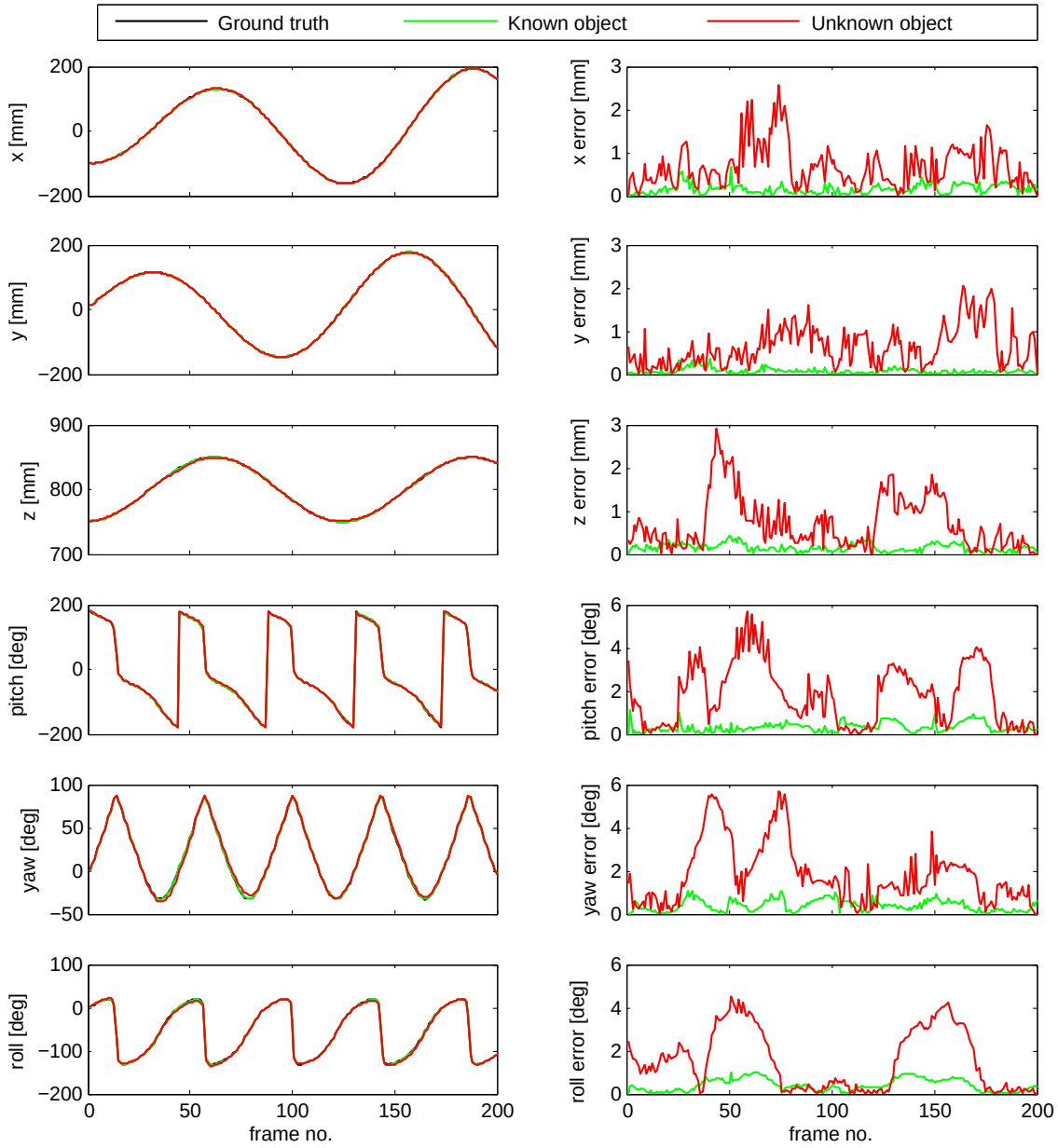



**Figure 4.13:** Quantitative evaluation of the accuracy our method for tracking 3D rigid object on synthetic data. The left column shows the absolute ground truth translation and rotation, and the right column shows the error in translation and rotation. The error in translation is measured in mm while rotation is measured in degree.

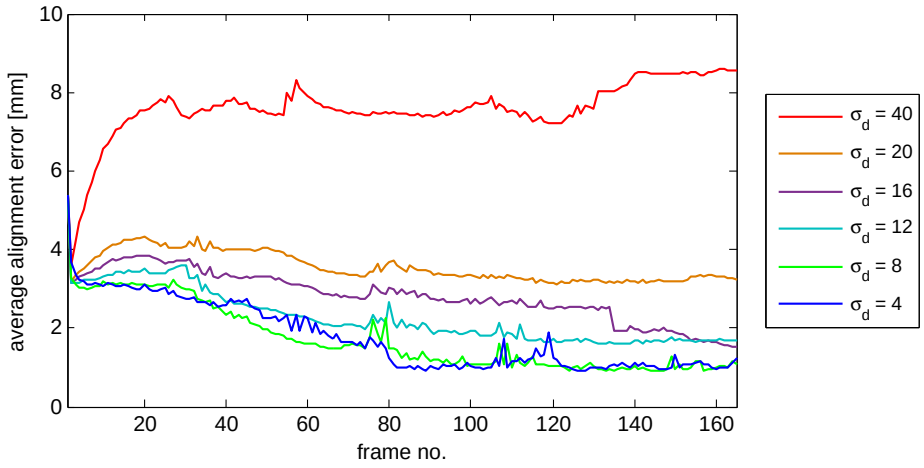



**Figure 4.14:** The effect of different $\sigma_d$ value on the accuracy of our method for 3D reconstruction measured using an average alignment error.

construction result with those from KinectFusion [92]. Since KinectFusion requires a static scene to perform reconstruction, we place the object (a piece of sponge) in the centre of a random scene and use the Kinect SDK to analysis a sequence, in which we move the Kinect around the object to obtain most views of the object. Some sample frames are shown in Fig. 4.15. The first row shows the colour frame, and the second row visualizes the reconstruction re-projected onto the colour image (after aligning with the depth frame). The third row shows the 'Fusion frame' from the Kinect SDK, which is the KinectFusion reconstruction result up to and including the current frame. The last two columns show the reconstruction results from our method compared to those of KinectFusion. Both are similarly visually pleasing. The 3D model produced by KinectFusion has an incorrect lip on the top surface, while that part is correctly reconstructed by our algorithm. Our method does however produce a more noisy surface below the reconstructed 3D shape, in the areas that have never been observed by the camera. This is because noisy outlier depth pixels can propagate incorrect membership probabilities to areas in the 3D volume where the related views of object has never be observed.

The KinectFusion result are directly obtained from the Kinect SDK, using a

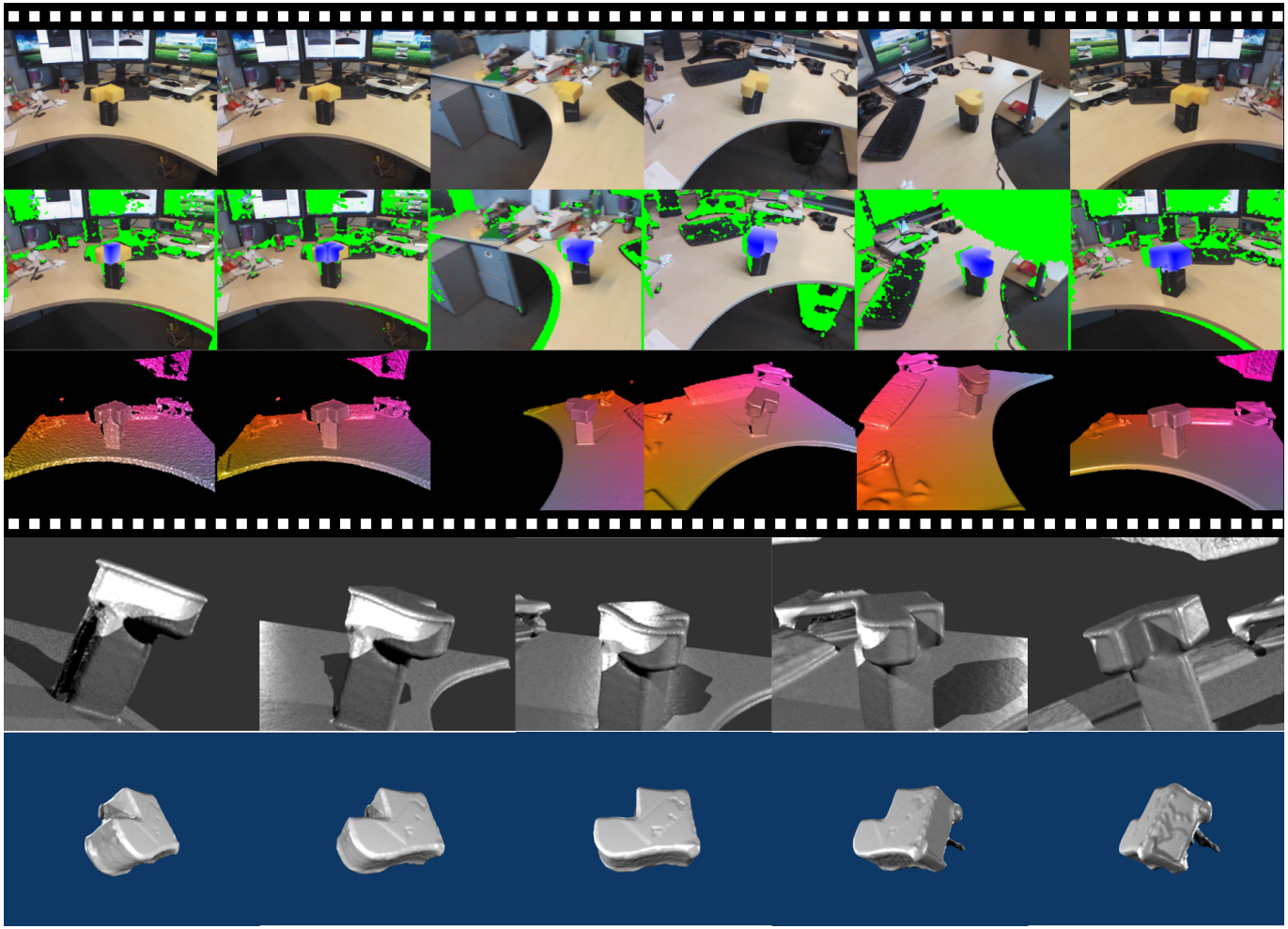


**Figure 4.15:** Film strip showing a comparison between our method and KinectFusion for 3D reconstruction. Each column shows a frame, the first row shows the color frame, the second row visualized our reconstruction result on re-projected color image (aligned with depth image), the third row shows the KinectFusion fusion frame. Last two rows shows the final reconstruction results (Top: Kinect fusion. Bottom: ours). Whole sequence see Video B.2 in Appendix B.

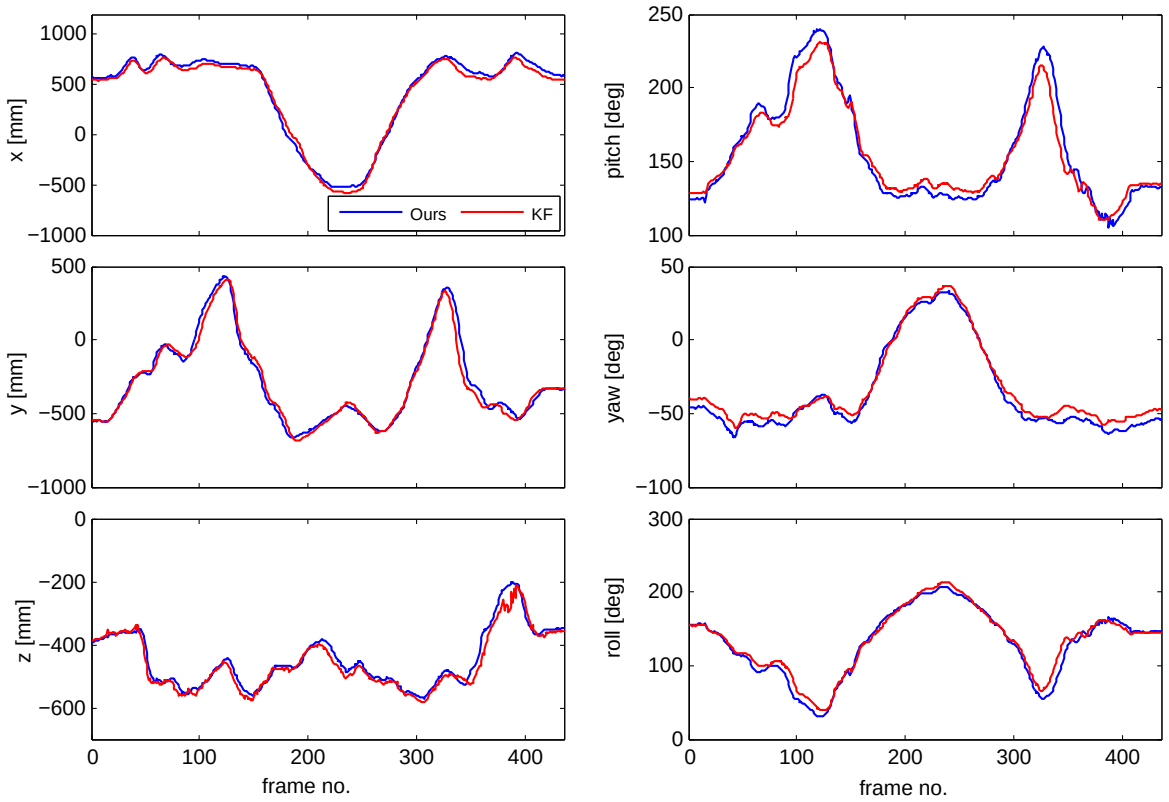



**Figure 4.16:** Quantitative comparison between camera pose output when using KinectFusion[92] and our method. Translation is measured in mm while rotation is measured in degree.

$384 \times 384 \times 384$ volume and 384 voxels/metre. In Fig. 4.16, we compare the camera pose produced by KinectFusion with that from our method. The fixed Euclidean transform between the two sets of poses is removed by aligning the two camera poses using the trajectories of two camera centres. As shown in Fig. 4.16 our tracking result is very close to the output of KinectFusion despite relying only on the local geometry of the reconstructed object.

## 4.7 Conclusion

In this chapter, we have introduced a novel probabilistic framework for simultaneous tracking and reconstruction using RGB-D data. Our method is able to track and reconstruct a small moving object in unconstrained environment, and is robust to outliers, occlusion and missing data in the input imagery. The reconstruction module in our method evolves a 3D level-set embedding function on a per-voxel inside/outside probability volume, which is learned incrementally online. The probabilistic formulation of the reconstruction allows the introduction of a shape prior into the 3D shape evolution, and in turn this permits initialization of the whole tracking-reconstruction from simple 3D models. The shape evolution comprises independent per-voxel operations and can easily be implemented in a massively parallel fashion.

In the demanded stroke rehabilitation system, our tracking algorithm with RGB-D data is able to robustly track the control object in patients' hand, however, in order to track the patients feet, there still remain a few issues to be solved. First, two interacting feet usually share identical appearance, which can lead two instances of independent single object trackers being confused when two feet are close together. Secondly, the proposed single object tracker does not impose any physical constraint on the position of the object. This is a valid assumption when a sin-

gle object is moving free space. However, when tracking multiple objects, objects should not occupy the same space. In Chapter 5 we extend the current tracking algorithm again, to simultaneously track multiple objects while considering the physical constraints.

# 5

# 3D Tracking of Multiple Object with Identical Appearance

*A shortcoming of current approaches to 3D tracking of objects using both appearance and depth data is that they are confined to single objects. Multiple object tracking can be attempted with replicated single trackers, but this neglects the constraint that objects should not occupy the same space, and fails when multiple objects have highly similar appearance. In this chapter we develop a graphical model which takes account of similarity and proximity and leads to robust real-time tracking of multiple objects from RGB-D data, without recourse to bolt-on collision detection. An early version of the work has been published in Ren et al. [112].*

## 5.1 Introduction

A key limitation of the colour-depth 3D tracker described in Chapter 4 is its focus on *single* objects. One straightforward approach to tracking multiple objects would be to replicate single object trackers. However, this straightforward extension would be naive, ignoring two obvious pitfalls. First is similarity in appearance,

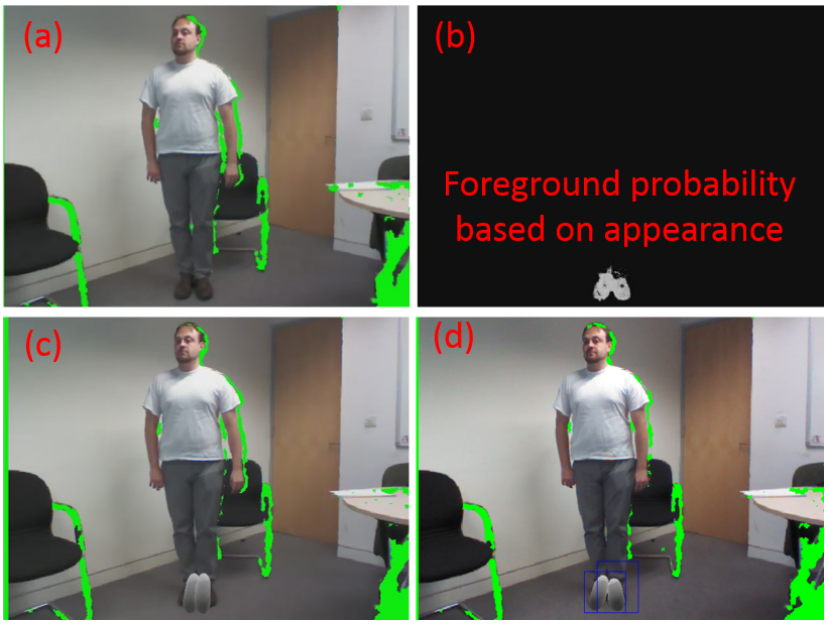



**Figure 5.1:** Sample results when using two independent single object trackers and the proposed multi-object tracker to track two feet with identical appearance. (a) input RGB-D image. (b) Foreground probability. (c) Failed tracking result using two single-object tracker. (d) Correct tracking result using the proposed multi-object tracker.

multiple objects may have similar shape and colour (for example, cars are usually followed by more cars, not by elephants, hands and feet come in pairs.) Second is the physical constraint between multiple bodies, that they must not intersect one another.

As an example of this problem, when we use two independent instances of the single-object tracker of Chapter 4 to track feet (as required in certain rehabilitation exercises). The trackers fail as soon as the two feet move close together. Fig. 5.1) illustrates what happens to the independent single-object trackers and the proposed multi-object tracker when two feet with identical appearance and shape move next to each other.

In this chapter we address these two problems, proposing a colour-depth 3D tracker that can recover the 3D pose of multiple objects with *identical* appearance, while preventing them from intersecting. Section 5.2 gives an overview of related

work. Section 5.3 describes our probabilistic formulation of the multiple object tracking problem. In Section 5.4, we describe the performance our implementation and in Section 5.5 we provide experimental insight into its operation. Conclusions are drawn in Section 5.6.

## 5.2 Related Work

Multi-object tracking and occlusion handling are two strongly connected problems. The respective literature includes various approaches that tackle both problems simultaneously. The majority of previous works are in 2D image domain, tracking 2D shapes or bounding boxes. Such methods are, by construction, limited to the estimation of the 2D poses. Although some exceptions of 2D tracking methods consider depth up to a level of layer ordering, they are still not suitable for our purpose of tracking full 6 DoF poses of 3D objects. A broader review of the general multi-target tracking has been made in Chapter 2 and in this section, we focus on reviewing recent works of full 6-DoF 3D multi-object tracking.

The work by Kim *et al.* [56] and Kyriazis and Argyros [67, 68] are the most relevant to the problem of recovering 3D poses of multiple objects across time. Kim *et al.* [56] performed simultaneous camera and multi object pose estimation in real-time using only colour image as input. First, all objects are placed statically in the scene and the system is initialized with a PTAM-like (PTAM, Klein and Murray [58]) structure-from-motion reconstruction step. A point cloud representation of the whole scene (including the objects and the static environment) is generated by triangulating matched SIFT points on each key frame. Secondly, the use specifies the object by drawing 3D boxes on a key frame and the object model is then represented by a set of coloured 3D points on the surface of the 3D boxes. Finally, at each frame, SIFT features are first used for object detection, then RANSAC is used to

compute the object pose that best matches the object model and the detected SIFT feature. However,the bottom-up nature of the work only allows for limited robustness and extensibility. With the planar model representation used, only cube-like objects can be tracked in the proposed system.

In Kyriazis and Argyos [67], the authors track multiple objects and a moving hand in the scene using RGB-D input in real-time. However, the basic assumption of the work - that there is a single actor - limits the system to only being able to track the motion of the object that is currently manipulated by the moving hand while the rest of the scene is static. More recently, in Kyriazis and Argyos [68], the author extend the method in [67] to track the poses of multiple 3D objects and two hand simultaneously using a "Ensemble of Collaborative Trackers" (ECT). A set of independent single trackers is initialized, and the tracking results of all trackers are fed to a global memory to allow all trackers to collaborate in the tracking in later frames. However, the hypothesis-test framework of the system requires discriminative appearance models for successful tracking of multiple objects, making it not suitable for the purpose of tracking of objects which share identical appearance.

Physical constraints in 3D object tracking are usually enforced by reducing the number of degrees of freedom in the state. An elegant example of tracking of always-connected objects (or sub-parts) in this way is given by Drummond and Cipolla [39]. However, when tracking multiple independently moving objects, physical constraints are introduced suddenly and intermittently by the collision of objects: they cannot be conveniently enforced by dof reduction. Indeed, rather few works explicitly model the physical collision between objects. In Oikonomidis *et al.* [97], the authors track two interacting hands with Kinect input. A penalty term measuring the inter-penetration of fingers is introduced to invalidate impossible articulated poses. In Kyriazis and Argyros [96, 67] a hand and a moving object are simultaneously tracked, and invalid configurations similarly penalized. In both cases the measure used is the minimum magnitude of 3D translation required to

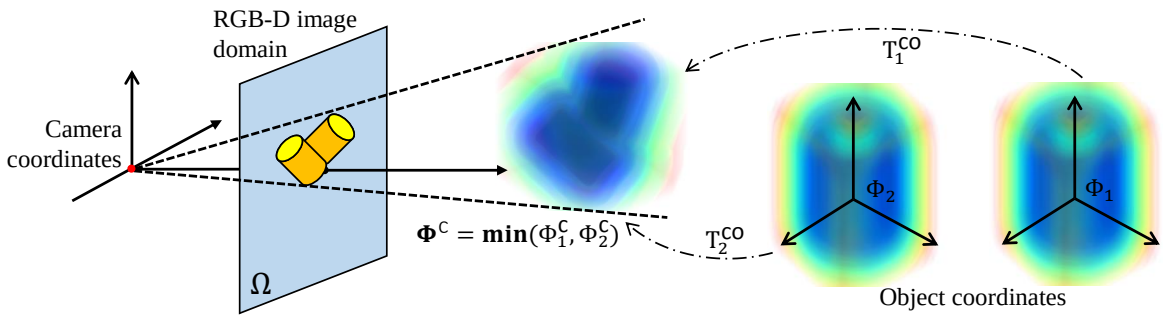



**Figure 5.2:** Illustration of the geometry of the scene and the fusion of multiple SDFs in camera coordinates.

eliminate intersection of the two objects, a measure computed using the Open Dynamic Engine library (due to Smith [131]). In contrast, in our proposed method the collision constraint is more naturally enforced through a probabilistic generative model, without the need of an additional physics simulation engine.

## 5.3 3D tracking of multiple objects

The theoretical underpinning of our multi-object tracker is again a probabilistic model, based on the one from Chapter 4. After introducing the graphical model and scene geometry in §5.3.1, we detail inference on the graphical model in §5.3.2 and §5.3.3. The optimization method and the online learning of appearance models are summarized in §5.3.4.

### 5.3.1 Generative Model and scene geometry

The scene geometry and notations for multi-object tracking are illustrated in Fig. 5.2. The $M$ objects being tracked are represented by their SDFs $\{\Phi_1 \dots \Phi_M\}$. Given the poses $\{\boldsymbol{p}_1 \dots \boldsymbol{p}_M\}$ at a particular time the shapes are transformed into the camera coordinates, where they are fused into a single shape union $\boldsymbol{\Phi}^c$. The formulation of

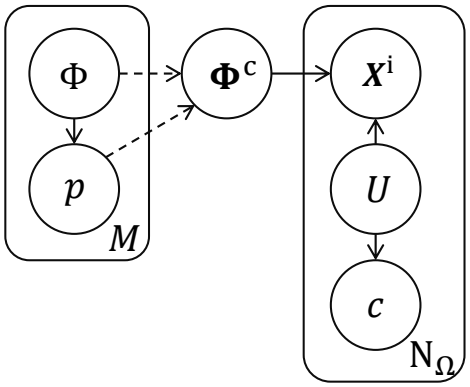



**Figure 5.3:** Graphical model for our multi-object tracker.

the shape union is explained later in the section. The same foreground/background appearance model $P^{\mathrm{f}}, P^{\mathrm{b}}$ as described in Chapter 4 is adopted in our multi-object tracking method. The RGB-D image is assumed to be generated from the shape union and the appearance model.

The objective of multi-object tracking is to find the optimal sequence of sets of poses $\{\boldsymbol{p}_{1,t} \dots \boldsymbol{p}_{M,t}\}_{1..T}$ given the set of object shapes $\{\Phi_1 \dots \Phi_M\}$ and observed RGB-D images $\{\Omega_1 ... \Omega_T\}$:

$$\max_{\{\boldsymbol{p}_{1,t} \cdots \boldsymbol{p}_{M,t}\}_{1..T}} P(\{\boldsymbol{p}_{1,t} \dots \boldsymbol{p}_{M,t}\}_{1..T} | \Phi_1 \dots \Phi_M, \Omega_1 \dots \Omega_t) \ . \tag{5.1}$$

We do not assume a motion model, hence all set of poses in the sequence are independently estimated at each time step. We drop the index $t$ for clarity, writing Eqn. 5.1 as.

$$\max_{\boldsymbol{p}_1 \cdots \boldsymbol{p}_M} P(\boldsymbol{p}_1 \dots \boldsymbol{p}_M | \Phi_1 \dots \Phi_M, \Omega) \tag{5.2}$$

The graphical model for a single pixel in RGB-D image is shown in Fig. 5.3. When tracking multiple objects in the scene, the generated RGB-D image $\Omega$ should be conditionally dependent on the set of 3D object shapes $\{\Phi_1 \dots \Phi_M\}$ and their corresponding poses $\{\boldsymbol{p}_1 \dots \boldsymbol{p}_M\}$. To make the generative process more intuitive,

we introduce an intermediate variable $\mathbf{\Phi}^{\mathsf{c}}$, which is the union of all 3D object shapes in *camera coordinates*. The generative process is as follows. The set of 3D shapes and their corresponding set of poses first generate a 'shape union' $\mathbf{\Phi}^{\mathsf{c}}$ in camera coordinates. Then, for each pixel location, the depth is drawn from the foreground/background model $U$ (same as in Chapter 4) and the shape union $\mathbf{\Phi}^{\mathsf{c}}$ and the color is drawn from the appearance model $P(c|U)$. Note that when the number of objects $M = 1$ the generative model deflates gracefully to the single object case given in Chapter 4.

The joint probability of the graphical model is

$$P(\Phi_1 \dots \Phi_M, \boldsymbol{p}_1 \dots \boldsymbol{p}_M, \mathbf{\Phi}^{\mathsf{c}}, \boldsymbol{X}^{\mathsf{i}}, U, c) = \tag{5.3}$$
$$P(\Phi_1 \dots \Phi_M) P(\mathbf{\Phi}^{\mathsf{c}} | \Phi_1 \dots \Phi_M, \boldsymbol{p}_1 \dots \boldsymbol{p}_M) P(\boldsymbol{X}^{\mathsf{i}}, U, c | \mathbf{\Phi}^{\mathsf{c}}) P(\boldsymbol{p}_1 \dots \boldsymbol{p}_M | \Phi_1 \dots \Phi_M) \ ,$$

where

$$P(\boldsymbol{X}^{\mathsf{i}}, U, c | \mathbf{\Phi}^{\mathsf{c}}) = P(\boldsymbol{X}^{\mathsf{i}} | U, \mathbf{\Phi}^{\mathsf{c}}) P(c|U) P(U) \ . \tag{5.4}$$

Since the generation of the shape union $\mathbf{\Phi}^{\mathsf{c}}$ is deterministic given the set of shapes and poses, we drop the term $P(\mathbf{\Phi}^{\mathsf{c}} | \Phi_1 \dots \Phi_M, \boldsymbol{p}_1 \dots \boldsymbol{p}_M)$. By marginalizing over the foreground/background model $U$, the posterior distribution of the set of poses $\{\boldsymbol{p}_1 \dots \boldsymbol{p}_M\}$ given all object shapes $\{\Phi_1 \dots \Phi_M\}$ for a single pixel $\{\boldsymbol{X}^{\mathsf{i}}, c\}$ follows

$$P(\boldsymbol{p}_1 \dots \boldsymbol{p}_M | \boldsymbol{X}^{\mathsf{i}}, c, \Phi_1 \dots \Phi_M) \propto P(\boldsymbol{X}^{\mathsf{i}}, c | \mathbf{\Phi}^{\mathsf{c}}) P(\boldsymbol{p}_1 \dots \boldsymbol{p}_M | \Phi_1 \dots \Phi_M) \ , \tag{5.5}$$

where

$$P(\boldsymbol{X}^{\mathsf{i}}, c | \mathbf{\Phi}^{\mathsf{c}}) = \sum_{u \in \{f,b\}} P(\boldsymbol{X}^{\mathsf{i}} | U = u, \mathbf{\Phi}^{\mathsf{c}}) P(c | U = u) P(U = u) \ . \tag{5.6}$$

We refer the first term $P(\boldsymbol{X}^{\mathsf{i}}, c | \mathbf{\Phi}^{\mathsf{c}})$, which describes how likely a pixel — both its

colour value and the pixel location — is generated by the current shape union, as the *data term*; and the second term $P(\boldsymbol{p}_1 \dots \boldsymbol{p}_M | \Phi_1 \dots \Phi_M)$, which puts a prior on the set of poses given the set of shapes, as the *physical constraint term*.

### 5.3.2 Data term

The first part of the data term $P(\boldsymbol{X}^{\mathsf{i}} | U = u, \boldsymbol{\Phi}^{\mathsf{c}})$ is similarly defined as the per-pixel likelihood term with normalized smoothed delta function $\delta^{\mathsf{on}}$ and a smoothed and shifted Heaviside function $H^{\mathsf{out}}$

$$P(\boldsymbol{X}^{\mathsf{i}} | U = f, \boldsymbol{\Phi}^{\mathsf{c}}) = \frac{\delta^{\mathsf{on}}(\boldsymbol{\Phi}^{\mathsf{c}}(\boldsymbol{X}^{\mathsf{c}}))}{\eta_f^{\mathsf{c}}} \tag{5.7}$$

$$P(\boldsymbol{X}^{\mathsf{i}} | U = b, \boldsymbol{\Phi}^{\mathsf{c}}) = \frac{H^{\mathsf{out}}(\boldsymbol{\Phi}^{\mathsf{c}}(\boldsymbol{X}^{\mathsf{c}}))}{\eta_b^{\mathsf{c}}} \tag{5.8}$$

$$\eta_f^{\mathsf{c}} = \sum_{j=1}^{N_\Omega} \delta^{\mathsf{on}}(\boldsymbol{\Phi}^{\mathsf{c}}(\boldsymbol{X}_j^{\mathsf{c}})) \tag{5.9}$$

$$\eta_b^{\mathsf{c}} = \sum_{j=1}^{N_\Omega} H^{\mathsf{out}}(\boldsymbol{\Phi}^{\mathsf{c}}(\boldsymbol{X}_j^{\mathsf{c}})) \ , \tag{5.10}$$

where $\boldsymbol{X}^{\mathsf{c}}$ is unprojected from $\boldsymbol{X}^{\mathsf{i}}$ in the camera coordinates, following

$$\boldsymbol{X}^{\mathsf{c}} = \mathtt{K}^{-1} \boldsymbol{X}^{\mathsf{i}} \ . \tag{5.11}$$

The functions $\delta^{\mathsf{on}}$ and $H^{\mathsf{out}}$ are the same smoothed Dirac delta function and the smoothed Heaviside step function as in Chapter 4. The per-pixel labelling $P(U{=}f)$ and $P(U{=}b)$ follows uniform distributions:

$$P(U{=}f) = \frac{\eta_f^{\mathsf{c}}}{\eta^{\mathsf{c}}} \ , \qquad P(U{=}b) = \frac{\eta_b^{\mathsf{c}}}{\eta^{\mathsf{c}}} \ , \qquad \eta^{\mathsf{c}} = \eta_f^{\mathsf{c}} + \eta_b^{\mathsf{c}} \ . \tag{5.12}$$

Substituting Eqn. 5.7∼5.12 into Eqn. 5.6, we obtain the likelihood of the shape union for a single pixel

$$P(\boldsymbol{X}^{\mathsf{i}}, c|\boldsymbol{\Phi}^{\mathsf{c}}) = P^{\mathsf{f}}\delta^{\mathsf{on}}(\boldsymbol{\Phi}^{\mathsf{c}}(\boldsymbol{X}^{\mathsf{c}})) + P^{\mathsf{b}}H^{\mathsf{out}}(\boldsymbol{\Phi}^{\mathsf{c}}(\boldsymbol{X}^{\mathsf{c}})), \tag{5.13}$$

where $P^{\mathsf{f}}{=}P(c|U{=}f)$ and $P^{\mathsf{b}}{=}P(c|U{=}b)$.

Now we describe how the shape union $\boldsymbol{\Phi}^{\mathsf{c}}$ is formed given the set of shapes and poses. Assuming multiple objects do not intersect, their SDFs can be fused together by taking the minimum value of all SDFs. Given a set of object shapes $\{\Phi_1 \dots \Phi_M\}$ and their corresponding set of poses $\{\boldsymbol{p}_1 \dots \boldsymbol{p}_M\}$, we transform each object shape $\Phi_m$ into camera coordinates as $\Phi_m^{\mathsf{c}}$ using $\mathrm{T}^{\mathsf{co}}$. Then the object shapes in *camera coordinates* $\{\Phi_1^{\mathsf{c}} ... \Phi_M^{\mathsf{c}}\}$ are fused into a single SDF $\boldsymbol{\Phi}^{\mathsf{c}}$ with a minimum function. We use an analytical relaxation to approximate the minimum function as

$$\boldsymbol{\Phi}^{\mathsf{c}} = \min(\Phi_1^{\mathsf{c}}, \Phi_2^{\mathsf{c}}, ..., \Phi_M^{\mathsf{c}}) \approx -\frac{1}{\alpha}\log\sum_{m=1}^{M}\exp\{-\alpha\Phi_m^{\mathsf{c}}\}\ . \tag{5.14}$$

where $\alpha$ is a constant parameter that controls the smoothness of the approximation. Theoretically, a larger $\alpha$ gives a better approximation of the minimum function, but empirically, we find that smaller $\alpha$ gives wider based of convergence for the tracker. We use $\alpha{=}2$ in our case. The per-voxel values of $\Phi_m$ and $\Phi_m^{\mathsf{c}}$ are:

$$\Phi_m^{\mathsf{c}}(\boldsymbol{X}^{\mathsf{c}}) = \Phi_m(\mathrm{T}^{\mathsf{oc}}\begin{bmatrix}\boldsymbol{X}^{\mathsf{c}}\\ 1\end{bmatrix}) = \Phi_m(\boldsymbol{X}^{\mathsf{o}}). \tag{5.15}$$

Using Eqn. 5.15 in Eqn. 5.14:

$$\begin{aligned}\boldsymbol{\Phi}^{\mathsf{c}}(\boldsymbol{X}^{\mathsf{c}}) &= -\frac{1}{\alpha}\log\sum_{m=1}^{M}\exp\{-\alpha\Phi_m(\mathrm{T}_m^{\mathsf{oc}}\boldsymbol{X}^{\mathsf{c}})\} \\ &= -\frac{1}{\alpha}\log\sum_{m=1}^{M}\exp\{-\alpha\Phi_m(\boldsymbol{X}_m^{\mathsf{o}})\}\ ,\end{aligned} \tag{5.16}$$

where $\boldsymbol{X}_m^{\circ}$ is the transformation of $\boldsymbol{X}^{\mathsf{c}}$ in the $m$-th object coordinate system. The likelihood of a single pixel now follows

$$P(\boldsymbol{X}^{\mathsf{i}}, c|\boldsymbol{\Phi}^{\mathsf{c}}) = \tag{5.17}$$
$$P^{\mathsf{f}}\delta^{\mathsf{on}}\left(-\frac{1}{\alpha}\log\sum_{m=1}^{M}\exp\{-\alpha\Phi_m(\boldsymbol{X}_m^{\circ})\}\right) + P^{\mathsf{b}}H^{\mathsf{out}}\left(-\frac{1}{\alpha}\log\sum_{m=1}^{M}\exp\{-\alpha\Phi_m(\boldsymbol{X}_m^{\circ})\}\right) .$$

Assuming pixel-wise independence, the log-likelihood of the whole RGB-D image gives us the data term of the overall energy function

$$\mathcal{E}_{\text{data}} = -\log P(\Omega|\boldsymbol{\Phi}^{\mathsf{c}}) = -\sum_{j=1}^{N_\Omega}\log P(\boldsymbol{X}_j^{\mathsf{i}}, c|\boldsymbol{\Phi}^{\mathsf{c}}) . \tag{5.18}$$

To optimize the energy function, we require the derivatives of this term w.r.t. the change of the set of pose parameters $\boldsymbol{\theta}^*=\{\boldsymbol{p}^*{}_1 \dots \boldsymbol{p}^*{}_M\}$ in which each $\boldsymbol{p}^*{}_m$, recall, is a 6-vector. They are found from (here we drop the pixel index $j$ for clarity)

$$\frac{\partial\mathcal{E}_{\text{data}}}{\partial\boldsymbol{\theta}^*} = -\sum_{\boldsymbol{X}^{\mathsf{i}}\in\Omega}\left\{\frac{P^{\mathsf{f}}\frac{\partial\delta^{\mathsf{on}}}{\partial\boldsymbol{\Phi}^{\mathsf{c}}} + P^{\mathsf{b}}\frac{\partial H^{\mathsf{out}}}{\partial\boldsymbol{\Phi}^{\mathsf{c}}}}{P(\boldsymbol{X}^{\mathsf{i}}, c|\boldsymbol{\Phi}^{\mathsf{c}})}\frac{\partial\boldsymbol{\Phi}^{\mathsf{c}}(\boldsymbol{X}^{\mathsf{c}})}{\partial\boldsymbol{\theta}^*}\right\} \tag{5.19}$$

where

$$\frac{\partial\boldsymbol{\Phi}^{\mathsf{c}}(\boldsymbol{X}^{\mathsf{c}})}{\partial\boldsymbol{\theta}^*} = -\frac{1}{\alpha}\sum_{m=1}^{M} w_m\frac{\partial\Phi_m}{\partial\boldsymbol{X}_m^{\circ}}\frac{\partial\boldsymbol{X}_m^{\circ}}{\partial\boldsymbol{\theta}^*} , \tag{5.20}$$

$$w_m = \frac{\exp\{-\alpha\Phi_m(\boldsymbol{X}_m^{\circ})\}}{\sum_{k=1}^{M}\exp\{-\alpha\Phi_k(\boldsymbol{X}_k^{\circ})\}} , \tag{5.21}$$

and

$$\frac{\partial\boldsymbol{X}_m^{\circ}}{\partial\boldsymbol{\theta}^*} = \left[\frac{\partial\boldsymbol{X}_m^{\circ}}{\partial\boldsymbol{p}^*{}_1}\cdots\frac{\partial\boldsymbol{X}_m^{\circ}}{\partial\boldsymbol{p}^*{}_M}\right] \tag{5.22}$$

The derivatives $\frac{\partial\boldsymbol{X}_m^{\circ}}{\partial\boldsymbol{p}^*{}_k}$ follow the same formula as in Eqn. 3.14 from Chapter 3 if $m = k$, otherwise it equals to zero. We use central finite differences to pre-compute the gradients of the SDFs $\frac{\partial\Phi_m}{\partial\boldsymbol{X}_m^{\circ}}$ for each object as in Chapter 3.

Note that the derivative of this energy term has a very clear meaning. Given a pixel $\boldsymbol{X}^{\mathrm{i}}$ in the RGB-D image domain, instead of assigning this pixel deterministically to a certain object, we back-projected $\boldsymbol{X}^{\mathrm{i}}$ (i.e. $\boldsymbol{X}^{\mathrm{c}}$ in camera coordinates) into all objects' coordinate systems with the current set of poses $\{\boldsymbol{p}_1 \dots \boldsymbol{p}_M\}$. Then, a membership weight $w_m$ is automatically computed according to Eqn. 5.21. For example, consider a point in camera coordinates $\boldsymbol{X}^{\mathrm{c}}$. If the back-projection $\boldsymbol{X}^{\mathrm{o}}_m$ is close to the $m$-th object's surface ($\Phi(\boldsymbol{X}^{\mathrm{o}}_m) \approx 0$) and other back-projections $\boldsymbol{X}^{\mathrm{o}}_k$ are further away from the surfaces ($\Phi(\boldsymbol{X}^{\mathrm{o}}_k) \gg 0$), then we will find $w_m \to 1$ and the other $w_k \to 0$, with $\sum_{m=1}^{M} w_m = 1$. Thus $w_m$ can be interpreted as the probability that a pixel $\boldsymbol{X}^{\mathrm{c}}$ belongs to the $m$-th object.

### 5.3.3 Physical constraint term

Now we discuss the physical constraint term $P(\boldsymbol{p}_1 \dots \boldsymbol{p}_M | \Phi_1 \dots \Phi_M)$ in Eqn. 5.6. We decompose the joint probability of all object poses given all 3D object shapes into a product of per-pose probabilities:

$$P(\boldsymbol{p}_1 \dots \boldsymbol{p}_M | \Phi_1 \dots \Phi_M) = P(\boldsymbol{p}_1 | \Phi_1 \dots \Phi_M) \prod_{j=2}^{M} P(\boldsymbol{p}_j | \boldsymbol{p}_1 \dots \boldsymbol{p}_{j-1}, \Phi_1 \dots \Phi_M) \quad (5.23)$$

If we do not have any pose priors on any single objects, we can ignore the probability term $P(\boldsymbol{p}_1 | \Phi_1 \dots \Phi_M)$. The remaining products can be used to enforce pose-related constraints. Here we use them to avoid collisions in the tracker by defining them such that objects do not penetrate each other.

The probability $P(\boldsymbol{p}_j | \boldsymbol{p}_1 \dots \boldsymbol{p}_{j-1}, \Phi_1 \dots \Phi_M)$ is defined based on the fact that a surface point on one object should not move inside any other objects. For each object $\Phi_m$ we uniformly and sparsely sample a set of $K$ *collision points* $\mathcal{C}_m = \{\boldsymbol{C}^{\mathrm{o}}_{m,1} \dots \boldsymbol{C}^{\mathrm{o}}_{m,K}\}$ from its surface in object coordinates. At each timestep these are transformed into the camera frame as $\{\boldsymbol{C}^{\mathrm{c}}_{m,1} \dots \boldsymbol{C}^{\mathrm{c}}_{m,K}\}$ using the current pose $\boldsymbol{p}_m$.

Let us denote the partial union of SDFs $\{\Phi_1^c \dots \Phi_{m-1}^c\}$ by $\boldsymbol{\Phi}_{-m}^c$. Our proposition now is that we can write

$$P(\boldsymbol{p}_m | \{\boldsymbol{p}_1 \dots \boldsymbol{p}_{m-1}\}, \Phi_1 \dots \Phi_M) \sim \frac{1}{K} \sum_{k=1}^{K} H^{\text{out}} \left( \boldsymbol{\Phi}_{-m}^c (\boldsymbol{C}_{m,k}^c) \right) \tag{5.24}$$

where $H^{\text{out}}$ is the off-set smoothed Heaviside function defined in Chapter 4. The rationale is that if all the collision points on object $m$ lie *outside* the shape union of objects 1 to $m-1$, this quantity will tend to unity, whereas if progressively more and more of the collision points lie *inside* the partial shape union, the quantity tends to zero. The negative log-likelihood of Eqn. 5.24 gives us the second part of the overall cost

$$\mathcal{E}_{\text{coll}} = -\sum_{m=1}^{M} \log \left( \frac{1}{K} \sum_{k=1}^{K} H^{\text{out}} \left( \boldsymbol{\Phi}_{m-}^c (\boldsymbol{C}_{m,k}^c) \right) \right) . \tag{5.25}$$

### 5.3.4 Optimization and appearance learning

The overall cost is the sum of the data term and the collision constraint term

$$\mathcal{E} = \mathcal{E}_{\text{data}} + \mathcal{E}_{\text{coll}} . \tag{5.26}$$

To optimize the set of poses $\{\boldsymbol{p}_1 \dots \boldsymbol{p}_M\}$, we use the same Levenberg-Marquardt iterations and local frame pose updates as given in Chapter 3 .We also follow the same online appearance adaptation scheme as in Chapter 4 after the tracking is completed for each frame. We use the pixels that have $|\boldsymbol{\Phi}^c(\boldsymbol{X}^c)| \leqslant 3$ to compute the surface appearance model and the pixels in the immediate surrounding region of the objects to compute the background model. This involves points that best fit the surfaces of multiple objects.

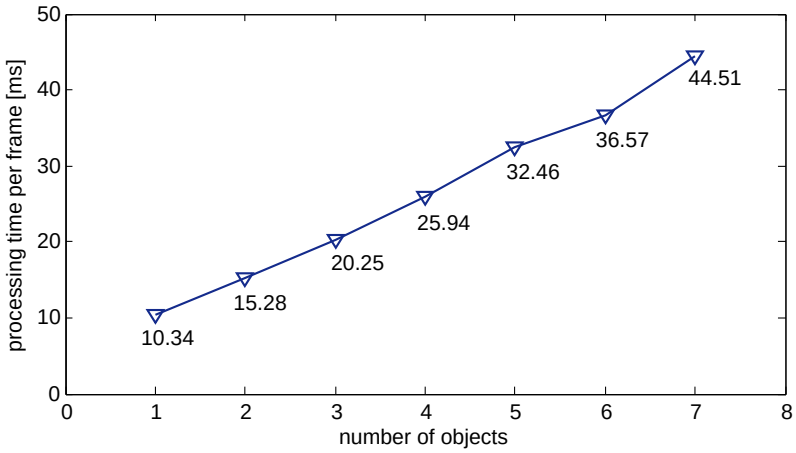



**Figure 5.4:** Quantitative evaluation of the processing speed of our tracker on CPU with respect to the number of objects

## 5.4 Implementation and performance

We have implemented our multi-object tracker on an Intel Core i7 3.4GHz CPU, where a 30Hz frame rate is maintained when tracking up to 5 objects. We have also implemented a specialized version of our tracker on GPU for 2 objects, to track patients' hands and feet. On a NVDIA GTX 680 GPU, the speed of CPU and GPU implementations is similar at 15 ms per frame because (i) we only use pixels that are close to the projection of the object in the depth image and (ii) most of the tracked objects occupy a small region of the RGB-D image. This means we only leverage a few thousand points, which does not take full advantage of the computational power of the GPU. If a larger number of points is used (i.e when tracking objects that occupy larger image regions or when using denser depth), we expect the GPU implementation to outperform the CPU implementation by a larger margin. The processing time on CPU w.r.t. the number of object is shown in Fig. 5.4. As expected, the complexity of the method grows linear in the number of objects.


**Figure 5.5:** Sample frames from our synthetic sequence for experiment.

## 5.5 Evaluation

We have performed a variety of experimental evaluations, both qualitative and quantitative. Further qualitative examples of our algorithm tracking different types of objects in real-time and under significant occlusion and missing data are provided in Video B.3 in Appendix B.

### 5.5.1 Quantitative evaluation

We begin with two sets of quantitative experiments to evaluate the proposed method. For the first (Fig. 5.6) we follow a standard benchmarking strategy from the markerless tracking literature and evaluate our tracking results on synthetic data, since ground truth information for real data is very difficult to obtain. We move two objects of known shape with similar appearance in front of a virtual camera and generate RGB-D frames. The objects periodically move further apart then closer to each other. We add zero-mean Gaussian noise to both the rendered colour and the depth images. Four sample frames from the test sequence are shown in Fig. 5.5. Using this sequence we compare the tracking accuracy of our multi-object track-

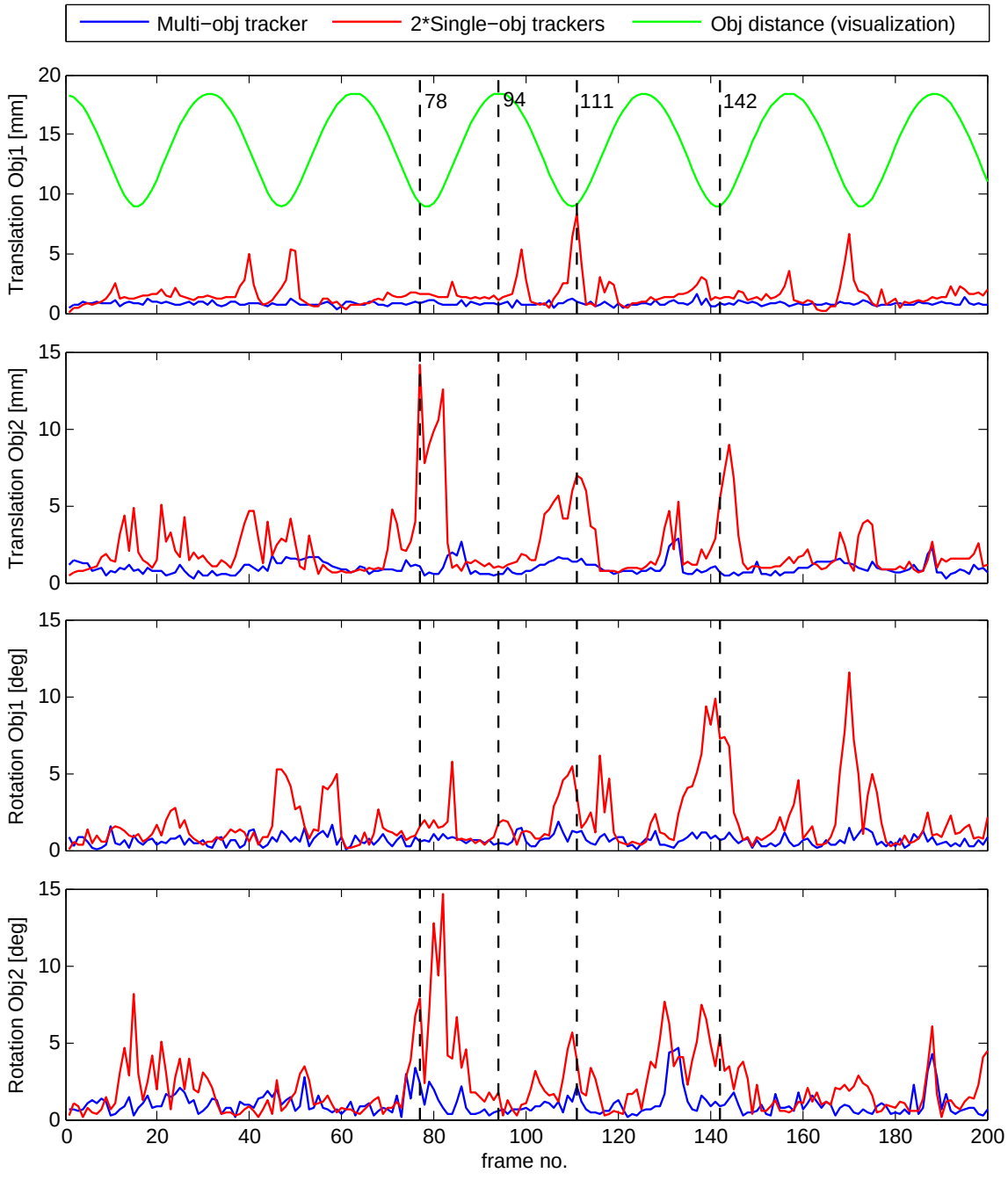



**Figure 5.6:** Comparison of pose estimation error between our multi-object tracker and two instances of the single-object tracker described in Chapter 4

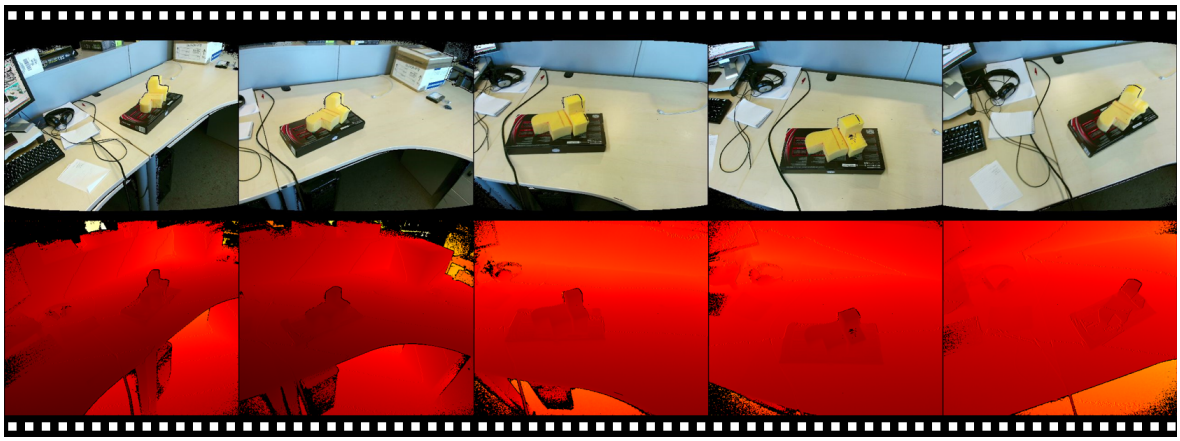


**Figure 5.7:** Sample real world RGB-D frames used in our experiment.

er with two instances of the single object tracker that we described in Chapter 4. Translations are measured in mm and rotations are measured in degrees. In the plot of Fig. 5.6 the green line shows the relative distance between the two objects. Note that this value has been scaled and offset for visualization. The dotted vertical lines with numbers correspond to the frames shown in Fig. 5.5. It can be seen that when the two objects with similar appearance are neither overlapping nor close (*e.g.* frame 94), using either two single object trackers or our multi-object tracker provide similarly accurate results. However, once the two objects move close together (*e.g.* frames 78, 111 and 142), the two separate single object trackers produce a very large error. The single object tracker fails to model the pixel membership, which leads to an incorrect pixel association when the two objects are close together. Our soft pixel membership in the multi-object tracker solves this problem.

The second quantitative experiment (Fig. 5.8) makes a similar comparison, but with real imagery. As before, it is difficult to obtain the *absolute* ground truth pose of the objects, and instead we measure the consistency of the *relative* pose between two static objects, while moving the camera around. The camera is always aimed towards the two objects. Example frames are shown in Fig. 5.7. When the two recovered poses are accurate we would expect consistent relative translation and rotation through the whole sequence. As shown in Fig. 5.8, our multi-object tracker

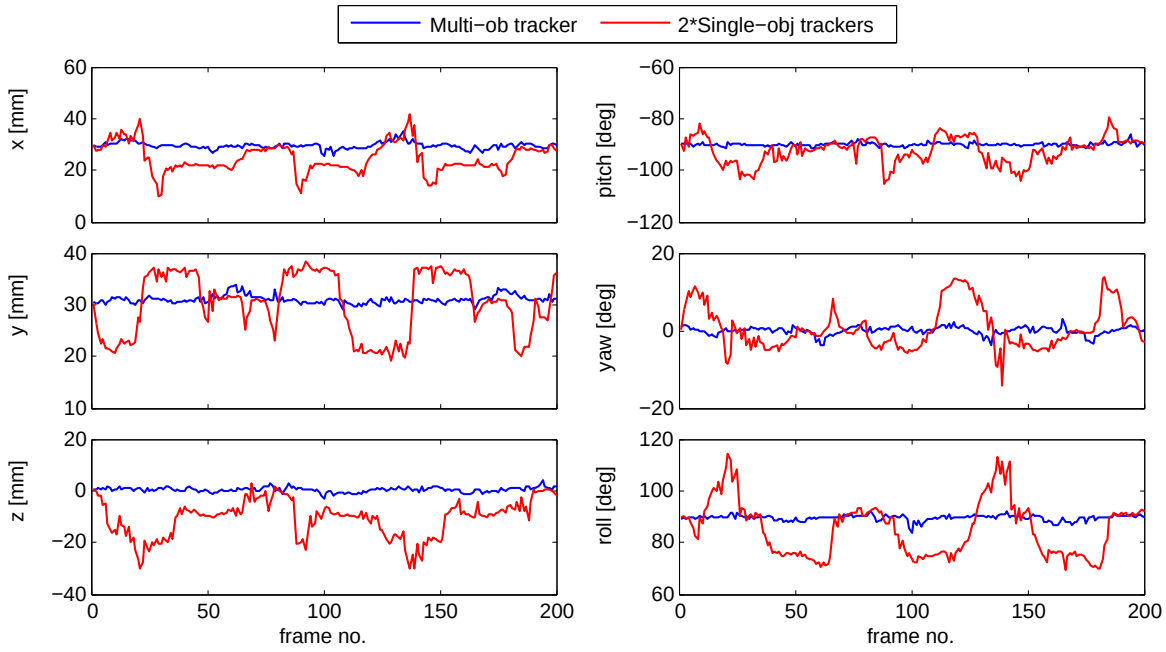



**Figure 5.8:** Comparison of the variance in relative pose estimation between our multi-object tracker and two instances of the single-object tracker

is able to recover much more consistent relative translation and rotation than two independent one-body trackers of Chapter 4.

### 5.5.2 Qualitative evaluation

We use five challenging real sequences to demonstrate the robust performance of our multi-object tracker. The results are shown in Figs. 5.9∼5.13. For Figs. 5.9 to 5.12, rows 1 and 2 show the colour and the depth image inputs and row 3 shows the per-pixel foreground probability $P^{\mathrm{f}}$. Row 4 shows the per-pixel membership weight $w_m$, the magenta colour ($w_1 \gg 0.5$, $w_2 \ll 0.5$) and cyan colours ($w_2 \gg 0.5$, $w_1 \ll 0.5$) corresponding to the two objects, and the blue coloured pixels have ambiguous membership ($w_1 \approx w_2 \approx 0.5$) . Row 5 shows the final tracking result.

In Figs. 5.9 and 5.10, we use *well-carpentered* models for tracking. Fig. 5.9 shows the tracking of two pieces of sponge with identical shape and appearance model-

s. Our tracker is able to track through heavy occlusions and handle challenging motions. In Fig. 5.10 we simultaneously track a white cup and a white ball to demonstrate the effectiveness of the physical collision constraint. Note that even though there is no depth observation from the ball owing to significant occlusion from the cup, our algorithm can still estimate the location of the ball using the physical constraint alone.

In contrast, Figs. 5.11 and 5.12 show our tracker using reconstructed and hence somewhat *inaccurate* 3D shapes for tracking. First in Fig. 5.11, we track two interacting hands (fixed hand articulation pose). Even though the hand models do not fit the observation perfectly, the tracker still recovers the poses of both hands by finding the local minimum that best explains the colour and depth observations.

In Fig. 5.12 we track two interacting feet with a pair of approximate shoe models. Throughout most of the sequence our tracker successfully recovers the two poses. However, we do also encounter two failure cases here. The first one is shown in column 4 of Fig. 5.12, where the shoe is incorrectly rotated. This happens because the 3D model is somewhat rotationally ambiguous around its long axis. The second failure case can be seen in column 6. Here, the ground pixels (*i.e.* the black shadow) have very high foreground probability, as can be clearly seen in row 3. With most of one foot occluded, the tracker incorrectly tries to fit the model to the pixels with high foreground probability, which leads to failure.

Note however that, our tracker does automatically recover from both failure cases. As soon as the feet move out of the ambiguous position, the multi-object tracker use the previous incorrect pose as initialization and converges to the correct pose at the current frame.

Finally, in Fig. 5.13 we show a challenging sequence where 5 pieces of toy bricks are tracked to prove that the proposed tracker is able to handle larger number of objects. The set of toy bricks has difference shapes but identical appearance. The

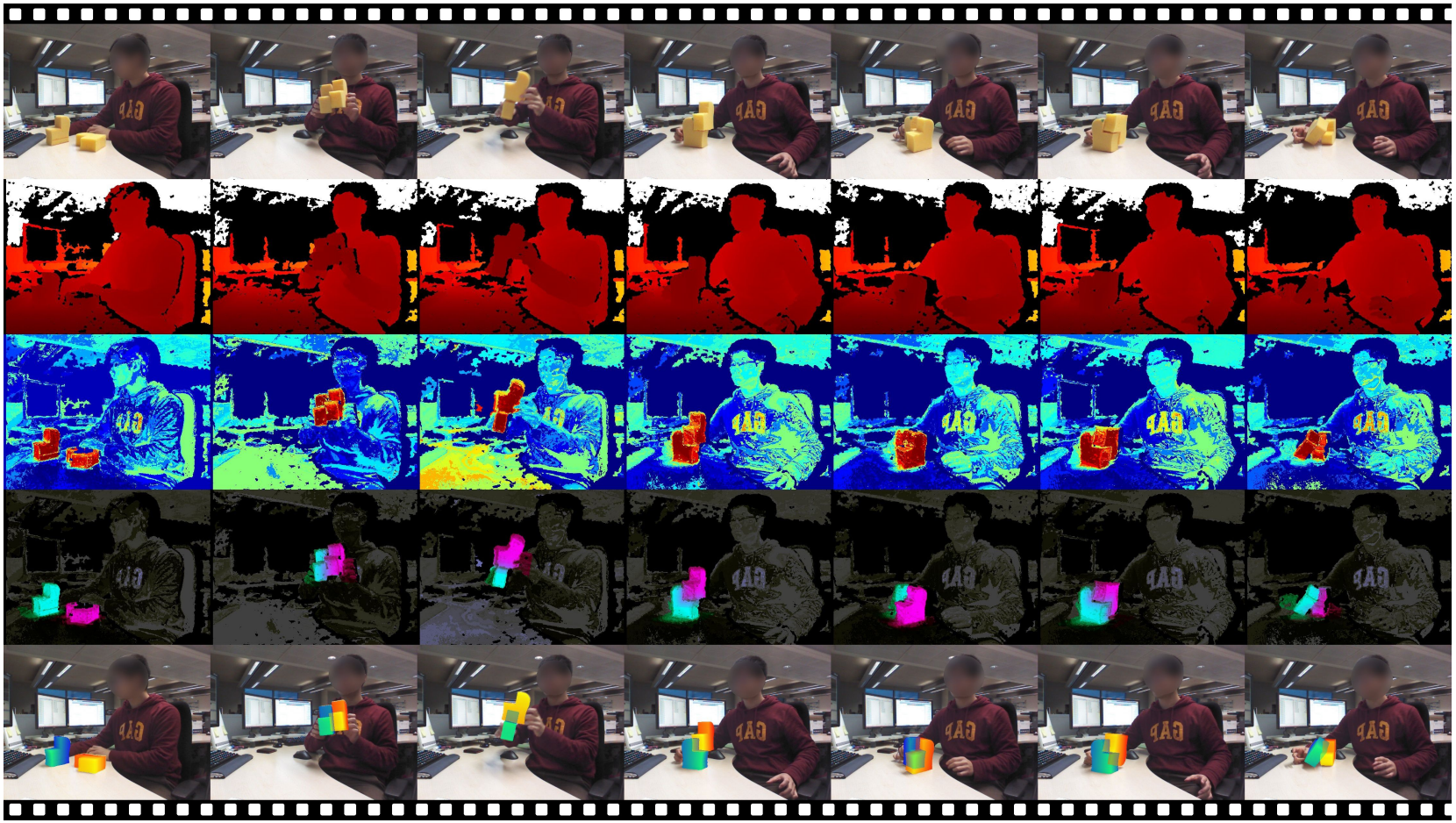


**Figure 5.9:** Film strip showing our algorithm tracking two pieces of sponge with the accurate object models. Row 1,2 show the colour and the depth image inputs. Row 3 shows the per-pixel foreground probability $P_f$. Row 4 shows the per-pixel membership weight $w_i$, magenta and cyan colour correspond to the two objects, the blue coloured pixels are with ambiguous membership. In row 5, we show the tracking result. Whole sequence see Video B.3 in Appendix B.

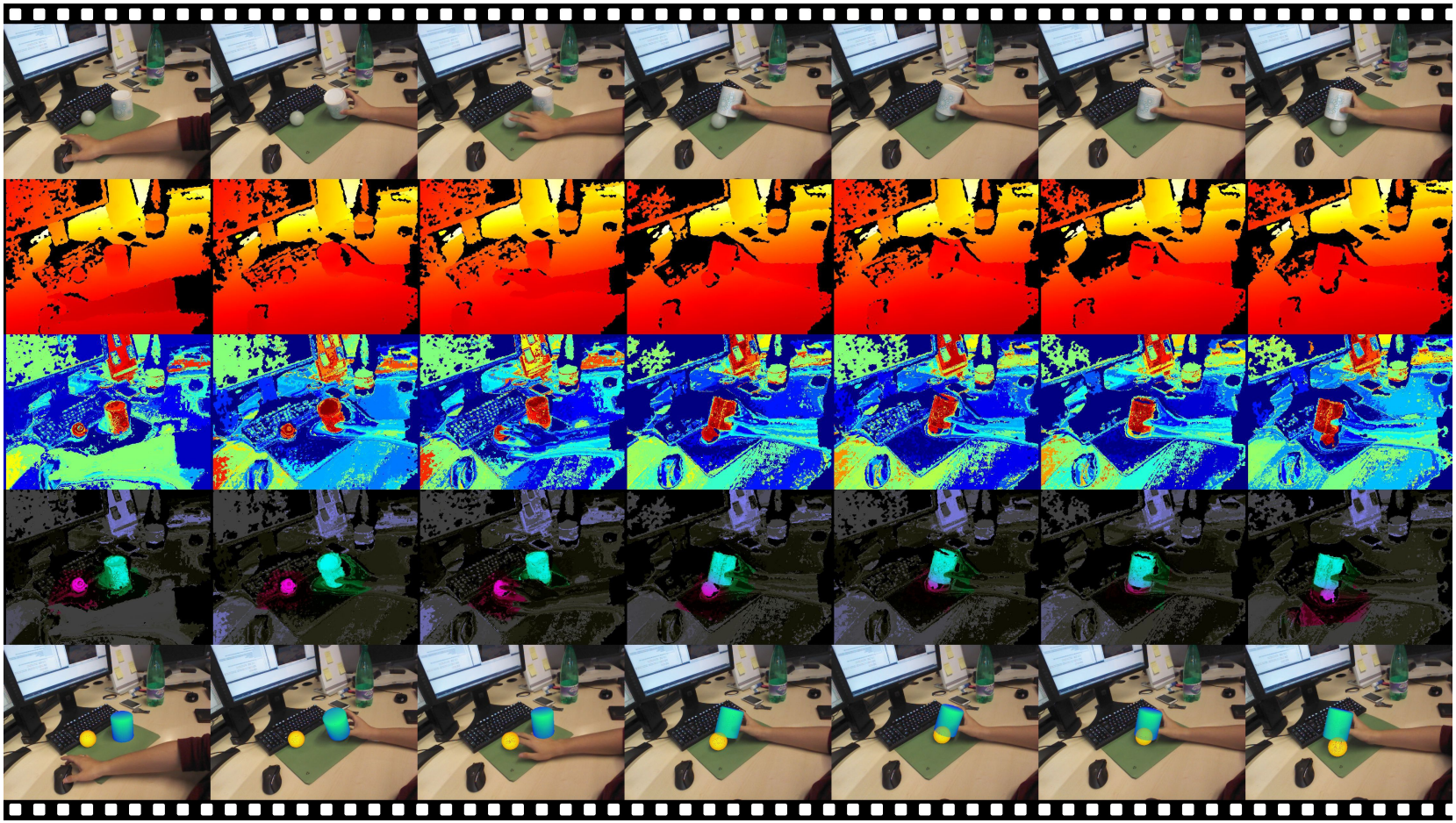


**Figure 5.10:** Film strip showing our algorithm tracking a ball and a cup with the accurate object models. Row 1,2 show the colour and the depth image input. Row 3 shows the per-pixel foreground probability $P_f$. Row 4 shows the per-pixel membership weight $w_i$, magenta and cyan colour correspond to the two objects, the blue coloured pixels are with ambiguous membership. In row 5, we show the tracking result. Whole sequence see Video B.3 in Appendix B.

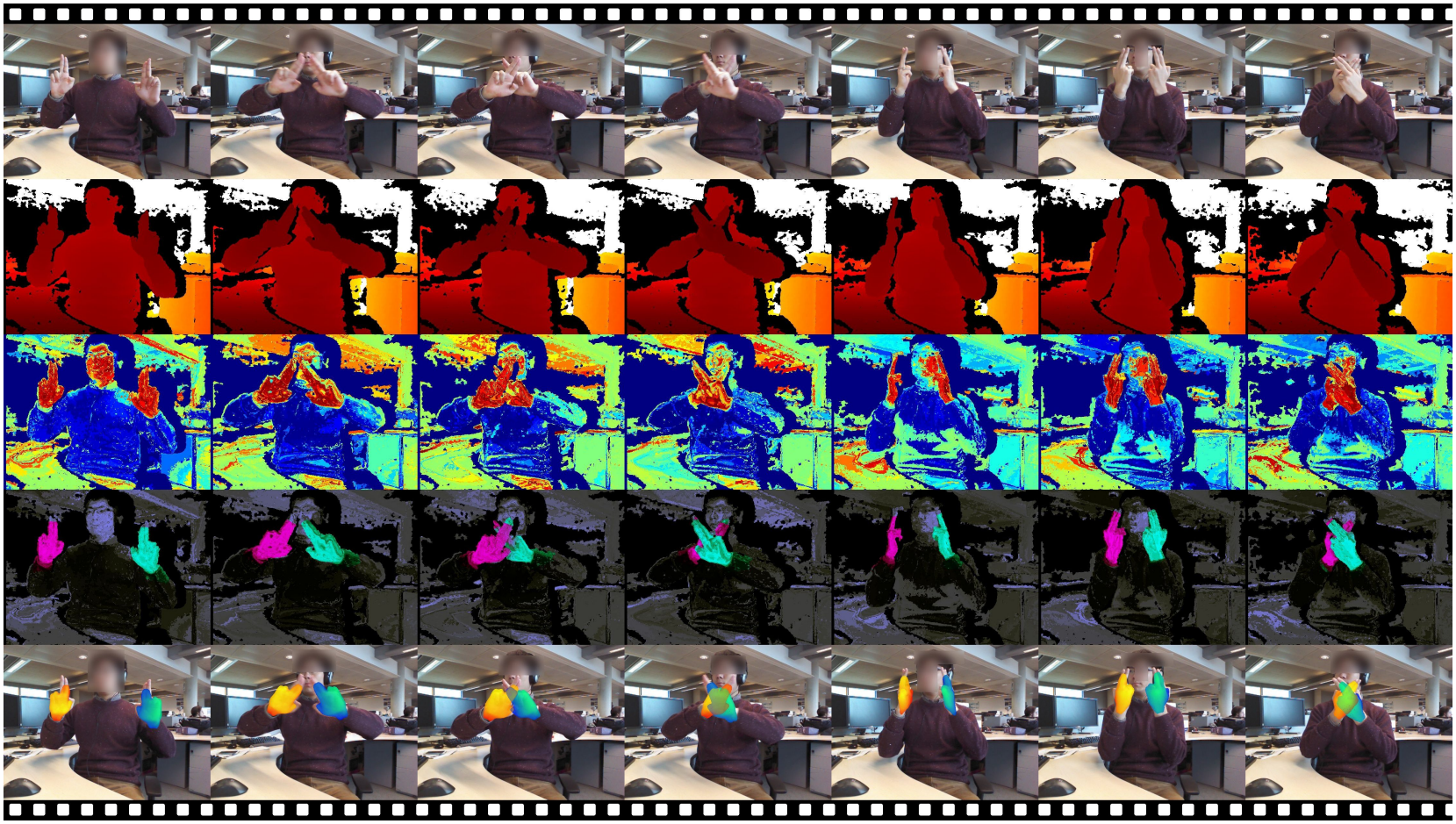


**Figure 5.11:** Film strip showing our algorithm tracking two interacting hands with two *inaccurate* models. Row 1,2 show the colour and the depth image input. Row 3 shows the per-pixel foreground probability $P_f$. Row 4 shows the per-pixel membership weight $w_i$, magenta and cyan colour correspond to the two objects, the blue coloured pixels are with ambiguous membership. In row 5, we show the tracking result. Whole sequence see Video B.3 in Appendix B.

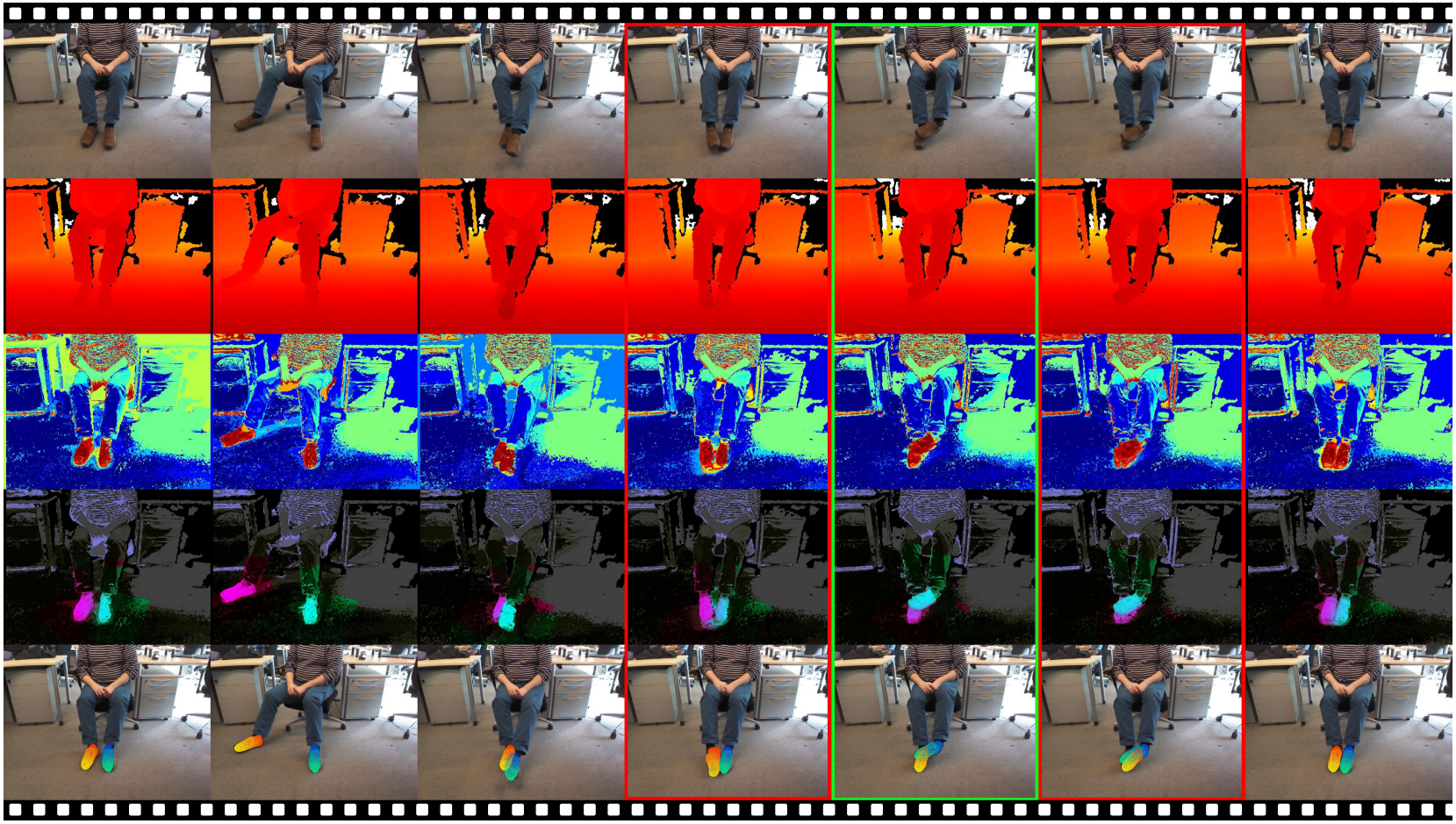


**Figure 5.12:** Film strip showing our algorithm tracking two interacting feet with two *inaccurate* models. Row 1,2 show the colour and the depth image input. Row 3 shows the per-pixel foreground probability $P_f$. Row 4 shows the per-pixel membership weight $w_i$, magenta and cyan colour correspond to the two objects, blue coloured pixels are with ambiguous membership. In row 5, we show the tracking result. The tracker failed on the frames of column 4 and 6 but finally recovered on the frames of column 7. Whole sequence see Video B.3 in Appendix B.

**Figure 5.13:** Film strips showing a challenging sequence where 5 pieces of toy bricks with identical colour are tracked. The top sequence shows the tracking result rendered on the colour image and the sequence below shows the original colour images. Our multi-object tracker manage to track through the whole sequence without tracking failure. Whole sequence see Video B.3 in Appendix B.

top sequence shows the tracking result and the bottom sequence shows the original colour input. In spite of the heavy self-occlusion and the occlusion introduced by hands, the multi-object tracker is able to track robustly.

## 5.6 Conclusions

This Chapter has presented a novel framework for tracking multiple 3D objects from a sequence of RGBD images. The formulation has several advantages over instantiating multiple individual trackers, both from a theoretical point of view and from a practical one. First, our method is particularly well suited to tracking several objects with similar or identical appearance, which is a common case in many applications, such as tracking cars or pairs of hands or feet. Our method is grounded in a rigorous probabilistic framework, so we naturally obtain weights that indicate the probability of individual image observations being generated by each of the tracked objects, thus implicitly solving the data association problem. Furthermore, our formulation naturally leads to what we call a physical constraint term, which allows us to specify prior knowledge about the world. We have used this term to indicate that it is unlikely that several objects occupy the same locations in 3D space. In addition to collision avoidance, the formulation would allow for generic interaction forces between objects to be modeled.

We validate our claims with several experiments, showing that the combined tracking of multiple objects exhibits superior performance over instantiating the same tracker multiple times independently. For this evaluation we used an implementation that easily tracks multiple objects at 30 Hz without the use of any GPU acceleration. Our system is therefore well suited for real time applications.

Since our tracker is region-based and currently uses simple histograms as appearance models, it is particularly well suited to objects where the texture is unin-

formative. A possible direction of research is to transfer our tracking framework to different appearance models, such as texture-based models. In line with other model based 3D trackers our approach currently also requires 3D models of the tracked objects to be known and given to the algorithm. While we do explicitly show good performance even with crude and inaccurate models, this might be considered another shortcoming to be resolved in future work. In particular dynamic objects, such as hands, could be an interesting direction, since tracking individual fingers might greatly benefit from a tracker that can deal with near-identical appearance and nicely integrated physical constraints.

# 6

# Shape Regression for Online Segmentation and Tracking

*We propose a novel regression based framework that uses online learned shape information to recover occluded 2D object contours. Our key insight is to regress the global, coarse, properties of shape from its local properties, i.e. it-s details. We do this by representing shapes using their 2D discrete cosine transforms and by regressing low frequency from high frequency harmonics. We learn this regression model using Locally Weighted Projection Regression (LWPR) which expedites online, incremental learning. After sufficient observation of a set of unoccluded shapes, the learned model can detect occlusion and recover the full shapes from the occluded ones. We demonstrate the ideas using a level-set based tracking system that provides shape and pose, however, the framework could be embedded in any segmentation-based tracking system. Our experiments demonstrate the efficacy of the method on a variety of objects using both real data and artificial data. A previous version of this work has been published by Ren et al. [116, 111]*

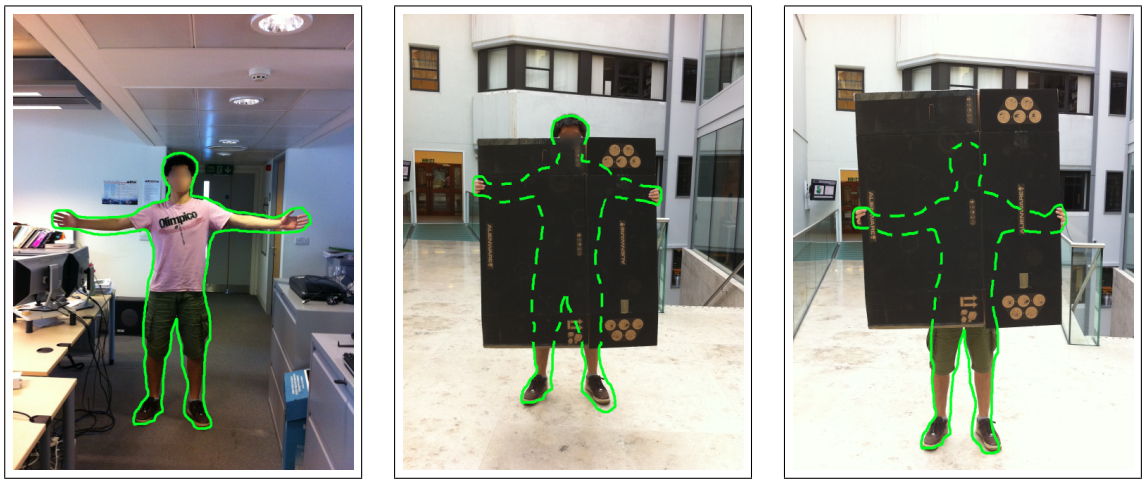


**Figure 6.1:** Left: full human shape, from which we learn the relationship between local properties and global ones. Middle & Right: when occlusion happens, we can reconstruct the global shape from observed local properties based on the learnt relationship. (This an illustrative example of our idea, for real examples, please refer to Fig. 6.7 and Fig. 6.6)

## 6.1 Introduction

In the previous chapters, we discussed the use of implicit shape representations (*e.g.* SDFs) for real-time 3D tracking and reconstruction from RGB-D data. In this chapter, we further explore the possibility of using implicit shape representations for real-time *2D* tracking, segmentation and occlusion detection & recovery with *only* RGB image input.

In the field of 2D tracking, there has been substantial research in segmentation based tracking, in such works as Yilmaz *et al.* [147], Bibby and Reid [13], Mirmehdi *et al.* [89], *etc.*. These methods extract an active contour at each frame and use it to update the shape of a tracked object. Such methods result in the efficient tracking of previously unseen objects. However, a challenge that remains is occlusion, because, unless the tracked shape is constrained in some way, the resulting contour will have an incorrect shape. Our aim in this chapter, then, is to show how to learn the set of legal shapes of a potentially deformable object *incrementally* and *online* and then use this learned model to detect occlusion and recover the original shape of the

object at each frame.

We consider the problem of occlusion detection and shape recovery by modeling the relationship between the local and global properties of shape. The motivation behind our idea is illustrated in Fig. 6.1, where we show an occluded human (with only the legs visible). Even though the bulk of the person is occluded, a human observer can reconstruct the shape (*i.e.* the global property) from the relationship between the hands, arms, legs etc. (*i.e.* the local properties). We describe a method to formalize this insight by learning the relationship between the local and global properties of shapes. Specially, we show how Locally Weighted Projection Regression (LWPR Vijayakumar *et al.* [142]) can be used to learn a regression from the high frequency harmonics to the low frequency ones of a shape, and how this regression can be used to detect and recover occlusions on previously seen shapes. We link our shape regression to the pixel-wise posteriors (PWP) level set-based tracker of Bibby and Reid [13]. The PWP tracker obtains the target pose (a 6 DoF 2D affinity or 4 DoF 2D similarity transform) and figure/ground segmentation at each frame. We use the pose obtained from the tracker to align the shapes and then add them to the learning framework, as they are received. After a burn-in period, the framework is able to recover occluded shapes at real time.

The remainder of the chapter is structured as follows: we begin in Section 6.2 by reviewing some related work, then in Section 6.3 detail our occlusion detection and shape regression framework. Specifically, in §6.3.1 we discuss the discrete cosine transform shape representation and its advantages. In §6.3.2, §6.3.3 and §6.3.4 we detail the LWPR-DCT algorithm and describe how we detect occlusion, discriminate between occlusion and a new shape, and recover occluded shapes. We show qualitative and quantitative evaluations of our method in Section 6.4, and conclude in Section 6.5.

## 6.2 Related works

Due to the importance of occlusion reasoning in visual tracking, there has been substantial related research in various aspects, however, there are very few general frameworks to identify occlusion explicitly. Many appearance-based tracker solve the occlusion problem directly by statistical analysis. In Jepson *et al.* [54], Han and Davis [47] and Ross *et al.* [118], the authors learn an adaptive appearance model online, but when long-term occlusion occurs, the appearance models can be easily contaminated due to the blind update strategies. In Adam *et al.* [6] and Chockalingam *et al.* [26], the target is decomposed into multiple components or patches, and occlusion is implicitly reasoned by robust statistics. In the more recently work of Kwak *et al.* [66], the authors divide the object into several cells and train a classifier using patch likelihood. This way, they are able to explicitly detect occlusion efficiently, thus, improve the accuracy of tracking. However, all the works above are based on object appearance models, which does not explicitly model the object shape, and in presence of occlusion, they are unable to recover the complete shape of the object.

A typical solution to recover the complete shape in the presence of occlusions is to put constraints on the minimization of the level set energy function. Such methods roughly fall into two categories: the first category comprises of methods which try to capture the variance in the space of legal embedding functions (*e.g.* Leventon *et al.* [74], Tsai *et al.* [136], Rousson and Paragios [120], Cremer *et al.* [34], Dambreville [36], Prisacariu and Reid [105]). This was first attempted by Leventon *et al.* [74], where PCA was used to learn the space of zero level set embedding functions. To segment the image, the contour was evolved by minimising an energy function which combined three terms, one for the image data, one for the shape and one for the pose. The minimisation sought alignment of the curve with the image gradients, while at the same time maximising the probability of the shape.

Replacing edges with regions as the main source of image information, the method of Leventon *et al.* [74] was extended by Tsai *et al.* [136]. Here PCA is again used on level set functions, but the energy function can be, for example, the region based one of Vese and Chan [141]. The minimisation is done directly in the shape space i.e. by differentiation with respect to the position in the lower dimensional latent space and the 2D pose. Nonlinear dimensionality reduction was used first by Rathi *et al.* [109] and Dambreville *et al.* [36], in the form of Kernel PCA (Schölkopf *et al.* [124]). This has been shown to greatly improve the learning capabilities of the shape space. Here a segmentation quality measure is maximised in the space of embedding functions, aiming to minimise the distance between the projection of those embedding functions and the known lower dimensional latent points. The opposite is true for Prisacariu and Reid [105, 104], where the optimisation process is kept in the lower dimensional space and a closed form generative process is used to generate high dimensional shapes. This is achieved by replacing Kernel PCA with Gaussian Process Latent Variable Models (Lawrence [71]). Prisacariu and Reid [104] represent shapes explicitly using elliptic Fourier descriptors (Kuhl and Giardina [64]) and generate the level set embedding functions at runtime, whereas Prisacariu and Reid [105] learn spaces of level set embedding functions directly, compressed using the discrete cosine transform (DCT, Watson [143]). All of the above presented methods are robust to occlusions, since the evolution of the contour is limited to the space of possible shapes. None of them however explicitly considers occlusion modelling or recovery. Furthermore, inference tends to be very slow (in the order of seconds or minutes per frame) and training is always done *off-line*. This means that, when new contours are added, the *whole model* must be re-trained, an operation that can take up to several hours.

The second school of methods attempts to influence the shape in the current frame by comparing it with a number of recently observed shapes. Mirmehdi *et al.* [89] proposed to incrementally build a dynamic space of good shape hypothe-

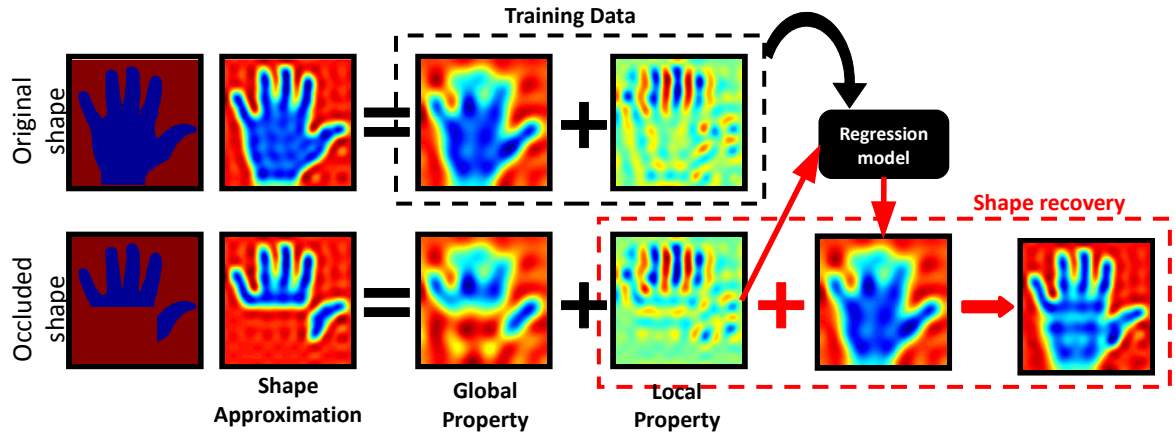



**Figure 6.2:** An overview of our whole shape recovery algorithm.

ses from all frames leading up to the current one. The shape of the current frame is thus constrained by minimizing its distance from a locally Gaussian weighted shape expectation of the learned space. By continually updating a weight matrix, this method can incrementally update the space of good shapes without re-training. However, in practice, (i) both the size of the weight matrix and the time it takes to update it grows as $n^2$ (where $n$ is the number of observed good shapes), and (ii) in order to keep track of this matrix, all previously observed shapes need to be stored. Alternatively, when using a fixed size weight matrix, the method suffers from rapid forgetting. The authors also note that this method is very slow, making it unsuitable for real-time operation. In another work, Yilmaz *et al.* [147], a dense level set function is embedded in the shape, with the background area set to zero. A probability distribution for each grid point on the level set is modelled with a single Gaussian, which is updated only where no occlusion is present. Once occlusion is detected (using area and distance heuristics), the method uses the Gaussian model on each grid point to cast an expansion force on the level set, to recover the missing parts. However, the updating rate is difficult to tune when the shape of a deformable object is learned: updating too quickly will result in recovering the current shape simply based on the previous few shape, while updating too slowly will suffer from large uncertainty.

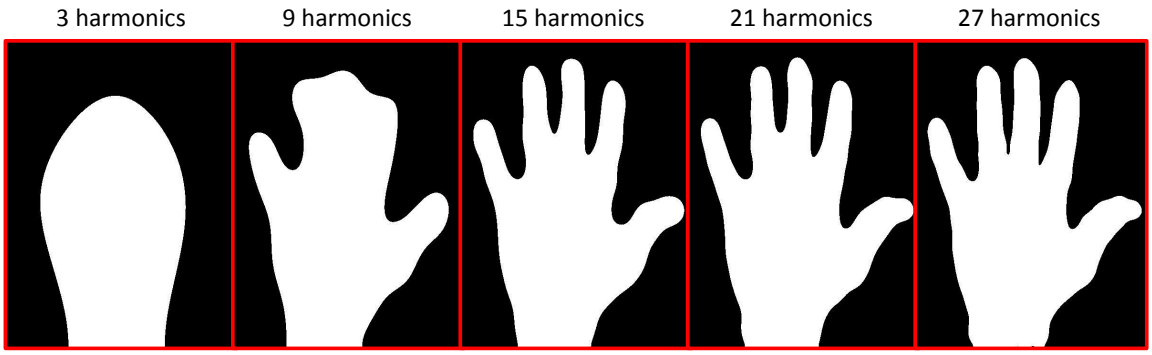



**Figure 6.3:** Using a different number of harmonics to approximate shape: as the number of harmonics that are used to encode the shape increases, more details of the shape are captured. Only the first few tens of harmonics are sufficient to recover full information of the shape, higher frequency harmonics contains more image noise than information of the shape.

## 6.3 Shape regression for online segmentation and tracking

### 6.3.1 Shape representation via DCT

The 2D discrete cosine transform (DCT, Watson [143]) is a special case of the discrete Fourier transform, which represents an image using a series of orthogonal cosine basis functions known as harmonics, each with its own frequency and amplitude. A common use for the DCT is image compression, it being the basis for the JPEG format. Similarly, in Prisacariu and Reid [105], the authors used it to compress level set embedding functions. Our work is based on a different property of the DCT, namely the fact that the low frequency harmonics contain the coarse "bulk" properties of the information in the signal, while high frequency ones contain the "details". When applied to shapes, this means that, often, when an object is occluded, parts of its main body may be missing, but many high frequency details remain. Our experiments suggest that occlusions introduce relatively minor changes to the high frequency DCT coefficients. Based on this observation, we train a regression model from higher frequency harmonics to lower frequency harmon-

ics using previously observed complete shapes. Thus, when an occluded shape is observed, we extract its high-frequency harmonics and use the regressor to determine the expected low frequency harmonics, and hence recover the whole shape by adding the low frequency harmonics to the high frequency ones. Fig. 6.2 gives an overview of our framework.

The 2D DCT of a $N \times N$ image is defined as:

$$C(u,v) = \alpha(u)\alpha(v) \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x,y) \cos\left[\frac{\pi(2x+1)u}{2N}\right] \cos\left[\frac{\pi(2y+1)v}{2N}\right] \tag{6.1}$$

for $u, v \in 0, 1, 2, \ldots, N-1$, $\alpha(u)$ and $\alpha(v)$ are defined as:

$$\alpha(u) = \begin{cases} \sqrt{\frac{1}{N}} & \text{for } u = 0 \\ \sqrt{\frac{2}{N}} & \text{for } u \neq 0 \end{cases} \tag{6.2}$$

$$\tag{6.3}$$

The inverse transform is defined as:

$$f(x,y) = \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} \alpha(u)\alpha(v)C(u,v) \cos\left[\frac{\pi(2x+1)u}{2N}\right] \cos\left[\frac{\pi(2y+1)v}{2N}\right] \tag{6.4}$$

for $x, y \in [0, \ldots, N-1]$. Since the basis functions are orthogonal, the coefficients can be computed independently, as above. The transform yields a natural hierarchical representation of the original image in which the top-left, low frequency coefficients in the DCT capture the overall signal, while the high frequency coefficients (further away from top-left) capture the details of the image .

We use the DCT to represent a silhouette mask image, i.e. a binary image of the figure/ground segmentation, with 1 for foreground and -1 for background. This is in contrast with works such as Prisacariu and Reid [105] where the DCT is computed from the level set (SDF) embedding function. Our representation insures

that the high frequency harmonics capture variability *in the contour* rather then, for example, in the structure of the level set embedding the contour.

Note that, the first several tens of harmonics are sufficient to recover the whole shape, very high frequency harmonics contains far more noise than information of the shape. For example, as is shown Fig. 6.3, when we increase the number of harmonics from 3 to 15, we observe significant increase in the shape detail, however, from 15 to 27, very little detail information is added to the recovered shape. Thus, we use only 10-15 DCT harmonics (depending on complexity of the shape) to encode the shape, and divide the selected harmonics into low-frequency ones and high frequency ones. Taking the inverse DCT transform of the selected harmonics then thresholding at zero yields an approximation for the silhouette.

### 6.3.2 Locally Weighted Projection Regression(LWPR)

In this work we aim to recover the missing part of a shape in a discriminative manner, by using an online trained regression (as opposed to learning a generative shape space) from the high frequency DCT coefficients to the low frequency ones. We therefore need to learn an incremental approximation of a highly non-linear and high dimensional function. Established methods for fitting non-linear functions globally already exist, a few examples being Support Vector Machine Regression (SVMR, Smola and Schölkopf [132]), Gaussian Process Regression (GPR, Rasmussen [108]) and Variational Bayes Mixture Models (VBM, Corduneanu and Bishop[32]). These methods are however not generally suited for online learning in high dimensional spaces. First, they require a priori determination of the right function space, in terms of basis or kernel functions (GPR, SVMR) or number of latent variables (VBM). Second, all these methods are developed primarily for off-line batch training, rather than for incremental learning. For instance, in the case of SVMR, when a new training point is added, the outcome of the global optimisation

**Table 6.1:** Legend of indexes and symbols used

| Notation | Affectation |
|---|---|
| $M$ | No. of training data points |
| $N$ | Input dimensionality (dim of $\mathbf{x}^{hf}$) |
| $H$ | Output dimensionality (dim of $\mathbf{x}^{lf}$) |
| $k = 1 : K$ | No. of local models |
| $r = 1 : R$ | No. local projections in each local model |
| $\{\mathbf{x}_i^{hf}, \mathbf{x}_i^{lf}\}_{i=1}^{M}$ | Training data |
| $\{x_{i,j}^{lf}\}_{j=1}^{H}$ | Element of $\mathbf{x}_i^{lf}$ |
| $\{\mathbf{s}_i\}_{i=1}^{M}$ | Lower dim projection of input data $\mathbf{x}_i^{hf}$ |
| $\{s_{i,r}\}_{r=1}^{R}$ | Element of $\mathbf{s}_i$ |
| $\mathbf{e}$ | Prediction error in LWPR |
| $\{e_j\}_{j=1}^{H}$ | Elements of $\mathbf{e}$ |
| $\mathbf{c}_k$ | Field centre of the $k$-th Receptive Field |
| $\mathbf{D}_k$ | Distance metric of the $k$-th Receptive Field |
| $\{\mathbf{u}_r^n\}_{r=1}^{R}$ | The $r$-th projection of a local model after observing $n$ training point (local model index omitted) |
| $\{\beta_r^n\}_{r=1}^{R}$ | The weight for the $r$-th projection of a local model model after observing $n$ training point (local model index omitted) |
| $(\mathbf{a}_0^n, \beta_0^n)$ | The means of input and output of a local regression model after $n$ training points. |
| $w_k$ | Activation of the $k$-th RF as in Eqn.6.5 |
| $W_k^n$ | Cumulative weights of the $k$-th RF after $n$ training points |
| $\{SS_r^n, SR_r^n, SZ_r^n\}_{r=1}^{R}$ | Trace variables for the incremental computation of the $r$-th local regression after seeing $n$ training points |

can be changed greatly.

We use Locally Weighted Projection Regression (LWPR, Vijayakumar *et al.* [142]) as our regression model. LWPR is a nonlinear function approximator that learns rapidly from incrementally acquired data, without needing to store the training data. The computational complexity grows linearly with the number of inputs. LWPR can also deal with a large number of possibly redundant inputs, which is often the case when tracking rigid objects.

Each processing unit in LWPR is a multi-input, single-output regression model.

We therefore use multiple processing units to formulate our multi-input, multi-output shape regression (LWPR-DCT model). LWPR is based on the hypothesis that high dimensional data are characterised by locally low-dimensional distributions. A learned LWPR unit has $K$ local models, each comprising a Receptive Field (RF) characterised by: (1) a field centre $\mathbf{c}_k$; (2) a positive semi-definite distance metric $\mathbf{D}_k$ that determines the size and shape of the neighbourhood contributing to the local model and (3) a local projection regression model (a modified version of partial least squares) characterised by a set of projections and their respective weights. In the following sections, we detail how we learn a LWPR-DCT model from a set of training shapes and use the learnt model to detect and recover occluded shapes.

### 6.3.3 Incremental online learning with LWPR

The notations used in this and the following sections are listed in Table 6.1. The learning algorithm of a single LWPR unit is summarised in Table 6.2. The learning of a LWPR unit comprises (1) the incremental computation of projections and regressions in each local models and (2) the adjustment of the shapes and the sizes of the receptive fields (RFs). We start with the algorithm for updating the projections and regressions in each local model. Initialized with no RF, when a training shape is observed, we compute its high frequency and low frequency DCT coefficients as input and output to LWPR. Given a set of high frequency DCT coefficients $\mathbf{x}_i^{hf}$, each local model in a LWPR unit computes a RF weight, which is also known as the activation:

$$w = \exp\left\{-\frac{1}{2}(\mathbf{x}_i^{hf} - \mathbf{c})^T \mathbf{D} (\mathbf{x}_i^{hf} - \mathbf{c})\right\} \tag{6.5}$$

where we omitted the local model index $k$ for clarity.

If no RF is activated by more than $w_{gen}$, a new RF centered at $\mathbf{x}_i^{hf}$ will be created with the initial distance metric $\mathbf{D}_{def}$ and two projections. $w_{gen} \leq 1$ is a threshold that determines the distance between different RFs: the closer $w_{gen}$ is set to 1, the

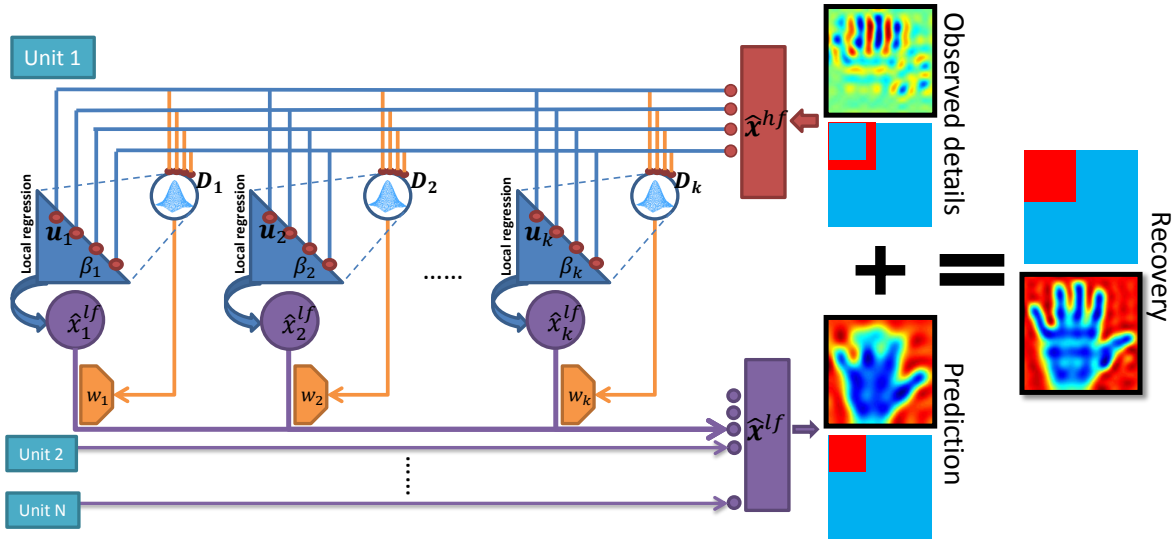



**Figure 6.4:** Structure and work flow of LWPR, inspired by Fig. 3 in Vijayakumar *et al.* [142].

**Table 6.2:** Pseudo code for the learning of LWPR-DCT model.

- Initialize the LWPR with no receptive field (RF).
- For the $i$-th training shape $\Phi_i$
  - compute the first $(N+H)$ DCT coefficients of $\Phi_i$
    - ⋆ Use No. $1 \sim H$ coefficients as output $\mathbf{x}_i^{lf}$
    - ⋆ Use No. $(H+1) \sim (N+H)$ coefficients as input $\mathbf{x}_i^{hf}$.
  - For the $k$-th out of $K$ existing RFs:
    - ⋆ Calculate the activation using Eqn. 6.5.
    - ⋆ Update the $k$-th local regression model.
    - ⋆ Update the distance metric $\mathbf{D}_k$
    - ⋆ Check the decreasing rate of $MSE$ at each projection to see if the number of projections needs to be increased.
  - If no RF was activated by more than $w_{gen}$:
    - ⋆ Create a new RF with initial number of projections $R = 2$ $\mathbf{c}_{K+1} = \mathbf{x}_i^{hf}$ and $\mathbf{D}_{K+1} = \mathbf{D}_{def}$, $K \leftarrow K+1$.

more overlap local models will have. $\mathbf{D}_{def}$ is a diagonal Gaussian distance metric, which determines the initial shape of the RF.

Each local model is initialized with input mean $\mathbf{a}_0^0 = \mathbf{0}$, output mean $\beta_0^0 = 0$ and weight $W^0 = 0$. With new training data $(\mathbf{x}_i^{hf}, x_{i,j}^{lf})$, these are updated as:

$$W^{n+1} = \lambda W^n + w \tag{6.6}$$

$$\mathbf{a}_0^{n+1} = (\lambda W^n \mathbf{a}_0^n + w\mathbf{x}_i^{hf}) / W^{n+1} \tag{6.7}$$

$$\beta_0^{n+1} = (\lambda W^n \beta_0^n + w x_{i,j}^{lf}) / W^{n+1} \tag{6.8}$$

Each LWPR local regression model is an incremental locally weighted version of partial least squares, parametrised by a set of projections $\{\mathbf{u}_r\}_{r=1}^R$ and corresponding weights $\{\beta_r\}_{r=1}^R$. Given new training data $(\mathbf{x}_i^{hf}, x_{i,j}^{lf})$, the local regression models

in the $j$-th LWPR unit are updated as follows:

$$\textbf{Initialize: } \mathbf{z} = \mathbf{x}_i^{hf} - \mathbf{a}_0^{n+1},\ res_1 = x_{i,j}^{lf} - \beta_0^{n+1}$$

$$\textbf{For} \quad r = 1 : R$$

$$1.\ s_{i,r} = \mathbf{z}^T \mathbf{u}_r^n / \sqrt{\mathbf{u}_r^{nT} \mathbf{u}_r^n} \tag{6.9}$$

$$2.\ SS_r^{n+1} = \lambda SS_r^n + w\, s_{i,r}^2$$

$$3.\ SR_r^{n+1} = \lambda SR_r^n + w\, s_{i,r}\, res_r$$

$$4.\ SZ_r^{n+1} = \lambda SZ_r^n + w\, \mathbf{z}\, s_{i,r}$$

$$5.\ \mathbf{u}_r^{n+1} = \lambda \mathbf{u}_r^n + w\, \mathbf{z}\, res_r$$

$$6.\ \beta_r^{n+1} = SR_r^{n+1} / SS_r^{n+1}$$

$$7.\ \mathbf{p}_r^{n+1} = SZ_i^{n+1} / SS_r^{n+1} \tag{6.10}$$

$$8.\ \mathbf{z} = \mathbf{z} - s_{i,r}\, \mathbf{p}_r^{n+1}$$

$$9.\ res_{r+1} = res_r - s_{i,r} \beta_r^{n+1}$$

$$10.\ MSE_i^{n+1} = \lambda MSE_i^n + w\, res_{r+1}^2 \tag{6.11}$$

$$e_j = res_{R+1} \tag{6.12}$$

where the variables $SS, SR, SZ$ are sufficient statistics that enable use to perform incremental regression learning without the need to explicitly store any training data. $\lambda \in [0, 1]$ is a forgetting factor that allows exponential forgetting of old data in the sufficient statistics.

The learning algorithm has a simple mechanism to determine whether the number of projections in a local model needs to be increased by recursively keeping track of the mean-square error (MSE, as recursively computed in Eqn. 6.11) as a function of the number of projections. The algorithm stops adding new projections if the MSE at next added projection does not decrease more than a certain percentage of the previous MSE (i.e. $MSE_{i+1}/MSE_i > \phi$).

The distance metric $\mathbf{D}$, which describes the locality of the receptive fields, can be learnt for each local model individually by stochastic gradient descent in a penalized leave-one-out cross validation cost function (indicated by the subscript $-i$):

$$J = \frac{1}{\sum_{i=1}^{M} w_i} \sum_{i=1}^{M} w_i (x_i^{lf} - \hat{x}_{i,-i}^{lf})^2 + \frac{\gamma}{N} \sum_{i,j=1}^{N} D_{ij}^2 \tag{6.13}$$

where $M$ denotes the number of training shapes, $N$ is the number of RFs. $\gamma$ is a trade-off parameter that can be determined empirically and the output dimension index $j$ has been omitted for clarity.

Minimisation of Eqn. 6.13 can be accomplished in a incremental way (without the need to store any training data) as well. Vijayakumar *et al.* [142] proposed to expend Eqn. 6.13 with the PRESS residual error and formulated $J$ in terms of the projected inputs $\mathbf{s}_i = [s_{i,1} ... s_{i,R}]^T$ in Eqn. 6.9:

$$J = \frac{1}{\sum_{i=1}^{M} w_i} \sum_{i=1}^{M} \frac{w_i (x_i^{lf} - \hat{x}_i^{lf})^2}{(1 - w_i \mathbf{s}_i{}^T \mathbf{P}_s \mathbf{s}_i)^2} + \frac{\gamma}{N} \sum_{i,j=1}^{N} D_{ij}^2 \tag{6.14}$$

where $\mathbf{P}_s$ corresponds to the inverted weighted covariance matrix of the projected input $\mathbf{s}_i$ for $R = N$. Given object function Eqn. 6.14, the distance metric $\mathbf{D}$ is learnt by gradient descent:

$$\mathbf{M}^{n+1} = \mathbf{M}^n - \alpha \frac{\partial J}{\partial \mathbf{M}} \quad \text{where} \quad \mathbf{D} = \mathbf{M}^T \mathbf{M} \tag{6.15}$$

where $\mathbf{M}$ is a upper triangular matrix from a Cholesky decomposition of $\mathbf{D}$. For further details on the proof and derivation of Eqn.6.14 and Eqn. 6.15, the interested reader is referred to Appendix B in Vijayakumar *et al.* [142].

### 6.3.4 Shape Regression with LWPR-DCT

In real world applications, the proposed LWPR-DCT learning framework requires a 'burn-in' period to acquire un-occluded shapes as training data. During this period we assume the shapes adopted by an object to be un-occluded and aligned by the PWP tracker of Bibby and Reid [13]. We then transform the observed shapes into high frequency and low frequency DCT coefficients $(\mathbf{x}^{hf}_{obs}, \mathbf{x}^{lf}_{obs})$ and train a LWPR on this sequence of shapes. Once the LWPR model has been learnt, we can use it to detect occlusion in the shape as well as recover the original un-occluded shape.

Fig. 6.4 shows the prediction workflow of a single LWPR processing unit. Given a set of novel input harmonics $\mathbf{x}^{hf}$, the prediction from a single local model follows the standard partial least squares regression:

$$\textbf{Initialize: } \hat{x}^{lf} = \beta_0, \mathbf{z} = \mathbf{x}^{hf} - \mathbf{a}_0 \tag{6.16}$$

$$\textbf{For } r = 1 : R \tag{6.17}$$

$$1.\ s = \mathbf{u}_r^T \mathbf{z} \tag{6.18}$$

$$2.\ \hat{x}^{lf} = \hat{x}^{lf} + \beta_r\, s \tag{6.19}$$

$$3.\ \mathbf{z} = \mathbf{z} - s\, \mathbf{p}_r \tag{6.20}$$

where $\mathbf{p}_r$ is computed using Eqn. 6.10 and the local model index $k$ omitted for clarity. The final output (i.e. a single low frequency DCT coefficient) is given by the weighted mean of all $K$ local outputs:

$$\hat{x}^{lf} = \frac{\sum_{k=1}^{K} w_k \hat{x}_k^{lf}}{\sum_{k=1}^{K} w_k} \tag{6.21}$$

where $w_k$ is the RF weight computed using Eqn. 6.5. Note that, since each LWPR unit is an multi-input, single-output regression model, we have the number of LWPR units equal to the number of output dimensions.

**Table 6.3:** Pseudo code for the occlusion detection and recovery using learnt LWPR-DCT model.

---

Given an observed shape $\Phi_{obs}$
- ◦ compute the first $H$ DCT coefficients of $\Phi_{obs}$:
  - ⋆ Use No. $1 \sim (H-N)$ coefficients as $\mathbf{x}_{obs}^{lf}$.
  - ⋆ Use No. $(H-N+1) \sim H$ coefficients as $\mathbf{x}_{obs}^{hf}$.
- ◦ $count = 0$ (# of '*Active*' LWPR units)
- ◦ For the $i$-th out of $(H-N)$ LWPR units:
  - ⋆ Calculate the activations for all local models with Eqn 6.5.
  - ⋆ If any local model is activated more than $w_{gen}$:
    - -Label this LWPR unit as *Active*.
    - -$count \leftarrow count + 1$
- ◦ If $count \geq (H-N)/2$ (more than half of the unites are activated):
  - ⋆ Predict the low frequency harmonics $\hat{\mathbf{x}}^{lf}$ using learnt LWPR
  - ⋆ If $\|\hat{\mathbf{x}}^{lf} - \mathbf{x}_{obs}^{lf}\|^2 \geq 2\|\mathbf{e}\|^2$
    - -Shape occluded, recover shape by replacing $\mathbf{x}_{obs}^{lf}$ with $\hat{\mathbf{x}}^{lf}$.
    - else
    - -Shape is not occluded.
  - else
  - ⋆ The high frequency details have not been learnt, stop.

---

The occlusion detection and shape recovery mechanism of LWPR-DCT is summarised in Table 6.3: when a new shape is observed, we first compute the activation of all local models in each LWPR unit using Eqn. 6.5. If any RF in a LWPR unit has been activated by more than $w_{gen}$ we label this LWPR unite as '*Active*'. If the number of '*Active*' LWPR units is larger than 50% of the overall number of LWPR unites, we believe the details of the shape to have been learnt before, otherwise, we classify the shape as 'not learnt' and proceed no further. For a shape whose details have been learnt, the system then makes a prediction of the low frequency components for the shape and calculate the difference between the prediction $\hat{\mathbf{x}}^{lf}$ and the observed low frequency harmonics $\mathbf{x}^{lf}_{obs}$. If the difference is smaller than twice the prediction error $\mathbf{e}$ (Eqn. 6.12) of the learnt LWPR-DCT model, we consider the shape as 'not occluded' and leave the tracker output unchanged. Otherwise, we classify the shape as being known but occluded and update it according to our prediction.

## 6.4 Experiments and performance analysis

We tested our method both qualitatively and quantitatively, on several video sequences and data sets. We used an Intel Core i7-870 (2.93GHz) machine to run all our experiments. We denote our method with *LWPR-DCT*.

We begin with the qualitative analysis. Examples of successful shape recovery using artificially generated occlusions are shown in Fig. 6.5. The images on the left of each column show the shape approximation using the inverse of the truncated DCT (red $\rightarrow$ 1, blue $\rightarrow$ -1). The images on the right show the recovered shape by thresholding the images on the left at zero level. We show the original silhouette in the left two columns, the occluded silhouette in the middle two columns and the recovered silhouette in the right two columns. The results show that LWPR-DCT

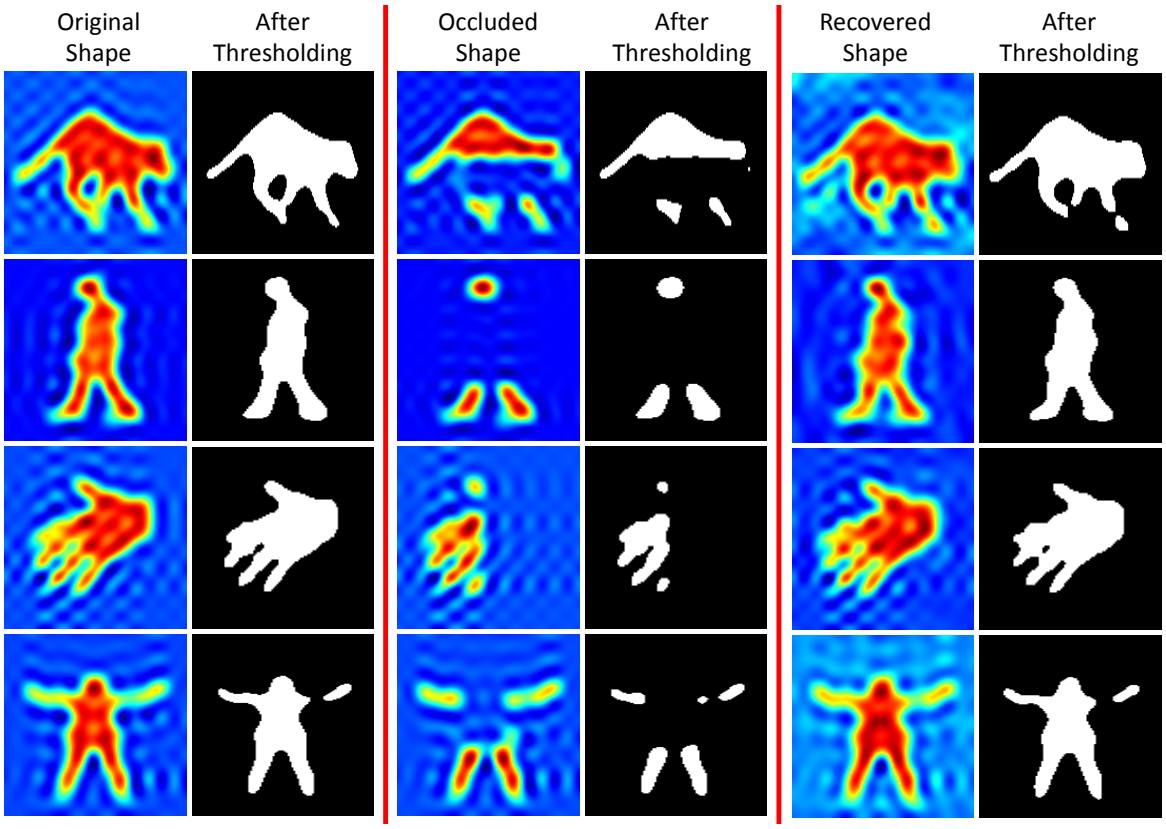



**Figure 6.5:** Examples of recovered shapes from artificially occluded images. The left column of each pair shows false color images(blue=-1, red=1) of the inverse "truncated DCT" (i.e.. the approximation of the silhouette via the first 10 to 15 harmonics), while the right column shows the silhouette obtained from thresholding the approximation at zero. From top row to bottom row: *Cat running*, *Man walking* , *Hand*, *Woman jumping*

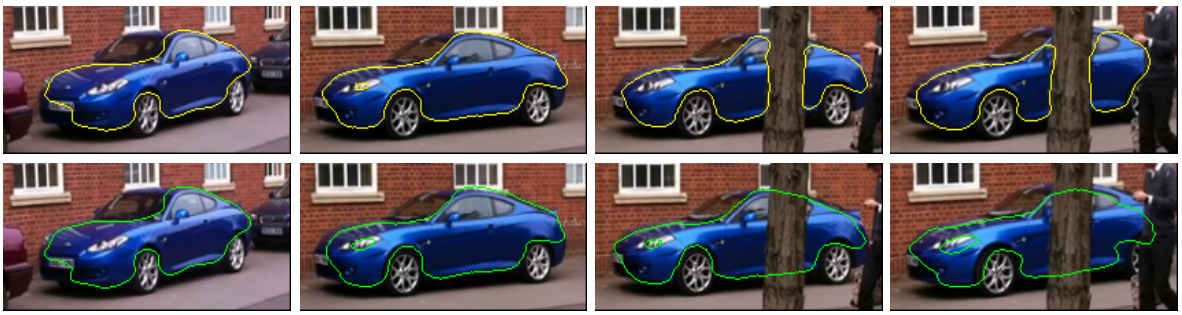


**Figure 6.6:** Example frames from a video tracking a car, comparing our method to the PWP tracker of [13]. When the car is not occluded both methods produce similar results. When the tree is in front of the car the segmentation produced by the PWP tracker is corrupted, while the one produced by our tracker is not.

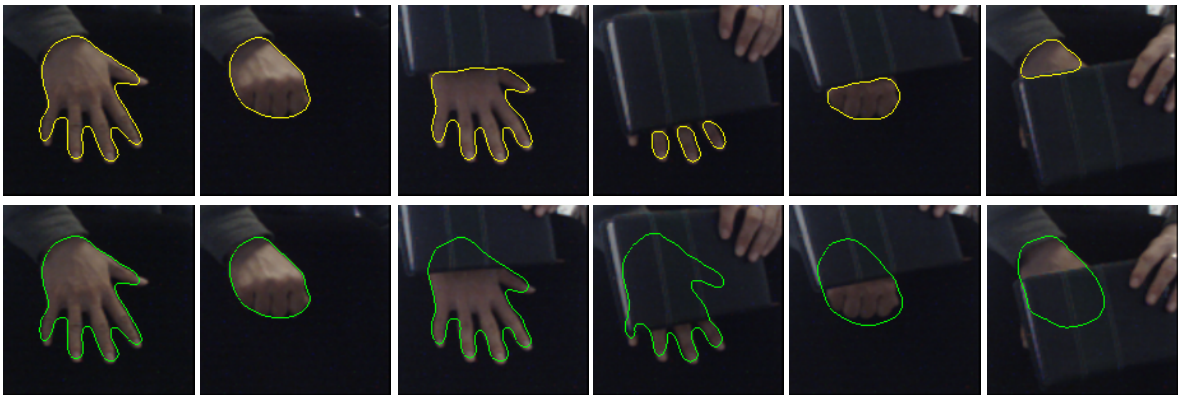


**Figure 6.7:** Example frames from a video tracking a hand, comparing our method to the PWP tracker of [13]. When no occlusions are present, both method produce similar results. However, as soon as the hand is occluded, the PWP tracker produces an incorrect segmentation, while our method still generates correct contours. Whole sequence see Video B.4 in Appendix B.

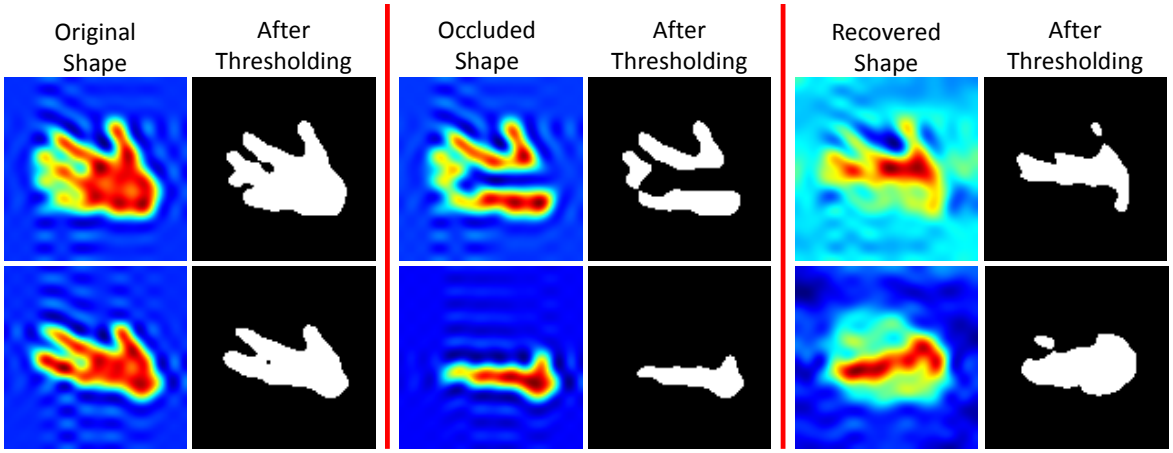



**Figure 6.8:** Example failure cases (from the *hand* video). Top line fails because noisy high frequency harmonics are introduced, while bottom line fails because details are missing.

is capable of recovering the shape in the presence of various types of artificially introduced occlusion.

In Fig. 6.7 and Fig. 6.6, we compare our algorithm to the standard pixel-wise posteriors (PWP) tracker of Bibby and Reid [13] on real video sequences and show that we are able to successfully recover the correct contour, in the presence of heavy occlusions. In the first 2 frames of Fig. 6.7 there are no occlusions, so both LWPR-DCT and the standard PWP tracker yield similar results. When the hand is occluded, in the other 4 frames, the output segmentations of the PWP tracker are corrupted, while ours are still correct. Similarly in Fig. 6.6, in the presence of the occlusion introduced by the tree, our LWPR-DCT framework can still recover the learnt car shape.

We show two failure cases of our method in Fig. 6.8. The failure modes are presented using artificially introduced occlusions. LWPR-DCT can fail in two ways. First, when too many small occlusions are present, the high frequency DCT harmonics may be affected, as is shown in the upper row of Fig. 6.8. The failure here happens because the input to LWPR-DCT has been changed considerably by the occlusions. In real world applications, this case is usually observed when the object being tracked is behind a fence or occluded by several small objects. The second failure case happens when too much detail is occluded, as is shown in the lower row of Fig. 6.8. This happens because the high-frequency DCT harmonics, which we rely on, are missing. Note that, in both failure cases, the input occluded shapes may be classified as 'not learnt' in the occlusion detection stage, however, this is not guaranteed, thus LWPR-DCT might produce incorrect results.

We designed three sets of experiments to evaluate our LWPR-DCT framework quantitatively. First we measure the effectiveness of the shape recovery using LWPR-DCT and show how many high frequency harmonics are required as input for occlusion recovery. We then evaluate the effectiveness of occlusion detec-

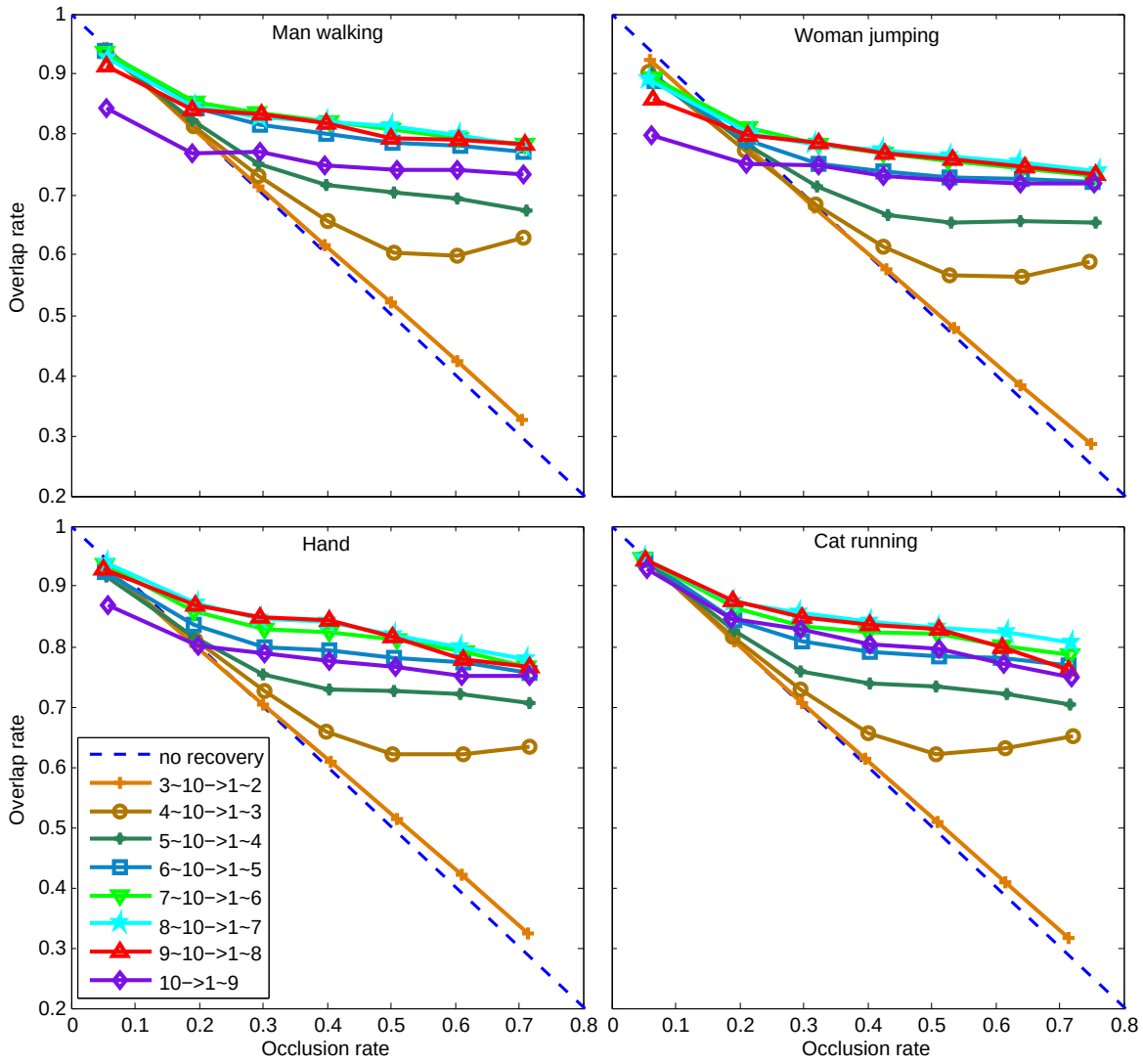



**Figure 6.9:** Shape recovery performance evaluation of LWPR-DCT on four datasets using a different number of harmonics as input and output.

tion with LWPR-DCT using a different number of input harmonics. Finally, we compare our algorithm with a state-of-the-art shape prior based method of [105] (denoted by GPLVM-DCT) on the performance of occlusion recovery and average processing time.

For the first two experiments, we used four datasets to evaluate the effectiveness of LWPR-DCT: *Cat running* (artificial video with few distinct poses, 398 frames), *Woman jumping* (real video with an average number of distinct poses, 410 frames), *Man walking* (real video with many distinct poses, 411 frames, the subject 2 walk of the HumanEva I dataset of Sigal *et al.* [129])) and *Hand* (real video with many distinct poses, 408 frames). For each video, all frames are segmented and aligned using the PWP tracker, then added to LWPR-DCT as training data. We then add different sizes of artificial occlusions (where each occlusion is rectangular and in a random location) to each frame. We chose to generate occlusions artificially in order to be able to control the percentage of occlusion and to have accurate knowledge of the ground truth. For each frame, we generate 7 levels of occlusion, ranging from 0.1 (10%) to 0.8 (80%).

In the first test, we use the overlap rate $R = \frac{S_{gt} \bigcap S_{rcv}}{S_{gt} \bigcup S_{rcv}}$ as our performance criteria, where $S_{gt}$ is the ground truth shape and $S_{rcv}$ is the recovered shape. We use the first 10 harmonics to approximate the segmented shapes and run tests on all possible combinations of the numbers of input and output harmonics (harmonic 10 generating 1 to 9, 9 and 10 generating 1 to 8, etc.). Fig. 6.9 shows the results. Our method gives sensible results just by using the 10-th harmonic to regress all 1∼9 harmonics. Using the harmonics 8∼10 to regress harmonics 1∼7 or 9∼10 to 1∼8 gives the best performance in all cases. Performance decreases again as we increase the number of known harmonics. This happens because we are using too many low frequency harmonics as high frequency input, and such input as been corrupted by occlusion.

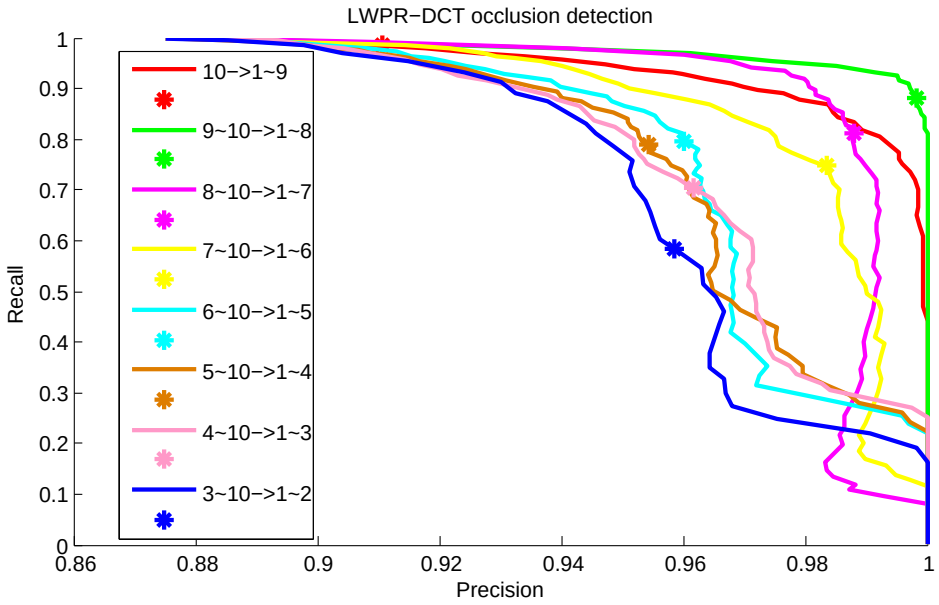



**Figure 6.10:** Precision-recall curve of occlusion detection using LWPR-DCT, '$*$' on each curve correspond to *threshold* = **e**.

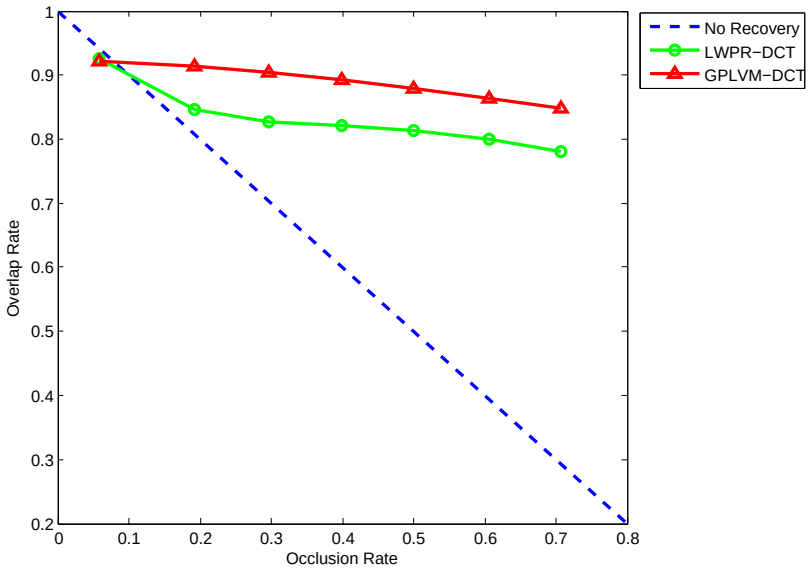



**Figure 6.11:** Comparing LWPR-DCT to GPLVM-DCT on recovery performance.

| Occlusion rate | 0.2927 | 0.3896 | 0.4947 | 0.6012 | 0.7042 | 0.8082 | 0.9409 |
|---|---|---|---|---|---|---|---|
| GPLVM-DCT (s/frame) | 2.6301 | 2.9180 | 3.0261 | 3.2784 | 3.5733 | 3.6994 | 3.9663 |
| LWPR-DCT (s/frame) | 0.0378 | 0.0380 | 0.0382 | 0.0381 | 0.0381 | 0.0379 | 0.0382 |

**Table 6.4:** Comparing LWPR-DCT to GPLVM-DCT on processing time

In the second test, we use the same training and test set as the first experiment, but in the testing set, we also added the original un-occluded shapes. We evaluate the effectiveness of our occlusion detection method by tracking the average precision and recall on all four data sets. We use the first 10 harmonics to approximate the shapes and run tests on all train–test combinations to see how many input/output harmonics are needed to obtain the best result. Note that, given the observed shape has been learnt before (Table 6.3), the occlusion detection in LWPR-DCT relies on a single threshold parameter, i.e. if $\|\hat{\mathbf{x}}^{lf} - \mathbf{x}^{lf}_{obs}\|^2 \geq 2\|threshold\|^2$ the observed shape will be classified as occluded. We use the prediction error $\mathbf{e}$ obtained in the training phase as this threshold (Table 6.3). In this experiment, only shapes that are detected as occluded are labelled as *positive*, shapes that are not learnt and shapes that are detected as un-occluded are labelled as *negative*. In Fig. 6.10 we track the precision-recall by varying the *threshold*. Also, the point $threshold = \mathbf{e}$ is plotted '$*$'. As is shown in Fig. 6.10, using the harmonics 9$\sim$10 to regress harmonics 1$\sim$8 gives the best performance. Using the prediction error $\mathbf{e}$ as the threshold also (approximately) gives the best precision-recall on the curve. As we increase the number of input harmonics, the detection performance decreases. This decrease in performance again shows that occlusions affect the low frequency harmonics more than the high frequency ones.

In the last quantitative experiment, we compare our algorithm to the shape prior method of Prisacariu and Reid [105], which generates embedding functions from a 2 dimensional GPLVM latent space. Here, segmentation (i.e. the recovery of the unoccluded shape), is an iterative non-linear minimization in the learned latent space. In our experiment, for each occluded shape, we run three separate

minimizations, we compute the recovery rate for each resulting shape, and we take an average of those values. We run multiple minimizations (rather than a single one) because each one can converge to a different shape, so to accurately measure the performance of Prisacariu and Reid [105] on our test data we need to consider all these results. As starting points for the minimization, we use the three points that generate the shapes most similar to the ground truth from the previous frame. We run both methods on the training and testing data from the *man walking* sequence from last experiment. Fig. 6.11 shows the time consumption and recovery rate of both methods. As a well trained, shape prior based method, GPLVM-DCT outperforms our method by an average of 10%. But, as is shown in the timings chart, the time consumption for LWPR-DCT stays constant at around 35ms per shape, while the processing time required by GPLVM-DCT increases with the occlusion rate and it is much larger than LWPR-DCT (up to 114 times higher). This happens because, when using LWPR-DCT, each shape recovery is a single (closed form) regression, while, in the GPLVM-DCT case, segmentation is an iterative process with the number of iterations being proportional to the percentage of occlusion in the image. Note that the GPLVM-DCT timings shown in Fig. 6.11 are for a single mode search. Since we use three such searches, the actual processing time per frame it three times as large. In this experiment we used the harmonics 8∼10 to regress the other 1∼7 harmonics. Note that the dot line corresponds to no-recovery, which, when less then 10% occlusion is introduced, actually has higher accuracy. This is an artefact caused by the fact that we only use a small number of DCT harmonics to represent shapes and affects the method of [105] as well. It can be easily avoided by using more DCT harmonics, at the expense of an increase in speed and memory usage.

## 6.5 Conclusions and discussions

In this chapter, we have presented a novel regression based framework for online shape learning and recovery. Shapes are represented by discrete cosine transform harmonics and the set of object shapes is modelled by a regression from the high frequency harmonics to the low frequency harmonics. Our method incrementally learns a shape model for an observed object and detects/recovers occlusions in real time. We integrated our method with a level-set based tracker, but it could be potentially linked to other types of segmentation and tracking algorithms.

Our method currently has two limitations. First, the DCT representation of shape is rotation dependent, i.e small rotation change of a shape can make the high frequency coefficient change greatly, resulting in very different prediction results. Currently we rely on the PWP tracker, which obtains camera pose and segmentation at each frame, to align the shapes. Secondly, some special types of occlusion are very difficult for LWPR-DCT to handle: (1) when noisy high frequency components are introduced by small occlusion and (2) when the details of the shape are occluded. In these two cases, LWPR-DCT might give incorrect predictions, while (much slower) shape prior based methods would be more applicable.

The chosen learning algorithm LWPR is a very powerful method applied widely in robotics to learn dynamic models for control(e.g. Kopicki *et al.* [62], Rusu *et al.et al.* [121], yet it has not been frequently applied to the field of computer vision. According to our knowledge, our proposed work is among the first that uses LWPR for online learning to solve vision problems. Aside from the preliminary version of our work in Ren *et al.* [116], the authors of Li *et al.* [77] use LWPR to learn an appearance to pose regression and then plug the pose estimator into particle filter framework for accurate tracking. In Li *et al.* [76] and Li *et al.* [78], LWPR is used to incrementally learn a geometrical model (appearance to object height), which is later used for removing outliers in the object detection stage. Many features of

LWPR (for example, not suffering from high-dimension feature space, not assuming the global regression function is linear because it can approximate nonlinear functions by piecewise linear models and the ability to be updated incrementally online) have great potential application in real-time, online solved vision problems and we are keen to explore these potentials much further.

While we have demonstrated the value of LWPR for shape recovery under occlusion, we believe that this general idea has wider applications. For example, we could consider regressing local appearance to global positions, which would have similarity to Blaschko *et al.* [15] and Fritz *et al.* [44], or more ambitiously regress local appearance to global appearance. The method would also extend to 3D shapes and 3D data.

# 7

# Conclusions, and REWIRE revisited

## 7.1 Summary

The objective motivation and applied objective of the research described in this thesis has been to enable tracking of a stroke patient's feet, hands and the control objects that they touch for a home rehabilitation system. To enable this using color & depth imagery, a thorough study has been made of how probabilistic methods and level set based implicit shape representation can be used for real-time tracking, 3D model building and occlusion reasoning. Furthermore, the practical value of the research has been taken into consideration, and all the algorithms developed have achieved either real-time or close to real-time performance.

After the introductory remarks of Chapter 1, we began in Chapter 2 with a broad overview of previous related methodologies for real-time tracking, 3D reconstruction and multi-target tracking. First, edge-, point- and region-based methods for model-based 3D tracking were reviewed. Edge-based methods are sensitive to motion blur and cluttered backgrounds, while point-based methods are restricted to textured objects. Region-based methods are more robust to motion blur and occlusion, but fail when the pose-silhouette mapping is ambiguous. Second, we looked at previous methods for 3D model building. Shape-from-X methods ei-

ther recover only depth measurement or require multiple calibrated views to work. Structure-from-motion based methods, on the other hand, cannot achieve real-time performance for dense reconstruction, due to the slow dense stereo step. Finally, we considered methods for multi-target tracking. Previous multi-target tracking methods are based on Bayesian filtering, which has limitations when searching in the high dimensional state spaces typically encountered. Simplification by representing object as points or bounding boxes has been made to reduce the dimensionality, but this makes the recovery of full DoF 3D poses of objects impossible.

Chapter 3∼5 are a trilogy which exploit the implicit shape representation for efficient 3D tracking and reconstruction.

In Chapter 3, we introduced the model representation involving a 3D SDF. Based on this, we developed a probabilistic graphical model for tracking a single known object in 3D with only depth data. The pose tracking problem was cast as a cost minimization problem by working though each pixel's contribution to the joint probability. Unlike previous model-based methods that rely on the noisy image gradients to guide the search, the present method takes advantage of the smooth gradients of the SDFs allowing the minimization to be carried out efficiently using second-order optimization methods. Furthermore, the method inherits the advantage of region-based methods that no explicit point correspondences are required, making the method inherently robust to motion blur. It was shown that the probabilistic framework can also be applied to more general 3D model registration problems. This was demonstrated in a number of applications, including intrinsic and extrinsic camera calibration, and point cloud modelling.

In Chapter 4, the probabilistic graphical model was extended for the purpose of simultaneous 3D tracking and model building with RGB-D imagery. Colour information was utilized to model the foreground/background likelihood at each pixel. The tracking problem was cast as the MAP estimate of the pose on the extended graphical model, and the use of colour information made the tracking method ro-

bust to close-to-surface outliers, missing data and occlusion. For reconstruction — and based on the idea of space carving — an inside/outside volumetric model of the object was learned online. Given a simple 3D model as prior, the 3D shape was reconstructed by evolving a 3D level set function in the volumetric space. The present reconstruction method comprises per-voxel operations only, and thus can be implemented in parallel fashion on GPU for real-time performance.

In Chapter 5 we extended the probabilistic graphical model again, now to allow for 3D tracking of multiple moving objects with identical appearance. By formulating a 'shape union' — the union of all SDFs in camera coordinates — the data association problem and the physical constraint are simultaneously solved through the extended graphical model. We demonstrated that the method is computationally efficient, with a linear increase in computational time w.r.t the number of objects.

Extensive experimentation has shown that all the algorithms proposed in Chapter 3∼5 have achieved real-time performance with GPU implementation. Furthermore, computational efficiency of the tracking algorithms has been achieved in two ways. First, the tracking algorithms back-project the few pixels in the object region to object coordinates, instead of projecting the vast mount of vertices onto image domain. Second, the pose optimization utilizes the precomputed gradients of the SDFs, instead of computing image gradients at each frame. With these mechanisms, the tracker have achieved real-time performance with CPU implementation. The multi-object tracker can track up to 5 objects in real-time on a single CPU thread.

In Chapter 6, we again explore implicit shape representations — but now for online shape learning. The problem of real-time occlusion detection and recovery for 2D tracking is considered. 2D shapes are represented by the harmonics in the 2D discrete cosine transform (DCT) of their contours. High and low frequency harmonics are decoupled to represent the coarse information and the details of the 2D shape respectively. A regression model is learned online to model the relationship between the high frequency harmonics of and those of the low frequency

using Locally Weighted Projection Regression (LWPR). We have demonstrated that the learned regression model is able to detect occlusion and recover the complete shape in real-time.

## 7.2 Future work

### 7.2.1 Scene understanding with dynamic objects

The contributions in this thesis lie generally in the fields of tracking and reconstruction. Throughout, along with a focus on the theoretical aspects of our research, we have also demonstrated the application of our methods to stroke patient home rehabilitation. Many other applications are possible, of course, most significantly, we suggest, in the field of robotics.

Current approaches for scene reconstruction assume a broadly static environment, with the motion of objects being modelled as noise, which needs to be filtered out. However, if the dynamic objects in the scene can be recognized and tracked effectively, their shapes and motions may also be leveraged to improve the reconstruction of the scene structure. To do this, objects need to be categorized into two principal types — deformable (such as humans or animals) and rigid (such as furniture or housewares). When using structure-from-motion algorithms to reconstruct the scene, the motions and shapes of the rigid objects can be used directly to improve not only the tracking of the camera, but also the mapping of the environment. Knowledge provided by deformable objects is much more difficult to use in augmenting the low-level geometry information. However, deformable objects can actually provide valuable information from a semantic point of view to potentially improve the reconstruction of the static scene structure. For example, standing or sitting human shapes can provide information about the supporting planes. In order to take full advantage of the information in both the stationary and the moving

parts of the environment, a higher level scene representation or one that mixes both low-level features and high-level semantic object knowledge will be crucial. A challenging direction of research would be to develop a multilevel scene representation to allow both rigid and deformable moving objects to be considered as semantic "landmarks" towards a better understanding of a naturally dynamic scene. The dynamic scene reconstruction problem also directly relates to the generic 3D object tracking and reconstruction problem discussed in next subsection.

### 7.2.2 3D tracking and reconstruction of generic objects

The thesis has developed a probabilistic framework for simultaneous tracking and reconstruction of an rigid 3D object with RGB-D data. The approach is able to reconstruct small moving objects in a dynamic scene using a single RGB-D sensor. However, there are still some limitations that, if addressed, would considerably increase the capabilities of the method. These include the assumptions of rigidity, known appearance models and position overlap between consecutive frames. Some of these limitations would be greatly mitigated if, first, the object class could be recognized effectively and, second, if prior shape and appearance knowledge could be used in the reconstruction. A very promising direction of research would be to capture the variance of shape and appearance inside low dimensional latent spaces. Next, given the depth and imagery input, we could jointly optimize for the pose and shape, but with the latter constrained within a learned latent space. For example, Dame , Prisacariu, Ren and Reid [38] have started to explore this idea by using a learned 3D shape space to improve dense SLAM reconstruction. The pose of the object and the shape is jointly optimized with the constraint that the 3D shape should only evolve within the learned shape space. However, if the variance of deformable shapes can be captured either off-line or on-line in a lower dimensional space, the information can be used to remove the assumption of rigidity towards more generic tracking and reconstruction of possibly deformable

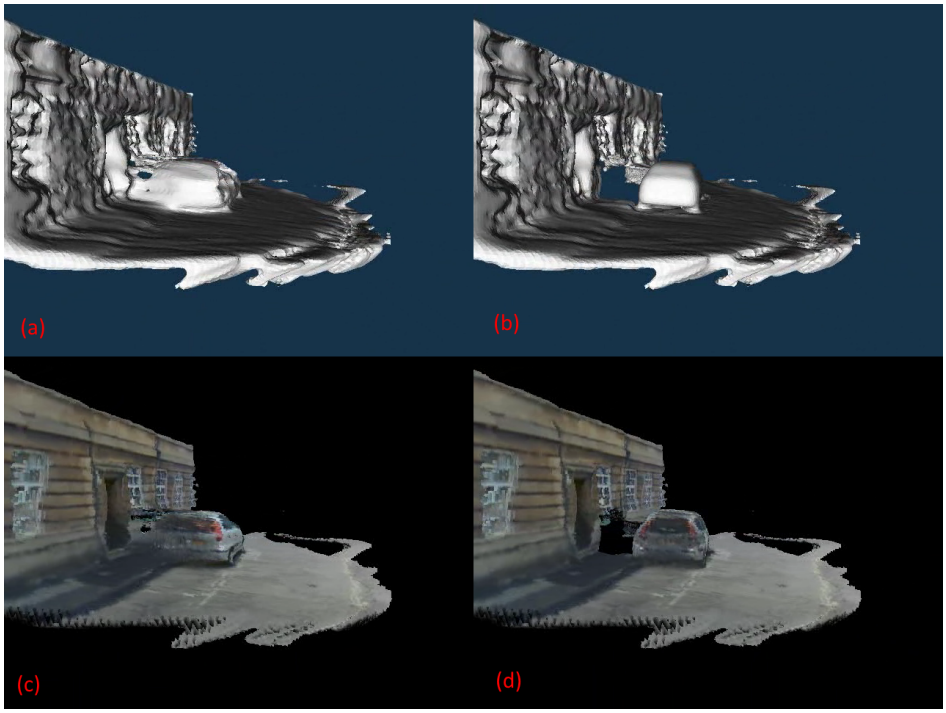



**Figure 7.1:** Results from Dame, Prisacariu, Ren and Reid [38]: (a, c) reconstruction results from a dense SLAM system. (b, d) improved result by estimating the car shape and pose with the learned latent car shapes space.

objects.

## 7.3 Current status of project REWIRE

Finally, we revisit the current status of project REWIRE. A prototype of the REWIRE platform is currently deployed in the Hospital Universitario Virgen del Rocio, Seville, Spain and the University Hospital Zurich, Switzerland, and is under clinical trail.

Because the trails are still underway, and because of patient confidentiality, we are unable to report detailed results in this thesis. The project partners report that the platform appears to be successful, that the patients are enjoying the various exercises, and that they are successfully performing their rehabilitation sessions both at home and within the hospital environment. We expect to have more objective

measurement of success by the end of the prototype deployment stage, *i.e.* early 2015.

# A

# gSLIC: A Real-time Implementation of SLIC Superpixel Segmentation

Over-segmentation of images has a wide range of applications in computer vision. Unfortunately, most state-of-the-art superpixel segmentation methods suffers from a high computational cost, which make them unable to be used in real-time systems. Achanta *et al.* [5] introduced simple iterative clustering (SLIC) algorithm to efficiently produce compact and nearly uniform superpixels. The simplicity, efficiency and the performance of the algorithm make it faster and more practical for real-time systems than other existing superpixel segmentation methods, like Normalized cuts [90] and QuickShift [140]. But still, the CPU-sequential implementation of SLIC works at 300∼400ms to segment a 640x480 image. Reducing the number of iterations for each clustering can make the algorithm faster, but this will suffer form loss of performance.

In this appendix, we detail an implementation of the SLIC algorithm using NVIDIA CUDA framework. We present improvement 10× ∼20× from the original CPU implementation of Achanta *et al.* [5]. We are not the first to attempt fast image segmentation on GPU, notably Fulkerson and Soatto [45], which presents exact GPU implementation of the quick shift superpixel segmentation algorithm.

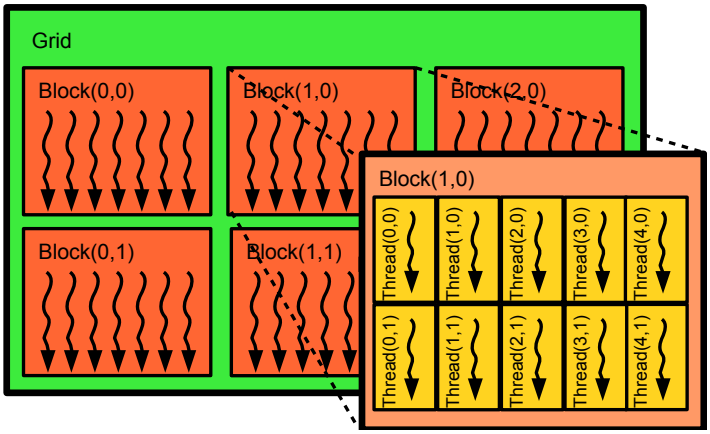



**Figure A.1:** NVIDIA CUDA thread model after after [95]

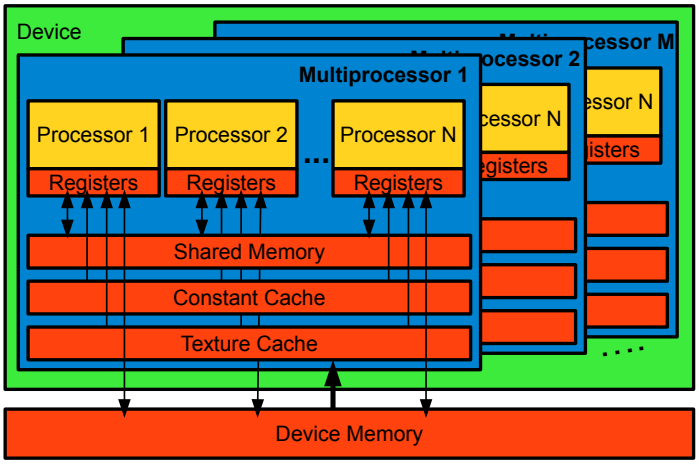



**Figure A.2:** NVIDIA CUDA memory model after [95]

Regardless of the difference in algorithm themselves at the moment, our gSLIC implementation is around 10× faster then Fulkerson and Soatto [45].

Our full source code with a simple example can be downloaded from http://www.robots.ox.ac.uk/~carl/code/gSLIC_with_Sample.zip, in the following sections, we will describe in detail our algorithm and implementation.

## A.1 GPU computing and NVIDIA CUDA

GPUs are traditionally been designed to be used for dense 3D graphic rendering, new-generation GPUs has made GPGPU (General-Purpose Computation on GPUs) available for sovling no-graphic computer vision problems. NVIDIA CUDA provides a set of SDK, software stack and compiler that allows for the implementation of programs in C for execution on GPU. The thread model of CUDA is shown in Fig. A.1. CUDA allows C functions (also called *kernels*) to be executed multiple times by multiple *threads*, on multiple GPUs. Each thread carries a *kernel*, and for complete utilization of GPU, thousands of *threads* will be used. *Threads* are grouped in *blocks*, and *blocks* are grouped into *grids* (Fig. A.1). Threads in a *block* share memory and synchronize while *blocks* in a *grid* are independent. Each thread block is executed on only one multiprocessor but a multiprocessor can execute several blocks at the same time. The memory model of CUDA is shown in Fig. A.2. As it is shown, a thread (executed on a processor within a multiprocessor) have a access to 6 different types of memory: register, local, shared, global (device), constant and texture memory. Each processor in multiprocessor has its own set of register and local memory; each multiprocessor has its on-chip shared, constant and texture cache and shared memory (constant texture cache are both read-only, shared memory is read-write). Global (device) memory access is much slower than on-chip and built-in memory. Consequently the bottle neck in CUDA based software is often global (device) memory access.

## A.2 Simple Linear Iterative Clustering (SLIC)

The Simple Linear Iterative Clustering (SLIC) algorithm for superpixel segmentation is proposed in Achanta *et al.* [5]. An example of segmentation result is shown in Fig. A.3.

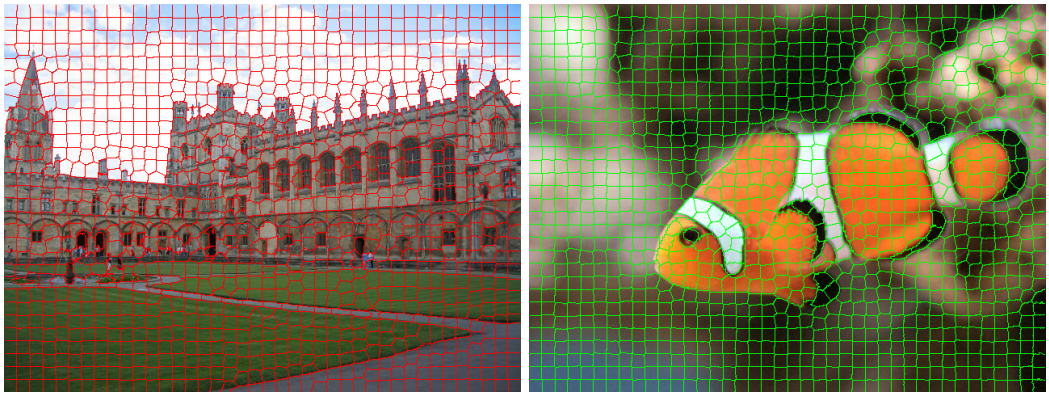


**Figure A.3:** example result of SLIC superpixel segmentation

The SLIC superpixel segmentation algorithm is a k-means-based local clustering of pixels in the 5-D $[l\ a\ b\ x\ y]$ space defined by the $L, a, b$ values of the CIELAB color space and the $x, y$ pixel coordinates. The reason why CIELAB color space is chosen is that it is perceptually uniform for small color distance. Instead of directly using the Euclidean distance in this 5-D space, SLIC introduce a new distance measure that considers superpixel size. The SLIC algorithm takes as input a desired number of approximately equally-sized superpixel $K$, then for a image with $N$ pixels, the approximate size of each superpixel is $N/K$. For roughly equally sized superpixels there would be a superpixel center at every grid interval $S = \sqrt{N/K}$. Let $[l_i, a_i, b_i, x_i, y_i]^T$ be the 5-D point of a pixel, cluster center $C_k$ should be of the same form $[l_k, a_k, b_k, x_k, y_k]^T$. The distance measure $D_k$ is defined as:

$$\begin{aligned} d_{lab} &= \sqrt{(l_k - l_i)^2 + (a_k - a_i)^2 + (n_k - b_i)^2}\ , \\ d_{xy} &= \sqrt{(x_k - x_i)^2 + (y_k - y_i)^2}\ , \\ D_s &= d_{lab} + \frac{m}{S} d_{xy}, \end{aligned} \tag{A.1}$$

where $D_s$ is the sum of the $lab$ distance and the $xy$ plane distance *normalized* by the grid interval $S$. Variable $m$ is introduced to control the compactness of superpixels. The greater the value of $m$, the more spatial proximity is emphasized and the more compact the cluster.

**Table A.1:** Modified SLIC superpixel segmentation algorithm.

Initialize cluster centers $[l_k, a_k, b_k, x_k, y_k]^T$ by sampling pixels at regular grid steps $S$.
Perturb cluster centers in to the lowest gradient position.
**for** each pixel
    Assign the pixel to the nearest cluster center based on initial grid interval $S$;
**repeat**
    **for** each pixel
        Locally search the nearby 9 cluster centers for the nearest one
        Label this pixel with the nearest cluster's index.
    Update each cluster center based on pixel assignment
    Compute residual error $E$($L1$ distance) between last and current iteration.
**until** $E \leq threadshould$
Enforce connectivity of superpixels

With the distance matric defined, the SLIC superpixel segmentation algorithm is simply local k-means algorithm, which is summarized in the **Algorithm 1** in Achanta *et al.* [5]. In order to make the most of the parallel computing on GPU, we modified the algorithm to enable one-thread-per-pixel computing, as summarized in Table A.1

## A.3 gSLIC implementation

As is shown in Fig. A.4, Our algorithm can be split into CPU and GPU two parts. The image is acquired by the host function running CPU, then transferred to GPU device memory. After color space transformation and segmentation has been done by GPU, segmentation mask is transferred back to host function again, where we run a recursion-based post processing function to enforce the connectivity of all superpixels.

The color space transformation part is naturally pixel-wise parallelizable, so we use 1 thread per pixel on $16 \times 16$ blocks. Then we use 1 thread per cluster to initialize cluster centers. The initial size of each cluster is determined by $S$

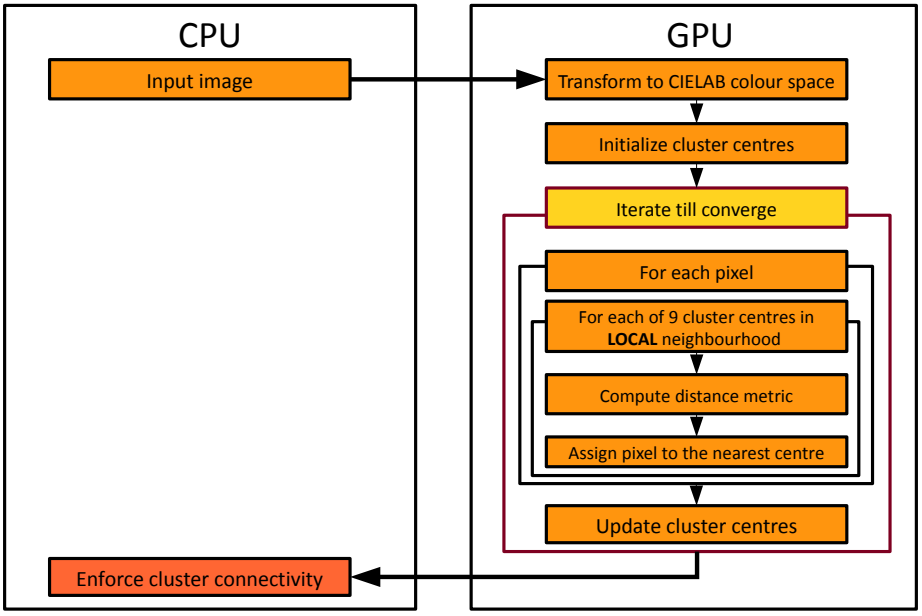



**Figure A.4:** Work flow of gSLIC

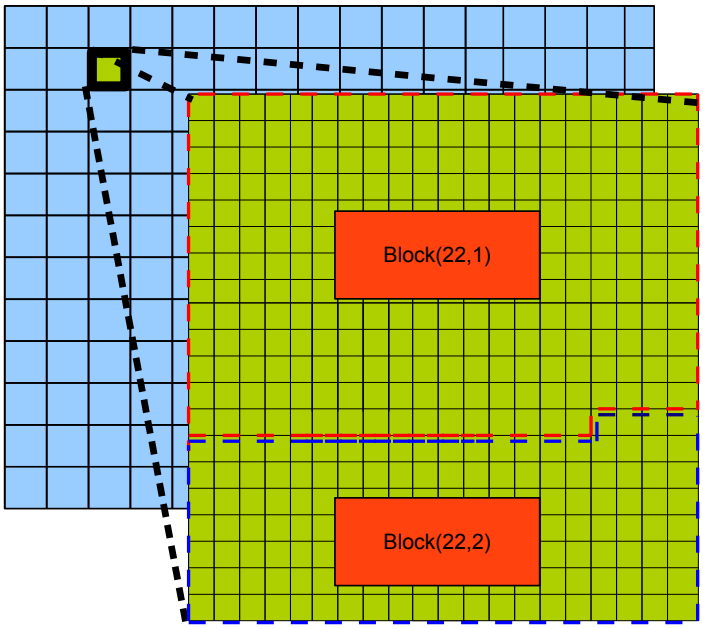



**Figure A.5:** Block arrangement example for gSLIC (Right)

defined in Section A.2. In order to keep compatibility with CUDA 1.0, in which the maximum number of thread per block is 512, we still use $16 \times 16$ fixed sized block in the local k-means iteration step. For most cases, the size of each cluster is larger than 256, thus clusters are consisted of multiple blocks, Fig. A.5 is an example of our block assignment. By using this block assignment, it is guaranteed that all threads within the same block need only to search the same set of cluster centers in neighborhood for the nearest one. Thus we pre-load the cluster centers' information into local shared memory for efficiency.In each iteration, after all pixels has been assigned a label (which is the index of the nearest center), we use one cluster per thread to update cluster center. The reason why we use on thread per cluster instead of on thread per pixel is to avoid atomic operation, which will slow down the whole algorithm greatly. Besides, This part could be accelerated by using parallel reduction algorithm, but in current version, since we have already obtained real-time performance, we did not implemented the parallel reduction. When the K-means iteration has converged, we transfer the labeled image back to host as segmentation mask. The post processing to enforce connectivity is the same algorithm as in Achanta *et al.* [5]. Since it is recursion based method, is not suitable for GPU computing, thus we put it on CPU host function.

## A.4 Library Usage

Because the source code for gSLIC is available online, we have decided to include a brief description of the usage of the library, as below.

**FastImgSeg** is the main class. It implements the full gSLIC superpixel segmentation algorithm. It need to be initialized by class constructor or by **initializeFastSeg**. Initialization take the size of image and number of segments as parameter. After initialization, call **LoadImg** to load user image. Note that current lib only take 4-channel image as input, the forth channel is reserved for depth informa-

| Implementation | 320×240 | 640×480 | 1280×960 |
|---|---|---|---|
| gSLIC | 9ms | 21ms | 86ms |
| SLIC [5] | 88ms | 354ms | 1522ms |

**Table A.2:** Example of full processing times for different image sizes

| Implementation | Kmeans Iteration (GPU) | Enforce Connectivity (CPU) |
|---|---|---|
| 320×240 | 6.5ms | 2.5ms |
| 640×480 | 13ms | 8ms |
| 1280×960 | 53ms | 33ms |

**Table A.3:** GPU and CPU time consumption for different image sizes

tion in later version. When user image has been loaded, call **DoSegmentation** to segment the image. Currently 3 methods are available: **SLIC** (SLIC in CIELAB space), **XYZ_SLIC** (SLIC in XYZ space), **RGB_SLIC** (SLIC in RGB space). The second parameter of **DoSegmentation** is the weight $m$ for spatial distance, as defined in Section A.2. When segmentation is finished, resulting segmentation mask will be stored in the public buffer **segMask**. User can also call **Tool_GetMarkedImg** to draw segmentation boundary on **markedImage** or call **Tool_WriteMask2File** to write segmentation mask to a file.

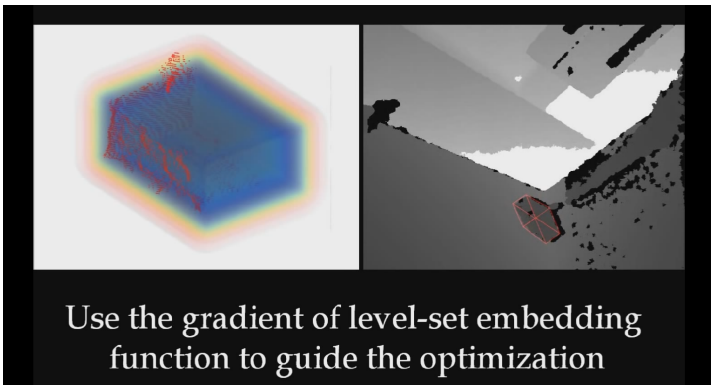
## A.5 Results

Our implementation is designed to produce the same result as the sequential SLIC implementation of Achanta *et al.* [5], thus we use their windows executable as our baseline method in our speed test experiment. We used an Intel Core i7-2600 (3.60GHz) machine with a Nvidia GTX460 GPU to run all our speed test. In Table A.2 we show the comparison between the processing time consumed by SLIC [5] and gSLIC for an single image at three sizes. The times of speeding up by using our GPU implementation increases with the size of image, achieving 10∼20 times faster than the original sequential implementation. In Table A.3 we show the processing time consumed by both the GPU Kmeans iteration part and CPU

enforce connectivity. The time consumption by the recursive enforce connectivity increases much faster than the GPU iteration part, so in the future work, we will introduce a parallel version of enforcing connectivity algorithm.

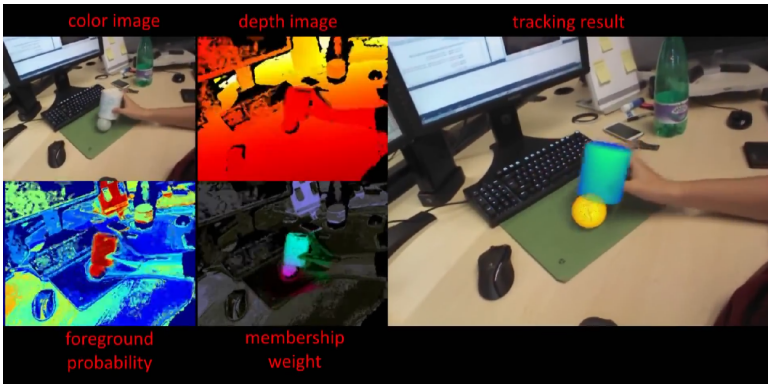
# B

# Video Material

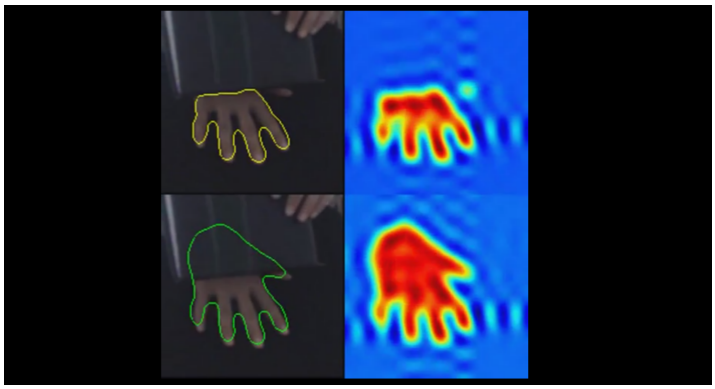



**Video B.1:** A Generalizable Probabilistic Framework for Model Fitting in Depth
http://youtu.be/Xh6Lyc_AaO8


**Video B.2:** STAR3D: Simultaneous Tracking And Reconstruction of 3D Objects
http://youtu.be/8ppQ4FtRNVc




**Video B.3:** 3D Tracking of Multiple Objects with Identical Appearance
http://youtu.be/BSkUee3UdJY



**Video B.4:** Shape Regression for Online Segmentation and Tracking
http://youtu.be/irqYhrgEilM

# Bibliography


[1] Kinect sensor. http://www.microsoft.com/en-us/kinectforwindows/.

[2] Nintendo balance board. http://wiibrew.org/wiki/Wii_Balance_Board.

[3] Nintendo wii remote. http://wiibrew.org/wiki/Wiimote.

[4] Royal college of physicians national sentinel stroke clinical audit 2010 round 7 public report for england, wales and northern ireland. Prepared on behalf of the Intercollegiate Stroke Working Party May 2011. page 43.

[5] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Ssstrunk. SLIC Superpixels. Technical report, EPFL, 2010.

[6] A. Adam, E. Rivlin, and I. Shimshoni. Robust fragments-based tracking using the integral histogram. In *Proc 19th IEEE Conf on Computer Vision and Pattern Recognition*, pages 798–805, 2006.

[7] D. Anguelov, P. Srinivasan, D. Koller, S. Thrun, J. Rodgers, and J. Davis. SCAPE: shape completion and animation of people. *ACM Trans Graph*, 24(3):408–416, 2005.

[8] M. Armstrong and A. Zisserman. Robust object tracking. In *Proc Asian Conference on Computer Vision*, 1995.

[9] B. G. Baumgart. *Geometric modeling for computer vision*. PhD thesis, Dept. of Computer Science, Stanford University, 1974.

[10] P. Beardsley, P. Torr, and A. Zisserman. 3D model acquisition from extended image sequences. In *Proc 4th European Conf on Computer Vision*, 1996.

[11] J. Beis and D. Lowe. Shape indexing using approximate nearest-neighbour search in high-dimensional spaces. In *Proc 1997 IEEE Computer Society Conf on Computer Vision and Pattern Recognition*, pages 1000–1006, 1997.

[12] P. J. Besl and N. D. McKay. A method for registration of 3-d shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239–256, 1992.

[13] C. Bibby and I. Reid. Robust real-time visual tracking using pixel-wise posteriors. In *Proc 10th European Conf on Computer Vision*, pages 831–844, 2008.

[14] C. Bibby and I. D. Reid. Robust real-time visual tracking using pixel-wise posteriors. In *Proc 10th European Conf on Computer Vision*, pages 831–844, 2008.

[15] M. B. Blaschko and C. H. Lampert. Learning to Localize Objects with Structured Output Regression. In *Proc 10th European Conf on Computer Vision*, pages 2–15, 2008.

[16] R. Boian, A. Sharma, C. Han, A. Merians, G. Burdea, S. Adamovich, M. Recce, M. Tremaine, and H. Poizner. Virtual reality-based post-stroke hand rehabilitation. *Studies in Health Technology and Informatics*, 85:64–70, 2002.

[17] N. A. Borghese, P. L. Lanzi, R. Mainetti, and M. Pirovano. IGER: An intelligent game engine for rehabilitation. In *Converging Clinical and Engineering Research on Neurorehabilitation, BIOSYSROB 1*, pages 947–950. Springer Berlin Heidelberg, 2013.

[18] N. A. Borghese, M. Pirovano, R. Mainetti, and P. L. Lanzi. Computational intelligence and game design for effective at-home stroke rehabilitation. *Games for Health Journal: Research, Development and Clinical Applications*, 2(2):81–88, 2013.

[19] A. Broadhurst, T. Drummond, and R. Cipolla. A probabilistic framework for the Space Carving algorithm. In *Proc 8th IEEE Int Conf on Computer Vision*, 2011.

[20] T. Brox, B. Rosenhahn, J. Gall, and D. Cremers. Combined region- and motion-based 3D tracking of rigid and articulated objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(3):402–415, 2009.

[21] J. W. Burke, M. D. J. McNeill, D. K. Charles, P. J. Morrow, J. H. Crosbie, and S. M. McDonough. Optimising engagement for stroke rehabilitation using serious games. *Visual Computer*, 25:1085–1099, 2009.

[22] D. Herrera C., J. Kannala, and J. Heikkila. Joint depth and color camera calibration with distortion correction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(10):2058–2064, 2012.

[23] M. S. Cameirao, S. B. Badia, E. D. Oller, and P. Verschure. Neurorehabilitation using the virtual reality based Rehabilitation Gaming System: methodology, design, psychometrics, usability and validation. *Journal of Neuroengineering and Rehabilitation*, 3:7–48, 2010.

[24] S. Chaudhuri and A. N. Rajagopalan. *Depth from defocus - a real aperture imaging approach.* Springer, 1999.

[25] J. Chen, D. Bautembach, and S. Izadi. Scalable real-time volumetric surface reconstruction. *ACM Trans. Graph.*, 32(4):113:1–113:16, 2013.

[26] P. Chockalingam, S. N. Pradeep, and S. Birchfield. Adaptive fragments-based tracking of non-rigid objects using level sets. In *Proc 12th IEEE Int Conf on Computer Vision*, pages 1530–1537, 2009.

[27] C. Choi and H.I. Christensen. Real-time 3d model-based tracking using edge and keypoint features for robotic manipulation. In *Proc 2010 IEEE Int Conf on Robotics and Automation*, pages 4048–4055, 2010.

[28] C. Choi, A. Trevor, and H. I. Christensen. RGB-D Object Tracking: A Particle Filter Approach on GPU. In *Proc IEEE/RSJ Conf on Intelligent Robots and Systems, Tokyo Big Sight, Japan, Nov 3-8, 2013*, 2013.

[29] R. Cipolla and A. Blake. Surface shape from the deformation of apparent contours. *International Journal of Computer Vision*, 9(2):83–112, 1992.

[30] R. A. Clark, Y. Pua, K. Fortin, C. Ritchie C, K. E, L. Denehy, and A. L. Bryant. Validity of the Microsoft Kinect for assessment of postural control. *Gait Posture*, 36(3):372–377, 2012.

[31] R. Colombo, F. Pisano, A. Mazzone, C. Delconte, S. Micera, M. C. Carrozza, P. Dario, and G. Minuco. Design strategies to improve patient motivation during robot-aided rehabilitation. *Journal of NeuroEngineering and Rehabilitation*, 4(1):3, 2007.

[32] A. Corduneanu and C. Bishop. Proc 8th International Conference on Artificial Intelligence and Statistics. In *Artifical Intelligence and Statistics*, pages 27–34, 2001.

[33] D. Cremers. Dynamical statistical shape priors for level set-based tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(8):1262–1273, Aug 2006.

[34] D. Cremers, S. Osher, and S. Soatto. Kernel density estimation and intrinsic alignment for knowledge-driven segmentation: Teaching level sets to walk. *International Journal of Computer Vision*, pages 36–44, 2004.

[35] B. Curless and M. Levoy. A volumetric method for building complex models from range images. In *Proc of 23rd Annual Conference on Computer Graphics and Interactive Techniques*, 1996.

[36] S. Dambreville, Y. Rathi, and A. Tannenbaum. A framework for image segmentation using shape models and kernel space shape priors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(8):1385–1399, 2008.

[37] S. Dambreville, R. Sandhu, A. Yezzi, and A. Tannenbaum. Robust 3D pose estimation and efficient 2D region-based segmentation from a 3D shape prior. In *Proc 10th European Conf on Computer Vision*, pages 169–182, 2008.

[38] A. Dame, V. A. Prisacariu, C. Y. Ren, and I. Reid. Dense reconstruction using 3D object shape priors. In *Proc 26th IEEE Conf on Computer Vision and Pattern Recognition*, 2013.

[39] T. Drummond and R. Cipolla. Real-time visual tracking of complex structures. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24:932–946, 2002.

[40] M. Fischler and R. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. In *Readings in computer vision: issues, problems, principles, and paradigms*, pages 726–740. Morgan Kaufmann Publishers Inc., 1987.

[41] A. Fitzgibbon. Robust registration of 2d and 3d point sets. In *Proc 12th British Machine Vision Conference*, 2001.

[42] A. W. Fitzgibbon and A. Zisserman. Automatic 3D model acquisition and generation of new images from video sequences. In *Proc 5th European Conf on Computer Vision*, pages 1261–1269, 1998.

[43] T. E. Fortmann, Y. Bar-Shalom, and M. Scheffe. Sonar tracking of multiple targets using joint probabilistic data association. *Oceanic Engineering, IEEE Journal of*, 8(3):173–184, Jul 1983.

[44] M. Fritz, B. Leibe, B. Caputo, and B. Schiele. Integrating representative and discriminant models for object category detection. In *Proc 10th IEEE Int Conf on Computer Vision*, pages 1363–1370, 2005.

[45] B. Fulkerson and S. Soatto. Really quick shift: Image segmentation on a gpu. Technical report, Department of Computer Science, University of California, Los Angeles, 2010.

[46] J. Gall, B. Rosenhahn, and H. Seidel. Drift-free tracking of rigid and articulated objects. In *Proc 21st IEEE Conf on Computer Vision and Pattern Recognition*, pages 1–8, 2008.

[47] B. Han and L. S. Davis. On-line density-based appearance modeling for object tracking. In *Proc 10th IEEE Int Conf on Computer Vision*, pages 1492–1499, 2005.

[48] C. Harris. Tracking with rigid models. *Active Vision*, pages 59–73, 1993.

[49] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, second edition, 2004.

[50] R. Held, A. Gupta, B. Curless, and M. Agrawala. 3D puppetry: a Kinect-based interface for 3D animation. In *Proc ACM Symposium on User Interface Software and Technology*, pages 423–434, 2012.

[51] B. K. P. Horn and M. J.(Eds) Brooks. *Shape from Shading*. MIT press, Cambridge, Massachusetts, 1989.

[52] N. Kyriazis I. Oikonomidis and A. Argyros. Efficient model-based 3d tracking of hand articulations using kinect. In *Proc 21st British Machine Vision Conference*, 2011.

[53] M. Isard and A. Blake. CONDENSATION - conditional density propagation for visual tracking. *International Journal of Computer Vision*, 29:5–28, 1998.

[54] A. D. Jepson, D. J. Fleet, and T. F. El-Maraghi. Robust online appearance models for visual tracking. In *Proc 15th IEEE Conf on Computer Vision and Pattern Recognition*, pages 415–422, 2001.

[55] Z. Khan, T. Balch, and F. Dellaert. Mcmc-based particle filtering for tracking a variable number of interacting targets. In *Proc 8th European Conf on Computer Vision*, 2004.

[56] K. Kim, V. Lepetit, and W. Woo. Keyframe-based modeling and tracking of multiple 3d objects. In *Proc 9th IEEE/ACM Int Symp on Mixed and Augmented Reality*, pages 193–198, 2010.

[57] G. Klein and D. Murray. Full-3D Edge Tracking with a Particle Filter. In *Proc 16th British Machine Vision Conference*, pages 1119–1128, 2006.

[58] G. Klein and D. Murray. Parallel tracking and mapping for small AR workspaces. In *Proc 6th IEEE/ACM Int Symp on Mixed and Augmented Reality*, 2007.

[59] D. Koller, K. Daniilidis, and H.-H. Nagel. Model-based object tracking in monocular image sequences of road traffic scenes. *International Journal of Computer Vision*, 10:257–281, 1993.

[60] D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, 2009.

[61] H. Kollnig and H.-H. Nagel. 3D pose estimation by directly matching polyhedral models to gray value gradients. *International Journal of Computer Vision*, 23:283–302, 1997.

[62] M. S. Kopicki, S. Zurek, R. Stolkin, T. Morwald, and J. L. Wyatt. Learning to predict how rigid objects behave under simple manipulation. In *Proc 2011 IEEE Int Conf on Robotics and Automation*, pages 5722–5729, 2011.

[63] H. S. Koppula, A. Anand, T. Joachims, and A. Saxena. Semantic labeling of 3d point clouds for indoor scenes. In *Advances in Neural Information Processing Systems*, 2011.

[64] F. Kuhl and C. Giardina. Elliptic fourier features of a closed contour. *Computer Graphics and Image Processing*, 18(3):236–258, 1982.

[65] K. N. Kutulakos and S. M. Seitz. A theory of shape by space carving. *International Journal of Computer Vision*, 38(3):199–218, 2000.

[66] S. Kwak, W. Nam, B. Han, and J. H. Han. Learning occlusion with likelihoods for visual tracking. In *Proc 13th IEEE Int Conf on Computer Vision*, pages 1551–1558, 2011.

[67] N. Kyriazis and A. Argyros. Physically plausible 3d scene tracking: The single actor hypothesis. In *Proc 26th IEEE Conf on Computer Vision and Pattern Recognition*, pages 9–16, 2013.

[68] N. Kyriazis and A. Argyros. Physically plausible 3d scene tracking: The single actor hypothesis. In *Proc 27th IEEE Conf on Computer Vision and Pattern Recognition*, 2014.

[69] A. Laurentini. The visual hull concept for silhouette-based image understanding. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(2):150–162, 1994.

[70] S. A. Lauterbach, M. H. Foreman, and J. R. Engsberg. Computer games as therapy for persons with stroke. *Game for Health*, 2(1):311–318, 2013.

[71] N. Lawrence. Probabilistic non-linear principal component analysis with gaussian process latent variable models. *Journal of Machine Learning Research*, 6:1783–1816, 2005.

[72] V. Lepetit, P. Lagger, and P. Fua. Randomized trees for real-time keypoint recognition. In *Proc 18th IEEE Conf on Computer Vision and Pattern Recognition*, pages 775–781, 2005.

[73] K. Levenberg. A method for the solution of certain non-linear problems in least squares. *Quarterly of Applied Mathematics*, 2(2):164C168, 1944.

[74] M. Leventon, E. Grimson, and O. Faugeras. Statistical shape influence in geodesic active contours. In *Proc 14th IEEE Conf on Computer Vision and Pattern Recognition*, volume 1, pages 316–323, 2000.

[75] C. Li, C.Xu, C. Gui, and M. D. Fox. Level set evolution without re-initialization: a new variational formulation. In *Proc 18th IEEE Conf on Computer Vision and Pattern Recognition*, 2005.

[76] M. Li, W. Chen, K. Huang, and T. Tan. Multi-target tracking by learning class-specific and instance-specific cues. In *Proc Asian Conference on Computer Vision*, pages 67–81, 2010.

[77] M. Li, W. Chen, K. Huang, and T. Tan. Visual tracking via incremental self-tuning particle filtering on the affine group. In *Proc 23rd IEEE Conf on Computer Vision and Pattern Recognition*, pages 1315–1322, 2010.

[78] M. Li, Q. Hu, Y. Wang, and W. Dong. Human detection based on pyramidal statistics of oriented filtering and online learned scene geometrical model. *Neurocomputing*, 101:338–346, 2013.

[79] D. Lowe. Robust model-based motion tracking through the integration of search and estimation. *International Journal of Computer Vision*, 8(2):113–122, 1992.

[80] D. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[81] J. MacCormick and A. Blake. A probabilistic exclusion principle for tracking multiple objects. In *Proc 7th IEEE Int Conf on Computer Vision*, 1999.

[82] P. C. Mahalanobis. On the generalised distance in statistics. In *Proceedings National Institute of Science, India*, volume 2, pages 49–55, 1936.

[83] R. Mainetti, A. Sedda, M. Ronchetti, G. Bottini, and N. A. Borghese. Duck-neglect: video-games based neglect rehabilitation. *Technology and Health Care*, 21:97–111, 2013.

[84] D. W. Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *SIAM Journal on Applied Mathematics*, 11(2):431–441, 1963.

[85] D. Marr and E. Hildreth. Theory of Edge Detection. *Proceedings of the Royal Society of London. Series B. Biological Sciences*, 207(1167):187–217, 1980.

[86] C. D. Mathers and D. Loncar. Projections of global mortality and burden of disease from 2002 to 2030. *PLoS Medicine*, 3(11):e442, 2006.

[87] W. Matusik, C. Buehler, R. Raskar, S.J. Gortler, and L. McMillan. Image-based visual hulls. In *Proc of 27th Annual Conference on Computer Graphics and Interactive Techniques*, 2000.

[88] D. Michel, X. Zabulis, and A. Argyros. Shape from interaction. *Machine Vision and Applications*, 25(4):1077–1087, 2014.

[89] M. Mirmehdi, J. Chiverton, and X. Xie. On-line learning of shape information for object segmentation and tracking. In *Proc 19th British Machine Vision Conference*, 2009.

[90] G. Mori. Guiding model search using segmentation. In *Proc 10th IEEE Int Conf on Computer Vision*, pages 1417–1423, 2005.

[91] S. K. Nayar, M. Watanabe, and M. Noguchi. Real-time focus range sensor. In *Proc 5th IEEE Int Conf on Computer Vision*, 1995.

[92] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohli, J. Shotton, S. Hodges, and A. W. Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *Proc 10th IEEE/ACM Int Symp on Mixed and Augmented Reality*, 2011.

[93] R. A. Newcombe, S. J. Lovegrove, and A. J. Davison. Dtam: Dense tracking and mapping in real-time. In *Proc 13th IEEE Int Conf on Computer Vision*, pages 2320–2327, 2011.

[94] M. Nießner, M. Zollhöfer, S. Izadi, and M. Stamminger. Real-time 3d reconstruction at scale using voxel hashing. *ACM Trans. Graph.*, 32(6):169:1–169:11, 2013.

[95] NVIDIA. *NVIDIA CUDA Programming Guide 3.0.*, 2010.

[96] I. Oikonomidis, N. Kyriazis, and A. A. Argyros. Full DOF tracking of a hand interacting with an object by modeling occlusions and physical constraints. In *Proc 13th IEEE Int Conf on Computer Vision*, 2011.

[97] I. Oikonomidis, N. Kyriazis, and A. A. Argyros. Tracking the articulated motion of two strongly interacting hands. In *Proc 25th IEEE Conf on Computer Vision and Pattern Recognition*, 2012.

[98] S. Osher and J. Sethian. Fronts propagating with curvature-dependent speed: Algorithms based on Hamilton-Jacobi formulations. *Journal of Computers*, 79(1):12–49, 1988.

[99] M. Ozuysal, V. Lepetit, F. Fleuret, and P. Fua. Feature harvesting for tracking-by-detection. In *Proc 9th European Conf on Computer Vision*, pages 592–605, 2006.

[100] T. Asfour P. Azad, D. Mnch and R. Dillmann. 6-dof model-based tracking of arbitrarily shaped 3d objects. In *Proc 2011 IEEE Int Conf on Robotics and Automation*, 2011.

[101] M. Pirovano, R. Mainetti, G. Baud-Bovy, P. L. Lanzi, and N. A. Borghese. Self-adaptive games for rehabilitation at home. In *Proc IEEE Conf Computational Intelligence and Games*, pages 179–186, 2012.

[102] M. Pirovano, C. Y. Ren, I. Frosio, P. Lanzi, V. Prisacariu, D. Murray, and N. A. Borghese. Robust silhouette extraction from kinect data. In *Proc 17th Int Conf on Image Analysis and Processing*, 2013.

[103] M. Pollefeys, D. Nistr, J.-M. Frahm, A. Akbarzadeh, P. Mordohai, B. Clipp, C. Engels, D. Gallup, S.-J. Kim, P. Merrell, C. Salmi, S. Sinha, B. Talton, L. Wang, Q. Yang, H. Stewnius, R. Yang, G. Welch, and H. Towles. Detailed real-time urban 3d reconstruction from video. *International Journal of Computer Vision*, 78(2-3):143–167, 2008.

[104] V. A. Prisacariu and I. Reid. Nonlinear shape manifolds as shape priors in level set segmentation and tracking. In *Proc 24th IEEE Conf on Computer Vision and Pattern Recognition*, pages 2185–2192, 2011.

[105] V. A. Prisacariu and I. Reid. Shared shape spaces. In *Proc 13th IEEE Int Conf on Computer Vision*, 2011.

[106] V. A. Prisacariu and I. D. Reid. PWP3D: Real-time segmentation and tracking of 3D objects. In *Proc 19th British Machine Vision Conference*, 2009.

[107] V. A. Prisacariu, A. V. Segal, and I. D. Reid. Simultaneous monocular 2D segmentation, 3D pose recovery and 3D reconstruction. In *Proc 10th Asian Conference on Computer Vision*, pages 593–606, 2012.

[108] C. E. Rasmussen and C. K. I. Williams. *Gaussian processes for machine learning*. The MIT Press, 2006.

[109] Y. Rathi, S. Dambreville, and A. Tannenbaum. Statistical shape analysis using kernel pca. In *Proc. Algorithms and Systems, Neural Networks, and Machine Learning*, volume 6064, pages 425–432, 2006.

[110] S. Ravela, B. Draper, J. Lim, and R. Weiss. Adaptive tracking and model registration across distinct aspects. *Intelligent Robots and Systems*, pages 174–180, 1995.

[111] C. Y. Ren, V. Prisacariu, and I. Reid. Regressing local to global shape properties for online segmentation and tracking. *International Journal of Computer Vision*, 106(3):269–281, 2014.

[112] C. Y. Ren, V. A. Prisacariu, O. Kaehler, I. Reid, and D. Murray. 3d tracking of multiple objects with identical appearance using RGB-D input. In *Proc 2nd Int Conf on 3D Vision*, 2014.

[113] C. Y. Ren, V. A. Prisacariu, and I. D. Reid. STAR3D: Simultaneous tracking and reconstruction of 3D objects using RGB-D data. In *Proc 14th IEEE Int Conf on Computer Vision*, 2013.

[114] C. Y. Ren and I. Reid. A unified energy minimization framework for model fitting in depth. In *Computer Vision – ECCV 2012. Workshops and Demonstrations*, pages 72–82. Springer Berlin Heidelberg, 2012.

[115] C.Y. Ren and I. Reid. gSLIC: a real-time implementation of slic superpixel segmentation. Technical report, University of Oxford, Department of Engineering Science, 2011.

[116] Y. Ren, V. Prisacariu, and I. Reid. Regressing Local to Global Shape Properties for Online Segmentation and Tracking. In *Proc 22nd British Machine Vision Conference*, 2011.

[117] B. Rosenhahn, T. Brox, and J. Weickert. Three-dimensional shape knowledge for joint image segmentation and pose tracking. *International Journal of Computer Vision*, 73(3):243–262, 2007.

[118] D. A. Ross, J. Lim, and M.-H. Yang. Adaptive probabilistic visual tracking with incremental subspace update. In *Proc 8th European Conf on Computer Vision*, pages 470–482, 2004.

[119] E. Rosten and T. Drummond. Machine learning for high-speed corner detection. In *Proc 9th European Conf on Computer Vision*, 2006.

[120] M. Rousson and N. Paragios. Shape priors for level set representations. In *Proc 7th European Conf on Computer Vision*, pages 78–92, 2002.

[121] R. B. Rusu, A. M., M. Beetz, and B. P. Gerkey. Extending player/stage/gazebo towards cognitive robots acting in ubiquitous sensor-equipped environments. In *IEEE International Conference on Robotics and Automation (ICRA) Workshop for Network Robot Systems*, 2007.

[122] J. Schell. *The Art of Game Design: A Book of Lenses*. Elsevier, 2008. ISBN: 978-0-12-369496-6.

[123] C. Schmaltz, B. Rosenhahn, T. Brox, D. Cremers, J. Weickert, L. Wietzke, and G. Sommer. Region-based pose tracking. In *Proc Iberian Conference on Pattern Recognition and Image Analysis*, pages 56–63, 2007.

[124] B. Schölkopf, A. Smola, and K.-R. Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10:1299–1319, 1998.

[125] D. Schulz, W. Burgard, D. Fox, and AB. Cremers. Tracking multiple moving targets with a mobile robot using particle filters and statistical data association. In *Proc 2001 IEEE Int Conf on Robotics and Automation*, 2001.

[126] S. Seitz, B. Curless, J. Diebel, D. Scharstein, and S. Richard. A comparison and evaluation of multi-view stereo reconstruction algorithms. In *Proc 19th IEEE Conf on Computer Vision and Pattern Recognition*, 2006.

[127] S. Seitz and C. R. Dyer. Photorealistic scene reconstruction by voxel coloring. *International Journal of Computer Vision*, 35(2), 1999.

[128] M. D. Shuster. A survey of attitude representations. *The Journal of the Astronautical Sciences*, 41(4):439–517, 1993.

[129] L. Sigal, A. Balan, and M. Black. HumanEva: Synchronized Video and Motion Capture Dataset and Baseline Algorithm for Evaluation of Articulated Human Motion. *International Journal of Computer Vision*, 87:4–27, 2010.

[130] I. Skrypnyk and D. Lowe. Scene modelling, recognition and tracking with invariant image features. In *Proc 3rd IEEE/ACM Int Symp on Mixed and Augmented Reality*, pages 110–119, 2004.

[131] R. Smith. Open Dynamics Engine, 2006. http://www.ode.org/.

[132] A. J. Smola and B. Schölkopf. A tutorial on support vector regression, 2004.

[133] A. Thayananthan, B. Stenger, P. H. S. Torr, and R. Cipolla. Shape context and chamfer matching in cluttered scenes. In *Proc 16th IEEE Conf on Computer Vision and Pattern Recognition*, 2003.

[134] N. Townsend, K. Wickramasinghe, P. Bhatnagar, K. Smolina, M. Nichols, J. Leal, R. Luengo-Fernandez, and M. Rayner. *Coronary heart disease statistics*, page 57. British Heart Foundation: London, 2012 edition, 2012.

[135] B. Triggs, P. McLauchlan, R. Hartley, and A. Fitzgibbon. Bundle adjustment – a modern synthesis. In *VISION ALGORITHMS: THEORY AND PRACTICE, LNCS*, pages 298–375. Springer Verlag, 2000.

[136] A. Tsai, A. Yezzi, W. Wells, C. Tempany, D. Tucker, A. Fan, E. Grimson, and A. Willsky. A shape-based approach to the segmentation of medical imagery using level sets. *Transactions on Medical Imaging*, 22(2):137–154, 2003.

[137] R. Ueda. Tracking 3d objects with point cloud library, 2012. http://www.pointclouds.org.

[138] M. Uenohara and T. Kanade. Vision-based object registration for real-time image overlay. In *Journal of Cognitive Neuroscience*, pages 13–22, 1991.

[139] L. Vacchetti, V. Lepetit, and P. Fua. Combining edge and texture information for real-time accurate 3d camera tracking. In *International Symposium on Mixed and Augmented Reality*, pages 48–57, 2004.

[140] A. Vedaldi and S. Soatto. Quick shift and kernel methods for mode seeking. In *Proc 10th European Conf on Computer Vision*, 2008.

[141] L. Vese and T. Chan. A multiphase level set framework for image segmentation using the mumford and shah model. *International Journal of Computer Vision*, 50(3):271–293, 2002.

[142] S. Vijayakumar, A. D'Souza, and S. Schaal. Incremental online learning in high dimensions. *Neural Computation*, 17:2602–2634, 2005.

[143] A. B. Watson. Image compression using the discrete cosine transform. *Mathematica Journal*, 4:81–88, 1994.

[144] A. Weiss, D. A. Hirshberg, and M. J. Black. Home 3D body scans from noisy image and range data. In *Proc 13th IEEE Int Conf on Computer Vision*, pages 1951–1958, 2011.

[145] M. Wuthrich, P. Pastor, M. Kalakrishnan, J. Bohg, and S. Schaal. Probabilistic object tracking using a range camera. In *Proc IEEE/RSJ Conf on Intelligent Robots and Systems, Tokyo Big Sight, Japan, Nov 3-8, 2013*, 2013.

[146] J. Xiao. Sfmedu: Multi-view 3d reconstruction for dummies. http://vision.princeton.edu/courses/SFMedu.

[147] A. Yilmaz, X. Li, and M. Shah. Contour-based object tracking with occlusion handling in video acquired using mobile cameras. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(11):1531–1536, 2004.

[148] R. Zhang, P.-S. Tsai, J. E. Cryer, and M. Shah. Shape-from-shading: a survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(8):690–706, 1999.

[149] Z. Zhang. A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(11):1330 – 1334, 2000.

[150] L. Zimmerli, C. Krewer, R. Gassert, F. Müller, R. Reiner, and L. Lünenburger. Validation of a mechanism to balance exercise difficulty in robot- assisted upper-extremity rehabilitation after stroke. *Journal of NeuroEngineering and Rehabilitation*, 9(6):1–13, 2012.