

Visual Object Perception in Unstructured Environments

A Thesis
Presented to
The Academic Faculty

by

Changhyun Choi

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy

Robotics Ph.D. Program
School of Interactive Computing
College of Computing
Georgia Institute of Technology
December 2014

Copyright © 2014 by Changhyun Choi

Visual Object Perception in Unstructured Environments

Approved by:

Professor Henrik I. Christensen, Advisor
School of Interactive Computing
College of Computing
Georgia Institute of Technology

Professor Anthony Yezzi
School of Electrical & Computer Engineering
College of Engineering
Georgia Institute of Technology

Professor James M. Rehg
School of Interactive Computing
College of Computing
Georgia Institute of Technology

Professor Dieter Fox
Department of Computer Science & Engineering
University of Washington

Professor Irfan Essa
School of Interactive Computing
College of Computing
Georgia Institute of Technology

Date Approved: August 22, 2014

*To my parents,
who have believed in my prolonged dream.*

ACKNOWLEDGEMENTS

First of all, I would like to thank my advisor, Henrik Christensen, for his kind advice and thoughtful guidance. I appreciate that he had believed in me all the time, even when I lacked confidence in my strength. I am also grateful to my doctoral thesis committee, Jim Rehg, Irfan Essa, Anthony Yezzi, and Dieter Fox, for their constructive comments and priceless suggestions. Thanks to you all, my thesis has been much improved since my proposal draft.

Meeting kind friends in a strange foreign place has been a valuable present. I would like to thank the CogRob lab mates: Alex Trevor, John Rogers, Jake Huckaby, Carlos Nieto, Akansel Cosgun, Sasha Lambert, Rahul Sawhney, Kimoon Lee, Sungtae An, Siddharth Choudhary, Pushkar Kolhe, and Jayasree Kumar. I sincerely thank Heni Ben Amor for giving valuable comments on both my thesis and my prospective career. It was also a precious experience to study with brilliant Robotics Ph.D. students, such as Richard Roberts, Yong-Dian Jian, Baris Akgun, Duy-Nguyen Ta, Tucker Hermans, Misha Novitzky, Neil Dantam, Maya Cakmak, Crystal Chao, etc. I would also like to thank Kathy Cheek and Nina White for their kind administrative help.

Outside of Gatech, I had the great opportunity to collaborate with amazing researchers, including Yuichi Taguchi, Ming-Yu Liu, Srikumar Ramalingam, and Oncel Tuzel at MERL as well as Ross Knepper, Andrew Spielberg, and Mehmet Dogar at MIT. Study and research funding during my Ph.D. from General Motors, Boeing Company, MERL, KFAS, and Google Summer of Codes is greatly appreciated.

Finally, I am deeply indebted to my father and mother for giving me the freedom of pursuing my dream and to my sister Goeun for her warm support in a distance.

TABLE OF CONTENTS

DEDICATION	iii
ACKNOWLEDGEMENTS	iv
LIST OF TABLES	ix
LIST OF FIGURES	x
ACRONYMS	xii
MATHEMATICAL NOTATION	xiv
SUMMARY	xvii
I INTRODUCTION	1
1.1 Motivation and Challenges	3
1.1.1 Objects with and without Textures	3
1.1.2 Clutter	4
1.1.3 Object Discontinuities	4
1.1.4 Real-time Constraints	4
1.2 Definition and Scope	5
1.3 Thesis Statement	6
1.4 Summary of Contributions	6
1.5 Outline	7
1.6 Publication Note	7
II RELATED WORK	8
2.1 Inception of Object Perception	8
2.2 Monocular-based Approaches	9
2.2.1 Pose Estimation using Keypoints	9
2.2.2 Pose Estimation using Edges	10
2.2.3 Pose Tracking with Single Pose Hypothesis	10
2.2.4 Pose Tracking using Multiple Pose Hypotheses	11
2.3 Depth-based Approaches	12
2.3.1 Pose Estimation using 3D Point Clusters	12
2.3.2 Pose Estimation using 3D Point Descriptors	12
2.3.3 Pose Estimation using Pair Features	13
2.3.4 Pose Estimation using Templates	13
2.3.5 Pose Tracking with Depth Information	13
2.4 Parallelized Object Perception	14

III	COMBINING TEXTURE AND BOUNDARY INFORMATION (2D)	16
3.1	Contributions	16
3.2	Camera and Object Models	17
3.2.1	Camera Model	17
3.2.2	3D Object Model and Salient Edge Selection	18
3.3	Particle Filter on the $SE(3)$ Group	19
3.3.1	State and Measurement Equations	19
3.3.2	Particle Filter	20
3.3.3	AR State Dynamics	22
3.3.4	Particle Initialization using Keypoint Correspondences	22
3.3.5	Edge-based Measurement Likelihood	24
3.3.6	Optimization using IRLS	26
3.3.7	Re-initialization based on \widehat{N}_{eff}	27
3.4	Experimental Results	27
3.4.1	Effectiveness of Considering Edge Orientations	28
3.4.2	Effectiveness of Considering Multiple Pose Hypotheses	31
3.4.3	Effectiveness of Performing RANSAC	33
3.4.4	Effectiveness of Employing AR State Dynamics	37
3.4.5	Re-initialization	37
3.4.6	Comparison with BLORT tracker	38
3.5	Summary	41
IV	EXTENDING TO TEXTURELESS OBJECTS (2D)	42
4.1	Edge-based Pose Estimation	42
4.1.1	Generating Edge Templates	42
4.1.2	Coarse Pose Estimation	43
4.1.3	Annealed Particle Filtering	45
4.1.4	Symmetric Objects	46
4.2	Experimental Results	46
4.3	Summary	48
V	VOTING-BASED POSE ESTIMATION (3D)	49
5.1	Exploiting Object Color Information	49
5.2	Contributions	49
5.3	RGB-D Pose Estimation	50
5.3.1	Point Pair Feature	51
5.3.2	Color Point Pair Feature	51
5.3.3	Object Learning	52

5.3.4	Voting Scheme	54
5.3.5	Pose Clustering	56
5.4	Experimental Results	57
5.4.1	Object Models	57
5.4.2	Experiment Setup	58
5.4.3	Gaussian Noise	61
5.4.4	Synthetic Scenes: Multiple Objects Single Instance (MOSI)	66
5.4.5	Synthetic Scenes: Single Object Multiple Instances (SOMI)	73
5.4.6	Real Cluttered Scenes	74
5.4.7	Computation Time	81
5.5	Summary	82
5.6	Exploiting Object Boundary Information	82
5.7	Contributions	82
5.8	Boundary-based Pose Estimation	82
5.8.1	System Overview	82
5.8.2	Pair Features for Boundaries	84
5.8.3	Object Representation	86
5.8.4	Voting Scheme for <i>L2L</i> Feature	87
5.8.5	Pose Clustering	87
5.9	Experimental Results	88
5.9.1	Synthetic Data	90
5.9.2	Real Data	91
5.9.3	Bin-Picking System Performance	92
5.10	Summary	93
VI	OBJECT POSE TRACKING (3D)	94
6.1	Contributions	94
6.2	Likelihood Evaluation	96
6.3	Implementation Details	97
6.4	Experiments	98
6.4.1	Object Models	98
6.4.2	Synthetic Sequences	99
6.4.3	Real Sequences	102
6.5	Summary	102
VII	CONCLUSIONS	104
7.1	Summary	104
7.2	Conclusions	104
7.3	Future Directions	106

7.4 Concluding Remarks	107
APPENDIX A – IRLS AND ITS JACOBIAN DERIVATION	108
BIBLIOGRAPHY	111

LIST OF TABLES

3.1	RMS errors and computation time in synthetic image sequences (Baseline vs. PF)	32
3.2	RMS errors and computation time in synthetic image sequences (PF vs. BLORT)	40
5.1	Average computation time of the five approaches on the real dataset	79
5.1	Average computation time of the five approaches on the real dataset	80
5.2	Average numbers of pair features in the synthetic scene dataset and relative processing time	91
5.3	Average absolute pose estimation errors	93
5.4	Pickup success rate	93
6.1	RMS errors and computation time in synthetic RGB-D sequences (PCL vs. Our tracking)	100

LIST OF FIGURES

1.1	Challenges in visual object perception	3
3.1	Original CAD models and selected salient edges of our target objects	18
3.2	Determining salient edges	18
3.3	Residual determination for calculating the likelihood	24
3.4	Tracking results showing the effectiveness of considering edge orientation	29
3.5	The 6-D pose and normalized residual plots of the results in Figure 3.4 (a)	30
3.6	Tracking results showing the effectiveness of considering multiple pose hypotheses	31
3.7	Tracking results showing the effectiveness of performing RANSAC	33
3.8	Tracking results showing the effectiveness of employing AR state dynamics	34
3.9	The 6-D pose and normalized residual plots of the results in Figure 3.8 (a)	35
3.10	Tracking results showing the capability of re-initialization	36
3.11	The \widehat{N}_{eff} plot of the results in Figure 3.10 (a)	37
3.12	The 6-D pose and normalized residual plots of the results of our particle filter and BLORT on the synthetic “Book” object sequence (simple background case)	38
3.13	A comparison between the tracking results of our particle filter and BLORT on real image sequences	39
4.1	Polygonal mesh models and edge templates	43
4.2	Tracking results showing effectiveness of considering multiple hypotheses	47
4.3	Tracking results showing effectiveness of suppressing the rotating motion about the axis of symmetry	47
4.4	Initialization and annealed particle filtering	48
4.5	Tracking results showing the re-initialization capability	48
5.1	The Color Point Pair Feature (CPPF)	50
5.2	Aligning pair features in the intermediate coordinate system	53
5.3	Bit-encoded 64-bit CPPF key	56
5.4	Bit-encoded 64-bit vote	56
5.5	Polygonal mesh models of the test objects	58
5.6	Adding Gaussian noise in the synthetic noise dataset	61
5.7	Some “Clorox” examples of the synthetic noise dataset	62
5.8	Precision-recall curves of the noise experiment	63
5.9	Recognition rates against Gaussian noise σ	64
5.10	Selected pose estimation results in the MOSI dataset	67
5.11	Precision-recall curves of the multiple objects single instance (MOSI) experiment	68
5.12	Recognition rates of top N pose results in the MOSI experiment	69
5.13	Selected pose estimation results in the SOMI dataset	70
5.14	Precision-recall curves of the single object multiple instances (SOMI) experiment	71
5.15	Recognition rates of top N pose results in the SOMI experiment	72

5.16	Selected pose estimation results in the real cluttered dataset	75
5.17	Precision-recall curves for the real cluttered scene experiments	76
5.18	Recognition rates of top N pose results in the real cluttered scene experiments	77
5.19	System overview	83
5.20	Pair features for voting-based pose estimation	84
5.21	Geometric primitives \mathcal{M} for the pair features	85
5.22	Aligning line pair features in the intermediate coordinate system	87
5.23	Test objects	88
5.24	Detection rates against occlusion rates for the synthetic dataset	89
5.25	Two example scenes from the 500 synthetic scenes	90
5.26	Two example scenes from the real scans	91
5.27	Histograms of pose estimation errors	92
6.1	A tracking example	94
6.2	Multiple renderings for the likelihood evaluation	95
6.3	Mesh models for objects and kitchen	98
6.4	Camera trajectories in synthetic sequences	98
6.5	Tracking results on the “Orange Juice” and the “Kinect Box” synthetic sequences	99
6.6	The 6-DOF pose plots of the “Kinect Box” results	101
6.7	Tracking results on the “Tide” and “Milk” real sequences	102
6.8	Boxplots showing computation time of our tracking and the PCL tracking on the “Tide” synthetic sequence	103

ACRONYMS

AR	Autoregressive.
B2B	Boundary-to-Boundary.
BBF	Best Bin First.
BLORT	BLOCKS world Robotic vision Toolbox.
BSP	Binary Space Partitioning.
CAD	Computer-Aided Design.
CPPF	Color Point Pair Feature.
CPU	Central Processing Unit.
CUDA	Compute Unified Device Architecture.
CVFH	Clustered Viewpoint Feature Histogram.
DOF	Degrees Of Freedom.
EPnP	Efficient Perspective- n -Point.
FBO	FrameBuffer Object.
GLSL	OpenGL Shading Language.
GPU	Graphics Processing Unit.
HOG	Histogram of Oriented Gradients.
HSV	Hue, Saturation, and Value.
ICP	Iterative-Closest Point.
IRLS	Iteratively Reweighted Least Squares.
L2L	Line-to-Line.
MERL	Mitsubishi Electric Research Labs.
MOSI	Multiple Objects Single Instance.
OpenCV	Open Source Computer Vision.
OpenGL	Open Graphics Library.
PCL	Point Cloud Library.
PF	Particle Filter.
PPF	Point Pair Feature.
PnP	Perspective- n -Point.
RAM	Random-Access Memory.
RANSAC	RANdom SAmples Consensus.
RGB	Red, Green, and Blue.
RGB-D	RGB with Depth.
RMS	Root Mean Square.

S2B	Surface-to-Boundary.
S2S	Surface-to-Surface.
SIFT	Scale-Invariant Feature Transform.
SIMD	Single Instruction Multiple Data.
SISD	Single Instruction Single Data.
SLAM	Simultaneous Localization And Mapping.
SOMI	Single Object Multiple Instances.
SSE	Streaming SIMD Extensions.
SURF	Speeded Up Robust Features.
TSDF	Truncated Signed Distance Function.
VFH	Viewpoint Feature Histogram.

MATHEMATICAL NOTATION

(x, y)	The center location of the detected edge template.
$\hat{\mathbf{X}}_t^{*(n)} \in SE(3)$	The optimized particle state of $\mathbf{X}_t^{*(n)}$ by the IRLS.
$\hat{\mathcal{C}}$	The set of keypoint correspondences after the ratio test.
α_l	The particle survival rate at layer l .
α_m	The intermediate angle.
$\angle(\mathbf{v}_1, \mathbf{v}_2) \in [0, \pi)$	The angle between two vectors, \mathbf{v}_1 and $\mathbf{v}_2 \in \mathbb{R}^3$.
β_l	The rate of annealing at layer l .
$\sigma \in \mathbb{R}^3$	The quantization levels for color vectors.
$\Sigma_n \in \mathbb{R}^{N_z \times N_z}$	The covariance matrix of \mathbf{n}_t .
$\Sigma_w \in \mathbb{R}^{6 \times 6}$	The covariance matrix of the Wiener process noise.
δ	The matching cost of the object detection.
δ	The quantization level for distance.
γ_s	The search radius for (color) point pair feature.
$\hat{\boldsymbol{\mu}} \in \mathbb{R}^6$	The motion velocity that minimizes the residual vector.
$\kappa \in \mathbb{Z}$	The 64-bit encoded key of either \mathbf{F}_{PPF} or \mathbf{F}_{CPPF} .
λ_a	The AR process parameter.
λ_c	The parameter that controls the sensitivity of the ratio of $N_p - N_i$ to N_p to the likelihood.
λ_r	The parameter that controls the sensitivity of the arithmetic mean of the residual vector $\bar{\mathbf{r}}$ to the likelihood.
λ_v	The parameter that controls the sensitivity of the ratio of $N_z - N_{\hat{z}}$ to N_z to the likelihood.
λ_δ	The parameter that controls the sensitivity of the cost δ to the initial particle weight.
$\mathbf{c} \in \mathbb{R}^3$	The color vector.
$d\mathbf{W}_t \in \mathfrak{se}(3)$	The Wiener process noise on $\mathfrak{se}(3)$ at time t .
$\mathbf{E}_i \in \mathfrak{se}(3)$	The i^{th} basis elements of $\mathfrak{se}(3)$.
\mathbf{e}_i	The i^{th} edge in mesh models.
$\mathbf{F}_{CPPF} \in \mathbb{R}^{10}$	The color point pair feature descriptor.
$\mathbf{F}_{PPF} \in \mathbb{R}^4$	The point pair feature descriptor.
$\mathbf{J} \in \mathbb{R}^{N_z \times 6}$	The Jacobian matrix of $\mathbf{n}_i^T \mathbf{p}_i$ with respect to $\boldsymbol{\mu}$ obtained by computing partial derivatives at the current pose.
$\mathbf{K} \in \mathbb{R}^{2 \times 3}$	The intrinsic camera parameters.
$\mathbf{n} \in \mathbb{R}^3$	A face normal unit vector in mesh models.
$\mathbf{n} \in \mathbb{R}^2$	The unit normal vector in 2D images.
\mathbf{n}_t	The Gaussian noise in the measurement at time t .

$\mathbf{p} \in \mathbb{R}^2$	Projected 2D image coordinates of a 3D point.
$\mathbf{P} \in \mathbb{R}^4$	3D homogeneous coordinates of a 3D point.
$\mathbf{p}_i^m \in \mathbb{R}^3$	The reference model point.
$\mathbf{p}_j^m \in \mathbb{R}^3$	The referred model point.
$\mathbf{p}_i^s \in \mathbb{R}^3$	The reference scene point.
$\mathbf{p}_j^s \in \mathbb{R}^3$	The referred scene point.
$\mathbf{r} \in \mathbb{R}^{N_z}$	The residual vector from the residuals of visible sample points \mathbf{p}_i .
$\mathbf{R} \in SO(3)$	A rotation matrix.
$\mathbf{R}_x(\alpha) \in SO(3)$	The rotation matrix around the \mathbf{x} -axis with angle α .
$\mathbf{T}_{m \rightarrow g} \in SE(3)$	The transformations from the model coordinate systems to the intermediate coordinate system.
$\mathbf{T}_{s \rightarrow g} \in SE(3)$	The transformations from the scene coordinate systems to the intermediate coordinate system.
$\mathbf{W} \in \mathbb{R}^{N_z \times N_z}$	A weighted diagonal matrix.
$\mathbf{X}_t \in SE(3)$	A rigid body transformation or pose on the $SE(3)$ group at time t .
$\mathbf{X}_t^{(n)} \in SE(3)$	The n^{th} particle state at time t .
\mathcal{A}	The intermediate angle array.
\mathcal{C}	The set of putative keypoint correspondences.
\mathcal{D}	The set of object detections.
\mathcal{D}_h	The array of duplication numbers in \mathcal{H} .
\mathcal{F}	The set of keyframes.
\mathcal{H}	The array of quantized hash keys.
\mathcal{H}_r	The reduced array of \mathcal{H} .
\mathcal{H}_s	The sorted array of \mathcal{H} .
\mathcal{I}	An input image.
\mathcal{I}_s^r	The indices from the reduced array \mathcal{H}_r to the sorted array \mathcal{H}_s .
\mathcal{I}_o^s	The indices from the sorted array \mathcal{H}_s to the original array \mathcal{H} .
\mathcal{M}	The object model point cloud.
\mathcal{S}	The scene point cloud.
\mathcal{S}_t	The set of weighted particles at time t .
\mathcal{V}	The votes array in global memory.
$\nu \in \mathbb{Z}$	The 64-bit encoded vote.
$\exp : \mathfrak{se}(3) \mapsto SE(3)$	The exponential map from $\mathfrak{se}(3)$ onto $SE(3)$.
$\log : SE(3) \mapsto \mathfrak{se}(3)$	The logarithmic map from $SE(3)$ onto $\mathfrak{se}(3)$.
\oslash	Component-wise division.

$\pi_t^{(n)}$	The normalized weight of the n^{th} particle state at time t .
ρ	The probability in which at least one set of randomly sampled m correspondences is from inliers.
σ	The scale of the detected edge template.
τ_n	The minimum required number of keypoint correspondences for the initialization.
τ_r	The threshold value for the ratio test.
τ_s	The threshold value for the salient edge selection.
τ_ϵ	The Euclidean distance threshold value for determining inlier keypoint correspondences.
θ	The quantization level for angle.
$\theta_e(\mathbf{q}_i)$	The orientation of the image edge point \mathbf{q}_i .
$\theta_m(\mathbf{p}_i)$	The orientation of the model edge to which the sample point \mathbf{p}_i belongs.
$\tilde{\pi}_t^{(n)}$	The unnormalized weight of the n^{th} particle state at time t .
$\hat{\mathcal{C}}$	The set of keypoint correspondences having the maximum number of inlier correspondences between the input image \mathcal{I} and the keyframes \mathcal{F} .
\widehat{N}_{eff}	The approximated effective particle size.
$\tilde{\mathcal{C}}$	A set of randomly chosen keypoint correspondences from $\hat{\mathcal{C}}$.
$\{(\mathbf{p}_i, \mathbf{n}_i), (\mathbf{p}_j, \mathbf{n}_j)\}$	The pair of two oriented points.
${}^c\mathbf{T}_{\mathcal{O}} \in SE(3)$	The pose of the object with respect to the camera frame estimated by our visual tracker.
${}^c\hat{\mathbf{T}}_{\mathcal{O}} \in SE(3)$	The pose of the rendered object with respect to the virtual camera frame saved in OpenGL rendering.
${}^o\mathbf{u}_y = (0, 1, 0, 0)^\top$	The unit vector of the axis of symmetry of the object.
$A : SE(3) \mapsto \mathfrak{se}(3)$	A possibly nonlinear map.
$g : SE(3) \mapsto \mathbb{R}^{N_z}$	The nonlinear measurement function.
m	The minimum number of keypoint correspondences to perform EPnP .
N	The number of particles.
N_i	The number of inlier keypoint correspondences.
N_p	The number of putative keypoint correspondences.
N_v	The size of the votes array \mathcal{V}
N_z	The number of visible sample points.
N_{eff}	The effective particle size.
N_{thres}	The threshold value for the re-initialization.
$N_{\hat{z}}$	The number of matched sample points.
r_i	The residual which is the Euclidean distance between \mathbf{p}_i and \mathbf{q}_i .

SUMMARY

As robotic systems move from well-controlled settings to increasingly unstructured environments, they are required to operate in highly dynamic and cluttered scenarios. Finding an object, estimating its pose, and tracking its pose over time within such scenarios are challenging problems. Although various approaches have been developed to tackle these problems, the scope of objects addressed and the robustness of solutions remain limited. In this thesis, we target a robust object perception using visual sensory information, which spans from the traditional monocular camera to the more recently emerged RGB-D sensor, in unstructured environments. Toward this goal, we address four critical challenges to robust 6-DOF object pose estimation and tracking that current state-of-the-art approaches have, as yet, failed to solve.

The first challenge is how to increase the scope of objects by allowing visual perception to handle both textured and textureless objects. A large number of 3D object models are widely available in online object model databases, and these object models provide significant prior information including geometric shapes and photometric appearances. We note that using both geometric and photometric attributes available from these models enables us to handle both textured and textureless objects. This thesis presents our efforts to broaden the spectrum of objects to be handled by combining geometric and photometric features.

The second challenge is how to dependably estimate and track the pose of an object despite the clutter in backgrounds. Difficulties in object perception rise with the degree of clutter. Background clutter is likely to lead to false measurements, and false measurements tend to result in inaccurate pose estimates. To tackle significant clutter in backgrounds, we present two multiple pose hypotheses frameworks: a particle filtering framework for tracking and a voting framework for pose estimation.

Handling of object discontinuities during tracking, such as severe occlusions, disappearances, and blurring, presents another important challenge. In an ideal scenario, a tracked object is visible throughout the entirety of tracking. However, when an object happens to be occluded by other objects or disappears due to the motions of the object or the camera, difficulties ensue. Because the continuous tracking of an object is critical to robotic manipulation, we propose to devise a method to measure tracking quality and to re-initialize tracking as necessary.

The final challenge we address is performing these tasks within real-time constraints. Our particle filtering and voting frameworks, while time-consuming, are composed of repetitive, simple and independent computations. Inspired by that observation, we propose to run massively parallelized frameworks on a GPU for those robotic perception tasks which must operate within strict time constraints.

CHAPTER I

INTRODUCTION

Robot systems are gradually getting deployed in everyday situations for assistance and collaboration with humans. It implies that robots are being operated in environments that are at best semi-structured. A typical scenario involves pick-and-place tasks, where a robot has to detect the presence of an object, pick it up, and move it to another location. An important challenge is, thus, designing of a system that can detect, recognize and perform pose estimation for a diverse set of objects. That is by no means a new problem. Early work dates back to the 60s ([Roberts 1965](#)). Since then, there have been significant efforts to improve the robustness of the solution ([Guzman 1971](#), [Binford 1971](#), [Underwood and Coates Jr 1975](#), [Murase and Nayar 1995](#), [Lowe 2004](#), [Collet et al. 2011](#), [Hinterstoisser et al. 2012a](#)). Nonetheless, most of the available techniques are limited in terms of the scope of objects, robustness to background clutter and occlusions, and computational efficiency. Whereas simple geometric models had been used in the early stage of machine perception, local appearance descriptor models are popular nowadays. The former models are good for textureless objects, while the latter models are preferred for well-textured objects. Although these two kinds of models complement each other, surprisingly there has been few efforts to exploit both, and thus the variety of objects existing approaches can deal with is still limited. In addition, despite significant progress, most of the existing approaches have poor degradation in the presence of significant occlusions and clutter. We also scarcely see efficient object perception algorithms which perform within time constraints.

Along with the robotic migration from controlled environments to unstructured surroundings, we recently observe a big transition in imaging devices. Microsoft introduced a motion sensing device, Kinect, which has an **RGB** video camera and an infra-red depth sensor. Since its first release in November 2010, it has been sold more than 24 million units over the world ([Epstein 2013](#)), and ASUS (Xtion Pro Live) and Intel (Creative Sens3D) also released similar **RGB** with depth (**RGB-D**) cameras on the market. Moreover, these sensors are getting smaller and are eventually about to be embedded in mobile phones. Google recently announced its initiative for mobile depth sensing throughout the Project Tango ([Google 2014](#)), and Apple Inc. also recently bought the PrimeSense company which is behind the 3D sensing technology of Kinect ([Chen 2014](#)). Since early in the 21st century, the majority of mobile phones came with cameras. Photos and videos captured from the camera have been one of the most important media to deliver information, and a large amount of the media has gathered on the Internet. For instance, as of 2013, more than 350 million photos are uploaded every day on average, and more than 250 billion photos have been uploaded to Facebook ([Internet.org 2013](#)). Given the trend, there is no doubt that the **RGB-D** sensors will augment the dominating media and enable us to capture 3D world, to exchange 3D visual information, and to organize 3D data on the Internet.

This trend is especially promising in robotics, since robots are existing in 3D space and are required to interact in 3D. The monocular camera has been widely adopted as a visual sensory input for object perception. When a 3D representation of an object and the intrinsic calibration of the camera are known, it is possible to recover the full six degrees-of-freedom (DOF) transformation between the object and the sensor coordinate systems via 2D-3D feature matching. Though the 2D-3D configuration is a sufficient condition for the problem of pose estimation and tracking, an additional depth modality can be significantly beneficial to the object perception problem. Especially, the 3D-3D configuration is preferred because we can rigorously exploit various geometric features available from both 3D models and 3D depth scenes, and in particular it will be beneficial to handling of less textured or textureless objects. Moreover, subtracting foreground objects is a trivial task with the depth information, while it is still a difficult task in 2D images. Stereo rigs have been actively studied in the computer vision literature. However, they are mainly limited due to the unreliable depth values around less textured or textureless regions. The aforementioned RGB-D sensors overcome this limitation by adopting an active illumination sensing technique, and more importantly these affordable sensors provide depth data along with color image in real-time speed¹ that is an important, necessary condition for robotic sensors. Given this big transition in sensing devices, it is of interest to investigate how these sensors can impact on visual perception research.

Since the pioneering work of Roberts (1965), visual object perception had progressed based on 3D geometric models for several decades, often known as the ‘Geometric Era’ (Guzman 1971, Binford 1971). Though there had been significant progress during this era, it could not help but yield attention to appearance-based approaches (Murase and Nayar 1995, Lowe 2004) because of its restricted performance in case of textured surfaces and the assumption about the 3D models. Interestingly, together with the 3D sensing trend, it seems that revisiting 3D models in visual perception is timely more than ever. Nowadays, hundreds of thousands of 3D object models are getting accumulated on the Internet, such as TurboSquid (TurboSquid 2014), Google 3D Warehouse (Google 2013), and KIT object models web database (Kasper et al. 2012). So it is highly anticipated that a significant number of new object models will be additionally clustered on the Internet, as obtaining 3D models is going to be a trivial task with the RGB-D sensor (Newcombe et al. 2011). In addition, unlike purely geometric objects used in the classic geometric era, a large portion of existing 3D object models on the web also accompany color or texture information, which is easily attainable via the RGB-D sensor. This fact implies that *exploiting both geometric shape information and photometric appearance attributes is naturally anticipated, and thus closing the loop of the geometric era and the currently dominant appearance age might be possible in the near future*. Note that these 3D model databases are significant prior information for visual perception. Therefore, an interesting challenge is to consider how such prior information about geometry and appearance can be utilized in the object perception problems.

¹Real-time speed means 30 frames per second or 30 Hz (about 33 ms).

1.1 Motivation and Challenges

While many approaches have addressed the pose estimation and tracking problem over the last five decades (Klein and Murray 2006, Aldoma et al. 2011, Collet et al. 2011, Hinterstoisser et al. 2012a), their scope of targeted objects is still narrowly bounded. In addition, the robustness of the existing approaches is still limited in that they do not work properly in heavily cluttered environments. Moreover, most of the work for object tracking does not actively address the object discontinuous cases. Object perception does not tend to perform in a timely manner due mainly to the heavy computation of underlying algorithms. In this thesis, we were mainly motivated by these challenges and actively address them to contribute toward robust and efficient object perception in unstructured environments.

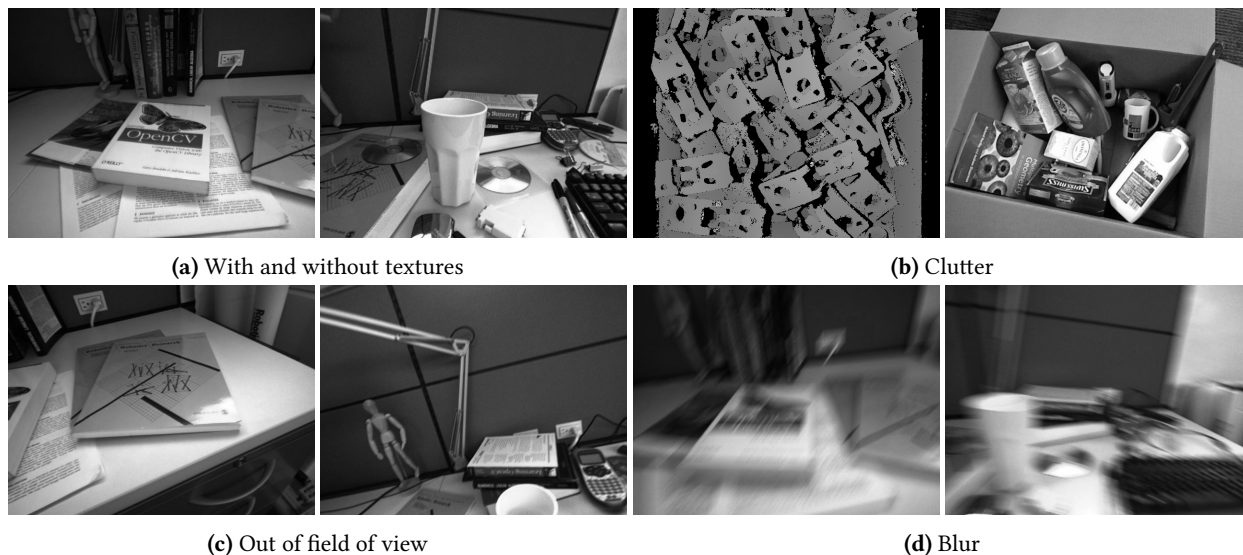


Figure 1.1. Challenges in visual object perception. (a) Perceiving both textured (book) and textureless (white cup) objects is still a challenging problem. (b) Cluttered backgrounds make data association harder, and hence pose estimates in significant clutter are often erroneous. (c, d) Object discontinuous cases may happen during tracking, such as going out of the field of view or image blur, due to the motions of objects or camera.

1.1.1 Objects with and without Textures

Handling both textured and textureless objects is still a challenging problem. Figure 1.1a depicts example objects with and without textures. When objects are well textured like the book object, some photometric features capturing texture are well suited. As invariant keypoints and their descriptors have been introduced, most of the recent progress in object recognition and pose estimation relies on keypoint matching. These keypoint-based approaches are applicable to highly textured objects, but many objects, such as dishes, glasses, industrial parts, still lack enough textures. For those textureless objects, it is required to exploit geometric features instead. For monocular approaches, edges from boundaries of objects are well suitable; for 3D depth-based approaches, depth discontinuities, point normals, or high curvature regions can be employed.

We employ both photometric and geometric features to handle both classes of objects. With a monocular camera,

we present our object pose estimation and tracking algorithm using keypoint and edge features. When an RGB-D sensor is available, we combine color, 3D points, depth discontinuities, and surface normals to perform pose estimation and tracking regardless of the degree of object texture.

1.1.2 Clutter

The difficulties of finding an object and tracking its pose depend on the degree of clutter in backgrounds. While most of the related efforts work well in backgrounds having a reasonable amount of clutter, it is still demanding to estimate the pose of an object with significant clutter as shown in Figure 1.1b. The clutter in backgrounds often causes false measurements, and these measurements tend to lead to false pose estimates. The false estimates commonly make tracking get stuck in local minima. When backgrounds are relatively simple, pose estimation approaches often adopt an object segmentation as a pre-processing step so that an object level matching scheme can be employed. However, this segmentation may not always be straightforward in heavily cluttered scenes. In addition, wrong segmented regions worsen precision of object perception.

For robust tracking, we devise a multiple pose hypotheses tracking based on a particle filtering framework. By maintaining multiple pose hypotheses, our tracking follows the global optimum despite significant background clutter. We also employ an additional refinement process based on the RANSAC algorithm in which wrong edge data associations are discarded to enhance the accuracy of pose estimates calculated from the measurement associations. In pose estimation, our approach does not hinge upon the top-down pre-processing segmentation which is easily deteriorated in cluttered settings. Instead, low-level pair features are used to determine feature correspondences between a model and a scene. Our approach is very robust with respect to background clutter by employing a voting framework, in which a set of most likely pose hypotheses is obtained as a bottom-up fashion.

1.1.3 Object Discontinuities

An ideal tracking scenario is that the tracked object is visible during the entire tracking. However, in reality the object happens to be occluded by other objects, human, or robots. As shown in Figure 1.1c and 1.1d, sometimes the object may be going out of the camera's field of view or blurred due to the motions of the object or the camera. These object discontinuous cases should be appropriately handled for realistic scenarios. However, few efforts addressed such cases. To tackle these challenges, the tracking is expected to re-initialize itself when the discontinuities occur.

We propose a re-initialization scheme based on a likelihood measure, which is calculated from the particle weights in our particle filter framework. From the measure, the local tracking triggers the global pose estimation when it is necessary. In this way, our pose estimation and tracking algorithms are systematically integrated with our particle filtering framework.

1.1.4 Real-time Constraints

Robotic perceptions are typically constrained by a timing limitation as they should reactively interact with objects or environments. Reliable robotic manipulations require high frame-rate tracking, and even a global pose estimation

algorithm should process a given scene in seconds to be effective in real scenarios. As we consider more objects and deal with more clutter, the search space of the problem is getting larger, and consequently, it is not straightforward to finish the computation under the time constraints.

One way to tackle the timing challenge is to exploit the power of parallel computation. Our particle filtering and voting frameworks are composed of a large number of repetitive yet simple operations, and thus, the frameworks can inherently be parallelizable. Modern graphics processing units (**GPU**) provide massive parallel computation power beyond rendering purposes for an affordable price. We investigate how to take advantage of the parallel power of the **GPU** to enable our approaches to comply with the time constraints.

1.2 Definition and Scope

The terminology in robotics and computer vision areas is convenient yet often confusing and even possibly misleading readers. To avoid that, we explicitly introduce important terms used in this thesis and mention the addressed scope of the thesis in this section.

In this thesis, we explore the problem of model-based visual object perception in unstructured environments. The meaning of **model-based** is that 3D object models are given *a priori*. Various 3D object model representations might be possible, but we choose the most general format, *3D polygonal mesh models*, since that representation has been widely adopted in the computer graphics area, and a significant number of geometric processing techniques on the model type are readily available. A broad definition of **object perception** is the task of identifying corresponding name or a unique identification number of a given object. In this thesis, however, it also includes **6-DOF** pose estimation and tracking because our main applications are in robotic perception. A **6-DOF pose** is a point on the smooth manifold, $SE(3)$ group, which represents a translation in 3D Euclidean space along with a 3D orientation in $SO(3)$ group (Ma et al. 2004). It usually describes a rigid body transformation between two coordinate systems, but here it mainly represents a rigid transformation from a sensor coordinate system to an object coordinate system. **Pose estimation** is the task of global searching for a set of most likely pose hypotheses complying with measurements while **pose tracking** is the locally recursive estimation of the posterior pose hypothesis given the prior pose hypothesis of the previous time step. These tasks can be achieved with various sensory modalities, but in this thesis we focus on **visual object perception**. The addressed visual sensors are **monocular** and **depth** cameras. The **unstructured environments** represent considerably *cluttered* scenes often with unexpected *occlusions* and any other *dynamic perturbations*.

Throughout this thesis, two sorts of visual features are employed: photometric and geometric features. **Photometric features** include image *intensity*, *color*, *edges from texture*, *keypoints* (including *corners*), and their local *descriptors*. These features are mainly obtained from photometric image formation in 2D. Another set of features is **geometric features** that are *edges from geometric shapes* (e.g. boundaries or sharp edges), *lines*, *planes*, *depth points*, *surface normals*, and *high curvatures*. While photometric features are mostly related to brightness, color, and texture,

geometric features are closely affiliated with geometric shapes. Please note that the photometric and geometric features are tightly interconnected. For instance, 2D edges from geometric shapes are typically calculated from gradient response images of an intensity image; color and intensity values are results of the interaction between lights and surface normals; keypoint and descriptors might depend on geometric texture.

1.3 Thesis Statement

A concise statement of this thesis is presented as follows:

“Multiple pose hypotheses frameworks, such as a particle filter for tracking and a voting procedure for pose estimation, incorporating both photometric and geometric features handle an increased spectrum of objects and are robust solutions for visual object perception in unstructured environments, wherein significant clutter and occlusions exist.”

1.4 Summary of Contributions

In this section, we summarize the main contributions of this thesis. The contributions are as follows:

- **Employing keypoint features to initialize tracking.** While most tracking approaches have not addressed the tracking initialization or initialize from scratch, we adopt a guided search based on a photometric feature (keypoint) matching between object models and an input image sequence. This initialization further enables the system to be capable of doing re-initialization. (Choi and Christensen 2010, 2011, 2012c)
- **Re-initialization for object discontinuities.** We proposed a re-initialization scheme that uses the number of effective particle size as a likelihood measure of tracking quality. (Choi and Christensen 2011, 2012c).
- **Refining edge correspondences with the RANSAC algorithm.** Edge data association between projected model wireframes and image edges is enhanced with a sampling-based hypothesize-and-testing framework to rule out wrong edge correspondences possibly from background clutter or non-Lambertian reflectance. (Choi and Christensen 2011, 2012c)
- **Coarse 3D pose estimation using an efficient chamfer matching.** An edge-based pose estimation is proposed for textureless objects and integrated with our particle filtering framework. (Choi and Christensen 2012b)
- **Color Point Pair Feature (CPPF) for voting-based pose estimation.** CPPF from RGB-D data was devised and applied in a voting framework. As a large portion of potentially false feature matches is skipped thanks to an additional color similarity measure, our approach turned out to be more robust and more efficient. (Choi and Christensen 2012a, 2014)
- **Parallel implementation of voting framework.** The time-consuming voting process is parallelized on GPU so that pose estimation is done within a second regardless of the degree of clutter. (Choi and Christensen 2014)

- **Pair features based on boundary information.** We propose novel pair features using object boundary information which are more effective for many industrial and real-world objects that are mostly planar. (Choi et al. 2012)
- **Employing rich features from an RGB-D sensor for visual tracking.** Unlike 2D edges or intensity differences on which most previous work has relied, we proposed a robust likelihood function that uses photometric (colors) and geometric (3D points and surface normals) features available from RGB-D channels. (Choi and Christensen 2013)
- **Parallel particle filter accelerated in GPU.** Inspired by the fact that the likelihood evaluation of each particle is independent of the other particles, our particle filtering framework is massively parallelized in GPU so that it can employ several thousands of particles yet show real-time performance. (Choi and Christensen 2013)

1.5 Outline

The thesis is organized as follows. In Chapter 2, an extensive literature survey of visual object perception is presented. In Chapter 3, our particle filtering framework on the $SE(3)$ group combining texture and boundary information is shown, and its extension to textureless objects is delineated in Chapter 4. Pose estimation algorithms based on our voting framework are described in Chapter 5, in which the color point pair feature is proposed for daily objects and the boundary pair features are devised for industrial objects. After showing our effort to real-time object tracking using an RGB-D camera in Chapter 6, we conclude our thesis in Chapter 7.

1.6 Publication Note

Our initial work on the monocular tracking combining edge and keypoint features appeared in Choi and Christensen (2010), and more robust visual tracking using a particle filter on the $SE(3)$ group was presented in Choi and Christensen (2011). This work was then published as a journal paper (Choi and Christensen 2012c). While the previous work only considered the keypoint-based initialization, an edge-based initialization was also addressed in Choi and Christensen (2012b) to take into account textureless objects. Our depth sensor-based pose estimation research (Choi et al. 2012) was initiated from the work in Mitsubishi Electric Research Labs (MERL) wherein I was a research intern. An extension of Choi et al. (2012) to RGB-D appeared in Choi and Christensen (2012a), and the work was further enhanced with GPU parallel computations. Lastly, a real-time RGB-D object tracking accelerated in GPU appeared in Choi and Christensen (2013).

CHAPTER II

RELATED WORK

In this chapter, we survey various efforts from historically seminal work to recent state-of-the-art approaches which are related to the thesis topic, 6-DOF object pose estimation and tracking. In Section 2.1, a set of pioneering efforts of the early computer vision era is revisited. Recent state-of-the-art research based on a monocular camera and a depth sensor is then examined in Section 2.2 and Section 2.3, respectively. We also review some initial efforts to accelerate computation of perception algorithms in Section 2.4.

2.1 Inception of Object Perception

The main theme of object recognition is to model objects and to register the models to a visual sensory input. The early stage, called the blocks world, goes back to the 60s where [Roberts \(1965\)](#) pioneered an object recognition system. The experimental setup was limited in that several bright and geometrically simple objects were placed in a dark uniform background. Even so, the work devised a perspective projection model and an edge detection method followed by a method to do line segmentation. From a polyhedral object representation, it generated various hypotheses for 3D vertices and edges through feature grouping. This early work assumed geometrically simple objects, so it could not solve for complex polygonal shaped or more generally curved objects. After this work, a series of efforts had tried to overcome the limitation by devising a part-based approach for generic curved objects ([Guzman 1971](#)) and introducing the generalized cylinder primitives ([Binford 1971](#), [Agin 1972](#), [Nevatia and Binford 1973, 1977](#), [Brooks 1982](#)). In automotive industries, [Perkins \(1978\)](#) introduced a recognition system for a bin-picking problem which is to estimate poses of 3D industrial parts in a uniform background. A general framework was, though there were some differences between approaches, to select a minimum set of matched features and to generate model hypotheses from the matchings ([Huttenlocher and Ullman 1987](#)). An efficient feature matching scheme, called geometric hashing, with a voting scheme appeared to address partial occlusions ([Lamdan and Wolfson 1988](#)). While most of the efforts considered matching from 3D models to 2D images, [Ikeuchi and Kanade \(1988\)](#) used 3D orientations from depth information along with 3D geometric models. 3D geometric models had prevailed for many reasons. They are viewpoint invariant in that it is possible to take into account geometric changes with respect to viewpoint variations, and hence, it is rather straightforward to predict the shape under perspective or affine projection. The fact that CAD models of a large number of manufactured objects were readily available also contributed to the popularity of the geometric models.

Interesting enough, there was an alternative avenue in which a set of multiple 2D views of a polyhedral object replaced the 3D geometric model. The set of views was often organized in a network structure, so-called *aspect*

graph, in which nodes and edges of the network represent polyhedral faces and adjacent relations of the faces respectively (Underwood and Coates Jr 1975). This aspect graph can be regarded as an ancestor of template matching approaches for generic object recognition and was succeeded by Dickinson et al. (1992) with *geons* primitives (Biederman 1985). One major problem of the aspect graph approach was the high complexity of the graph structure, since it significantly grows as the geometry of the object model gets complex.

A major limitation of purely geometric models is that surfaces on objects should be less textured or bland as performance of most of approaches relying on geometric models is significantly degraded with complex texture on surfaces. In the mid-90s, a totally new direction appeared based on appearance models. Murase and Nayar (1995) introduced appearance manifolds, often called *eigenspace*, which are low dimensional subspaces reduced from a large image set. It is interesting to note that recognition problem was reformulated as finding a nearest neighbor in the eigenspace of an object and coordinates in the eigenspace are corresponding to a pose of the object. Considering appearances of objects further results in local appearance features, such as affine patch features (Rothganger et al. 2006) or scale invariant keypoint descriptor (Lowe 2004), and these local descriptors have widely been used in object recognition and categorization research for the last decade (Pinz 2005). An extensive literature survey in object categorization can be found in Dickinson (2009), and a survey in the early geometric era is also available as a retrospective (Mundy 2006).

2.2 *Monocular-based Approaches*

One of the most popular sensors for robotic manipulation is monocular camera. Since monocular camera is cost effective sensor and allows fast image acquisition, it has been widely adopted for various visual object perception problems.

2.2.1 **Pose Estimation using Keypoints**

For the last decade, stable keypoint descriptors (Lowe 2004, Bay et al. 2008) have led to successful progress on object recognition. As these keypoint descriptors are invariant to changes in illumination and moderate geometric transformations, keypoint correspondences across different images can be reliably determined. For robotic manipulation, a set of 3D coordinates of the keypoints is required as an object model so that a full 6-DOF object pose can be recovered. These keypoint coordinates can be calculated via structure from motion (Collet et al. 2009) or back-projecting 2D keypoints to a 3D CAD model (Choi and Christensen 2012c). One of the major bottlenecks in the keypoint-based pose estimation is finding correspondences between an object model database and an input image sequence. Exact search tends to be an exhaustive search, and hence several efforts tackled the bottleneck by either using randomized tree structures to perform an approximated search (Lepetit and Fua 2006) or exploiting the parallel power of the GPU for a brute-force search (Collet et al. 2011).

2.2.2 Pose Estimation using Edges

Keypoint descriptors are suitable for well-textured objects, but a large number of daily or industrial objects lack texture. For less textured objects, as most of the classic approaches adopted, edge features are preferred since they correspond to the geometric shape of the objects. A common approach is that a set of edge image templates of an object is known *a priori*, and in the testing phase the templates are matched with a given edge image. Please note that this idea goes back to [Underwood and Coates Jr \(1975\)](#). In classic computer vision, the chamfer ([Barrow et al. 1977](#)) and Hausdorff ([Huttenlocher et al. 1993](#)) distances were proposed as robust metrics, and they were further enhanced by considering edge orientations ([Olson and Huttenlocher 1997](#), [Liu et al. 2010a](#)). Common methods to extract edge features from an image are image gradient-based, such as the Canny edge detector ([Canny 1986](#)). However, these methods often result in unnecessary edges from surface texture or non-Lambertian reflectance. Since meaningful boundary edges are often obtained from depth discontinuities, [Raskar et al. \(2004\)](#) introduced the multi-flash camera to detect depth edges by casting shadows from multiple flashes. The sensor was successfully employed in several robotic pose estimation algorithms ([Agrawal et al. 2010](#), [Liu et al. 2010b](#)). While that work has relied on edge feature, [Rodrigues et al. \(2012\)](#) exploited geometric surface shapes of industrial objects. To capture the surface shapes with a single camera, they also employed a multiple-light setting which was inspired by the photometric stereo ([Woodham 1980](#)). A random forest ([Amit and Geman 1997](#), [Breiman 2001](#), [Criminisi and Shotton 2013](#)) was then used to solve the data associations between training and testing local patches. The random forest is learned from a set of training local patches at a fixed distance, and thus the pose estimation performance of the approach often degrades in a different depth.

2.2.3 Pose Tracking with Single Pose Hypothesis

While pose estimation problem is a global search of a 6-DOF transformation from a given model to a scene image, pose tracking problem is a local search estimating inter-frame motions over multiple image frames. Typical scenario is that a tracking starts from a given pose and recursively estimates the 6-DOF pose vector in the sequence of images. Given a 3D CAD model of an object and a prior pose, the 3D model is projected to the current 2D image. Due to a small inter-frame motion, the 3D model and its corresponding region in the 2D image happen to be slightly misaligned, and tracking task is minimizing the misaligned errors. In the seminal work of [Harris \(1992\)](#), edge feature was used to measure perpendicular errors between the projected model and the image, and a least square method was employed for the minimization. Since the work of [Harris \(1992\)](#), there have been active efforts to enhance the early edge-based tracking system at general scale ([Drummond and Cipolla 2002](#), [Comport et al. 2004](#)) and at micro or nano scales ([Yesin and Nelson 2005](#), [Kratochvil et al. 2009](#), [Tamadazte et al. 2010](#)). Edges have been preferred because they are easy to compute and invariant to illumination and pose changes. However, a critical disadvantage of using edges is that they look similar to each other, and, as a result, it is tricky to perform data associations. In general, edge correspondences are determined by local search based on a prior pose estimate. So the tracking performance of an edge-based tracker directly depends on correct pose priors. To improve the pose priors, there have been various efforts to enhance the

pose accuracy by incorporating interest points (Vacchetti et al. 2004, Rosten and Drummond 2005, Pressigout and Marchand 2006) or employing additional sensors (Klein and Drummond 2004). While that work relies on edge data association between projected model edges and image edges, there was an effort based on a region segmentation approach (Bibby and Reid 2008). Prisacariu and Reid (2012) presented a real-time 3D object tracking based on a segmented region-based tracking. This work can be seen as an extension of the 2D free-form visual tracking using pixel-wise posterior (Bibby and Reid 2008) to 3D pose tracking, via optimizing the energy function with respect to 6-DOF pose parameters. For robust tracking, the approach maintains a generative probabilistic color model for both foreground and background, but it often backfires in the cases of occlusions and cluttered backgrounds that violate the generative color distribution. This work maintains a single hypothesis for the 6-DOF pose. When a target object is visually distinctive or its motion is rather simple, visual tracking is robust with these single pose hypothesis tracking approaches. However, these approaches are significantly challenged in the presence of background clutter and object discontinuous cases.

2.2.4 Pose Tracking using Multiple Pose Hypotheses

To overcome the limitation of the single pose hypothesis tracking, there have been active efforts to consider multiple pose hypotheses. Considering multiple edge correspondences was one of the efforts. Since edges are ambiguous and false edge correspondences directly lead the tracker to false pose estimates, some approaches have considered multiple hypotheses on edge correspondences (Vacchetti et al. 2004, Kemp and Drummond 2005). However, their work was still limited in that only one or two hypotheses were maintained from the multiple correspondences.

More general multiple pose hypotheses tracking has been implemented in particle filtering frameworks. Particle filter (Gordon et al. 1993, Doucet et al. 2001) is a Bayesian filtering method based on sequential simulation from posterior probability distributions. For the last two decades, this method has become popular since it can handle nonlinear and non-Gaussian dynamics, so it has often been regarded as a good alternative to the filtering methods designed upon Gaussian probability distributions. Isard and Blake (1998) introduced a 2D contour-based particle filter to computer vision community and showed great potential. Affine 2D visual trackers have also been proposed in a particle filter framework with incremental measurement learning (Ross et al. 2008, Kwon and Park 2010). For 3D visual tracking, Pupilli and Calway (2006) have shown the possibility of applying a particle filter to 3D edge-based tracking. While they demonstrated the tracking of simple 3D objects, Klein and Murray (2006) implemented a particle filtering approach which tracks a complex full 3D object in real-time by exploiting parallel power of GPU. Mörwald et al. (2010) also used GPU to implement a fast model-based 3D visual tracker. With edges from a 3D CAD model, they also employed edges from texture that possibly contribute to avoiding false edge correspondences as well as to enhance the accuracy of pose estimates. Teulière et al. (2010) addressed a similar problem by maintaining multiple hypotheses from low-level edge correspondences.

With a few exceptions (Kyrki and Kragic 2005, Mörwald et al. 2010), most of the work has made an assumption, in which trackers start from a given pose. Several efforts (Klein and Murray 2006, Pupilli and Calway 2006) used

annealed particle filters to find the true pose from scratch without performing an appropriate initialization. However, the search space might be too large to converge to the true pose in reasonable time, and it might not converge to the pose after enough time elapses.

2.3 *Depth-based Approaches*

As **RGB-D** sensors have recently been introduced at low cost, 3D based object perception can be more feasible than ever. The depth data from the sensors preserves 3D information, while 2D images lose 3D information of scenes via the perspective projection onto a 2D image plane. This 3D information enables us to exploit various geometric features, such as 3D surface normals, 3D edges, and so on, which are beyond the photometric features. As 3D models are given *a priori* and the geometric features are available in the models, we can rely on the direct geometric matching between the models and a 3D depth scene.

2.3.1 **Pose Estimation using 3D Point Clusters**

When the background of a given scene is known, we can easily segment foreground objects from the background. Imagine that several objects are placed on a table. If we find a dominant plane on the table by fitting a 3D plane model, separating object segments from the table is a straightforward task. This segmentation scheme significantly reduces the search space of object perception problem, and thus this planar background assumption has been widely adopted in robotic manipulations. [Aldoma et al. \(2011\)](#) devised the clustered viewpoint feature histogram (**CVFH**) which is an extension of the viewpoint feature histogram (**VFH**) ([Rusu et al. 2010](#)) to **6-DOF** pose estimation on segmented object clouds. [Lai et al. \(2011\)](#) proposed a tree structure for scalable object recognition and pose estimation, but the pose estimation is limited in that it can only estimate coarse **1-DOF** rotation of the object pose. Although these approaches can recognize object pose efficiently, they hinge upon perfect segmentation from the background. Thus, these approaches are only applicable to the well structured table-top manipulation, but they are not robust in cluttered environments.

2.3.2 **Pose Estimation using 3D Point Descriptors**

When the prior knowledge about the background structure is unavailable, it is required to match a set of object models directly with a given scene. Similar to 2D local image keypoint descriptors, several 3D point descriptors have been proposed based on the distribution of surface normal around a point ([Stein and Medioni 1992](#)), surface curvature ([Dorai and Jain 1997](#)), spin image ([Johnson and Hebert 1999](#)), relative angles between neighboring normals ([Rusu et al. 2009](#)), and neighboring range values around an interest point ([Steder et al. 2011](#)). While these features are invariant to rigid body transformation, they are in general sensitive to noise, feature parameters, and resolution difference of point clouds. Moreover, stable point descriptions on geometrically simple objects are limited, and hence estimating pose only with these point descriptors might not be a dependable solution.

2.3.3 Pose Estimation using Pair Features

A pair of primitive features is a more general way to describe characteristics of shape of objects since a set of pair features encodes both local and global shape descriptions. In addition, the pair features are inclusive in that they can represent both geometrically salient areas and dull regions. [Drost et al. \(2010\)](#) adopted a pair feature using two points and their normals. In the learning phase, a set of pair features from an object is calculated and saved in a hash table for fast retrieval. In the testing phase, points are randomly sampled from the sensor data, and each pair matched with pairs in the learned model casts a vote for a pose hypothesis. After the voting process, a set of high votes over a certain confidence level is aggregated to form possible object pose hypotheses. This approach was enhanced by incorporating the visibility context ([Kim and Medioni 2011](#)) or considering object boundary information ([Choi et al. 2012](#)). The overall system resembles the classical object perception systems ([Huttenlocher and Ullman 1987](#), [Lamdan and Wolfson 1988](#)) in the sense that a minimum set of local features is used to generate multiple pose hypotheses, and several consistent hypotheses are determined from them. The point pair feature can be seen as a successor of the surflet pairs ([Wahl et al. 2003](#)), and using a hash table for fast matching is also presented in [Mian et al. \(2006\)](#), [Lamdan and Wolfson \(1988\)](#). While the voting scheme is a popular idea in object recognition ([Ballard 1981](#), [Lamdan and Wolfson 1988](#), [Mian et al. 2006](#), [Pham et al. 2011](#), [Woodford et al. 2011](#)), the point pair feature was also used in a **RANSAC** ([Fischler and Bolles 1981](#))-based sampling approach in [Papazov et al. \(2012\)](#).

2.3.4 Pose Estimation using Templates

There are efforts to solve the pose estimation problem via augmented template matching approaches in which depth information is used for better recognition performance. [Park et al. \(2010\)](#) exploited the depth information to detect edges from depth discontinuities and adopted a template matching of the signed distance transformed images. [Hinterstoisser et al. \(2012a\)](#) combined the image gradient from **RGB** channels and the surface normal vectors calculated from the depth channel. The work employs a large number of templates for each object to take into account the variability of shape, and it uses the modern **SSE** instructions for parallel computation in **CPU**. While the initial work learns object templates online, the later work ([Hinterstoisser et al. 2012b](#)) automatically generates templates from 3D mesh models. Although this work can rapidly detect objects from **RGB-D** scenes, it is not free from the limitations of template matching: high false positive rates, low pose accuracy, and not reliable detection outside the learned depth range.

2.3.5 Pose Tracking with Depth Information

Reliable depth information is preferred to find correspondences in visually featureless environments, so it has been actively used in registration and tracking problems. The iterative-closest point (**ICP**) algorithm ([Besl and McKay 1992](#)) is well-known for the registration of 3D point clouds, but it strictly requires a good initial pose estimate so as to converge to the global optimum. Beam-based models ([Thrun et al. 2005](#), [Herbst et al. 2011](#)) were proposed to enhance the **ICP** algorithm by taking into account the viewpoint constraint and the noise model of the sensor.

Recently, these models were further improved by exploiting over-segmentation to take into account the free-space constraint (Krainin et al. 2012).

For environment mapping, Newcombe et al. (2011) presented a real-time mapping and pose tracking system, which fuses a live depth stream into a volumetric scene model called the truncated signed distance function (TSDF) and tracks the camera pose with respect to the scene model using the ICP algorithm. While this work is suitable for small scale mapping, Henry et al. (2010) showed a large 3D modeling approach relying on visual and shape information from the RGB-D camera.

For object tracking, several initial work using point clouds have been proposed using particle filters. Choi et al. (2011) combined multiple cues from both RGB and depth modalities in a particle filter for robust people tracking. A particle filter-based tracking in the Point Cloud Library (PCL) (Rusu and Cousins 2011) can track the 6-DOF pose of a reference point cloud model over a sequence of RGB-D images by the using point and color likelihood functions (Bersch et al. 2012). Pauwels et al. (2013) recently showed a real-time tracking which utilizes dense motion from optical flow and depth information calculated via stereo matching. Though the approach maintains single pose hypothesis, GPU-accelerated computation makes the approach robust with respect to occlusions and tracking.

For the sake of pose tracking over intra-class variations, random forest (Amit and Geman 1997, Breiman 2001, Criminisi and Shotton 2013) is a popular approach due mainly to its efficiency and generalization capability. Shotton et al. (2011) addressed the human pose estimation problem as a body part classification problem where each part is predicted by learned random forests using a simple depth image feature. While this work transformed from the regression to the classification problem, Fanelli et al. (2011) used the random forests for a regression problem so as to directly estimate the pose of objects.

2.4 *Parallelized Object Perception*

Visual perception requires processing of massive visual data. The traditional computing architecture is the single instruction single data (SISD) which restricts perception algorithms to iterating a set of operations over entire data, and hence executing typical perception algorithms takes a significant amount of time. Interestingly, a lot of visual perception algorithms are composed of repetitive yet simple computations. As computing architecture evolves, single instruction multiple data (SIMD) architecture enables these perception algorithms to run in parallel. Hinterstoisser et al. (2012a) is one example of using SIMD instructions to parallelize its computation in CPU. In addition to CPU, modern graphics processing unit (GPU) provides massive parallel computation power beyond rendering purposes, and it is thus a natural idea to design perception algorithms in parallel to take advantage of the computing power of the GPU. Collet et al. (2011) exploited the parallel power of GPU to perform a brute-force keypoint matching, and Prisacariu and Reid (2012) also implemented time-consuming routines in the GPU in which finding the 2D contour of 3D object projection and performing pose optimization were parallelized. Pauwels et al. (2013) also borrowed the GPU power to accelerate optical flow, stereo matching, and pose estimation routines.

Our main computing frameworks, voting and particle filtering, are composed of an amount of repetitive operations, and hence the algorithms are inherently possible to be parallelized. Particle filter (Isard and Blake 1998) is straightforward to be parallelized as the likelihood evaluation of each particle is independent of the other particles. Montemayor et al. (2004) showed a preliminary design of a particle filter on the GPU for a simple 2D object tracking, and some approaches presented GPU accelerated particle filters for 3D object pose tracking, for which they render a 3D object model to the GPU and a CUDA kernel directly accesses the rendering results to calculate importance weights of the particles (Azad et al. 2011, Mateo Lozano and Otsuka 2009). The work of Park et al. (2010) was also parallelized on GPU, where the distance transform and the downhill simplex optimization were accelerated. Salas-Moreno et al. (2013) recently showed a parallel implementation of Drost et al. (2010) for an object-based camera localization and mapping by employing a GPU parallel library, AMD Bolt C++ library (AMD 2012).

CHAPTER III

COMBINING TEXTURE AND BOUNDARY INFORMATION (2D)

In this chapter, we delineate a 3D model-based visual tracking approach using edge and keypoint features in a particle filtering framework. Recently, particle filtering-based approaches have been proposed to address very challenging visual tracking problems and have shown good performance, but most of the work has made an assumption that an initial pose is given. To ameliorate this limitation, we employ keypoint features for initialization of the filter. Given 2D-3D keypoint correspondences, we randomly choose a set of minimum correspondences to calculate a set of possible pose hypotheses. Based on the inlier ratio of correspondences, a set of poses is drawn to initialize particles. After the initialization, edge points are employed to estimate inter-frame motions. While we follow a standard edge-based tracking, an additional refinement process is performed to improve the edge correspondences between sampled model edge points and image edge points. For better tracking performance, we employ a first order autoregressive (AR) state dynamics, which propagates particles more effectively than Gaussian random walk models. The proposed system re-initializes particles by itself when the tracked object moves out of the field of view or is occluded. The robustness and accuracy of our approach are demonstrated via comparative experiments on synthetic and real image sequences.

This chapter is organized as follows. Our main contributions are explained in Section 3.1. In Section 3.2, we define the monocular camera model and explain the automatic salient edge selection process on 3D object models. In Section 3.3.1 and 3.3.2, we introduce a particle filtering framework with state and measurement equations. The AR state dynamics adopted in the framework is then presented in Section 3.3.3. After explaining how particles are initialized and their likelihoods are evaluated in Section 3.3.4 and 3.3.5, respectively, the optimization performed for each particle is explained in Section 3.3.6. Lastly, the re-initialization scheme is presented in Section 3.3.7, followed by experimental results on both synthetic and real image sequences in Section 3.4.

3.1 Contributions

The key contributions of the work in this chapter are as follows:

- We employ keypoint features as additional visual cues. While Klein and Murray (2006), Pupilli and Calway (2006) have used an annealing particle filter to find the initial pose, we initialize particles to highly probable states based on pose estimates calculated from keypoint correspondences so that initialized particles tend to converge faster than the usual annealed particle filtering.
- We refine edge correspondences between the projected model edges and the image edges via a RANSAC

(Fischler and Bolles 1981). Most of the edge-based tracking approaches have used the nearest edge correspondences without performing an appropriate refining process (Harris 1992, Drummond and Cipolla 2002, Comport et al. 2004, Choi and Christensen 2010), except a few work in Armstrong and Zisserman (1995), Teulière et al. (2010). Given that the edge correspondences directly affect the measurement likelihood and thus entire tracking performance, we employ a RANSAC approach to ensure consistent edge data associations. While Armstrong and Zisserman (1995) applied a RANSAC on each 2D line segments individually, we perform that on 3D sampled points and their corresponding 2D closest edge points.

- While most of the existing edge-based trackers (Klein and Murray 2006, Teulière et al. 2010) have employed random walk models as a motion model, we apply a first-order autoregressive (AR) state dynamics on the $SE(3)$ group to guide particles more effectively.
- To be fully automatic and reliable in practical settings, our approach monitors the number of effective particle size and uses the measure to decide when the tracker requires re-initialization.

3.2 Camera and Object Models

In this section, we introduce the monocular camera model and the visible edge determination method from 3D mesh models.

3.2.1 Camera Model

Our system employs a calibrated monocular camera so that we have the intrinsic and lens distortion parameters known *a priori*. Given the known camera parameters, we rectify input images in order to remove the lens distortion effect. Thus, as a camera model we use the standard pin-hole model given by:

$$\mathbf{p} = \text{Project}(\mathbf{K}, \mathbf{X}_t, \mathbf{P}^O) = \mathbf{K} \begin{pmatrix} \frac{x^c}{z^c} \\ \frac{y^c}{z^c} \\ 1 \end{pmatrix} \quad (3.1)$$

$$\mathbf{P}^c = \mathbf{X}_t \mathbf{P}^O \quad (3.2)$$

where $\mathbf{p} = (u, v)^\top$ is the projected 2D image coordinates, $\mathbf{P}^O = (x^O, y^O, z^O, 1)^\top$ and $\mathbf{P}^c = (x^c, y^c, z^c, 1)^\top$ are the 3D homogeneous coordinates of a point in object and camera coordinate systems, respectively, and $\mathbf{X}_t \in SE(3)$ is the pose of the camera at time t . The matrix \mathbf{K} represents the intrinsic parameters of the camera:

$$\mathbf{K} = \begin{pmatrix} f_u & 0 & u_0 \\ 0 & f_v & v_0 \end{pmatrix} \quad (3.3)$$

where f_u and f_v are the focal length in pixel dimensions, and u_0 and v_0 represent the position of the principal point.

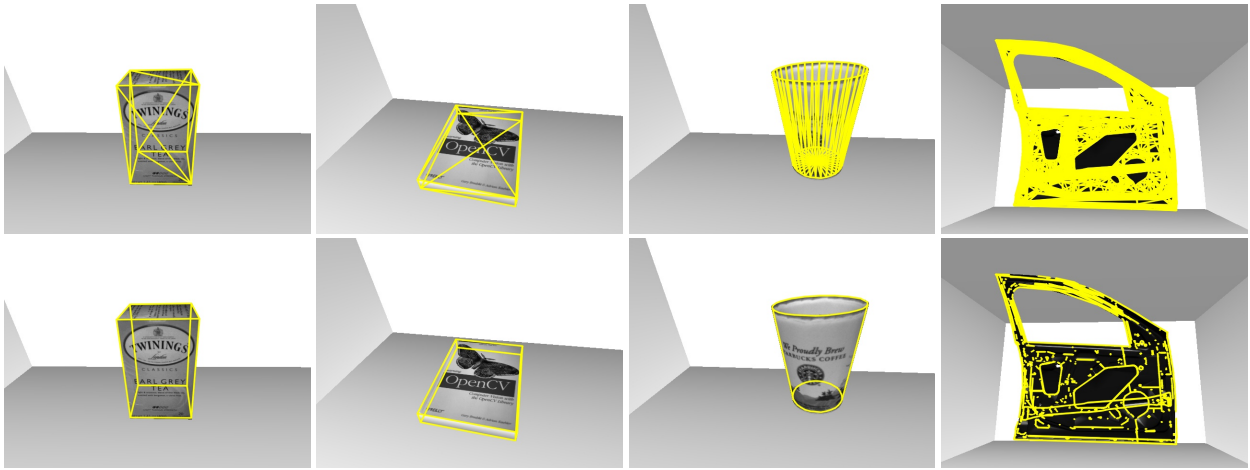


Figure 3.1. Original CAD models (upper row) and selected salient edges (lower row) of our target objects. Our approach automatically selects sharp edges that are likely to be visible in real images. From left to right, “Teabox”, “Book”, “Cup”, and “Car door”.

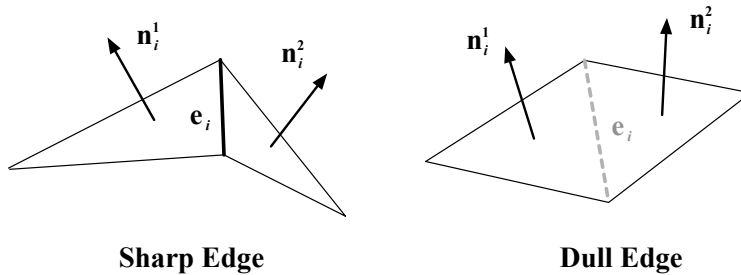


Figure 3.2. Determining salient edges. We use the face normal vectors available in the model.

3.2.2 3D Object Model and Salient Edge Selection

Most objects existing in our environments are manufactured, so their CAD models are likely to be available or at least obtainable via either manual 3D modeling or automatic 3D scanning (Newcombe et al. 2011). Although there are various formats for CAD models, most of them can be represented as a polygon mesh. A polygon mesh is usually composed of *vertices*, *edges*, *faces*, *polygons* and *surfaces*. To estimate the pose difference between two consecutive frames, we employ edge features in images coming from a monocular camera. So we should determine which *edges* in the CAD model of a targeted object would be visible in the images. Here we make an assumption that sharp edges in the model are more likely to be salient in images. To identify sharp edges, we use the face normal vectors from the model. As illustrated in Figure 3.2, if the face normal vectors of two adjacent faces are close to perpendicular, the edge shared by the two faces is regarded as a sharp edge. Similarly, if two face normal vectors are close to parallel, the edge is regarded as a dull edge. For the decision, we employ a simple thresholding scheme using the value of the inner product of two normal vectors. More formally, we can define an indicator function with respect to the edges in the model by:

$$I(\mathbf{e}_i) = \begin{cases} 1 & \text{if } \mathbf{n}_i^1 \top \mathbf{n}_i^2 \leq \tau_s \\ 0 & \text{otherwise} \end{cases} \quad (3.4)$$

where \mathbf{n}_i^1 and \mathbf{n}_i^2 are the face normal unit vectors of the two adjacent faces which share the i^{th} edge, \mathbf{e}_i . We found the threshold $\tau_s = 0.3$ is a reasonable value in most cases. This salient edge selection is performed offline, and the selected edges are saved for online tracking. The original 3D CAD model edges and the selected sharp edges are displayed in Figure 3.1. In general, the salient edges are only considered in edge-based tracking, but when dull edges constitute the object’s boundary, such as for the “Cup” object, they are also considered. To determine these boundary edges, we find edges shared by a front face and a back face. Testing front or back faces is done by calculating inner products of the face normal vectors and the z-axis of the camera. Testing boundary of the dull edges is performed at run-time, and it is thus desirable to avoid using dull edges when dull boundary edges are not the dominant edges in the target object.

3.3 Particle Filter on the $SE(3)$ Group

In 3D visual tracking, a state represents a 6-DOF pose of a tracked object, and tracking estimates time-varying change of coordinates. It is well known that the trajectory is not on a general vector space, rather it is on Lie groups, in general, the Special Euclidean group $SE(3)$ and the affine group $Aff(2)$ in 3D and 2D visual tracking, respectively. Since the trajectory we want to estimate is on a Lie group, the particle filter should be applied on the Lie group. Monte Carlo filtering on Lie groups is explicitly addressed in (Chiuso and Soatto 2000, Kwon et al. 2007, Kwon and Park 2010). It is well known that if a local coordinate system, an ad-hoc representation of motions (e.g. Euclidean embedding), is employed, the same perturbation on different states often results in different motions. Thus filtering performance and noise distribution of local coordinate-based particle filtering approaches are dependent on the choice of the local coordinates, while particle filtering on Lie groups is coordinate-invariant. This coordinate-invariance issue is well addressed in Kwon et al. (2007) and Kwon and Park (2010).

3.3.1 State and Measurement Equations

From the continuous general state equations on the $SE(3)$ group, discrete system equations are acquired via the first-order exponential Euler discretization (Kwon et al. 2007):

$$\begin{aligned} \mathbf{X}_t &= \mathbf{X}_{t-1} \cdot \exp(A(\mathbf{X}, t)\Delta t + \mathbf{dW}_t\sqrt{\Delta t}), \\ \mathbf{dW}_t &= \sum_{i=1}^6 \epsilon_{t,i} \mathbf{E}_i, \\ \epsilon_t &= (\epsilon_{t,1}, \dots, \epsilon_{t,6})^\top \sim \mathcal{N}(\mathbf{0}_{6 \times 1}, \Sigma_w) \end{aligned} \quad (3.5)$$

where $\mathbf{X}_t \in SE(3)$ is the state at time t , $A : SE(3) \mapsto \mathfrak{se}(3)$ is a possibly nonlinear map, \mathbf{dW}_t represents the Wiener process noise on $\mathfrak{se}(3)$ with a covariance $\Sigma_w \in \mathbb{R}^{6 \times 6}$, and \mathbf{E}_i are the i^{th} basis elements of $\mathfrak{se}(3)$:

$$\begin{aligned} \mathbf{E}_1 &= \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \mathbf{E}_2 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \mathbf{E}_3 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \\ \mathbf{E}_4 &= \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \mathbf{E}_5 = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \mathbf{E}_6 = \begin{pmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}. \end{aligned} \quad (3.6)$$

Algorithm 3.1: Particle Filtering on the $SE(3)$ group

Data: $\mathcal{I} = \{\mathcal{I}_0, \mathcal{I}_1, \dots, \mathcal{I}_I\}, \mathcal{F} = \{(\mathbf{p}_1, \mathbf{P}_1), \dots, (\mathbf{p}_F, \mathbf{P}_F)\}$

Result: $\mathcal{S} = \{\mathcal{S}_0, \mathcal{S}_1, \dots, \mathcal{S}_I\}$

Params: $N, \Sigma_w, \lambda_a, \lambda_v, \lambda_r, N_{thres}$

```
1:  $t \leftarrow 0$ 
2:  $init \leftarrow 1$ 
3:  $\mathbf{A}_0 \leftarrow \mathbf{0}_{4 \times 4}$ 
4: while  $\mathcal{I}_t \neq 0$  do
5:   if  $init = 1$  then
6:      $\mathcal{S}_t \leftarrow \text{InitParticle}(\mathcal{I}_t, \mathcal{F})$  (3.2)
7:     if  $\mathcal{S}_t \neq \{\phi\}$  then
8:        $init \leftarrow 0$ 
9:   else
10:    for  $n \leftarrow 1$  to  $N$  do
11:       $\mathbf{X}_t^{*(n)} \leftarrow \text{Propagate}(\mathbf{X}_{t-1}^{(n)}, \mathbf{A}_{t-1}^{(n)}, \Sigma_w)$  (3.13)
12:       $\mathbf{A}_t^{*(n)} \leftarrow \text{AR\_vel}(\mathbf{X}_t^{*(n)}, \mathbf{X}_{t-1}^{*(n)}, \lambda_a)$  (3.14)
13:       $\mathbf{Z}_t^{*(n)} \leftarrow \text{Measurement}(\mathbf{X}_t^{*(n)}, \mathcal{I}_t)$  (3.7)
14:       $\widehat{\mathbf{Z}}_t^{*(n)} \leftarrow \text{RANSAC}(\mathbf{Z}_t^{*(n)})$  (3.3)
15:       $\tilde{\pi}_t^{*(n)} \leftarrow \text{Likelihood}(\mathbf{Z}_t^{*(n)}, \widehat{\mathbf{Z}}_t^{*(n)}, \lambda_v, \lambda_r)$  (3.21)
16:       $\hat{\mathbf{X}}_t^{*(n)} \leftarrow \text{IRLS}(\mathbf{X}_t^{*(n)}, \widehat{\mathbf{Z}}_t^{*(n)})$  (3.22)(3.23)
17:       $\tilde{\pi}_t^* \leftarrow \text{Normalize}(\tilde{\pi}_t^{*(n)})$  (3.16)
18:       $\widehat{N}_{eff} \leftarrow \text{Neff}(\tilde{\pi}_t^*)$  (3.25)
19:      if  $\widehat{N}_{eff} \geq N_{thres}$  then
20:         $\mathcal{S}_t \leftarrow \text{Resampling}(\mathcal{S}_t^*)$ 
21:      else
22:         $init \leftarrow 1$ 
23:     $t \leftarrow t + 1$ 
```

Note that the stochastic state equation in (3.5) is equivalent to a convolution of probability densities (Park et al. 2008, Wang and Chirikjian 2006), except that the latter does not assume Gaussian noise.

The corresponding measurement equation is then:

$$\mathbf{Z}_t = g(\mathbf{X}_t) + \mathbf{n}_t, \mathbf{n}_t \sim \mathcal{N}(\mathbf{0}_{N_z \times 1}, \Sigma_n) \quad (3.7)$$

where $g : SE(3) \mapsto \mathbb{R}^{N_z}$ is a nonlinear measurement function and \mathbf{n}_t is a Gaussian noise with a covariance $\Sigma_n \in \mathbb{R}^{N_z \times N_z}$.

3.3.2 Particle Filter

In a generic particle filtering framework, the posterior density function $p(\mathbf{X}_t | \mathbf{Z}_{1:t})$ is represented as a set of weighted particles by

$$\mathcal{S}_t = \{(\mathbf{X}_t^{(1)}, \pi_t^{(1)}), \dots, (\mathbf{X}_t^{(N)}, \pi_t^{(N)})\} \quad (3.8)$$

where the particles $\mathbf{X}_t^{(n)} \in SE(3)$ represent samples of the current state \mathbf{X}_t , the normalized weights $\pi_t^{(n)}$ are proportional to the likelihood function $p(\mathbf{Z}_t | \mathbf{X}_t^{(n)})$, and N is the number of particles. The current state \mathbf{X}_t can be estimated

Algorithm 3.2: InitParticle(\mathcal{I}, \mathcal{F})

Data: $\mathcal{I}, \mathcal{F} = \{(\mathbf{p}_1, \mathbf{P}_1), \dots, (\mathbf{p}_F, \mathbf{P}_F)\}$
Result: $\mathcal{S} = \{(\mathbf{X}^{(1)}, \pi^{(1)}), \dots, (\mathbf{X}^{(N)}, \pi^{(N)})\}$
Params: $\tau_r, \tau_n, N, m, \mathbf{K}, \tau_\epsilon, \lambda_c$

- 1: $\hat{l} \leftarrow 0$
- 2: $\hat{\mathcal{C}} \leftarrow \{\phi\}$
- 3: $\mathbf{p}_i \leftarrow \text{ExtractSURF}(\mathcal{I})$ (Bay et al. 2008)
- 4: **for** $f \leftarrow 1$ **to** F **do**
- 5: $\{\hat{\mathbf{p}}_i, \hat{\mathbf{P}}_f\} \leftarrow \text{BBF}(\mathbf{p}_i, \mathbf{p}_f, \mathbf{P}_f, \tau_r)$ (Beis and Lowe 1997)
- 6: $\mathcal{C} \leftarrow \{\hat{\mathbf{p}}_i, \hat{\mathbf{P}}_f\}$
- 7: $\hat{\mathcal{C}} \leftarrow \text{RatioTest}(\mathcal{C}, \tau_r)$ (Lowe 2004)
- 8: $l \leftarrow \text{length}(\hat{\mathcal{C}})$
- 9: **if** $l > \hat{l}$ **then**
- 10: $\hat{l} \leftarrow l$
- 11: $\hat{\mathcal{C}} \leftarrow \mathcal{C}$
- 12: **if** $\hat{l} > \tau_n$ **then**
- 13: $N_p \leftarrow \hat{l}$
- 14: **for** $n \leftarrow 1$ **to** N **do**
- 15: $\tilde{\mathcal{C}} \leftarrow \text{RandomSample}(\hat{\mathcal{C}}, m)$
- 16: $\mathbf{X}^{*(n)} \leftarrow \text{EPnP}(\mathbf{K}, \tilde{\mathbf{p}}_i, \tilde{\mathbf{P}}_f)$ (Lepetit et al. 2009)
- 17: $\hat{\mathbf{p}} \leftarrow \text{Project}(\mathbf{K}, \mathbf{X}^{*(n)}, \hat{\mathbf{P}}_f)$ (3.1)(3.2)
- 18: $\mathcal{H} \leftarrow \{\phi\}$
- 19: **for** $s \leftarrow 1$ **to** N_p **do**
- 20: $\epsilon^{(s)} \leftarrow \|\hat{\mathbf{p}}_i^{(s)} - \hat{\mathbf{p}}^{(s)}\|_2$
- 21: **if** $\epsilon^{(s)} < \tau_\epsilon$ **then**
- 22: $\mathcal{H} \leftarrow \mathcal{H} \cup \{s\}$
- 23: $N_i \leftarrow \text{length}(\mathcal{H})$
- 24: $\tilde{\pi}^{*(n)} \leftarrow \text{KeypointLikelihood}(N_p, N_i, \lambda_c)$ (3.15)
- 25: $\pi^* \leftarrow \text{Normalize}(\tilde{\pi}^*)$ (3.16)
- 26: $\mathcal{S} \leftarrow \text{Resampling}(\mathcal{S}^*)$
- 27: **else**
- 28: $\mathcal{S} \leftarrow \{\phi\}$

by the weighted particle mean:

$$\mathbf{X}_t = \mathcal{E}[\mathcal{S}_t] = \sum_{n=1}^N \pi_t^{(n)} \mathbf{X}_t^{(n)} \quad (3.9)$$

or the mode:

$$\mathbf{X}_t = \mathcal{M}[\mathcal{S}_t] = \mathbf{X}_t^{(j)}, \quad j = \arg \max_j \pi_t^{(j)}. \quad (3.10)$$

When we apply the mean, however, there is a problem where the arithmetic average of $\mathbf{X}_t^{(n)}$ is not valid in the $SE(3)$. More specifically, let $\mathbf{R}_t^{(n)} \in SO(3)$ be the rotation part of the $\mathbf{X}_t^{(n)}$. Then the arithmetic mean $\bar{\mathbf{R}}_t = \frac{1}{N} \sum_{n=1}^N \mathbf{R}_t^{(n)}$ is not usually on the $SO(3)$ group. As an alternative, Moakher (2003) showed that a valid average of a set of rotations can be calculated by the orthogonal projection of $\bar{\mathbf{R}}_t$ onto $SO(3)$ as

$$\mathbf{R}_t = \begin{cases} \mathbf{V}\mathbf{U}^\top & \text{when } \det(\bar{\mathbf{R}}_t^\top) > 0 \\ \mathbf{V}\mathbf{H}\mathbf{U}^\top & \text{otherwise,} \end{cases} \quad (3.11)$$

where \mathbf{U} and \mathbf{V} are estimated via the singular value decomposition of $\overline{\mathbf{R}}_t^\top$ (i.e. $\overline{\mathbf{R}}_t^\top = \mathbf{U}\Sigma\mathbf{V}^\top$) and $\mathbf{H} = \text{diag}[1, 1, -1]$. Therefore, the valid arithmetic mean of the particles can be determined as

$$\mathbf{X}_t = \mathcal{E}_{SE(3)}[\mathcal{S}_t] = \begin{pmatrix} \mathbf{R}_t & \mathbf{T}_t \\ \mathbf{0}_{1 \times 3} & 1 \end{pmatrix} \quad (3.12)$$

where $\mathbf{T}_t = \frac{1}{N} \sum_{n=1}^N \mathbf{T}_t^{(n)}$ and $\mathbf{T}_t^{(n)} \in \mathbb{R}^3$ is the translation part of $\mathbf{X}_t^{(n)}$. A more detailed discussion of mean and covariance on the $SE(3)$ group is presented in Wang and Chirikjian (2008).

3.3.3 AR State Dynamics

The dynamics model for the state evolution is an essential part in that it has a significant impact on tracking performance. However, many particle filter-based trackers have been based on a random walk model because of its simplicity (Klein and Murray 2006, Teulière et al. 2010). The first-order autoregressive (AR) state dynamics is a good alternative since it is flexible, yet simple to implement. In the state equation (3.5), the term $A(\mathbf{X}, t)$ determines the state dynamics. A trivial case, $A(\mathbf{X}, t) = 0$, is a random walk model. Kwon and Park (2010) modeled this via the first-order AR process on the $Aff(2)$ as:

$$\mathbf{X}_t = \mathbf{X}_{t-1} \cdot \exp(\mathbf{A}_{t-1} + \mathbf{d}\mathbf{W}_t \sqrt{\Delta t}), \quad (3.13)$$

$$\mathbf{A}_{t-1} = \lambda_a \log(\mathbf{X}_{t-2}^{-1} \mathbf{X}_{t-1}) \quad (3.14)$$

where λ_a is the AR process parameter. Since the $SE(3)$ is a compact connected Lie group, the AR process model also holds on the $SE(3)$ group (Xavier and Manton 2006).

3.3.4 Particle Initialization using Keypoint Correspondences

Most of the particle filter-based trackers assume that an initial state is given. In practice, however, the initial setting of particles is crucial to ensure convergence to the true state. Several trackers (Klein and Murray 2006, Pupilli and Calway 2006) search for the true state from scratch, but it is desirable to initialize particle states by using other information. Keypoint features are good for wide baseline matching between images, and hence keypoint correspondences are well suited for filter initialization.

For initialization, we employ so-called keyframes $\mathcal{F} = \{(\mathbf{p}_1, \mathbf{P}_1), \dots, (\mathbf{p}_F, \mathbf{P}_F)\}$ which are composed of F sets of SURF keypoints (Bay et al. 2008) coordinates in 2D and 3D. The appearance of SURF keypoints is moderately variant with respect to viewpoint variations. As a result, it requires to capture multiple keyframes covering different views of an object. The keypoint coordinates in 2D (\mathbf{p}_f) are easily determined from SURF keypoints extraction, while 3D coordinates (\mathbf{P}_f) are determined by back-projecting 2D points onto surfaces of the tracked object. For the back-projection, we need to know the exact pose of the tracked object. It is possible to get the pose using a calibrated turn table, but we used our particle filter-based tracking started from a known pose. The keyframes are saved manually by user input during tracking. At runtime, an input image \mathcal{I} coming from a monocular camera is matched with the

Algorithm 3.3: RANSAC(\mathbf{X}, \mathbf{Z})

Data: $\mathbf{X}, \mathbf{Z} = \{\mathbf{p}, \mathbf{P}\}$
Result: $\hat{\mathbf{Z}} = \{\hat{\mathbf{p}}, \hat{\mathbf{P}}\}$
Params: $i_{max}, m, \mathbf{K}, \tau_\epsilon, \rho$

- 1: $i \leftarrow 0$
- 2: $\hat{n} \leftarrow 0$
- 3: $\kappa \leftarrow \infty$
- 4: $\hat{\mathcal{H}} \leftarrow \mathcal{H} \leftarrow \{\phi\}$
- 5: $S \leftarrow \text{length}(\mathbf{p})$
- 6: **while** $i < \kappa$ **and** $i < i_{max}$ **do**
- 7: $\tilde{\mathbf{Z}} \leftarrow \text{RandomSample}(\mathbf{Z}, m)$
- 8: $\tilde{\mathbf{X}} \leftarrow \text{IRLS}(\mathbf{X}, \tilde{\mathbf{Z}})$ (3.22)(3.23)
- 9: $\hat{\mathbf{p}} \leftarrow \text{Project}(\mathbf{K}, \tilde{\mathbf{X}}, \mathbf{P})$ (3.1)(3.2)
- 10: $\mathcal{H} \leftarrow \{\phi\}$
- 11: **for** $s \leftarrow 1$ **to** S **do**
- 12: $\epsilon^{(s)} \leftarrow \|\mathbf{p}^{(s)} - \hat{\mathbf{p}}^{(s)}\|_2$
- 13: **if** $\epsilon^{(s)} < \tau_\epsilon$ **then**
- 14: $\mathcal{H} \leftarrow \mathcal{H} \cup \{s\}$
- 15: $n \leftarrow \text{length}(\mathcal{H})$
- 16: **if** $n > \hat{n}$ **then**
- 17: $\hat{n} \leftarrow n$
- 18: $\hat{\mathcal{H}} \leftarrow \mathcal{H}$
- 19: $\kappa \leftarrow \log(1 - \rho) / \log(1 - (\hat{n}/S)^m)$
- 20: $i \leftarrow i + 1$
- 21: $\hat{\mathbf{Z}} \leftarrow \mathbf{Z}(\hat{\mathcal{H}})$

saved keyframes by extracting keypoints \mathbf{p}_i from \mathcal{I} and comparing them with \mathcal{F} . To find keypoint correspondences $\mathcal{C} = \{\hat{\mathbf{p}}_i, \hat{\mathbf{P}}_f\}$ efficiently, we employ the Best-Bin-First (BBF) algorithm using a kd-tree data structure (Beis and Lowe 1997) that allows execution of the search in $O(n \log n)$. As described in Lowe (2004), the ratio test is then performed to find distinctive feature matches $\hat{\mathcal{C}}$ with the threshold $\tau_r = 0.7$.

While we employed RANSAC (Fischler and Bolles 1981) after determining putative correspondences in our previous work (Choi and Christensen 2010), we skip this procedure because in the particle filter framework we can initialize particles in an alternative way. The basic idea is similar to RANSAC, but it considers multiple pose hypotheses. Instead of explicitly performing RANSAC, we randomly select a set of correspondences $\tilde{\mathcal{C}}$ from the given putative correspondences $\hat{\mathcal{C}}$ having a maximum number of correspondences over keyframes \mathcal{F} and estimate a possible set of poses $\mathbf{X}^{*(n)}$ from $\tilde{\mathcal{C}}$. Since we maintain 3D coordinates \mathbf{P}_f of keypoints in keyframes, we can get 2D-3D correspondences from the matching process. So we can regard this problem as the Perspective- n -Point (PnP) problem, in which the pose of a calibrated monocular camera is estimated from n 2D-3D point correspondences. To find a pose from the correspondences, we use the Efficient Perspective- n -Point (EPnP) algorithm (Lepetit et al. 2009) that provides a $O(n)$ time non-iterative solution for the PnP problem.

After a particle pose $\mathbf{X}^{*(n)}$ is initialized from randomly selected minimum correspondences, all 3D points $\hat{\mathbf{P}}_f$ from the putative correspondences $\hat{\mathcal{C}}$ are projected into 2D points $\hat{\mathbf{p}}$ and compared with 2D keypoints $\hat{\mathbf{p}}_i$. We then

count the number of inlier correspondences N_i whose Euclidean distances between $\hat{\mathbf{p}}$ and $\hat{\mathbf{p}}_i$ is within the threshold $\tau_\epsilon = 20.0$. For $n = 1, \dots, N$ where N is the number of particles, the weights of particles are assigned by the ratio of the number of putative correspondences N_p , which is the number of entries in $\hat{\mathcal{C}}$, and the number of inlier correspondences N_i as:

$$\tilde{\pi}^{*(n)} \propto p(\mathbf{Z}_t | \mathbf{X}_t^{*(n)}) \propto \exp\left(-\lambda_c \frac{N_p - N_i}{N_p}\right) \quad (3.15)$$

where λ_c is the parameter that controls the sensitivity of the ratio of $N_p - N_i$ to N_p to the likelihood. Then the weights $\tilde{\pi}^{*(n)}$ are normalized by:

$$\pi^{*(n)} = \frac{\tilde{\pi}^{*(n)}}{\sum_{i=1}^N \tilde{\pi}^{*(i)}}. \quad (3.16)$$

After normalization, particles are randomly drawn with probabilities proportional to these weights. Since weights are proportional to N_i , pose hypotheses having more inliers are likely to survive in the random sampling. This initialization process is similar to **RANSAC**, but it maintains multiple hypotheses of the initial poses while the **RANSAC** only finds the best hypothesis.

The initialization process is presented in Algorithm 3.2 where referred work and equations are cited in the comments area. Given an input image \mathcal{I} and keyframes \mathcal{F} , the algorithm returns the set of initialized particle states \mathcal{S} . Among the parameters, τ_n is the minimum required number of keypoint correspondences for the initialization, m is the minimum number of correspondences to perform **EPnP**, and \mathbf{K} is the intrinsic parameters of the camera in (3.3). We found that 7 and 9 are well suited to m and τ_n , respectively.

3.3.5 Edge-based Measurement Likelihood

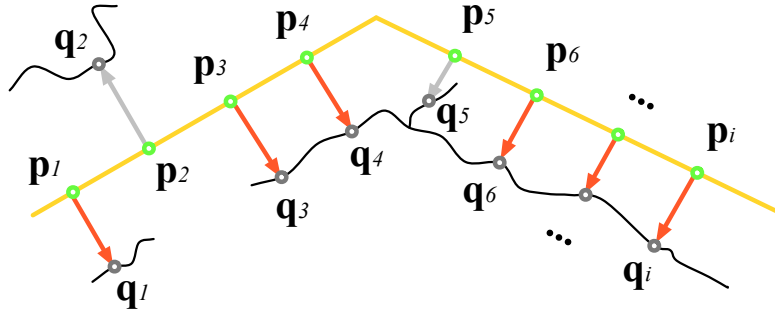


Figure 3.3. Residual determination for calculating the likelihood. Residual errors between projected model (yellow lines) and extracted image edges (black tortuous lines) are calculated. Sample points \mathbf{p}_i are generated along the model per a fixed distance and are matched to image edge points \mathbf{q}_i by performing 1D search along the direction orthogonal to the model edge.

Once each particle is initialized and propagated according to the **AR** state dynamics and Gaussian noise, it has to be evaluated based on its measurement likelihood. In general edge-based tracking, a 3D wireframe model is projected into a 2D image according to a pose hypothesis $\mathbf{X}_t^{*(n)}$. Then a set of points is sampled along edges in the wireframe model per a fixed distance. As some of sampled points are occluded by the object itself, a visibility test is necessary.

While Drummond and Cipolla (2002) used a Binary Space Partitioning (BSP) tree for hidden line removal, OpenGL occlusion query is an easy and efficient alternative in which the sampled points are tested for occlusion (Klein and Murray 2006). The visible sampled points are then matched to edge points, which are obtained by using the Canny edge detector (Canny 1986), from the input image by performing 1D perpendicular search (Drummond and Cipolla 2002, Choi and Christensen 2010). In the matching, most approaches have tried to match the sampled points to closest edge points without examining their orientation characteristics. However, it is well known that using edge orientation significantly enhances the quality of edge correspondences (Olson and Huttenlocher 1997, Liu et al. 2010a). Especially when the object contains relatively complex textures on its surfaces or it is located in cluttered environments, erroneous edge correspondences are often obtained from textures or background edges. These false correspondences result in a bad state hypothesis, and thus it is natural to exclude the edge correspondences having significant differences in orientation. This can be achieved by defining an indication function as

$$I(\mathbf{p}_i, \mathbf{q}_i) = \begin{cases} 1 & \text{if } |\theta_m(\mathbf{p}_i) - \theta_e(\mathbf{q}_i)| \leq \tau_\theta \\ 0 & \text{otherwise} \end{cases} \quad (3.17)$$

where θ_m and θ_e return the orientation of the model edge to which the sample point \mathbf{p}_i belongs and of the image edge point \mathbf{q}_i , respectively. Note that the computation burden is not significant since orientations of sampled points on the model edge and their corresponding edge points are only required to be determined.

After the orientation testing, the residual r_i which is the Euclidean distance between \mathbf{p}_i and \mathbf{q}_i is calculated. By stacking all of the residuals of visible sample points, the residual vector $\mathbf{r} \in \mathbb{R}^{N_z}$ is obtained as follows

$$\mathbf{r} = (r_1, r_2, \dots, r_{N_z})^\top \quad (3.18)$$

where N_z is the number of valid sample points (*i.e.* visible sample points correspond to the image edges). Along the residual \mathbf{r} , the unit normal vectors $\mathbf{n}_i \in \mathbb{R}^2$ of the residual are also saved to be used in the optimization explained in the next section:

$$\{\mathbf{n}_1, \mathbf{n}_2, \dots, \mathbf{n}_{N_z}\}. \quad (3.19)$$

Once we have edge correspondences, we refine them using RANSAC (Fischler and Bolles 1981). Although we discard matches having significant orientation differences, false matches may still remain. Some efforts tried to enhance these matches through maintaining multiple low-level edge clusters (Teulière et al. 2010) or applying a RANSAC on each 2D line segments (Armstrong and Zisserman 1995). One drawback of both work is the possibility of inconsistent refinement because edge or line segments are individually corrected. Another drawback is that their methods can be applied only to line segments and if the model of an object is composed of an amount of small line segments the effect of correction would be negligible or wrong. For consistent refinement of the edge correspondences, we perform a RANSAC on 3D sampled points \mathbf{P} and their corresponding 2D closest edge points

\mathbf{p} . Our approach consistently discards outliers by estimating the best 3D pose containing large number of inliers $\hat{\mathcal{H}}$. The **RANSAC** algorithm is presented in Algorithm 3.3 which finds the refined edge correspondences $\hat{\mathbf{Z}} = \{\hat{\mathbf{p}}, \hat{\mathbf{P}}\}$ given the current pose hypothesis \mathbf{X} and the original edge correspondences $\mathbf{Z} = \{\mathbf{p}, \mathbf{P}\}$. Among the parameters, i_{max} is the maximum number of iterations, again m is the minimum number of correspondences required for **EPnP**, and ρ is the probability in which at least one set of randomly sampled m correspondences is from inliers. The ρ , typically set to 0.99, is used to estimate the required number of iterations κ which is adaptively adjusted in the iteration.

Figure 3.3 illustrates the residual calculation in which each residual arrow represents $r_i \mathbf{n}_i$. Note that the second edge correspondence of \mathbf{p}_2 and \mathbf{q}_2 is wrong since \mathbf{q}_2 comes from background clutter, but it is soon discarded via the **RANSAC** refinement. The fifth edge correspondence is also erroneous because \mathbf{q}_5 does not belongs to edges from the object, but it is excluded via the orientation comparison.

After the edge correspondences are refined, the measurement likelihood can be calculated from the ratio between the number of matched sample points $N_{\hat{\mathbf{z}}}$ which are left after the **RANSAC** and the number of visible sample points N_z which pass a self-occlusion test as:

$$p(\mathbf{Z}_t | \mathbf{X}_t) \propto \exp\left(-\lambda_v \frac{(N_z - N_{\hat{\mathbf{z}}})}{N_z}\right) \quad (3.20)$$

where λ_v is the parameter that controls the sensitivity of the ratio of $N_z - N_{\hat{\mathbf{z}}}$ to N_z to the likelihood. This likelihood has been similarly used in Klein and Murray (2006). Another choice is employing $\bar{\mathbf{r}}$ which is an arithmetic average of the residual \mathbf{r} (Teulière et al. 2010):

$$p(\mathbf{Z}_t | \mathbf{X}_t) \propto \exp(-\lambda_r \bar{\mathbf{r}})$$

where λ_r is the parameter that controls the sensitivity of the arithmetic mean of the residual vector $\bar{\mathbf{r}}$ to the likelihood. We noticed that both likelihoods are valid, and we empirically found that using both terms shows better results. Therefore, in our approach the measurement likelihood is evaluated as:

$$p(\mathbf{Z}_t | \mathbf{X}_t) \propto \exp\left(-\lambda_v \frac{(N_z - N_{\hat{\mathbf{z}}})}{N_z}\right) \exp(-\lambda_r \bar{\mathbf{r}}). \quad (3.21)$$

3.3.6 Optimization using **IRLS**

One of the challenges in particle filtering for 3D visual tracking is the large state space, and hence a large number of particles is usually required for reliable tracking performance. To reduce the number of particles, Klein and Murray (2006) have used an annealed particle filter, while Bray et al. (2004) and Teulière et al. (2010) have selectively employed local optimizations in a subset of particles. For more accurate results, we optimize states of particles as well, for which Iteratively Reweighted Least Squares (**IRLS**) is employed (Drummond and Cipolla 2002, Choi and Christensen 2010). From **IRLS**, the optimized particle $\hat{\mathbf{X}}_t^{*(n)}$ is calculated as follows:

$$\hat{\mathbf{X}}_t^{*(n)} = \mathbf{X}_t^{*(n)} \cdot \exp\left(\sum_{i=1}^6 \hat{\mu}_i \mathbf{E}_i\right) \quad (3.22)$$

$$\hat{\boldsymbol{\mu}} = (\mathbf{J}^\top \mathbf{W} \mathbf{J})^{-1} \mathbf{J}^\top \mathbf{W} \hat{\mathbf{r}} \quad (3.23)$$

where $\hat{\boldsymbol{\mu}} \in \mathbb{R}^6$ is the motion velocity that minimizes the residual vector $\hat{\mathbf{r}} \in \mathbb{R}^{N_z}$, $\mathbf{J} \in \mathbb{R}^{N_z \times 6}$ is a Jacobian matrix of $\mathbf{n}_i^T \mathbf{p}_i$ with respect to $\boldsymbol{\mu}$ obtained by computing partial derivatives at the current pose, and $\mathbf{W} \in \mathbb{R}^{N_z \times N_z}$ is a weighted diagonal matrix. The derivation of the Jacobian matrix \mathbf{J} and more detailed information about IRLS are explained in the Appendix.

Note that the measurement likelihood in (3.21) is calculated before the IRLS optimization. To assign weights of particles, we have to evaluate the likelihood again with the optimized state $\hat{\mathbf{X}}_t^{*(n)}$. However, computing the likelihood again is computationally expensive because it requires the self-occlusion test and 1D perpendicular search for each particle. As an alternative, we can note that IRLS is a local optimization, and hence it highly depends on the previous state. Thus particles having higher likelihood tend to exhibit still higher likelihood after IRLS. But lower likelihood particles are likely to get stuck in local minima due to erroneous or insufficient edge correspondences. Therefore, we can approximate $p(\mathbf{Z}_t | \hat{\mathbf{X}}_t^{*(n)})$ as:

$$\tilde{\pi}_t^{*(n)} \propto p(\mathbf{Z}_t | \hat{\mathbf{X}}_t^{*(n)}) \approx p(\mathbf{Z}_t | \mathbf{X}_t^{*(n)}) \quad (3.24)$$

While this approximation leads to slight discrepancy, it is an efficient alternative especially when computation time does matter. After assigning weights of the particles, the weight $\tilde{\pi}_t^{*(n)}$ is then normalized to $\pi_t^{*(n)}$ by (3.16).

3.3.7 Re-initialization based on \widehat{N}_{eff}

Ideally a tracked object should be visible during an entire tracking session. In reality, however, it is quite common that the object moves out of frame or is occluded by other objects. In these cases, the tracker is required to re-initialize tracking. In general sequential Monte Carlo methods, the effective particle size N_{eff} has been used as a suitable measure of degeneracy (Doucet et al. 2000). Since it is hard to evaluate N_{eff} exactly, an alternative estimate \widehat{N}_{eff} is defined in Doucet et al. (2000):

$$\widehat{N}_{eff} = \frac{1}{\sum_{i=1}^N (\tilde{\pi}^{(i)})^2}. \quad (3.25)$$

Often it has been used as a measure to execute the resampling procedure. But, in our tracker we resample particles every frame, and hence we use \widehat{N}_{eff} as a measure to do re-initialization. When the number of effective particles is below a fixed threshold N_{thres} , the re-initialization procedure is performed.

The overall algorithm describing the particle filtering on the $SE(3)$ group is shown in Algorithm 3.1 where referred algorithms and equations are cited as $\langle \cdot \rangle$ and (\cdot) in the comments area, respectively. It requires a sequence of images \mathcal{I} and the keyframes \mathcal{F} as an input and estimates the posterior density as a set of weighted particles \mathcal{S} in each time t .

3.4 Experimental Results

In this section, we validate our proposed particle filter-based tracker via comprehensive experiments. We compare the performance of our approach with that of the single pose hypothesis tracker (Choi and Christensen 2010) and a

state-of-the-art tracker, **BLORT** (Mörwald et al. 2010). For the comparison, we obtained a set of real image sequences as well as synthetic image sequences for quantitative analysis.

To obtain the synthetic image sequences, we first tuned the projection matrix in **OpenGL** with the intrinsic camera parameters of our monocular camera. We then rendered the objects with texture mapping to visualize our target objects as realistic as possible. In addition, we prepared two backgrounds, simple white and complex textured, so that the performance comparison between two different background is possible. During rendering, we was continuously changing the camera position and orientation in **OpenGL** to simulate the real camera motion. The object poses during the rendering were saved to be used as known ground truth. For the real image sequences, we placed the monocular camera around the target objects and moved the camera.

Our system is composed of a standard desktop computer (Intel Core2 Quad **CPU** Q9300, 3.25G **RAM**, NVIDIA Quadro FX 570) and a Point Grey Research’s Flea 1394 camera (640 × 480 resolution). The **CAD** models of the “Teabox”, “Book” and “Cup” were designed by using Blender™ which is an open source 3D modeling tool. The “Car door” model was provided by an automobile company. We converted all of the models to the OBJ format to be used in our C++ implementation.

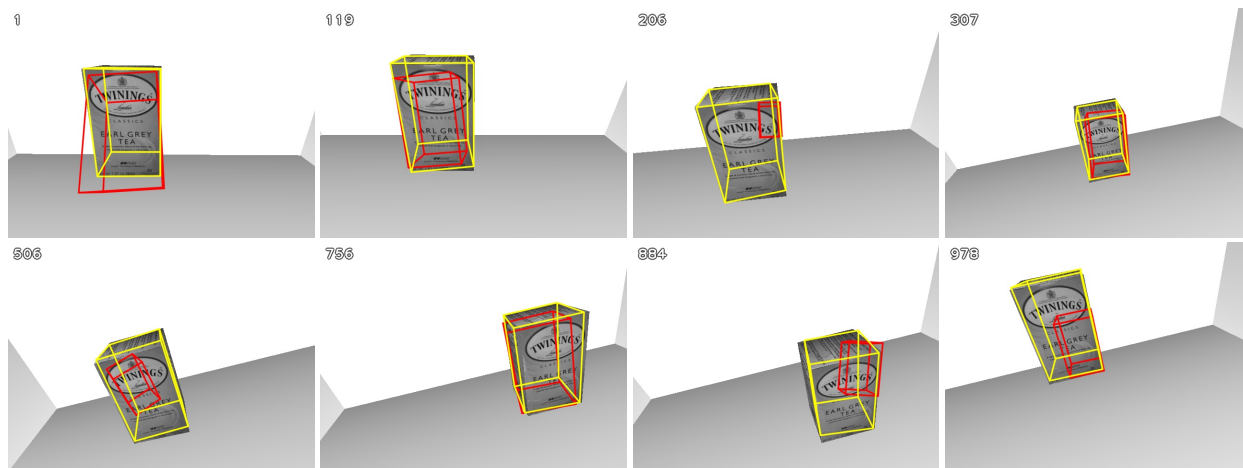
To verify our approach, we presents a series of comparative experiments:

1. Effectiveness of considering edge orientations
2. Effectiveness of considering multiple pose hypotheses
3. Effectiveness of performing **RANSAC**
4. Effectiveness of employing **AR** state dynamics

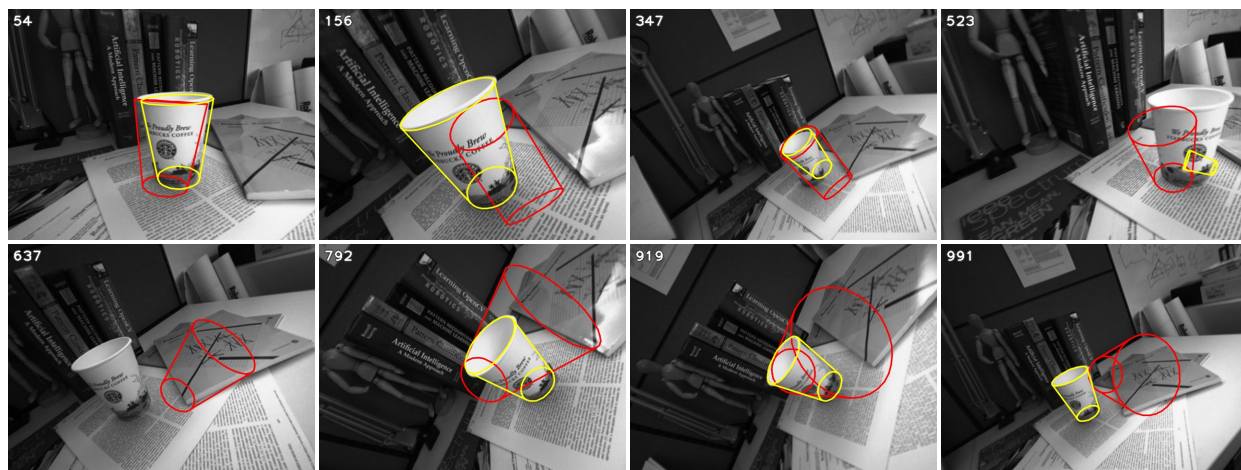
First, we examine the effect of considering edge orientations in §3.4.1 where our approach with only one particle is compared with the baseline in Choi and Christensen (2010). Second, the effectiveness of considering multiple pose hypotheses is verified through a comparison between trackers having single and multiple particles in §3.4.2. In §3.4.3, the effectiveness of the refinement process using **RANSAC** is scrutinized. Lastly, the benefit of employing the first-order **AR** state dynamics is discussed in §3.4.4. Each comparative experiment represents results of synthetic followed by real image sequences. For fair performance comparison, all parameters are the same except the compared parameter. After showing the comparative experiments, we show the re-initialization capability of our approach in §3.4.5. Finally, we compare the performance of our approach with that of a state-of-the-art tracker in §3.4.6.

3.4.1 Effectiveness of Considering Edge Orientations

When we find edge correspondences, we consider orientations of the edge points. By looking up orientations, we exclude edge correspondences having large differences in orientation between the sample points on model edges and the corresponding edge points from an input image. For this orientation testing, a simple indication function was defined in (3.17). It seems a simple enhancement, but even this modification significantly enhance tracking performance.



(a) The synthetic image sequence of the “Teabox” object: simple background case



(b) The real image sequence of the “Cup” object

Figure 3.4. Tracking results showing the effectiveness of considering edge orientation. Results of our approach with only one particle (i.e. $N = 1$) are depicted in yellow wireframes. As a baseline, tracking results of [Choi and Christensen \(2010\)](#) are shown in red wireframes. In spite of the limited number of particles, the proposed approach shows good tracking performance in the simple background sequence and acceptable tracking performance in the cluttered background real sequence, while the baseline is frequently stuck in local minima, and hence it drifts over the textured regions of the object or the background clutter. This difference is mainly due to false edge correspondences in the baseline, while the proposed approach discards some portion of the false edge correspondences by comparing edge orientation.

To investigate the effectiveness of considering edge orientations, we run our approach with only one particle in the synthetic and real image sequences. Since only one particle is considered in here, Gaussian noise is not added in propagation. As a reference, a similar single pose hypothesis tracker of [Choi and Christensen \(2010\)](#) that does not consider the edge orientations was also run in the same image sequences. The tracking results are presented in Figure 3.4 where the results with and without considering edge orientations, which is equivalent to the baseline, are depicted in yellow and red wireframes respectively. Note that both of the pose results are depicted in the same image sequences to clearly show the difference of the two approaches. Although the proposed approach uses the limited number of particles, it shows good tracking performance in the simple background synthetic sequence (Figure 3.4 (a)) and acceptable tracking performance in the cluttered background real sequence (Figure 3.4 (b)). However,

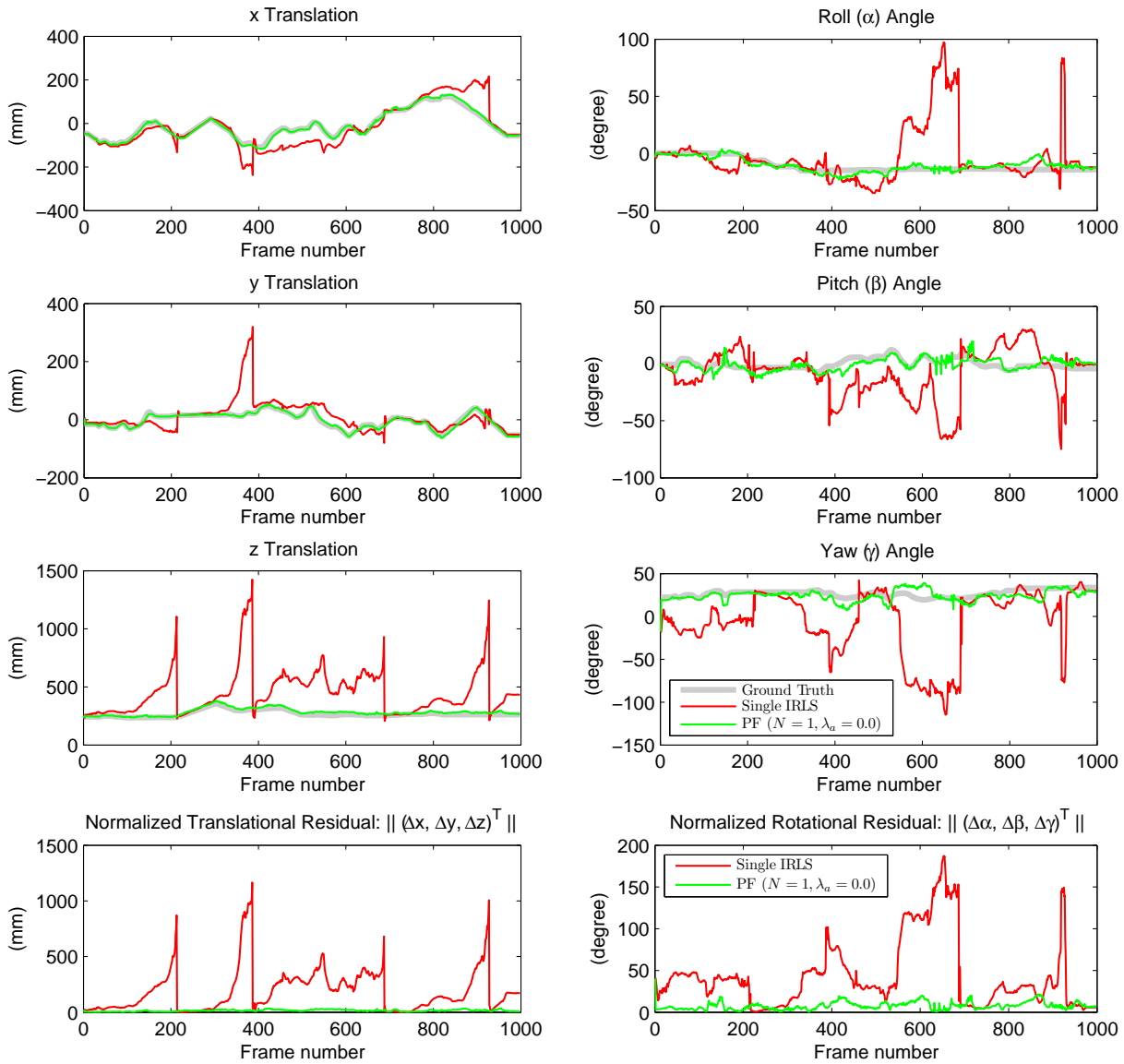
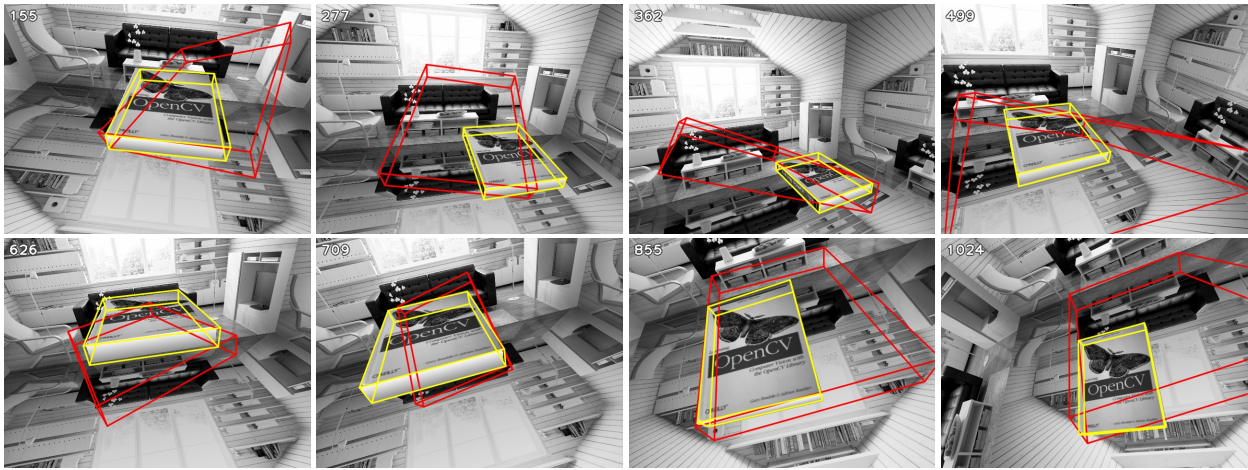


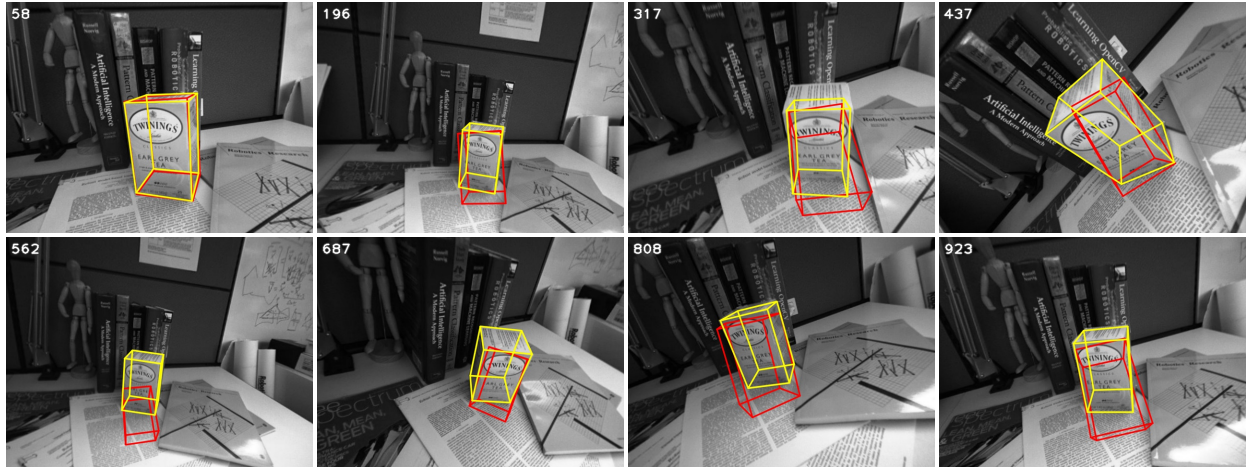
Figure 3.5. The 6-D pose and normalized residual plots of the results in Figure 3.4 (a). The proposed approach, particle filter ($N = 1, \lambda_a = 0.0$), shows reasonable accuracy, while the baseline from Choi and Christensen (2010) has significant errors. The baseline recovers from local minima via the re-initialization described in Choi and Christensen (2010), but it soon drifts again.

the baseline is frequently stuck in local minima, and hence it drifts over the textured regions of the object or the background clutter. To decompose pose results, 6-D pose and residual plots of the “Teabox” object are presented in Figure 3.5. According to the plots, we can easily see the difference where the proposed approach, PF ($N = 1, \lambda_a = 0.0$), shows much better results than the previous approach, Single IRLS. A quantitative analysis of this and following tests on synthetic sequences is presented in Table 3.1 which shows the root mean square (RMS) errors. For each image sequence, the upper row shows the RMS errors of the baseline and the lower five rows are the RMS errors of the proposed approach in various parameter settings for which the number of particles N , the AR parameter λ_a , and operation of RANSAC were altered.

The difference between the previous and the proposed approaches is mainly due to false edge correspondences



(a) The synthetic image sequence of the “Book” object: complex background case



(b) The real image sequence of the “Teabox” object




Figure 3.6. Tracking results showing the effectiveness of considering multiple pose hypotheses. Results of our approach maintaining 1 and 100 particles are depicted in red and yellow wireframes, respectively. For fair comparison, the same parameters are used except the number of particles. The complex background textures often cause false edge correspondences. Thus the single pose hypothesis tracker drifts during the entire tracking, while the tracker considering 100 pose hypotheses shows robust tracking results.

in the baseline, while the proposed approach discards some portion of the false edge correspondences by comparing edge orientation. These experimental results clearly support the argument that considering edge orientations enhances the quality of the matching between edge points.

3.4.2 Effectiveness of Considering Multiple Pose Hypotheses

As shown in §3.4.1, when the background is relatively simple, single pose hypothesis edge-based tracking shows reasonable performance. But it is quite challenging to reliably track an object when the background is complex or there is an amount of clutter. These challenging situations often make erratic edge correspondences, and hence single pose hypothesis tracking can be a fragile solution in these cases. To validate this argument, we compare two versions of our tracker using 1 and 100 particles. The comparative tracking results are shown in Figure 3.6, where results of the proposed tracker with 1 and 100 particles are drawn in red and yellow wireframes, respectively. For

Table 3.1. RMS errors and computation time in synthetic image sequences (Baseline vs. PF).

Objects	Mode	RMS Errors*						Time§
		X	Y	Z	Roll	Pitch	Yaw	
	Baseline (single IRLS)†	51.131	44.509	263.258	26.280	24.330	48.323	20.62
	PF ($N = 1, \lambda_\alpha = 0.0$)	5.234	3.323	16.034	3.555	4.626	6.727	18.11
	PF ($N = 100, \lambda_\alpha = 0.0$)	4.616	2.498	13.623	2.558	3.940	3.867	450.57
	PF ($N = 100, \lambda_\alpha = 0.5$)	4.272	2.255	12.430	2.375	3.882	3.628	449.39
	PF ($N = 100, \lambda_\alpha = 0.0$, RANSAC)	3.503	1.930	8.917	1.936	3.227	2.597	459.49
	PF ($N = 100, \lambda_\alpha = 0.5$, RANSAC)	3.234	1.829	8.062	1.843	3.057	2.530	464.69
	Baseline (single IRLS)†	63.074	27.075	83.377	25.028	38.839	62.695	23.05
	PF ($N = 1, \lambda_\alpha = 0.0$)	20.209	21.130	55.466	35.613	28.153	61.006	22.61
	PF ($N = 100, \lambda_\alpha = 0.0$)	4.521	8.357	13.974	4.431	7.717	5.593	471.78
	PF ($N = 100, \lambda_\alpha = 0.5$)	3.028	4.846	11.739	3.313	5.965	3.637	504.13
PF ($N = 100, \lambda_\alpha = 0.0$, RANSAC)	3.125	4.398	10.207	2.712	4.976	3.324	497.05	
PF ($N = 100, \lambda_\alpha = 0.5$, RANSAC)	2.795	4.953	9.503	2.410	4.854	3.400	519.03	
	Baseline (single IRLS)†	16.980	37.375	131.320	8.372	22.691	20.273	21.83
	PF ($N = 1, \lambda_\alpha = 0.0$)	76.376	95.839	92.012	3.571	14.141	27.524	20.90
	PF ($N = 100, \lambda_\alpha = 0.0$)	4.633	3.570	22.881	1.107	1.718	3.661	802.08
	PF ($N = 100, \lambda_\alpha = 0.5$)	4.502	3.402	22.240	0.954	1.508	3.791	799.30
	PF ($N = 100, \lambda_\alpha = 0.0$, RANSAC)	4.298	3.420	21.107	1.037	1.618	3.563	815.76
	PF ($N = 100, \lambda_\alpha = 0.5$, RANSAC)	4.140	3.171	20.084	0.876	1.408	3.545	811.00
	Baseline (single IRLS)†	242.794	71.234	168.683	35.455	42.016	18.776	25.61
	PF ($N = 1, \lambda_\alpha = 0.0$)	53.528	61.926	177.906	13.346	15.161	17.974	25.03
	PF ($N = 100, \lambda_\alpha = 0.0$)	11.663	14.519	28.967	3.276	2.771	4.189	771.41
	PF ($N = 100, \lambda_\alpha = 0.5$)	10.343	15.288	27.654	4.208	2.881	3.904	816.39
PF ($N = 100, \lambda_\alpha = 0.0$, RANSAC)	4.554	4.751	9.156	1.772	1.967	3.746	819.32	
PF ($N = 100, \lambda_\alpha = 0.5$, RANSAC)	3.195	3.843	8.239	1.589	1.979	3.697	819.85	
	Baseline (single IRLS)†	36.317	31.552	122.737	-	78.843‡	-	33.65
	PF ($N = 1, \lambda_\alpha = 0.0$)	189.488	19.947	58.342	-	25.917	-	32.63
	PF ($N = 100, \lambda_\alpha = 0.0$)	3.842	3.098	14.837	-	2.116	-	1915.49
	PF ($N = 100, \lambda_\alpha = 0.5$)	3.574	3.008	13.820	-	1.960	-	1910.34
	PF ($N = 100, \lambda_\alpha = 0.0$, RANSAC)	3.112	2.544	12.076	-	1.777	-	1928.42
	PF ($N = 100, \lambda_\alpha = 0.5$, RANSAC)	2.933	2.422	11.063	-	1.515	-	1925.59
	Baseline (single IRLS)†	126.662	57.309	127.284	-	33.180‡	-	37.67
	PF ($N = 1, \lambda_\alpha = 0.0$)	65.645	48.978	100.036	-	71.873	-	36.53
	PF ($N = 100, \lambda_\alpha = 0.0$)	8.856	11.778	22.064	-	7.041	-	1913.69
	PF ($N = 100, \lambda_\alpha = 0.5$)	8.312	11.417	20.045	-	6.186	-	1911.11
PF ($N = 100, \lambda_\alpha = 0.0$, RANSAC)	7.111	18.625	15.577	-	6.197	-	1936.90	
PF ($N = 100, \lambda_\alpha = 0.5$, RANSAC)	6.997	18.516	17.134	-	6.333	-	1949.41	
	Baseline (single IRLS)†	6.446	8.435	9.615	0.398	0.771	1.529	33.50
	PF ($N = 1, \lambda_\alpha = 0.0$)	5.206	6.821	11.921	0.333	0.974	1.523	40.93
	PF ($N = 100, \lambda_\alpha = 0.0$)	4.645	6.437	11.129	0.292	0.841	1.209	2699.23
	PF ($N = 100, \lambda_\alpha = 0.5$)	4.049	6.003	11.113	0.254	0.797	1.114	2702.93
	PF ($N = 100, \lambda_\alpha = 0.0$, RANSAC)	4.388	6.166	9.189	0.318	0.718	0.967	2691.41
	PF ($N = 100, \lambda_\alpha = 0.5$, RANSAC)	3.831	5.778	9.281	0.281	0.658	0.944	2676.74
	Baseline (single IRLS)†	93.991	70.093	323.827	16.753	7.972	60.336	37.16
	PF ($N = 1, \lambda_\alpha = 0.0$)	12.550	29.420	80.789	1.636	5.055	14.382	45.46
	PF ($N = 100, \lambda_\alpha = 0.0$)	9.872	14.614	28.212	0.913	3.257	5.960	2796.06
	PF ($N = 100, \lambda_\alpha = 0.5$)	7.488	11.584	24.185	0.731	1.865	4.096	2793.18
PF ($N = 100, \lambda_\alpha = 0.0$, RANSAC)	6.167	11.721	17.653	0.822	2.384	4.417	2733.86	
PF ($N = 100, \lambda_\alpha = 0.5$, RANSAC)	4.787	9.137	15.942	0.617	1.400	2.758	2737.89	

* The error units of translation and rotation are millimeter and degree, respectively. Better results are indicated in bold type.

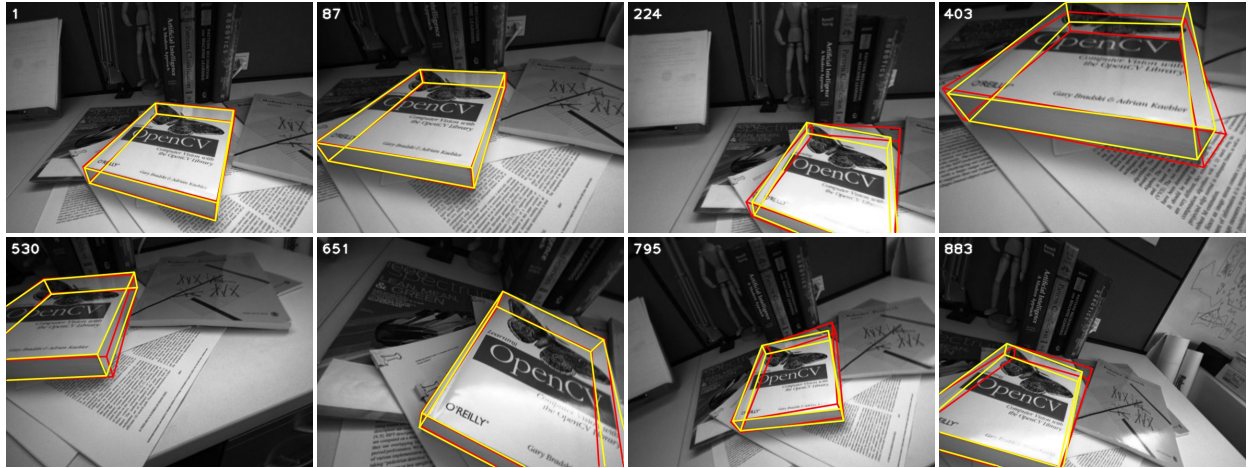
§ The unit of computation time is milliseconds per frame.

† The results of the previous approach (Choi and Christensen 2010) are compared as a baseline.

‡ Since the ‘‘Cup’’ object is symmetric and thus exact orientation cannot be estimated, we calculate the angle of the axis of symmetry.



(a) The synthetic image sequence of the “Cup” object: complex background case



(b) The real image sequence of the “Book” object

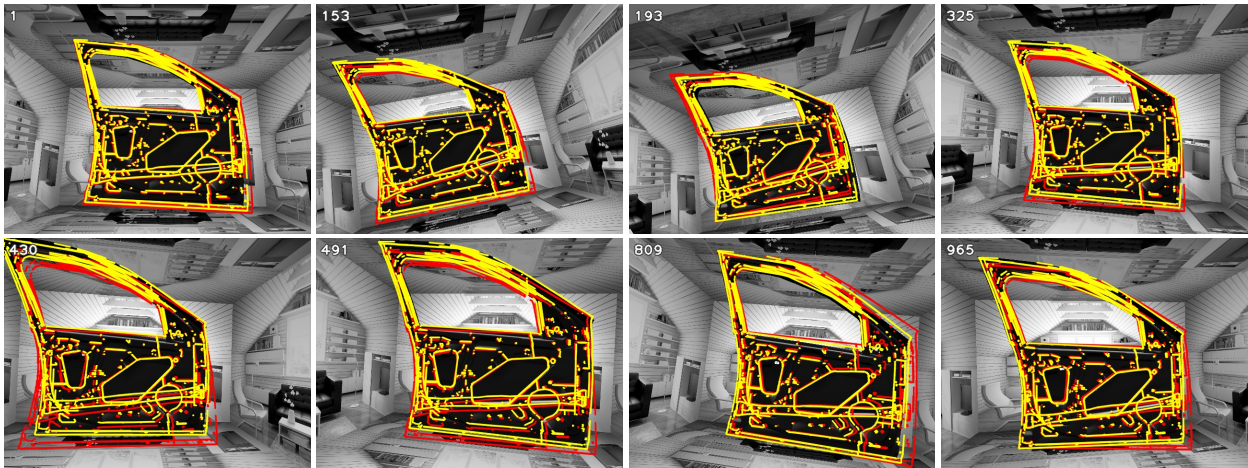
Figure 3.7. Tracking results showing the effectiveness of performing **RANSAC**. The yellow wireframes represent estimated poses of our approach with **RANSAC** refinement; The red wireframes are the results of the same approach without **RANSAC**. Both use the same number of particles and the **AR** process parameter ($N = 100, \lambda_a = 0.0$). While both show good tracking results, the one with **RANSAC** clearly exhibits better results. By performing **RANSAC**, some false edge correspondences which are inconsistent with the best pose hypothesis are discarded. Thus our approach with **RANSAC** tracks robustly, while the one without **RANSAC** is frequently misled by the complex background clutter.

clear visualization, we only display the mean of particles calculated via the valid mean of particles in (3.12). For fair comparison, the same parameters are used except the number of particles.

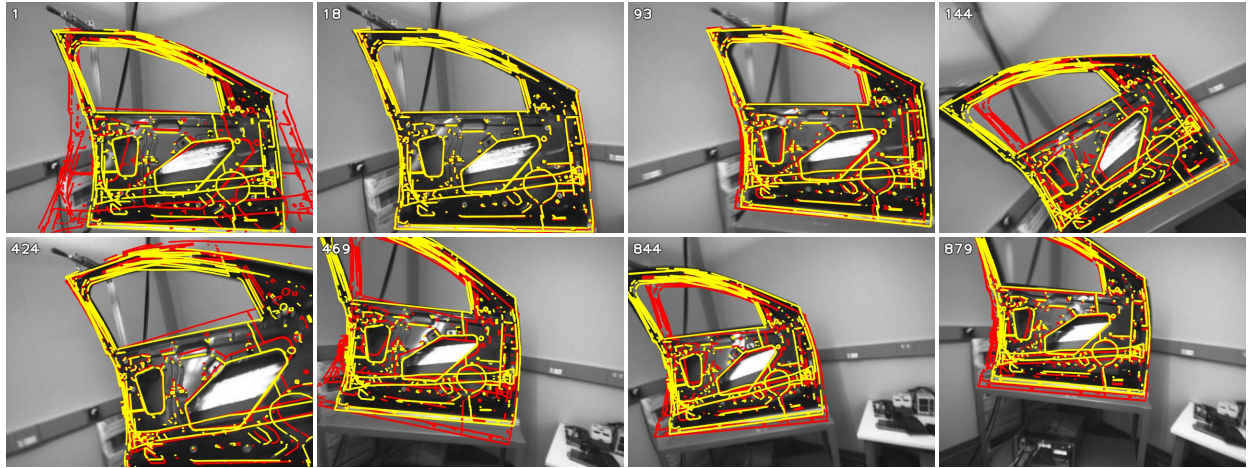
The complex background textures often cause false edge correspondences. Thus the single pose hypothesis tracker drifts during the entire tracking, while the tracker considering multiple pose hypotheses shows robust tracking results. Since our particle filter considers multiple pose hypotheses and resamples based on measurement likelihood, it is quite robust to false edge correspondences from which the single pose hypothesis tracker is often suffered.

3.4.3 Effectiveness of Performing **RANSAC**

As comparing orientations of edge points, our approach discards some false edge correspondences. But it still tend to have false edge matches because it might happen to have similar orientations but false correspondences. Our



(a) The synthetic image sequence of the “Car door” object: complex background case



(b) The real image sequence of the “Car door” object

Figure 3.8. Tracking results showing the effectiveness of employing AR state dynamics. To investigate the advantage of incorporating the AR state dynamics in our framework, we run our approach with ($\lambda_a = 0.5$) and without ($\lambda_a = 0.0$) the dynamics. Results of our approach with and without AR state dynamics are depicted in yellow and red wireframes, respectively. Both maintain 100 particles and perform RANSAC. The only difference is the AR parameter λ_a . As the linear state dynamics propagates particles with respect to their velocities, the particle filtering evolves effectively, and hence mean of particles represented in the yellow wireframes follows the global optimum. But the one without AR state dynamics which is equivalent to a random walk model is occasionally stuck in local minima.

approach takes an additional refinement process based on RANSAC (Fischler and Bolles 1981). In this section, we examine the advantage of performing RANSAC refinement. The results of our approach with ($N = 100, \lambda_a = 0.0$, RANSAC) and without ($N = 100, \lambda_a = 0.0$) RANSAC are shown in yellow and red wireframes in Figure 3.7 respectively.

In the RMS error calculation, it is straightforward to compare rotational errors except the “Cup” object because it is symmetric and thus an exact set of three orientations cannot be estimated. Instead of estimating the three rotations, we calculate the angle between unit vectors whose directions are coincide with the axis of symmetry of

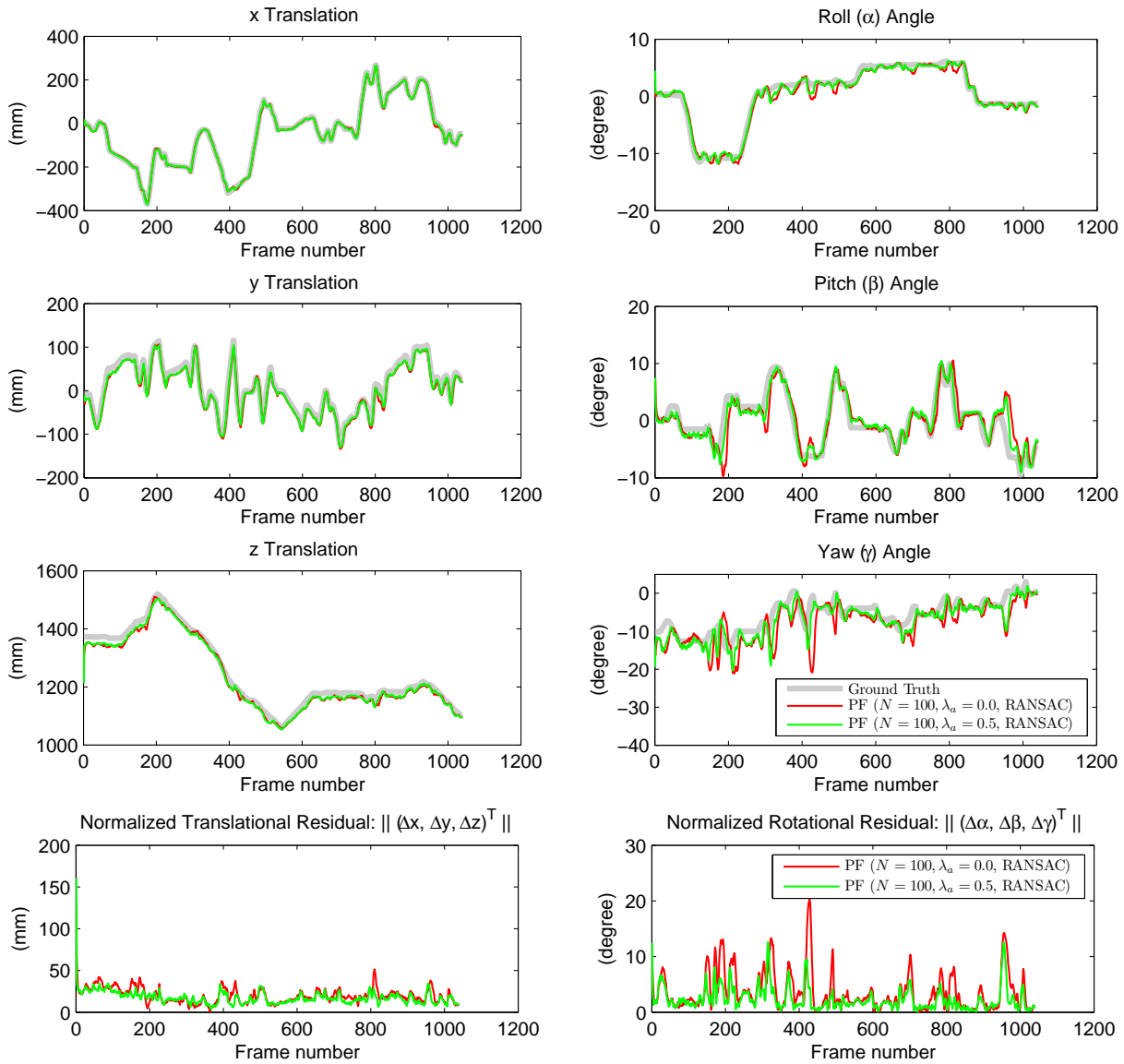


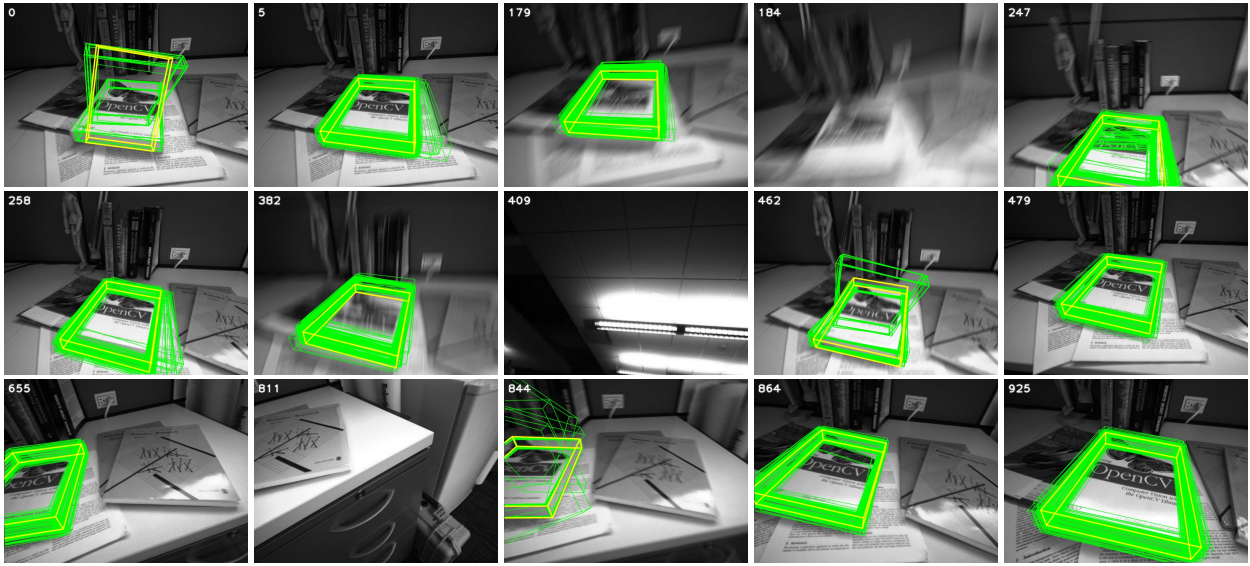
Figure 3.9. The 6-D pose and normalized residual plots of the results in Figure 3.8 (a). With the AR state dynamics, our approach shows better performance in terms of accuracy than the one without the dynamics. Especially, when the camera undergoes abrupt changes in position or orientation, the approach with the linear state dynamics exhibits agile responses, while the approach without the state dynamics results in bigger errors.

the object in the object coordinate system via

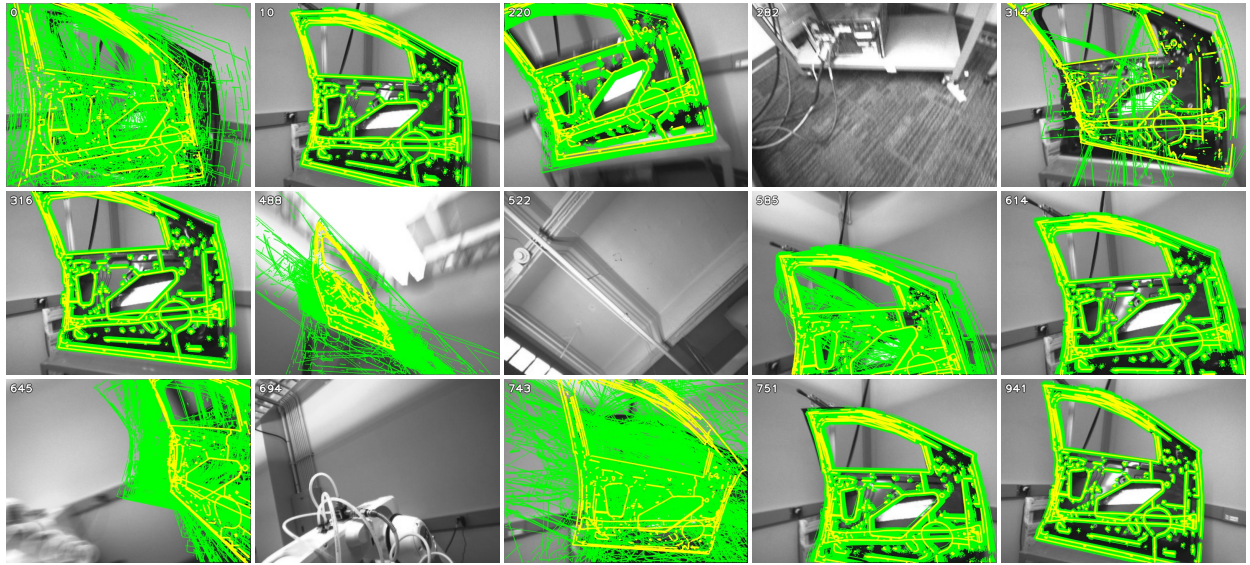
$${}^c \mathbf{u}_y = {}^c \mathbf{T}_O \cdot {}^O \mathbf{u}_y \quad (3.26)$$

$${}^c \hat{\mathbf{u}}_y = {}^c \hat{\mathbf{T}}_O \cdot {}^O \mathbf{u}_y \quad (3.27)$$

where ${}^c \mathbf{T}_O \in SE(3)$ is the pose of the object with respect to the camera frame estimated by our visual tracker, ${}^c \hat{\mathbf{T}}_O \in SE(3)$ is the pose of the rendered object with respect to the virtual camera frame saved in OpenGL rendering, and ${}^O \mathbf{u}_y = (0, 1, 0, 0)^T$ is the unit vector of the axis of symmetry of the object. Note that the fourth element of ${}^O \mathbf{u}_y$ is 0, and hence only rotation is considered. The angle between these unit vectors can be calculated by the inner



(a) The “Book” object



(b) The “Car door” object

Figure 3.10. Tracking results showing the capability of re-initialization. The set of green wireframes represent 100 particles and the yellow thick wireframe shows the mean of particles on each image. When the tracked object moves out of the field of view and images are blurred because of camera shaking, our tracker re-initializes by itself. At frame number $t = 0, 247, 462, 844$ at the “Book” and $t = 0, 314, 585, 743$ at the “Car door”, our approach is (re-)initialized. Since some keypoint correspondences were wrong, there are erroneous initial pose hypotheses. But the tracker quickly converges to the global optimum as the bad pose hypotheses are removed in the importance resampling process.

product as

$$\theta = \arccos({}^C \mathbf{u}_y^T {}^C \hat{\mathbf{u}}_y). \quad (3.28)$$

The rotational **RMS** errors of the “Cup” object are calculated with this angle in Table 3.1.

According to results in Figure 3.7 and Table 3.1, the one with **RANSAC** clearly exhibits better results. By performing **RANSAC**, some false edge correspondences which are inconsistent with the best pose hypothesis are discarded.

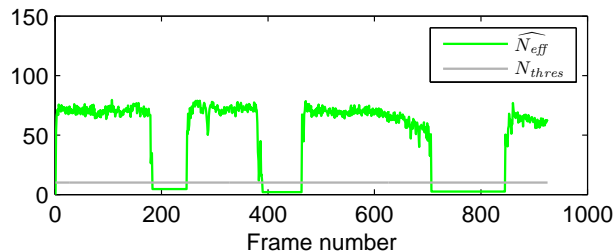


Figure 3.11. The \widehat{N}_{eff} plot of the results in Figure 3.10 (a). When the number of effective particles is under the threshold N_{thres} , our system re-initializes by itself.

Thus our approach with **RANSAC** tracks robustly, while the one without **RANSAC** is frequently misled by the complex background clutter.

3.4.4 Effectiveness of Employing **AR** State Dynamics

To verify the effect of the **AR** state dynamics, we execute the proposed approach with and without the **AR** state dynamics. To disable the dynamics, we set the parameter λ_a in the **AR** state dynamics equation (3.14) as 0 which is equivalent to a random walk model. For fair comparison, we use the same parameters except the **AR** parameter. We test in the synthetic and real image sequences of the “Car door” object. The tracking results are presented in Figure 3.8.

Although both use the same number of particles, Gaussian noise, and measurement likelihood, the tracking performances are quite distinctive. This difference is mainly due to the **AR** state dynamics which propagates particles according to the camera motion.

3.4.5 Re-initialization

In our previous system (Choi and Christensen 2010), we used a simple heuristic in which the difference in position of the object between frames and the number of valid sample points are monitored to trigger re-initialization. While that heuristic works well when the pose hypothesis drifts fast, it might not always be the case when the hypothesis stuck in local minima. We propose another way for re-initialization by taking advantage of multiple pose hypotheses. As in Algorithm 3.1, our system re-initializes when the number of effective particles \widehat{N}_{eff} is below a threshold.

To verify this method, we run the proposed tracker on two real image sequences. The tracking results are shown in Figure 3.10, and the number of effective particles for the first image sequence (Figure 3.10 (a)) is plotted over the frame numbers in Figure 3.11. When the tracked object moves out of the field of view or images are blurred because of camera shaking, the \widehat{N}_{eff} decreases significantly below the threshold value N_{thres} , and that triggers the re-initialization. This also works when the object is disappearing slowly. When the object gradually moves out of the field of view, the particles still follow the real trajectory of the object. After the object is completely disappeared, the particles have no visible sample points. Since it is impossible to evaluate the likelihood (3.21) without visible sample points and edge correspondences from them, the weight of the particle is set to zero. This reduces the number of effective particles, and thus it enables the re-initialization process. During re-initialization, the tracker

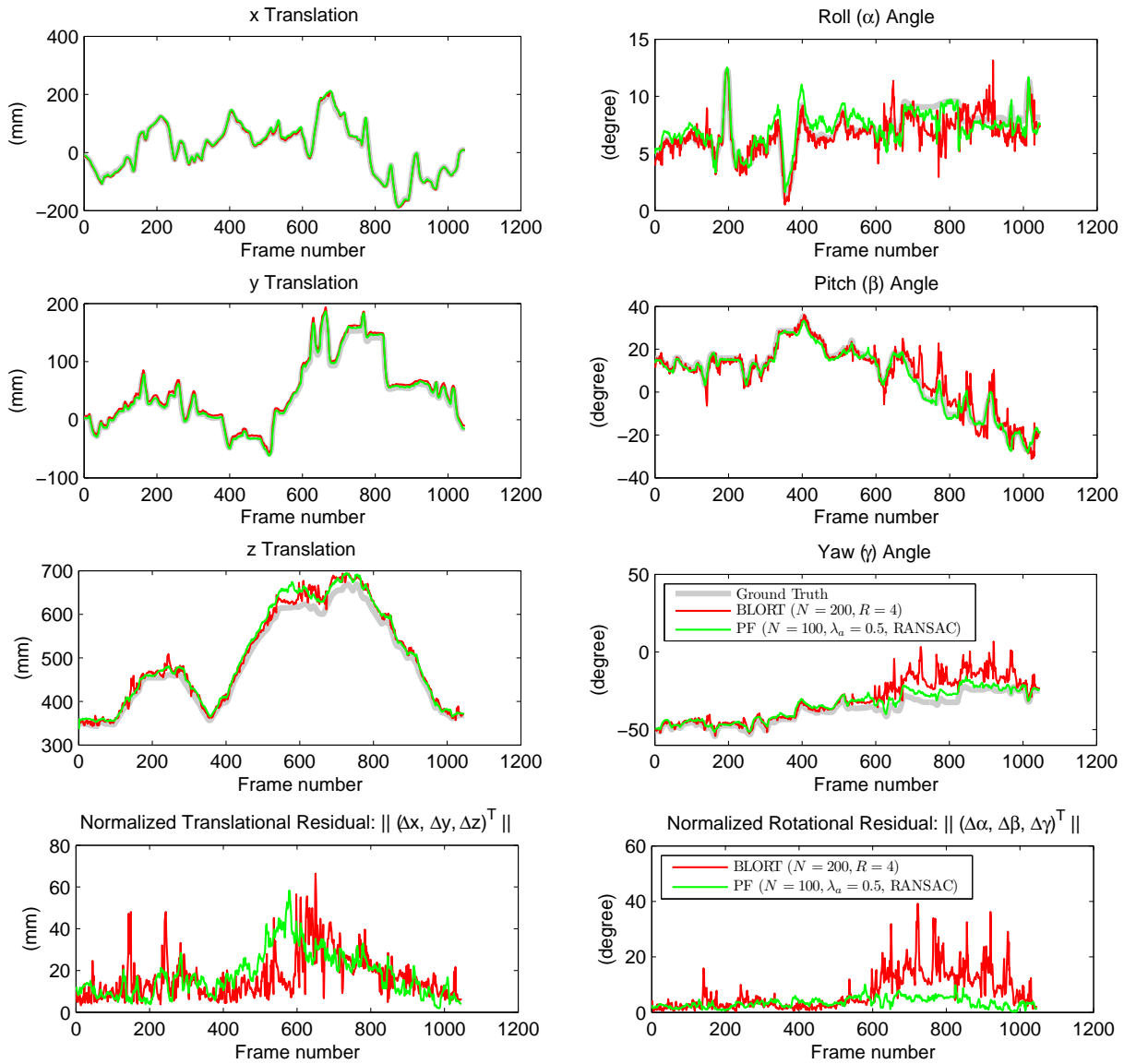
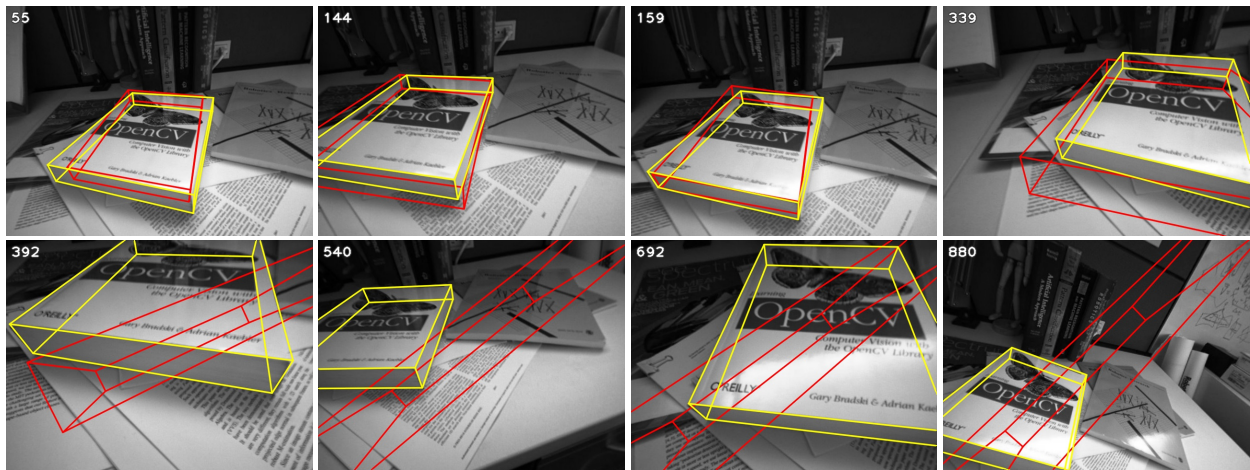


Figure 3.12. The 6-D pose and normalized residual plots of the results of our particle filter and **BLORT on the synthetic “Book” object sequence (simple background case).** With the sufficient number of particles and recursions ($N = 200, R = 4$), **BLORT** reports similar performance to our particle filter. **BLORT** even shows slightly better results in z translation possibly due to considering edges from texture of the object. However, **BLORT** severely suffers from jitter noise in rotations.

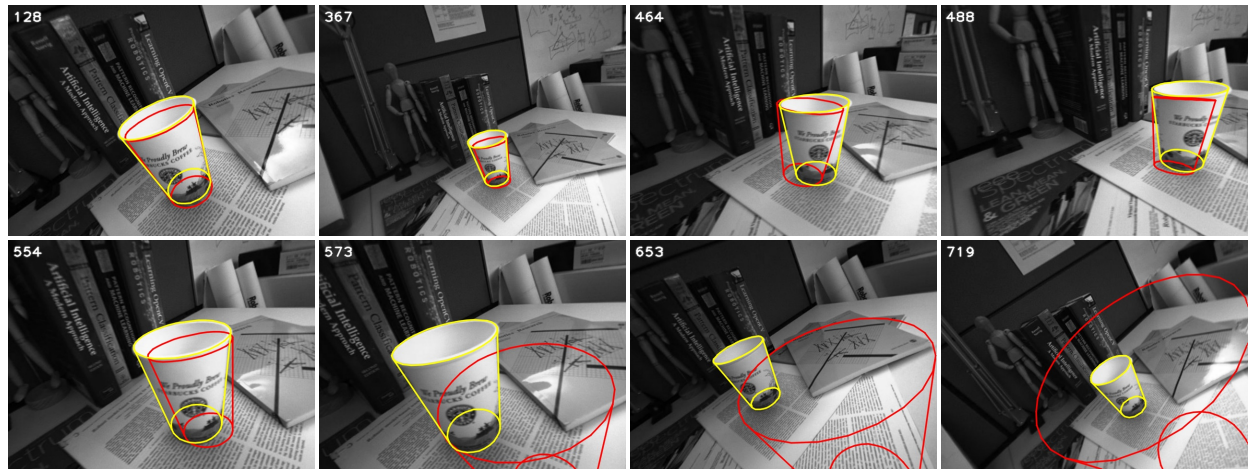
matches keypoints until it has at least the minimum number of keypoint correspondences τ_n . Once enough keypoint correspondences are acquired, the proposed system initializes particles as explained in §3.3.4 and Algorithm 3.2.

3.4.6 Comparison with **BLORT** tracker

So far we have shown a series of comparative experiments with Choi and Christensen (2010) or our proposed approach with several different parameters. However, it would be more convincing if we compare our proposed approach with the state-of-the-art tracker. To compare the performance of our approach with an existing solution, we chose **BLORT** tracker (Mörwald et al. 2010) because it combines state-of-the-art methods for object recognition



(a) The real image sequence of the “Book” object



(b) The real image sequence of the “Cup” object



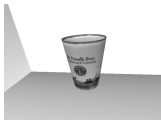
Figure 3.13. A comparison between the tracking results of our particle filter (yellow wireframes) and **BLORT** (red wireframes) on real image sequences. In both image sequences, our particle filter ($N = 100$, $\lambda_a = 0.5$, **RANSAC**) is compared with **BLORT** ($N = 200$, $R = 4$). While our particle filter well tracks the target objects, **BLORT** loses the targets in the middle of the sequences. Even before the tracking failures, pose results from **BLORT** are not well aligned to the global optimum.

and tracking as well as is publicly available.¹ By combining **SIFT**-based object recognition and edge-based particle filtering, **BLORT** provides reliable solution for robotic research. For robustness, **BLORT** considers edges from surface texture of objects in particle likelihood evaluation. Time consuming parts, such as hidden face removal, image processing, texture mapping, and particle filtering, are efficiently processed by **GPU**.

We used default parameters provided in the **BLORT** software package except the number of particles N and recursions R . Since **BLORT** employs the recursive particle filtering (Mörwald et al. 2009), the number of recursion also determines the robustness of tracking. Thus we changed these parameters in order to compare our approach with various settings of **BLORT**. Since **BLORT** cannot handle arbitrary complex shaped objects, we excluded “Car door” object in this experiments. We converted our OBJ formatted 3D polygonal mesh models to PLY format, which **BLORT** accepts. In addition, texture images of these objects were learned to be used in the likelihood evaluation

¹**BLORT** - The Blocks World Robotic Vision Toolbox: <http://users.acin.tuwien.ac.at/mzillich/?site=4>

Table 3.2. RMS errors and computation time in synthetic image sequences (PF vs. BLORT).

Objects	Mode	RMS Errors*						Time§	
		X	Y	Z	Roll	Pitch	Yaw		
Teabox		PF ($N = 100, \lambda_a = 0.5$, RANSAC)	3.234	1.829	8.062	1.843	3.057	2.530	464.69
		BLORT ($N = 100, R = 2$)	6.344	3.570	6.825	1.915	8.753	4.146	99.72
		BLORT ($N = 200, R = 2$)	4.747	3.164	5.878	1.528	6.879	3.228	174.36
		BLORT ($N = 100, R = 4$)	3.897	3.169	5.605	1.511	6.127	3.353	173.96
		BLORT ($N = 200, R = 4$)	3.684	2.289	4.821	1.276	5.572	1.946	325.03
		PF ($N = 100, \lambda_a = 0.5$, RANSAC)	2.795	4.953	9.503	2.410	4.854	3.400	519.03
		BLORT ($N = 100, R = 2$)	12.639	5.808	18.893	3.559	16.471	11.607	104.37
		BLORT ($N = 200, R = 2$)	42.314	25.175	110.392	4.246	28.717	10.883	101.15
		BLORT ($N = 100, R = 4$)	5.101	2.630	7.608	2.550	7.379	3.836	190.70
		BLORT ($N = 200, R = 4$)	6.411	3.180	16.986	3.745	6.537	3.744	331.18
Book		PF ($N = 100, \lambda_a = 0.5$, RANSAC)	4.140	3.171	20.084	0.876	1.408	3.545	811.00
		BLORT ($N = 100, R = 2$)	6.651	6.194	29.169	2.439	11.056	14.747	85.71
		BLORT ($N = 200, R = 2$)	4.736	5.989	22.240	1.904	7.898	11.589	147.94
		BLORT ($N = 100, R = 4$)	3.962	6.119	21.459	1.399	5.777	9.643	150.55
		BLORT ($N = 200, R = 4$)	3.643	5.893	18.073	1.132	4.346	8.780	275.86
		PF ($N = 100, \lambda_a = 0.5$, RANSAC)	3.195	3.843	8.239	1.589	1.979	3.697	819.85
		BLORT ($N = 100, R = 2$)	73.011	79.695	262.610	27.580	47.031	38.527	37.32
		BLORT ($N = 200, R = 2$)	73.198	90.931	216.233	44.918	17.872	23.865	51.41
		BLORT ($N = 100, R = 4$)	73.824	86.040	232.869	38.145	31.934	23.138	48.50
		BLORT ($N = 200, R = 4$)	59.819	76.913	229.491	56.344	37.451	29.128	75.40
Cup†		PF ($N = 100, \lambda_a = 0.5$, RANSAC)	2.933	2.422	11.063	–	1.515 †	–	1925.59
		BLORT ($N = 100, R = 2$)	41.878	24.509	187.267	–	57.789	–	36.91
		BLORT ($N = 200, R = 2$)	30.419	26.715	93.366	–	83.586	–	63.46
		BLORT ($N = 100, R = 4$)	56.216	45.414	610.823	–	55.184	–	52.97
		BLORT ($N = 200, R = 4$)	3.979	5.006	27.852	–	5.091	–	222.57
		PF ($N = 100, \lambda_a = 0.5$, RANSAC)	6.997	18.516	17.134	–	6.333 †	–	1949.41
		BLORT ($N = 100, R = 2$)	86.748	75.334	226.736	–	69.987	–	37.03
		BLORT ($N = 200, R = 2$)	79.777	35.124	114.818	–	69.422	–	56.70
		BLORT ($N = 100, R = 4$)	36.628	73.228	179.439	–	85.200	–	77.02
		BLORT ($N = 200, R = 4$)	64.853	52.676	132.043	–	105.518	–	98.50

* The error units of translation and rotation are millimeter and degree, respectively. Better results are indicated in bold type.

§ The unit of computation time is milliseconds per frame.

† Since the “Cup” object is symmetric and thus exact orientation cannot be estimated, we calculate the angle of the axis of symmetry.

of its particle filtering, and SIFT features were also saved to be used in the SIFT-based object recognition. For our tracker, 100 particles, the AR state dynamics, and RANSAC refinement were employed (*i.e.* $N = 100, \lambda_a = 0.5$, RANSAC).

Our approach and BLORT were tested in the synthetic image sequences. While we fixed the parameters of our approach, we tried several different numbers of particles ($N = 100$ or $N = 200$) and recursions ($R = 2$ or $N = 4$) for BLORT so that we can compare our approach with various settings of BLORT.² Figure 3.12 shows the 6-D pose and residual plots of the result in the “Book” simple background image sequence. With the sufficient number of particles and recursions ($N = 200, R = 4$), BLORT reports similar performance to our particle filter. BLORT even shows slightly better results in z translation possibly due to considering edges from texture of the object. However, BLORT severely suffers from jitter noise in rotations.

These plots in Figure 3.12 are the best results from BLORT. In many cases, BLORT lost tracking in the middle of image sequences. In Table 3.2, RMS errors and computation time of all experiments for the synthetic image sequences are presented. As shown in the table, BLORT reports poor accuracy, especially in complex background

²The default number of particles and recursions in BLORT are $N = 200$ and $R = 2$ respectively.

sequences (e.g. “Book” and “Cup”). While our approach outperforms **BLORT** in terms of accuracy in many cases, **BLORT** shows comparable results in “Teabox” object. The reason why **BLORT** gives better results for “Teabox” object is possibly due to the amount of texture. Since **BLORT** takes advantage of edges coming from surface texture in its particle filtering, sufficient texture information is crucial to a robust tracking. “Teabox” object has relatively plentiful textures compared to “Book” and “Cup” objects. Thus **BLORT** reports less **RMS** errors even in the “Teabox” complex background sequence, while it even shows large errors in “Cup” simple background sequence. On the contrary, our particle filter relies on sharp edges in the 3D polygonal model, and thus our approach is independent of textures. Better accuracy of our tracker is further achieved via **IRLS** optimization, **AR** state dynamics, and **RANSAC** refinement. Nor surprisingly, accuracy of **BLORT** gets better as the number of particles N or recursions R increases, but at the same time the computation time increases. As **BLORT** exploited the parallel power of **GPU**, it shows higher frame rates than our approach.

Similar comparative experiments are done in the real image sequences. The results in the “Book” and “Cup” image sequences are presented in Figure 3.13. While our particle filter (yellow wireframes) well tracks the target objects, **BLORT** (red wireframes) loses the targets in the middle of the sequences. Once **BLORT** failed to track the target, it remained as lost without any successful recovery until the end of sequence. Even before the tracking failures, pose results from **BLORT** are not well aligned to the global optimum.

BLORT reported good performance in simple background image sequences with well textured objects. However, it did poorly perform in relatively complex background with less textured objects, and its tracking was not as agile as our tracker. The pose results of **BLORT** showed severe jitter noise, especially in rotation, which is not desirable for robotic applications. We note the high frame rates of **BLORT** due mainly to the parallelization of the algorithm in **GPU**. It might imply that exploiting the parallel power from **GPU** would be worthwhile to accelerate the processing time of our approach.

3.5 Summary

We have presented an approach to 3D visual object tracking based on a particle filtering algorithm on the $SE(3)$ group. For fast particle convergence, we employed keypoint features and initialized particles by solving the **PnP** problem. Particles are propagated by the state dynamics which is given by the **AR** process on the $SE(3)$, and the state dynamics distributed particles more effectively. Edge correspondences were first enhanced by considering orientations of the edge points, and they were further refined through the **RANSAC** process. Measurement likelihood was calculated from both the residual and the number of valid sample points of the edge correspondences. During the tracking, the proposed system appropriately re-initialized by itself when the number of effective particles was below the threshold. Our approach has been tested via various experiments in which our multiple pose hypotheses tracker has shown notable performance on challenging background and clutter.

CHAPTER IV

EXTENDING TO TEXTURELESS OBJECTS (2D)

While the previous chapter described the textured object pose tracking, this chapter presents an extension of our previous approach to textureless objects. For the initialization of a textureless object pose, an efficient chamfer matching is employed so that a set of coarse pose hypotheses is estimated from matching results between 2D edge templates of an object and a query image, and particles are then initialized from the coarse pose hypotheses. To ensure the initialized particles are at or close to the global optimum, an annealing process is further performed after the initialization.

Our key contributions are as follows:

- We employ an efficient chamfer matching to find a set of starting states. Most particle filtering approaches assume that an initial state is given or is searched from scratch with the simulated annealing (Klein and Murray 2006). Several approaches have presented keypoint-based initialization (Choi and Christensen 2011), but keypoints are not usually applicable to textureless objects. Thus, we present a 3D pose estimation from a chamfer matching (Liu et al. 2010a) using a set of 2D edge templates.
- Although initial particles are assigned via the coarse pose hypotheses, they would be occasionally stuck in local optima right after the initialization. To ensure that initial states are at or close to the global optimum, we run a particle annealing method (Deutscher et al. 2000) right after the (re-)initialization.

This chapter is organized as follows. The edge template generation by rendering a CAD model of an object is described in Section 4.1.1, followed by the coarse pose estimation in Section 4.1.2. The annealed particle filtering, which is employed for more robust convergence of the initialized particles, is presented in Section 4.1.3. We also address how to reduce the search space of the particle filter in the case of symmetric objects in Section 4.1.4. In Section 4.2, comparative results for several image sequences are shown to validate the effectiveness of our approach.

4.1 Edge-based Pose Estimation

To initialize particles, coarse poses are estimated by employing an efficient chamfer matching (Liu et al. 2010a) that provides a sub-linear time solution for the matching and shows fewer false positive rates via the piecewise smooth cost function. For this, a set of edge templates is obtained offline from polygonal mesh models as training data.

4.1.1 Generating Edge Templates

We obtain a set of edge templates $\mathcal{T} = \{\mathcal{T}_1, \mathcal{T}_2, \dots, \mathcal{T}_T\}$ from a polygonal mesh model of an object, where T is the number of templates. To generate these templates, the projection matrix in OpenGL is set from the intrinsic

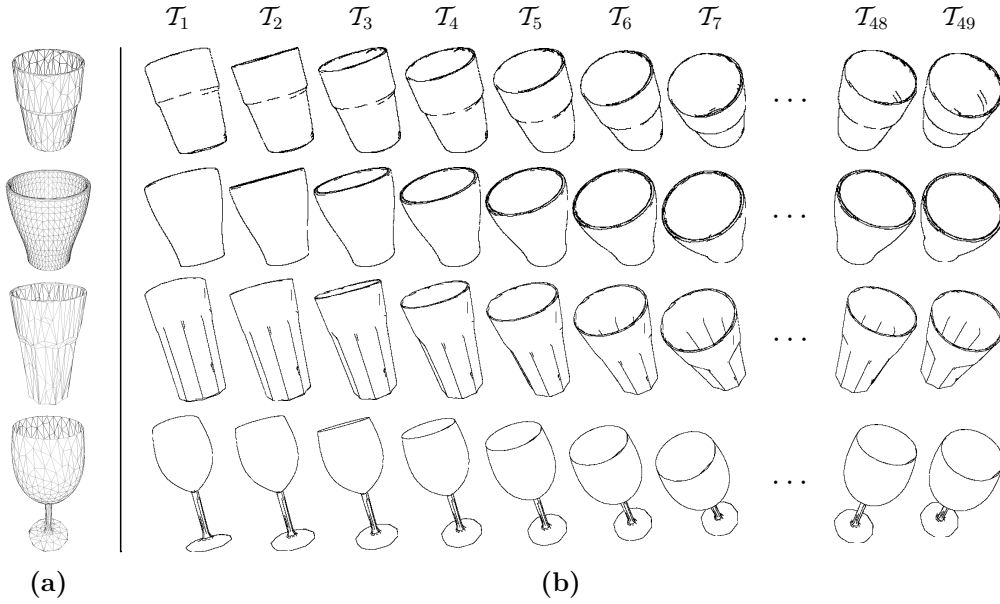


Figure 4.1. Polygonal mesh models and edge templates. (a) We chose 4 IKEA objects so that replicating our experiments would be easier. From top to bottom, REKO glass, FARGRIK glass, POKAL glass, and SVALKKA red wine glass. (b) Only visible edges were determined from the mesh models. To handle pose variations, the objects were rotated in \mathbf{x} and \mathbf{z} axes.

camera parameters of the monocular camera, which will be used in real experiments. The model is then rendered in **OpenGL** at a fixed depth Z_0 . To identify visible edges, we use the face normal vectors from mesh models as described in Section 3.2.2. As appearances of the models change with respect to rotational variations, multiple templates are obtained as in Figure 4.1 (b). To cover usual shape variations, the objects are rotated in \mathbf{x} and \mathbf{z} axes per 10° and 5° , respectively. Seven steps of rotations are sampled in each axis so that $T = 49$ templates are obtained per object.

4.1.2 Coarse Pose Estimation

With the set of templates generated in Section 4.1.1, the chamfer matching is performed on an input image \mathcal{I}_t . Since the edge templates were obtained with a fixed depth Z_0 , we run the chamfer matching in multiple scale to address different scales of the object in the image depending on the distance between the camera and the object. As a result of the chamfer matching, a set of detection windows are returned. Among the detection windows, we first consider the windows whose detection cost is under a threshold δ_{th} , then the non-maxima suppression is performed to have the lowest cost detection among the overlapped detection. As a result, we have a set of detections \mathcal{D} for $m = 1, \dots, M$:

$$\mathcal{D} = \{x^{(m)}, y^{(m)}, \delta^{(m)}, \mathcal{R}^{(m)}, \sigma^{(m)}\} \quad (4.1)$$

where $x^{(m)}$ and $y^{(m)}$ are the center location of the detected template in the input image, $\delta^{(m)}$ means the cost from the chamfer matching, $\mathcal{R}^{(m)} \in SO(3)$ is the corresponding rotation matrix saved in the template generation, $\sigma^{(m)}$ represents of the scale of the detected edge template, and M is the number of detections. The set of detections is sorted in order of increasing cost $\delta^{(m)}$.

Given \mathcal{D} , a set of coarse poses is estimated. Please note that we can only approximate the current pose from the edge templates, as the edge templates do not cover the entire appearance variations. For this approximation,

Algorithm 4.1: ChamferPose(\mathcal{I}, \mathcal{T})

Data: $\mathcal{I}, \mathcal{T} = \{\mathcal{T}_1, \mathcal{T}_2, \dots, \mathcal{T}_T\}$ **Result:** $\mathcal{S} = \{(\mathbf{X}^{(1)}, \pi^{(1)}), \dots, (\mathbf{X}^{(N)}, \pi^{(N)})\}$ **Params:** $Z_0, u_0, v_0, f_x, f_y, \delta_{th}, \lambda_\delta$

```
1:  $\mathcal{D} \leftarrow \{x^{(\phi)}, y^{(\phi)}, \delta^{(\phi)}, \mathcal{R}^{(\phi)}, \sigma^{(\phi)}\}$  (4.1)
2: for  $t \leftarrow 1$  to  $T$  do
3:   for  $\sigma \leftarrow \sigma_{min}$  to  $\sigma_{max}$  do
4:      $\{x', y', \delta', \mathcal{R}'\} \leftarrow \text{ChamferMatch}(\mathcal{I}, \mathcal{T}_t, \sigma, \delta_{th})$  Liu et al. (2010a)
5:      $\mathcal{D}' \leftarrow \{x', y', \delta', \mathcal{R}'\} \cup \{\sigma\}$ 
6:      $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}'$ 
7:  $\text{Sort}(\mathcal{D})$ 
8:  $M \leftarrow \text{length}(\mathcal{D})$ 
9: for  $m \leftarrow 1$  to  $M$  do
10:   $Z^{(m)} \leftarrow Z_0 / \sigma^{(m)}$  (4.2)
11:   $X^{(m)} \leftarrow (x^{(m)} - u_0)Z^{(m)} / f_x$  (4.3)
12:   $Y^{(m)} \leftarrow (y^{(m)} - v_0)Z^{(m)} / f_y$  (4.4)
13:   $\mathcal{P}^{(m)} \leftarrow \text{CoarsePose}(X^{(m)}, Y^{(m)}, Z^{(m)}, \mathcal{R}^{(m)})$  (4.5)
14: for  $n \leftarrow 1$  to  $N$  do
15:   $\mathbf{X}^{*(n)} \leftarrow \mathcal{P}^{(n \bmod M + 1)}$  (4.6)
16:   $\pi^{*(n)} \leftarrow \exp(-\lambda_\delta \delta^{(n \bmod M + 1)})$  (4.7)
17:  $\pi^* \leftarrow \text{Normalize}(\pi^*)$  (4.8)
18:  $\mathcal{S} \leftarrow \text{Resampling}(\mathcal{S}^*)$ 
```

two assumptions are considered. The first assumption is that although the center location $(x^{(m)}, y^{(m)})$ might be slightly far from the principal point (u_0, v_0) , the rotation matrix can be adopted from one at the principal point. Thus rotation of the object can be determined by $\mathcal{R}^{(m)}$. The second assumption is that the 3D center location of the object can be estimated via similar triangles in the perspective projection. Under this assumption, we can determine the z coordinate of the object $Z^{(m)}$ with respect to the camera by

$$Z^{(m)} = \frac{Z_0}{\sigma^{(m)}}. \quad (4.2)$$

Once $Z^{(m)}$ is determined, it is straightforward to calculate $X^{(m)}$ and $Y^{(m)}$ using similar triangles:

$$X^{(m)} = \frac{(x^{(m)} - u_0)}{f_x} Z^{(m)} \quad (4.3)$$

$$Y^{(m)} = \frac{(y^{(m)} - v_0)}{f_y} Z^{(m)} \quad (4.4)$$

where f_x and f_y are focal length of the camera in \mathbf{x} and \mathbf{y} directions, respectively. If we apply (4.2) to (4.3) and (4.4), we can represent the approximate pose hypothesis $\mathcal{P}^{(m)} \in SE(3)$ of the object with respect to the camera coordinate system as follows

$$\mathcal{P}^{(m)} = \begin{pmatrix} \mathcal{R}^{(m)} & \begin{matrix} \frac{(x^{(m)} - u_0)}{f_x} \frac{Z_0}{\sigma^{(m)}} \\ \frac{(y^{(m)} - v_0)}{f_y} \frac{Z_0}{\sigma^{(m)}} \\ \frac{Z_0}{\sigma^{(m)}} \end{matrix} \\ 0 & 0 & 0 & 1 \end{pmatrix}. \quad (4.5)$$

Algorithm 4.2: ParticleAnnealing(\mathcal{I}, \mathcal{S})

Data: $\mathcal{I}, \mathcal{S} = \{(\mathbf{X}^{(1)}, \pi^{(1)}), \dots, (\mathbf{X}^{(N)}, \pi^{(N)})\}$
Result: $\mathcal{S}_0 = \{(\mathbf{X}_0^{(1)}, \pi_0^{(1)}), \dots, (\mathbf{X}_0^{(N)}, \pi_0^{(N)})\}$
Params: $\alpha = \{\alpha_0, \dots, \alpha_L\}, \beta = \{\beta_0, \dots, \beta_L\}, \Sigma_{w,0}$

- 1: $\mathcal{S}_{L+1} \leftarrow \mathcal{S}$
- 2: **for** $l \leftarrow L$ **to** 0 **do**
- 3: **for** $n \leftarrow 1$ **to** N **do**
- 4: $\mathbf{X}_l^{*(n)} \leftarrow \text{Propagate}(\mathbf{X}_{l+1}^{(n)}, \Sigma_{w,l}, \alpha)$ (4.11)(4.12)
- 5: $\mathbf{Z}^{*(n)} \leftarrow \text{Measurement}(\mathbf{X}_l^{*(n)}, \mathcal{I})$ (3.7)
- 6: $\hat{\mathbf{Z}}^{*(n)} \leftarrow \text{RANSAC}(\mathbf{Z}^{*(n)})$
- 7: $\pi_l^{*(n)} \leftarrow \text{Likelihood}(\hat{\mathbf{Z}}^{*(n)}, \lambda_v, \lambda_e, \beta_l)$ (3.21)(4.10)
- 8: $\pi_l^* \leftarrow \text{Normalize}(\pi_l^*)$ (4.8)
- 9: $\mathcal{S}_l \leftarrow \text{Resampling}(\mathcal{S}_l^*)$

After $\mathcal{P}^{(m)}$ is calculated for all M detection, the N particles and their weights are initialized as

$$\mathbf{X}^{*(n)} = \mathcal{P}^{(n \bmod M+1)} \quad (4.6)$$

$$\pi^{*(n)} = \exp(-\lambda_\delta \delta^{(n \bmod M+1)}) \quad (4.7)$$

where λ_δ is a parameter, which controls the sensitivity to the costs. The weights are normalized via

$$\pi^{*(n)} = \frac{\pi^{*(n)}}{\sum_{i=1}^N \pi^{*(i)}}. \quad (4.8)$$

The particles are then randomly drawn with probability proportional to these weights, and we finally have a set of weighted particles \mathcal{S}_t after the initialization. This pose estimation procedure is presented in Algorithm 4.1 where relevant work and equations are cited in the comments area.

4.1.3 Annealed Particle Filtering

Although our particle filter starts from the most likely pose hypotheses, we cannot always guarantee that the filter converges to the global optimum. Since the sparse edge templates could not cover all possible ranges of pose variations, the errors come from this discrepancy might lead to local optima. Another limitation comes from the low precision of the chamfer matching. In cluttered backgrounds, the chamfer matching may return false positives which lead to poor initial states. Aside from these limitations, it is well known that even if a number of particles are employed, the particle filter might be stuck in local maxima.

To ensure that our particle filter starts near the global maximum, a simulated annealing (Deutscher et al. 2000) is performed after every initialization or re-initialization (Section 3.3.7). The set of weighted particles in (3.8) is augmented with the annealing layer l as

$$\mathcal{S}_{t,l} = \{(\mathbf{X}_{t,l}^{(1)}, \pi_{t,l}^{(1)}), \dots, (\mathbf{X}_{t,l}^{(N)}, \pi_{t,l}^{(N)})\}. \quad (4.9)$$

The annealing starts at layer $l = L$ where L is the number of annealing layers and the weights are determined by

$$\pi_{t,l}^{(n)} \propto p(\mathbf{Z}_t | \mathbf{X}_t^{(n)})^{\beta_l} \quad (4.10)$$

where β_l ($1 = \beta_0 > \beta_1 > \dots > \beta_L$) controls the rate of annealing at each layer. After normalization of the weights, N particles are randomly drawn from $\mathcal{S}_{t,l}$ with the probability of their weights $\pi_{t,l}^{(n)}$. The particles of the next layer $\mathcal{S}_{t,l-1}$ are then propagated as

$$\mathbf{X}_{t,l-1}^{(n)} = \mathbf{X}_{t,l}^{(n)} \cdot \exp(\mathbf{dW}_{t,l} \sqrt{\Delta t}) \quad (4.11)$$

where $\mathbf{dW}_{t,l}$ is the Wiener process noise with covariance $\Sigma_{w,l}$. This annealing process is iterated until it arrives at $\mathbf{X}_{t,0}^{(n)}$. In [Deutscher et al. \(2000\)](#), the $\Sigma_{w,l}$ was defined by

$$\Sigma_{w,l} = \Sigma_{w,0}(\alpha_L \alpha_{L-1} \dots \alpha_l) \quad (4.12)$$

where α_l represents the particle survival rate which is equivalent to \widehat{N}_{eff}/N in (3.25). They argued that $\alpha_0 = \alpha_1 = \dots = \alpha_L = 0.5$ provide sufficient results. In the parameter β_l , one can determine to adjust an initial rate of α_{init} to α_l using a gradient descent method ([Deutscher et al. 2000](#)). As a simple alternative, we empirically found $\beta_l = (0.5)^l$ shows good performance as well. The annealing algorithm is shown in [Algorithm 4.2](#).

4.1.4 Symmetric Objects

Some of our objects ([Figure 4.1](#)) are symmetrical so that rotation about the axis of symmetry, y -axis in our object models, cannot be uniquely determined. It is problematic when our particle filter searches the 6D pose space because it may result in a ridge posterior distribution. Thus, it is more efficient to search a 5D pose space instead of the full 6D space. This can be easily modified through our Lie group formulation. Recall that $\mathfrak{se}(3)$ has 6 basis elements as shown in (3.6), and exponentiating the term of the fifth basis \mathbf{E}_5 results in the rotation about y -axis in $SE(3)$:

$$\exp(\gamma \mathbf{E}_5) = \begin{pmatrix} \cos \gamma & 0 & \sin \gamma & 0 \\ 0 & 1 & 0 & 0 \\ -\sin \gamma & 0 & \cos \gamma & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}. \quad (4.13)$$

Therefore, it is possible to suppress rotating motion about the axis of symmetry by setting 0 in the fifth coefficient for \mathbf{E}_5 corresponding to the term $\mathbf{A}(\mathbf{X}, t)$ and \mathbf{dW}_t in (3.5), $\mathbf{dW}_{t,l}$ in (4.11), and $\boldsymbol{\mu}$ in (3.23).

4.2 Experimental Results

In this section, we validate our proposed solution via several experiments. REKO, SVALKA, POKAL glasses were chosen from the KIT ObjectModels Web Database ([Kasper et al. 2012](#)) in which more than 100 object models of household items are provided in 3D polygonal meshes and stereo images, and FARGRIK glass model obtained from Google 3D warehouse ([Google 2013](#)). We only use the provided mesh models in our experiments which were scanned via a high-accuracy laser scanner. The mesh models are represented in [Figure 4.1 \(a\)](#). We only use the provided mesh models in our experiments which are shown in [Figure 4.1 \(a\)](#). From these models, we prepare 49 edge templates per each object offline as shown in [Figure 4.1 \(b\)](#). These templates are used in the chamfer matching to initialize particles. To obtain test image sequences, a calibrated monocular camera was placed around the target objects, and the camera was moved so that the captured image sequences show significant variations in translation, rotation, and velocity.

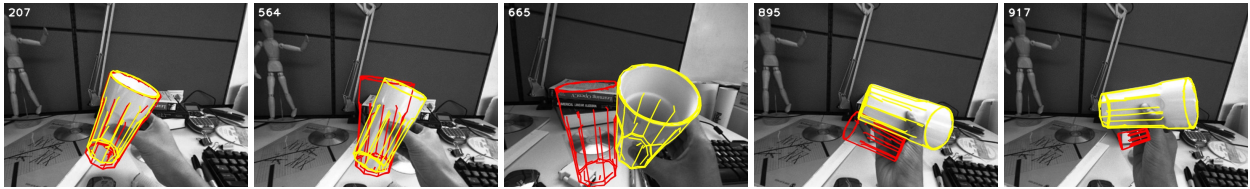


Figure 4.2. Tracking results showing effectiveness of considering multiple hypotheses. Results with 100 particles (yellow wireframe) and 1 particle (red wireframe) are shown in the sequence of the POKAL glass. Note that the yellow wireframe is well localized by calculating the mean of multiple hypotheses, while the red wireframe is drifted during entire tracking. The frame number is shown in the top left corner of each image.

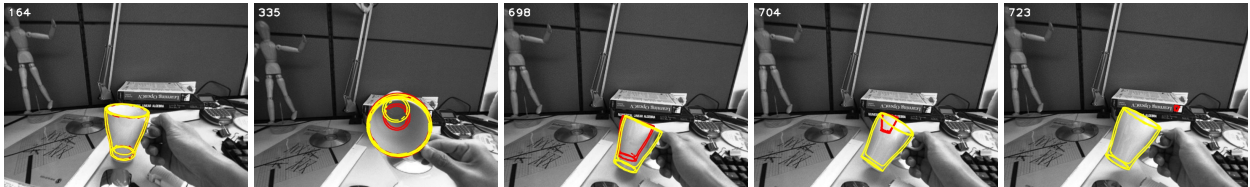


Figure 4.3. Tracking results showing effectiveness of suppressing the rotating motion about the axis of symmetry. Results with (yellow wireframe) and without (red wireframe) the suppression are shown in the sequence of the FARGRIK glass. Although they use the same number of particles and the parameters, the red wireframe starts to drift before the frame number 698 due to larger search space.

To validate our particle filtering approach, we first executed our system on the sequence of the POKAL glass with 1 and 100 particles. For fair comparison, we set the same parameters except the number of particles. In Figure 4.2, results of the system using 1 and 100 particles are depicted in red and yellow wireframes, respectively. While the red wireframes are suffered from drifting, the yellow ones are well fitted to the object. Since the particle filter considers multiple hypotheses, it is not stuck in local optima.

To verify the effectiveness of suppressing the rotating motion discussed in Section 4.1.4, the proposed approach was executed with and without the suppression. For fair comparison, the suppression was only altered. The tracking results are presented in Figure 4.3. The tracking difference is possibly due to different search spaces. With the same number of particles ($N = 100$), the suppressed version only searches for the global optimum in the 5D space, while the version without the suppression fails to find the optimum in the 6D space.

We prove the effectiveness of annealed particle filtering by turning on and off the annealing stage after the initialization. The experiment was executed with the same parameters except the annealing. The comparison of the two tracking results are presented in Figure 4.4. It is clear that employing the annealing process helps the tracker to start from the global optimum.

As monitoring the effective number of particles \widehat{N}_{eff} , the proposed system can re-initialize by itself when it is required. To verify this capability, we tested on a challenging image sequence, in which the object is often disappeared because of the camera motion (Figure 4.5). When these cases are occurred, \widehat{N}_{eff} falls significantly. Thus our system re-initializes the tracking and successfully recovers from the failure cases.

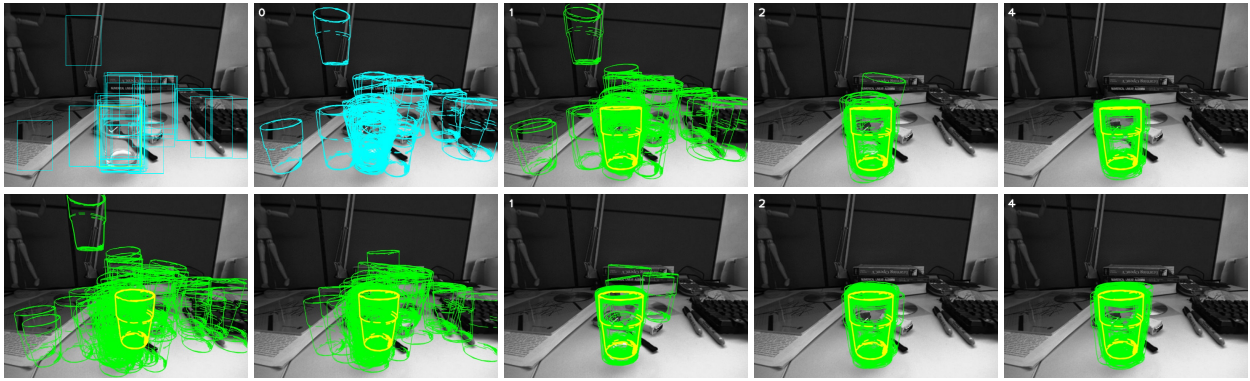


Figure 4.4. Initialization and annealed particle filtering. The top-left image shows the chamfer matching results which are depicted in cyan bounding boxes, except that the lowest cost window (*i.e.* best result) is drawn in the yellow box. The next image shows initial states in cyan wireframes determined by our algorithm. The upper and lower rows of the center to right columns present results without and with the annealing, respectively. Intermediate annealing results are shown in the bottom-left two images (annealing layer l is 4 and 2 from total $L = 5$ layers). The particle filter without annealing is often stuck in local optima, and thus it could not recover to the global optimum, while our annealed particle filter can converge.

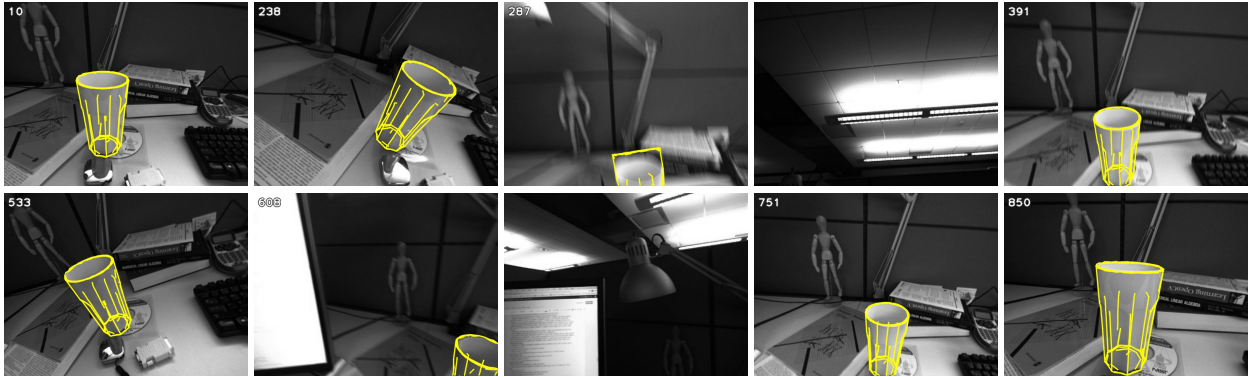


Figure 4.5. Tracking results showing the re-initialization capability. Based on the value of \widehat{N}_{eff} , our system can re-initialize when the object goes out of the field of view.

4.3 Summary

We presented a particle filtering approach using edge features for the textureless object detection and tracking. Our approach started with possible pose hypotheses via the chamfer matching followed by the coarse pose estimation. The initial poses were further refined through the annealed particle filtering to ensure they are close to the global maximum. The proposed approach was qualitatively validated in various experiments.

CHAPTER V

VOTING-BASED POSE ESTIMATION (3D)

In this chapter, we describe our object pose estimation algorithm for two classes of objects. The first object class is daily objects, which possess rich variations in color, texture, and shape. For this object class, we describe our approach exploiting both geometric (depth) and photometric (color) features in Section 5.1. The second object class is industrial parts, which lack sufficient color or texture variations. In Section 5.6, we delineate our approach for industrial objects relying on geometric features (boundaries from depth discontinuities and depth points). We show experimental results for daily and industrial objects in Section 5.4 and 5.9 respectively.

5.1 Exploiting Object Color Information

In this section, we present an object pose estimation approach exploiting both depth and color information available from an RGB-D sensor. There have been various attempts to solve the pose recognition problem, yet most of them have relied on an object segmentation which removes the background and clusters the remained points. This approach might be effective with a known background structure, but it is not promising when the background is cluttered or has an unknown geometric shape. In these cases of unstructured environments, it is necessary to hypothesize all possible transformations from an object model to a given scene. Recently, a voting approach, using 4D oriented point pair features, was introduced to find a set of 3D rigid transformations between the model and the test scene. We found that exploiting color information significantly enhances the performance of the voting process in terms of both time and accuracy. To exploit the color information, we define a color point pair feature (CPPF), which is utilized in the voting scheme for effective and robust pose computation. We further enhance the computational performance by parallelizing in a modern computational architecture. The proposed approach is extensively evaluated with four state-of-the-art approaches on both synthetic and real datasets.

5.2 Contributions

Our main contributions are as follows:

- We utilize color information available from the RGB-D sensor. Our color augmented point pair feature enables the voting scheme to be more efficient by reducing the number of duplicated features in the object learning phase. Thanks to the color information in CPPFs, a large number of potentially false feature matches are skipped, and this makes our approach more robust to background clutter.
- Our algorithm is significantly parallelized on a GPU. Although our CPPF-based pose estimation is more efficient than the approach without color information (Drost et al. 2010), the speed is still prohibited for the

Algorithm 5.1: RGB-D Pose Estimation

Data: $\mathcal{M} = \{(\mathbf{p}_1^m, \mathbf{n}_1^m, \mathbf{c}_1^m), \dots, (\mathbf{p}_{N_m}^m, \mathbf{n}_{N_m}^m, \mathbf{c}_{N_m}^m)\}, \mathcal{S} = \{(\mathbf{p}_1^s, \mathbf{n}_1^s, \mathbf{c}_1^s), \dots, (\mathbf{p}_{N_s}^s, \mathbf{n}_{N_s}^s, \mathbf{c}_{N_s}^s)\}$

Result: $\mathcal{P} = \{(\mathbf{P}_1, v_1), (\mathbf{P}_2, v_2), \dots, (\mathbf{P}_{\widehat{N}_p}, v_{\widehat{N}_p})\}$

Params: $N_v, \gamma_s, \delta, \theta, \sigma, N_c$

1: $\{\mathcal{A}, \mathcal{I}_o^s, \mathcal{H}_r, \mathcal{D}_h, \mathcal{I}_s^r\} \leftarrow \text{ObjectLearning}(\mathcal{M})$ (5.2)

2: $\{\mathcal{V}, i_v\} \leftarrow \text{Voting}(\mathcal{M}, \mathcal{S}, \mathcal{A}, \mathcal{I}_o^s, \mathcal{H}_r, \mathcal{D}_h, \mathcal{I}_s^r)$ (5.3)

3: $\mathcal{P} \leftarrow \text{ComputePoses}(\mathcal{M}, \mathcal{S}, \mathcal{V}, i_v)$ (5.4)

4: $\mathcal{P} \leftarrow \text{ClusterPoses}(\mathcal{P})$ (5.5)

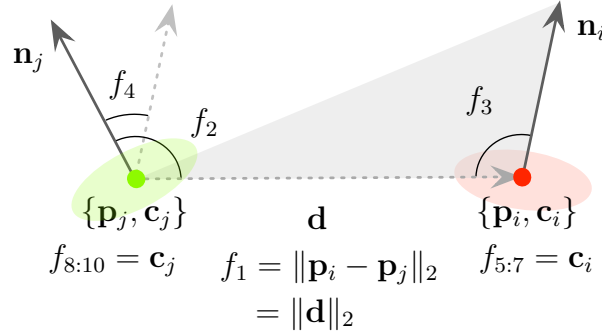


Figure 5.1. The Color Point Pair Feature (CPPF). The feature is defined by the distance (f_1), relative orientations (f_2, f_3 , and f_4), and colors ($f_{5:7}$ and $f_{8:10}$) of the pair of color oriented points $\{(\mathbf{p}_i, \mathbf{n}_i, \mathbf{c}_i), (\mathbf{p}_j, \mathbf{n}_j, \mathbf{c}_j)\}$.

general robotic manipulation scenarios. By exploiting the parallel power of an off-the-shelf GPU, our algorithm processes an RGB-D scene in several hundred milliseconds.

- We present an extensive dataset for the RGB-D pose estimation problem with ground truth pose information. The dataset is composed of synthetic and real RGB-D scenes along with ten object mesh models, and it is designed for various evaluations, such as noise, multiple objects single instance, single object multiple instances, and highly cluttered scenes.

5.3 RGB-D Pose Estimation

In this section, our RGB-D pose estimation algorithm is presented. By revisiting the point pair features of Drost et al. (2010) in Section 5.3.1, we introduce our color point pair feature in Section 5.3.2. In Section 5.3.3, the parallel object learning algorithm is presented, and the voting process and pose clustering are explained in Section 5.3.4 and 5.3.5, respectively. The overall algorithm is shown in Algorithm 5.1 which takes an object model point cloud \mathcal{M} and an scene point cloud \mathcal{S} as inputs and returns a set of pose hypotheses \mathcal{P} as an output. The more details of the parameters and the algorithms, referred as $\langle \cdot \rangle$ in the comments area, will be described in the subsequent sections.

5.3.1 Point Pair Feature

The point pair feature (PPF) is defined by paring two oriented points, and it has been employed in shape-based object recognition (Wahl et al. 2003, Drost et al. 2010). Let $\{(\mathbf{p}_i, \mathbf{n}_i), (\mathbf{p}_j, \mathbf{n}_j)\}$ denote the pair of two oriented points where \mathbf{p}_i and $\mathbf{p}_j \in \mathbb{R}^3$ are the *reference* and *referred* points on the object surface, and \mathbf{n}_i and $\mathbf{n}_j \in \mathbb{R}^3$ are their normals respectively. The point pair feature vector $\mathbf{F}_{\text{PPF}} \in \mathbb{R}^4$ is then defined as

$$\begin{aligned} \mathbf{F}_{\text{PPF}} &= \text{PPF}(\mathbf{p}_i, \mathbf{p}_j, \mathbf{n}_i, \mathbf{n}_j) \\ &= \begin{pmatrix} \|\mathbf{d}\|_2 \\ \angle(\mathbf{n}_i, \mathbf{d}) \\ \angle(\mathbf{n}_j, \mathbf{d}) \\ \angle(\mathbf{n}_i, \mathbf{n}_j) \end{pmatrix} \end{aligned} \quad (5.1)$$

where $\mathbf{d} = \mathbf{p}_i - \mathbf{p}_j$, and $\angle(\mathbf{v}_1, \mathbf{v}_2) \in [0, \pi)$ represents the angle between two vectors, \mathbf{v}_1 and $\mathbf{v}_2 \in \mathbb{R}^3$. The first dimension, $\|\mathbf{d}\|_2 = \|\mathbf{p}_i - \mathbf{p}_j\|_2$, represents the Euclidean distance between the two points. The second and third components are angles between the vector \mathbf{d} and the surface normal vectors \mathbf{n}_i and \mathbf{n}_j , respectively. The last component is the angle between the two normal vectors. This feature effectively encodes geometric constraints of point cloud surfaces so that efficient matching between model and scene point clouds is possible, especially when these point clouds contain rich surface normal variations.

5.3.2 Color Point Pair Feature

Even though the PPF might be suitable for objects having rich variations in surface normals, it is generally not discriminative enough to describe planar or self-symmetric objects. Hence it is required to augment the pair feature so that the feature will be more effective to these types of objects. The color point pair feature (CPPF) vector $\mathbf{F}_{\text{CPPF}} \in \mathbb{R}^{10}$ is defined by concatenating two 3D color vectors of the points:

$$\begin{aligned} \mathbf{F}_{\text{CPPF}} &= \text{CPPF}(\mathbf{p}_i, \mathbf{p}_j, \mathbf{n}_i, \mathbf{n}_j, \mathbf{c}_i, \mathbf{c}_j) \\ &= \begin{pmatrix} \text{PPF}(\mathbf{p}_i, \mathbf{p}_j, \mathbf{n}_i, \mathbf{n}_j) \\ \mathbf{c}_i \\ \mathbf{c}_j \end{pmatrix} \end{aligned} \quad (5.2)$$

where \mathbf{c}_i and $\mathbf{c}_j \in \mathbb{R}^3$ are color vectors. For generality, each color channel is normalized as $c \in [0, 1]$. Figure 5.1 illustrates the CPPF. When it comes to using color information, it is crucial to choose a proper color space to be less variable to illumination changes. The original color information from the RGB-D sensor is in the RGB color space. However, the RGB values tend to change much as illumination intensity varies. As an alternative, HSV (Hue, Saturation, and Value) color space was well studied and proven to be more invariant than the RGB space with respect to illumination changes. Thus we adopt the HSV color space in our experiments.

Algorithm 5.2: ObjectLearning(\mathcal{M})

Data: \mathcal{M}
Result: $\mathcal{A}, \mathcal{I}_o^s, \mathcal{H}_r, \mathcal{D}_h, \mathcal{I}_s^r$
Params: δ, θ, σ

- 1: $\mathcal{H} \leftarrow \mathbf{0}_{N_m \times N_m}$
- 2: $\mathcal{A} \leftarrow \mathbf{0}_{N_m \times N_m}$
- 3: **for** $i \leftarrow 1$ **to** N_m **do**
- 4: **for** $j \leftarrow 1$ **to** N_m **do**
- 5: **if** $i \neq j$ **then**
- 6: $\mathbf{F} \leftarrow \text{CPPF}(\mathbf{p}_i^m, \mathbf{p}_j^m, \mathbf{n}_i^m, \mathbf{n}_j^m, \mathbf{c}_i^m, \mathbf{c}_j^m)$ (5.2)
- 7: $\mathbf{k} \leftarrow \text{HashKey}(\mathbf{F}, \delta, \theta, \sigma)$ (5.3)
- 8: $\kappa \leftarrow \text{BitEncodeCPPF}(\mathbf{k})$ (5.4)
- 9: $\alpha_m \leftarrow \text{PlanarRotAngle}(\mathbf{n}_i, \mathbf{p}_i^m, \mathbf{p}_j^m)$ (5.3.4)
- 10: $\mathcal{H}(i, j) \leftarrow \kappa$
- 11: $\mathcal{A}(i, j) \leftarrow \alpha_m$
- 12: $\{\mathcal{H}_s, \mathcal{I}_o^s\} \leftarrow \text{gpu}::\text{Sort}(\mathcal{H})$
- 13: $\{\mathcal{H}_r, \mathcal{D}_h\} \leftarrow \text{gpu}::\text{Reduce}(\mathcal{H}_s)$
- 14: $\mathcal{I}_s^r \leftarrow \text{gpu}::\text{ExclusiveScan}(\mathcal{D}_h)$

5.3.3 Object Learning

In the object learning phase, an object representation is learned globally by calculating all possible CPPFs from an object point cloud. When there are N_m points in the point cloud \mathcal{M} , $N_m \times N_m$ CPPFs are calculated including N_m self pairs. Once the set of CPPF features are calculated, it is saved in a data structure for the later feature matching. Hash tables have been widely adopted for the purpose due to its fast search time (Mian et al. 2006, Drost et al. 2010, Kim and Medioni 2011, Choi et al. 2012, Papazov et al. 2012). To use the CPPF as the key for hash table, we first need to quantize the feature vector as

$$\begin{aligned} \mathbf{k} &= \text{HashKey}(\mathbf{F}_{\text{CPPF}}, \delta, \theta, \sigma) \\ &= \left(\lfloor \frac{\|\mathbf{d}\|_2}{\delta} \rfloor, \lfloor \frac{\angle(\mathbf{n}_i, \mathbf{d})}{\theta} \rfloor, \lfloor \frac{\angle(\mathbf{n}_j, \mathbf{d})}{\theta} \rfloor, \lfloor \frac{\angle(\mathbf{n}_i, \mathbf{n}_j)}{\theta} \rfloor, \lfloor \mathbf{c}_i \oslash \sigma \rfloor^\top, \lfloor \mathbf{c}_j \oslash \sigma \rfloor^\top \right)^\top \end{aligned} \quad (5.3)$$

where $\delta \in \mathbb{R}$, $\theta \in \mathbb{R}$, $\sigma \in \mathbb{R}^3$ are quantization levels for distance, angle, and color vectors, respectively. The symbol \oslash denotes component-wise division. A key of the CPPF, $\mathbf{k} \in \mathbb{Z}^{10}$, is then bit-encoded as

$$\kappa = \text{BitEncodeCPPF}(\mathbf{k}) \quad (5.4)$$

where the 64-bit key $\kappa \in \mathbb{Z}$ is encoded as in Figure 5.3.

Although the hash table search is designed for constant time search $O(1)$, the performance of the hash tables highly depends on the choice of hash function. Even though we carefully design the hash function, lots of point pairs are inserted in a relatively small number of hash slots due to the symmetric regions of objects. As a result, each hash search does not guarantee the optimal search time. In our approach, we employ the modern parallel computing architecture which is called GPU. For computational acceleration on the GPU, it is required to maintain the data in a simple structure so that a large number of threads can efficiently process the data in parallel. In this respect, we

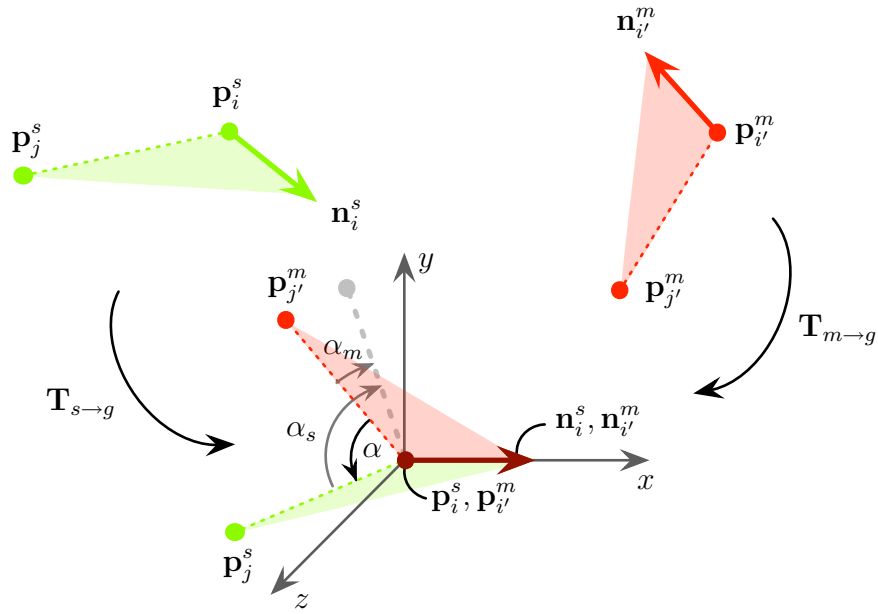


Figure 5.2. Aligning pair features in the intermediate coordinate system. By $\mathbf{T}_{s \rightarrow g}$, the scene reference point \mathbf{p}_i^s is moved to the origin and its orientation (normal or direction) \mathbf{n}_i^s is aligned to the \mathbf{x} -axis. The model reference point is similarly transformed by $\mathbf{T}_{m \rightarrow g}$ so that the positions and orientations of the reference points are aligned. The referred points \mathbf{p}_j^s and $\mathbf{p}_{j'}^m$ are then aligned by a rotation with angle α around the \mathbf{x} -axis.

maintain the set of keys in an array storage and use parallel operations for faster search later.

The object learning process is presented in Algorithm 5.2 where referred equations and sections are marked as (\cdot) and (\S) in the comments area, respectively. Given an object model point cloud \mathcal{M} , the algorithm returns the reduced hash keys \mathcal{H}_r with the intermediate angle array \mathcal{A} and other data required for the later voting process. The N_m designates the number of points in \mathcal{M} , and the α_m is the intermediate angle stored in \mathcal{A} that will be explained in Section 5.3.4. The for-loop in line 3 is parallelized so that each thread takes care of each (i, j) pair. Once the keys are calculated and saved in \mathcal{H} , they are first sorted (\mathcal{H}_s) and then reduced so that duplicated keys are removed. By reducing, the sorted keys \mathcal{H}_s are shrunk to the reduced keys \mathcal{H}_r with the array of duplication numbers \mathcal{D}_h . The \mathcal{D}_h is further utilized to come up with the indices from the reduced array to the sorted array \mathcal{I}_s^r . Similarly, the \mathcal{I}_o^s has the indices from the sorted array to the original array. These two arrays of indices are necessary to map from the reduced array \mathcal{H}_r to the original array \mathcal{H} for the voting process. The parallel primitive operations, such as sort, reduce, and exclusive scan, are readily available via GPU C++ template libraries, such as NVIDIA Thrust library (Hoberock and Bell 2010) or AMD Bolt library (AMD 2012). Since all data resides in GPU, we can easily call these parallel operations on GPU and process the massive data very efficiently.

The quantization parameters δ, θ, σ are important to set. While choosing very large levels reduce the discriminative power of the feature, using very small levels make the algorithm sensitive to noise. For the color quantization levels σ , we use the HSV color space. The V (value or intensity) channel is not generally invariant to illumination changes, and hence a relatively larger level is used. In our experiment, we empirically found these values and more details will be described in Section 5.4.

5.3.4 Voting Scheme

Let's assume that we found a correct match of CPPFs between scene and model point clouds. As described in Figure 5.2, we can align two normal vectors $\{\mathbf{n}_i^s, \mathbf{n}_{i'}^m\}$ of the two reference points $\{\mathbf{p}_i^s, \mathbf{p}_{i'}^m\}$ in an intermediate coordinate system. The alignment of two reference points constrains 3-DOF translation and the alignment of the two normals further constrains 2-DOF rotation. Therefore, there is only 1-DOF rotation ambiguity $\alpha \in \mathbb{R}$ around the \mathbf{x} -axis of the intermediate coordinate system. Once the α is determined by the two vectors $\mathbf{p}_j^s - \mathbf{p}_i^s$ and $\mathbf{p}_{j'}^m - \mathbf{p}_{i'}^m$, we can recover the pose of the object, $\mathbf{P} \in SE(3)$, which is the full 6-DOF rigid body transformation from the model coordinate system to the scene coordinate system via

$$\begin{aligned} \mathbf{P} &= \mathbf{T}_{m \rightarrow s} \\ &= \mathbf{T}_{s \rightarrow g}^{-1} \mathbf{R}_{\mathbf{x}}(\alpha) \mathbf{T}_{m \rightarrow g} \end{aligned} \quad (5.5)$$

where $\mathbf{R}_{\mathbf{x}}(\alpha) \in SO(3)$ is the rotation around the \mathbf{x} -axis with angle α , $\mathbf{T}_{s \rightarrow g} \in SE(3)$ and $\mathbf{T}_{m \rightarrow g} \in SE(3)$ are the transformations from the scene and model coordinate systems to the intermediate coordinate system, respectively. For a quick verification, the referred points $\{\mathbf{p}_j^s, \mathbf{p}_{j'}^m\}$ can be aligned by \mathbf{P} as

$$\begin{aligned} \mathbf{p}_{j'}^s &= \mathbf{P} \mathbf{p}_{j'}^m \\ &= \mathbf{T}_{s \rightarrow g}^{-1} \mathbf{R}_{\mathbf{x}}(\alpha) \mathbf{T}_{m \rightarrow g} \mathbf{p}_{j'}^m. \end{aligned}$$

It is possible to choose any arbitrary intermediate coordinate system, but a trivial choice is choosing the sensor coordinate system.

Unfortunately, the aforementioned assumption of a correct correspondence between two CPPFs is not always valid. In reality, there is a nontrivial amount of similar geometric and colored surfaces between the actual object and cluttered background point clouds. Due in part to sensor noise and to illumination changes, it happens that these similar regions result in incorrect pose hypotheses. To address this issue, a voting process is performed so that it finds the most likely pose hypothesis from the bin earned the maximum number of votes (Drost et al. 2010). The usual approach is employing two dimensional accumulator space for the voting process, in which the rows are the model reference points and the columns are discretized bins of α (Drost et al. 2010, Choi et al. 2012). Though it is possible to allocate a big memory space in CPU memory, it is technically impossible to assign the big voting space to each thread in GPU. As a workaround, a big chunk of global memory \mathcal{V} , where the maximum size is N_v , is allocated and every votes are accumulated in \mathcal{V} . Algorithm 5.3 carefully describes the voting process. Given the model point cloud \mathcal{M} , scene point cloud \mathcal{S} , and other data obtained from Algorithm 5.2, each reference scene point \mathbf{p}_i^s is paired with the other scene points \mathbf{p}_j^s . When the Euclidean distance between these two points are within the given search radius γ_s , the CPPF from the two points is calculated and is searched in the model keys \mathcal{H}_r via the binary search. Since \mathcal{H}_r is already sorted and reduced, the binary search is done in $O(\log n)$ where n is the number of reduced keys. If the corresponding key is found, the number of duplicated keys N_d is looked up via \mathcal{D}_h and the model point index

Algorithm 5.3: Voting($\mathcal{M}, \mathcal{S}, \mathcal{A}, \mathcal{I}_o^s, \mathcal{H}_r, \mathcal{D}_h, \mathcal{I}_s^r$)

Data: $\mathcal{M}, \mathcal{S}, \mathcal{A}, \mathcal{I}_o^s, \mathcal{H}_r, \mathcal{I}_s^r, \mathcal{D}_h$ **Result:** \mathcal{V}, i_v **Params:** $N_v, \gamma_s, \delta, \theta, \sigma$

```
1:  $\mathcal{V} \leftarrow \mathbf{0}_{1 \times N_v}$ 
2:  $i_v \leftarrow 0$ 
3: for  $i \leftarrow 1$  to  $N_s$  do
4:   for  $j \leftarrow 1$  to  $N_s$  do
5:     if  $i \neq j$  and  $\|\mathbf{p}_i^s - \mathbf{p}_j^s\| < \gamma_s$  then
6:        $\mathbf{F} \leftarrow \text{CPPF}(\mathbf{p}_i^s, \mathbf{p}_j^s, \mathbf{n}_i^s, \mathbf{n}_j^s, \mathbf{c}_i^s, \mathbf{c}_j^s)$  (5.2)
7:        $\kappa \leftarrow \text{HashKey}(\mathbf{F}, \delta, \theta, \sigma)$  (5.3)
8:        $\alpha_s \leftarrow \text{PlanarRotAngle}(\mathbf{n}_i^s, \mathbf{p}_i^s, \mathbf{p}_j^s)$  (5.3.4)
9:        $i_r \leftarrow \text{BinarySearch}(\mathcal{H}_r, \kappa)$ 
10:      if  $i_r \neq \{\phi\}$  then
11:         $N_d \leftarrow \mathcal{D}_h(i_r)$ 
12:         $i_s \leftarrow \mathcal{I}_s^r(i_r)$ 
13:        for  $d \leftarrow 1$  to  $N_d$  do
14:           $i_o \leftarrow \mathcal{I}_o^s(i_s + d)$ 
15:           $\alpha_m \leftarrow \mathcal{A}(i_o)$ 
16:           $\alpha \leftarrow \alpha_m - \alpha_s$  (5.7)
17:           $\hat{i}_v \leftarrow \text{AtomicAdd}(i_v, 1)$ 
18:           $\mathcal{V}(\hat{i}_v) \leftarrow \text{BitEncodeVote}(i, \frac{i_o}{N_m}, \lfloor \frac{\alpha}{\theta} \rfloor)$  (5.6)
```

of the sorted array i_s is found from \mathcal{I}_s^r . Finally, the original index in the original key array i_o is determined from \mathcal{I}_o^s , and this index is used to refer the pre-calculated intermediate angle from \mathcal{A} . The set $\{i, \frac{i_o}{N_m}, \lfloor \frac{\alpha}{\theta} \rfloor\}$ comprises a vote for a pose hypothesis, and for the later computation each vote is bit-encoded as

$$\nu = \text{BitEncodeVote}(i, \frac{i_o}{N_m}, \lfloor \frac{\alpha}{\theta} \rfloor) \quad (5.6)$$

where the 64-bit vote $\nu \in \mathbb{Z}$ is encoded as in Figure 5.4, i and $\frac{i_o}{N_m}$ are indices of scene and model reference points, respectively, and the last term $\lfloor \frac{\alpha}{\theta} \rfloor$ is the discretized angle of α . To avoid race condition in the array \mathcal{V} , the atomic add operation is required. Note that the α could be calculated online, but it is more efficient if α_m , the angle between the vector $\mathbf{p}_{j'}^m - \mathbf{p}_{i'}^m$ and the upper **xy** half-plane, is pre-calculated and saved in \mathcal{A} (Drost et al. 2010). Then all α for every corresponding model features can be determined by one calculation of α_s and minus operations as

$$\alpha = \alpha_m - \alpha_s. \quad (5.7)$$

The set of votes \mathcal{V} is then used to compute the final pose hypotheses in Algorithm 5.4 which shows how a set of pose hypotheses is obtained. Given the votes array \mathcal{V} and its associated size i_v , it calculates a set of poses and vote numbers \mathcal{P} , in which each element is the pair of a pose hypothesis $\mathbf{P}_i \in SE(3)$ and its number of votes $v_i \in \mathbb{Z}$. As in Algorithm 5.2, \mathcal{V} is sorted and then reduced to avoid redundant calculations on the same votes. In this process, duplicated elements in \mathcal{V} are removed and the number of duplication is updated in \mathcal{D}_v . Inside the for-loop, each unique vote element is bit-decoded so that the index of scene reference point i_s (i in Figure 5.4), the index of model reference point i_m ($\frac{i_o}{N_m}$ in Figure 5.4), and the index of alpha i_α ($\lfloor \frac{\alpha}{\theta} \rfloor$ in Figure 5.4) can be obtained. From the indices,

Algorithm 5.4: ComputePoses($\mathcal{M}, \mathcal{S}, \mathcal{V}, i_v$)

Data: $\mathcal{M}, \mathcal{S}, \mathcal{V}, i_v$ **Result:** $\mathcal{P} = \{(\mathbf{P}_1, v_1), (\mathbf{P}_2, v_2), \dots, (\mathbf{P}_{N_p}, v_{N_p})\}$ **Params:** δ, θ, σ

- 1: $\mathcal{V}_s \leftarrow \text{gpu}::\text{Sort}(\mathcal{V}, i_v)$
 - 2: $\{\mathcal{V}_r, \mathcal{D}_v\} \leftarrow \text{gpu}::\text{Reduce}(\mathcal{V}_s)$
 - 3: **for** $i \leftarrow 1$ **to** $\text{Length}(\mathcal{V}_r)$ **do**
 - 4: $v \leftarrow \mathcal{V}_r(i)$
 - 5: $i_s \leftarrow \text{ScenePointIndex}(v)$
 - 6: $i_m \leftarrow \text{ModelPointIndex}(v)$
 - 7: $i_\alpha \leftarrow \text{AlphaIndex}(v)$
 - 8: $\mathbf{T}_{s \rightarrow g} \leftarrow \text{InterTransform}(\mathbf{p}_{i_s}^s, \mathbf{n}_{i_s}^s)$
 - 9: $\mathbf{T}_{m \rightarrow g} \leftarrow \text{InterTransform}(\mathbf{p}_{i_m}^m, \mathbf{n}_{i_m}^m)$
 - 10: $\mathcal{P} \leftarrow \mathcal{P} \cup \{\text{GetPose}(\mathbf{T}_{s \rightarrow g}, \mathbf{T}_{m \rightarrow g}, i_\alpha), \mathcal{D}_v(i)\}$
- (5.5)
-

Elements	$\lfloor \frac{c_j(3)}{\sigma(3)} \rfloor$	$\lfloor \frac{c_j(2)}{\sigma(2)} \rfloor$	$\lfloor \frac{c_j(1)}{\sigma(1)} \rfloor$	$\lfloor \frac{c_i(3)}{\sigma(3)} \rfloor$	$\lfloor \frac{c_i(2)}{\sigma(2)} \rfloor$	$\lfloor \frac{c_i(1)}{\sigma(1)} \rfloor$	$\lfloor \frac{\angle(\mathbf{n}_i, \mathbf{n}_j)}{\theta} \rfloor$	$\lfloor \frac{\angle(\mathbf{n}_j, \mathbf{d})}{\theta} \rfloor$	$\lfloor \frac{\angle(\mathbf{n}_i, \mathbf{d})}{\theta} \rfloor$	$\lfloor \frac{\ \mathbf{d}\ _2}{\delta} \rfloor$	
Bits	63	59 58	54 53	49 48	44 43	39 38	34 33	28 27	22 21	16 15	0

Figure 5.3. Bit-encoded 64-bit CPPF key. The quantized CPPF descriptor \mathbf{k} in Equation (5.3) is bit-encoded so that the discretized distance $\lfloor \frac{\|\mathbf{d}\|_2}{\delta} \rfloor$, discretized angles $\lfloor \frac{\angle(\mathbf{n}_i, \mathbf{d})}{\theta} \rfloor$, $\lfloor \frac{\angle(\mathbf{n}_j, \mathbf{d})}{\theta} \rfloor$, $\lfloor \frac{\angle(\mathbf{n}_i, \mathbf{n}_j)}{\theta} \rfloor$, and the discretized color values $\lfloor \mathbf{c}_i \oslash \sigma \rfloor$, $\lfloor \mathbf{c}_j \oslash \sigma \rfloor$ are stored in a 64-bit key.

Elements	i	$\frac{i_m}{N_m}$	$\lfloor \frac{\alpha}{\theta} \rfloor$	
Bits	63	32 31	6 5	0

Figure 5.4. Bit-encoded 64-bit vote. Each vote is composed of three values: the index of scene reference point i , the index of model reference point $\frac{i_m}{N_m}$, and the quantized α angle $\lfloor \frac{\alpha}{\theta} \rfloor$. These values are encoded in a 64-bit vote, and a set of votes is further processed to result in a set of pose hypotheses.

the intermediate transforms $\mathbf{T}_{s \rightarrow g}$ and $\mathbf{T}_{m \rightarrow g}$ are recovered, and finally the pose estimate is calculated via Equation (5.5). The sorting and reducing are performed via the parallel library, and the for-loop is parallelized on GPU.

5.3.5 Pose Clustering

In Algorithm 5.4, it is important to note that each pair of pose and vote number in \mathcal{P} is calculated from the pair of i_s -th scene point and i_m -th model point. Since the object has N_m model points, many pairs in \mathcal{P} should represent a consistent pose hypothesis. For instance, if a half of N_m points are visible in the given scene, then the ideal number of pairs in \mathcal{P} should be $\frac{N_m}{2}$. But, in reality, the number tends to be much more or less due mainly to scene noise, the occlusion ratio of the object, false matches from clutter, and multiple instances of the same object. To take into account these cases, we need to aggregate similar pose hypotheses. Since advanced clustering methods such as mean shift (Tuzel et al. 2005, Pham et al. 2011) are computationally expensive, we employ an efficient agglomerative clustering. While Drost et al. (2010), Kim and Medioni (2011), Choi et al. (2012) have done the similar clustering, we further enhance this process on GPU so that hundreds of thousands elements in \mathcal{P} can be clustered in parallel.

The clustering process on GPU is shown in Algorithm 5.5. It takes unclustered pose hypotheses \mathcal{P} as an input, and it sorts \mathcal{P} in descending order of the number of votes v_i to make sure that the elements are grouped together to the several most likely pose hypotheses. The algorithm aggregates pose hypotheses until the number of clustered

Algorithm 5.5: ClusterPoses(\mathcal{P})

Data: $\mathcal{P} = \{(\mathbf{P}_1, v_1), (\mathbf{P}_2, v_2), \dots, (\mathbf{P}_{N_p}, v_{N_p})\}$ **Result:** $\mathcal{P} = \{(\mathbf{P}_1, v_1), (\mathbf{P}_2, v_2), \dots, (\mathbf{P}_{\widehat{N}_p}, v_{\widehat{N}_p})\}$ **Params:** N_c

```
1:  $\mathcal{P} \leftarrow \text{gpu}::\text{SortDescending}(\mathcal{P})$ 
2:  $prev \leftarrow next \leftarrow 1$ 
3:  $\widehat{N}_p \leftarrow 0$ 
4: while  $1 \leq next \leq N_p$  and  $\widehat{N}_p < N_c$  do
5:    $next \leftarrow \text{gpu}::\text{Partition}(\{\mathcal{P}(i) \mid prev \leq i \leq N_p\}, \mathcal{P}(prev))$ 
6:    $\mathcal{P}(\widehat{N}_p) \leftarrow \text{gpu}::\text{Reduce}(\{\mathcal{P}(i) \mid prev \leq i < next\})$ 
7:    $\widehat{N}_p \leftarrow \widehat{N}_p + 1$ 
8:  $\mathcal{P} \leftarrow \text{gpu}::\text{SortDescending}(\mathcal{P}, \widehat{N}_p)$ 
```

pose hypotheses \widehat{N}_p reaches the maximum number of pose clusters N_c or every elements are grouped together. The main operation in here is the partition operation which reorders the elements close to the compared pose $\mathcal{P}(prev)$. The similar elements to the compared one are then placed between $prev$ and $next$, and they are aggregated to result in a pose hypothesis $\mathcal{P}(\widehat{N}_p)$ by the reduce operation. After the clustering, it ends with the final descending sorting on the clustered pose hypotheses so that the most likely pose hypothesis comes first.

5.4 Experimental Results

In this section, we present a set of comparative experiments of [Drost et al. \(2010\)](#), [Papazov et al. \(2012\)](#), [Hinterstoisser et al. \(2012b\)](#), and our approach. We briefly explain how we build the object models from an RGB-D camera in Section 5.4.1. Section 5.4.2 describes our experiment setup, such as the used implementations of the approaches, the chosen parameters associated with the implementations, and the machine specifications used for the evaluations. We start the evaluations with a synthetic dataset to compare the performance of the approaches with respect to Gaussian noise in Section 5.4.3. The performance of the approaches are further evaluated in two sets of synthetic cluttered scenes: multiple objects single instance setting in Section 5.4.4 and single object multiple instances setting in Section 5.4.5. In Section 5.4.6, the approaches are compared in a set of highly cluttered real RGB-D scenes, captured by placing the random selections of the target objects with several other objects as clutter in a paper box. Lastly, we compare the computation time of the approaches and discuss the reasons of the different efficiency in Section 5.4.7.

5.4.1 Object Models

As test objects, 10 daily objects were chosen and their polygonal mesh models were generated as shown in Figure 5.5. To generate the mesh models, we used an RGB-D camera to capture multiple RGB-D views containing one of the objects. For simplicity, ARTags ([Fiala 2005](#)) were employed to register the multiple RGB-D scenes, followed by a planar segmentation which removes the background plane to result in the segmented object of interest. The mesh models were then obtained by running the Poisson reconstruction algorithm ([Kazhdan et al. 2006](#)). Since our approach requires color information of the objects, color attributes of the point clouds were transferred to their closest



Figure 5.5. Polygonal mesh models of the test objects. Ten daily objects were chosen. Each object model is obtained by combining multiple views of object point clouds, followed by the Poisson reconstruction algorithm (Kazhdan et al. 2006). From left to right: Clorox, Flash, Kuka Mug, Milk, MVG Book, Orange Juice, Pringles, Starbucks Mug, Tide, and Wrench.

vertices in the mesh models so that the mesh models maintain the color information.

As in Algorithm 5.1, the object model \mathcal{M} is a model point cloud, and hence we need to convert from the mesh models to model point clouds. For the conversion, we employ a sampling approach that randomly samples a set of points from the faces of an object mesh model with the probability proportional to the areas of the faces. Since our approach requires color information, the color attribute of each point is also determined by averaging the color values of the three vertices of the randomly selected face. To generate sufficient samples, we sample 1,000,000 points per object mesh and subsample using a voxel grid with the leaf size 5 mm. As a large number of CAD models are available from web nowadays, such as the Google 3D Warehouse (Google 2013) or the KIT object models web database (Kasper et al. 2012), our approach can easily learn any object as long as its 3D model is available from the database.

5.4.2 Experiment Setup

For experiments, we generated a set of synthetic and real datasets. The synthetic datasets were generated by rendering the polygonal mesh models (Figure 5.5) in the OpenGL. The projection matrix in the OpenGL was set in accordance with the known intrinsic parameters of the RGB-D camera, the ASUS Xtion Pro in our experiments, so that the rendered scenes simulate the captured scenes from the RGB-D camera. The color and depth buffers from

each rendering were accessed to save as an **RGB-D** pcd file which is a *de facto* standard file format of point cloud in the Point Cloud Library (**PCL**) (Rusu and Cousins 2011). When we save the point cloud, the pose values of the rendered objects were also saved as ground truth pose files for quantitative evaluations. More detailed descriptions of the dataset generation will be stated in each following section.

Our approach is compared with Drost et al. (2010), Papazov et al. (2012), and Hinterstoisser et al. (2012b). The approach using spin images (Johnson and Hebert 1999) is a possible baseline, but it is not considered here since both Drost et al. (2010) and Papazov et al. (2012) reported better results. We use our **GPU** accelerated version of Drost et al. (2010), while the original implementation of Papazov et al. (2012) contributed in the **PCL** is chosen. Papazov et al. (2012) has two important parameters which determines its performance significantly: the pair width d which constraints the first dimension (distance between two points in a pair) of learned **PPFs** and the probability P_M of recognizing the model in a single iteration. The suggested value of d by authors is about the half of the visible object points. So we calculate the maximum point distance in each model d_{max}^m which is defined as

$$d_{max}^m = \max_{i,j \in [1, N_m]} \|\mathbf{p}_i^m - \mathbf{p}_j^m\|_2 \quad (5.8)$$

where N_m is the number of model points \mathcal{M} and $\mathbf{p}_i^m, \mathbf{p}_j^m \in \mathbb{R}^3$ are the i, j -th model points. Then d is set as

$$d = \gamma \cdot d_{max}^m \quad (5.9)$$

where γ is the ratio. Although the authors recommended $\gamma = 0.5$, we evaluated the performance of Papazov et al. (2012) with different values $\gamma = 0.3, 0.5, 0.7$, and we found $\gamma = 0.3$ was the best parameter. The other parameter P_M decides the number of **RANSAC** iterations. The default value of P_M is 0.0125, but when we evaluated with this value, the random sampling approach of Papazov et al. (2012) barely reported true positives. The number of **RANSAC** iterations with $P_M = 0.0125$ is

$$\begin{aligned} N_{iter} &= \frac{\ln(1 - P_S)}{\ln(1 - P_M)} \\ &= 366.1062 \end{aligned} \quad (5.10)$$

where $P_S = 0.99$ is the probability of recognizing the model in N_{iter} trials. Since the approach searches for pose hypotheses only 366 times, the chance of having true positive recognitions is very low. We reduced the parameter as low as $P_M = 0.0005$ so that the number of iterations is sufficiently high as

$$\begin{aligned} N_{iter} &= \frac{\ln(1 - 0.99)}{\ln(1 - 0.0005)} \\ &= 9208.04 \end{aligned} \quad (5.11)$$

which means the approach of Papazov et al. (2012) will randomly search for the pose hypotheses about 9000 times. In this experiments, we set $\gamma = 0.3$ and $P_M = 0.0005$ for Papazov et al. (2012).

The LINEMOD of Hinterstoisser et al. (2012a) is a template matching approach for pose estimation. Since each template encodes the shape information of an object with a certain pose, it is generally required to have a large

set of templates to take into account the shape variations. While the work of [Hinterstoisser et al. \(2012a\)](#) learned the templates with a manual interaction from a user, the later work ([Hinterstoisser et al. 2012b](#)) exploited 3D CAD models to generate the templates automatically. Following the setup in [Hinterstoisser et al. \(2012b\)](#), we used 15 deg and 10 cm for rotation and depth step sizes. We generated multiple templates for each object by rotating the 3D object model with 15 deg step in $\mathbf{x} \in [-\frac{\pi}{2}, \frac{\pi}{2}]$, $\mathbf{y} \in (-\pi, \pi]$, and $\mathbf{z} \in (-\pi, \pi]$ axes, which results in both in-plane and out-of-plane rotations. To address scale changes depending on the distance between the sensor and the object, templates were generated with six levels of depth: from 40 cm to 90 cm with 10 cm step size. Therefore, the total number of templates per each object is $13 \times 24 \times 24 \times 6 = 44,928$. Please note that [Hinterstoisser et al. \(2012b\)](#) sampled only the upper hemisphere of the objects, whereas we sampled the whole sphere as objects in our datasets are randomly placed. This template matching approach requires a large number of templates, which takes an amount of time to generate (in our experiment, it takes about 1.76 hours, on average, to generate 44,928 templates). However, our model for each object is just a subsampled 3D point cloud, and learning CPPFs is very efficient (in our experiment, it takes less than 50 milliseconds). In [Hinterstoisser et al. \(2012b\)](#), they did not explicitly show how to recover the set of $SE(3)$ poses from the template matching, but we used the coarse pose estimation shown in [Choi and Christensen \(2012b\)](#) which estimates the rotation and translation from both the corresponding template’s rotation information and the similar triangles. [Hinterstoisser et al. \(2012b\)](#) employed two post-processing stages in which false positive detections are removed via checking color value followed by the ICP refinement ([Besl and McKay 1992](#)). For fair comparison, the additional color checking is not performed in this experiment, but we run LINEMOD with and without the ICP to examine the effect of performing the ICP. For the ICP parameters, we set 1 cm for the max correspondence distance, 50 for the maximum number of iterations, and 10^{-6} for the transformation epsilon. We chose the LINEMOD implementation in the [OpenCV \(Bradski and Kaehler 2008\)](#) and the ICP implementation in the [PCL](#). There is another implementation of LINEMOD in the [PCL](#), but in our evaluation it reported much worse performance than the LINEMOD implementation in the [OpenCV](#).

All approaches except [Papazov et al. \(2012\)](#) are deterministic in the sense that the approaches always result in the same pose estimates if identical object model and scene point cloud are given. [Papazov et al. \(2012\)](#) is, however, stochastic as it employs the RANSAC approach based on random sampling. For statistically meaningful results, we ran 10 trials for [Papazov et al. \(2012\)](#) and averaged over the results of the multiple trials, while our approach, [Drost et al. \(2010\)](#), and [Hinterstoisser et al. \(2012b\)](#) with and without the ICP were run only once.

We evaluate the performance of the five approaches quantitatively using the ground truth information. If the difference between an estimated pose and its corresponding ground truth pose is less than 15 mm for translation and 10° for rotation, it is counted as a true positive. As some objects are self-symmetric, symmetry of each object is also taken into account. For example, “Clorox” is a cylinder shape, so any rotation in the axis of symmetry is ignored for the comparison.

For the parameters in [Algorithm 5.2](#), we set $\delta = 10$ mm, $\theta = 6^\circ$. The default color quantization levels are

$\sigma = (0.25, 0.25, 1.0)^\top$, but σ are set differently per object to take into account the different color distributions of the target objects as follows

$$\begin{aligned}\sigma_{Clorox} &= (0.2, 0.33, 1.0)^\top \\ \sigma_{Flash} &= (0.1, 0.1, 0.5)^\top \\ \sigma_{Kuka\ Mug} &= (1.0, 0.2, 1.0)^\top \\ \sigma_{Milk} &= (1.0, 0.2, 0.5)^\top \\ \sigma_{MVG\ Book} &= (0.2, 0.2, 0.5)^\top \\ \sigma_{Orange\ Juice} &= (0.2, 0.2, 0.5)^\top \\ \sigma_{Pringles} &= (0.15, 0.25, 1.0)^\top \\ \sigma_{Starbucks\ Mug} &= (1.0, 0.33, 1.0)^\top \\ \sigma_{Tide} &= (0.25, 0.25, 1.0)^\top \\ \sigma_{Wrench} &= (0.25, 0.25, 1.0)^\top.\end{aligned}$$

We found these values empirically, but it would be of interest to determine these quantization parameters automatically, though we leave this as future work. We also set the maximum size of the global vote memory $N_v = 40,000,000$ and the search radius $\gamma_s = 1.0 \cdot d_{max}^m$ in Algorithm 5.3. Lastly, the maximum number of pose clusters is set as $N_c = 10$ in Algorithm 5.5.

All the experiments were evaluated in a standard desktop computer with an Intel Core2 Quad CPU Q9300, 8G RAM, and an off-the-shelf GPU, NVIDIA GeForce GTX 590 with CUDA 4.1.

5.4.3 Gaussian Noise

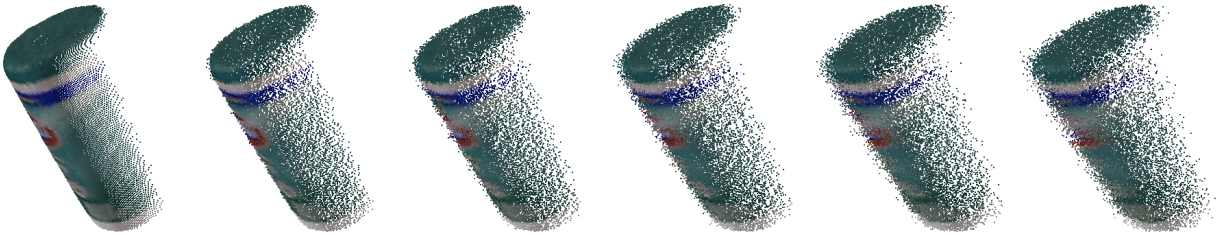


Figure 5.6. Adding Gaussian noise in the synthetic noise dataset. To simulate the noise of RGB-D cameras, Gaussian noise is added in the direction of the camera ray. From left to right: $\sigma = 0, 2, 4, 6, 8, 10$ mm.

To examine the performance with respect to noise, we generate a set of synthetic noise scenes by adding Gaussian noise with different standard deviations as shown in Figure 5.6. To mimic the noise of RGB-D cameras, the Gaussian noise is added in the direction of the camera ray as follows

$$\mathbf{p}^s = \mathbf{p}^s + \frac{\mathbf{p}^s}{\|\mathbf{p}^s\|_2} \cdot \mathcal{N}(0, \sigma) \quad (5.12)$$

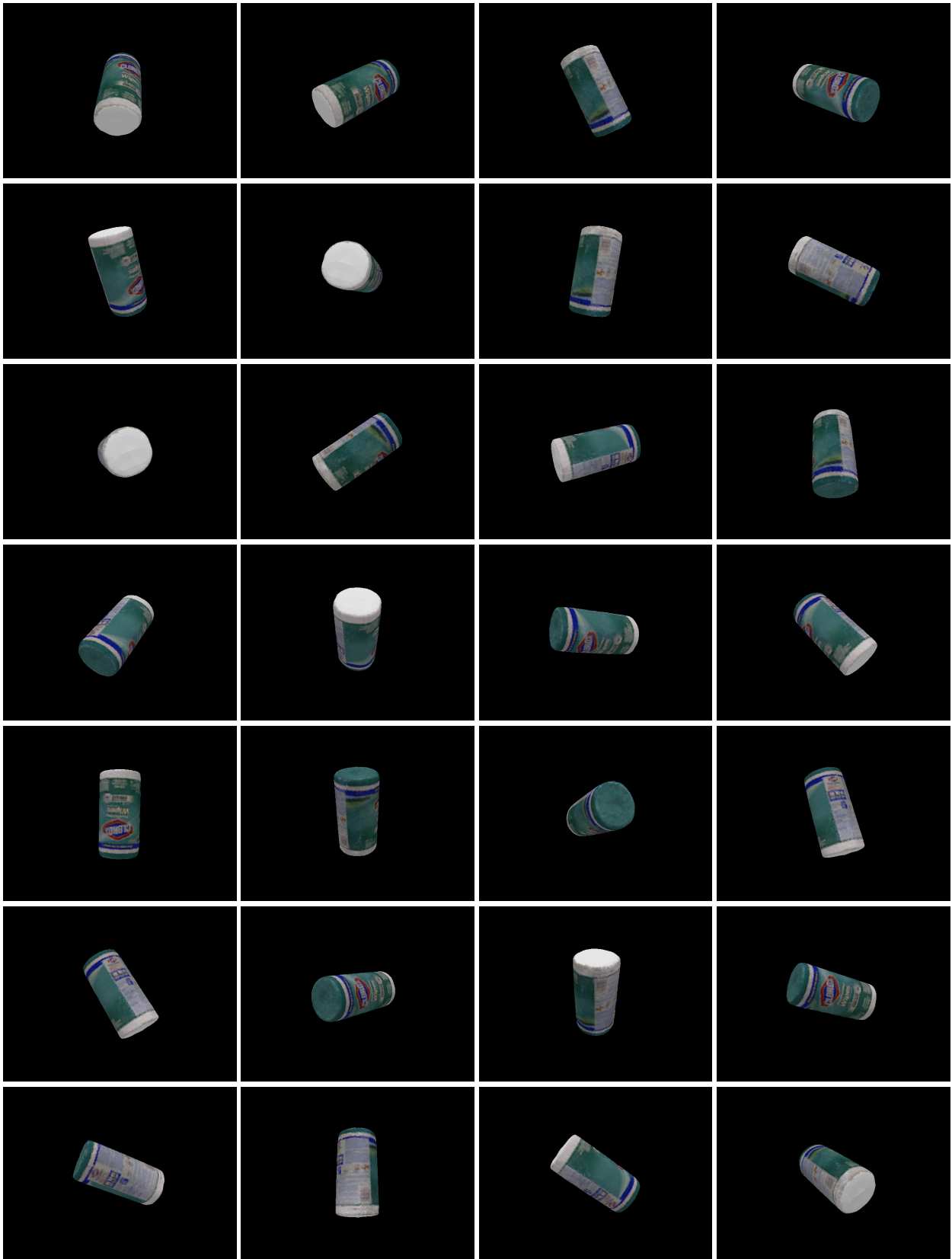


Figure 5.7. Some “Clorox” examples of the synthetic noise dataset. To evaluate the performance of five approaches with respect to noise, a large set of synthetic dataset was generated. For each object, its mesh model was drawn in the **OpenGL** with a fixed translation (0.5 meter apart from the virtual camera) yet with random rotations. Some scenes of “Clorox” object are shown here.

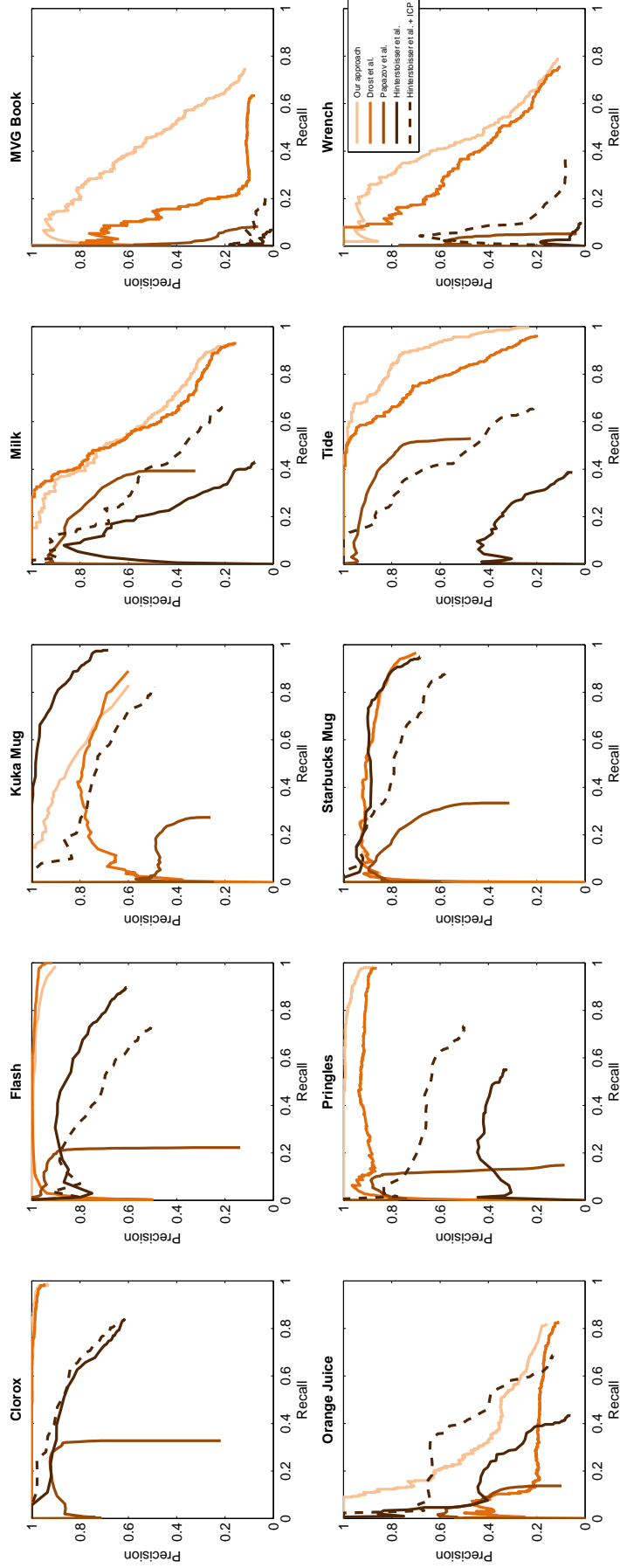


Figure 5.8. Precision-recall curves of the noise experiment. Our approach outperforms the other approaches in most cases. The performance of [Drost et al. \(2010\)](#) is quite similar to ours in some cases, but in general our approach reports higher precision. According to the results of [Hinterstoisser et al. \(2012b\)](#) with and without the **ICP**, the **ICP** algorithm generally enhances both precision and recall but not always. Note that [Papazov et al. \(2012\)](#) is suffered from low recall, while [Papazov et al. \(2012\)](#) reports about 50% recall at best, our approach shows at least about 80% recall in every case.

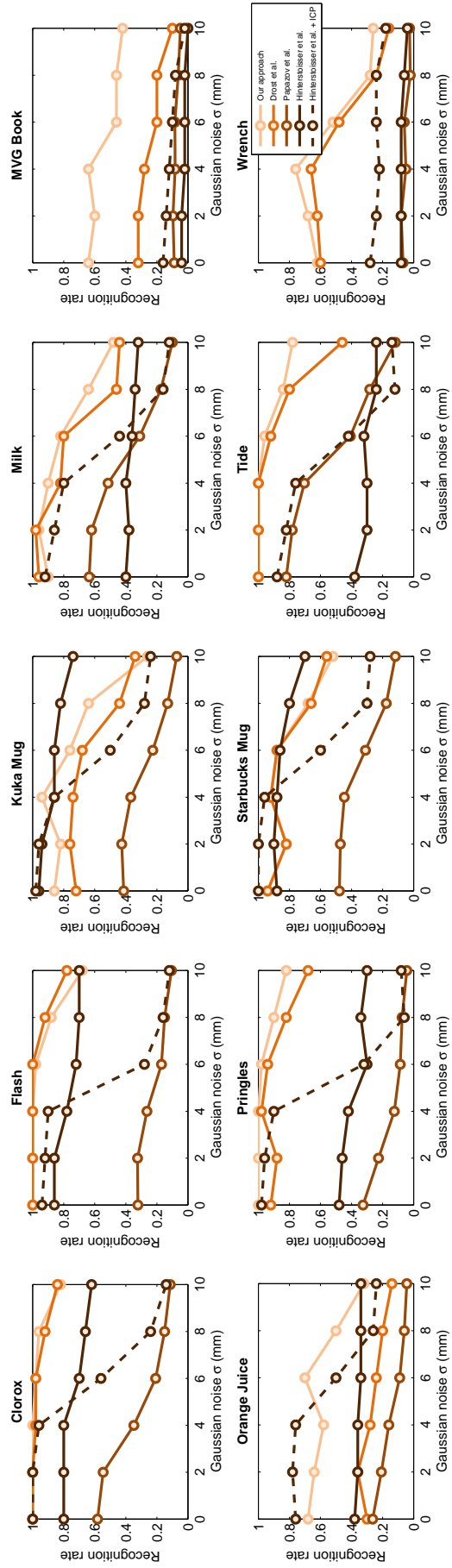


Figure 5.9. Recognition rates against Gaussian noise σ . As σ increases, the performance of the five approaches decrease. Though the performance of the approaches slightly vary, overall our approach outperforms the four compared approaches in most objects.

where $\mathbf{p}^s \in \mathbb{R}^3$ is a point in the synthetic scene, $\frac{\mathbf{p}^s}{\|\mathbf{p}^s\|_2}$ is the unit vector of the camera ray, and $\mathcal{N}(0, \sigma)$ is the zero-mean Gaussian noise with the standard deviation σ . The range of the standard deviations are 0, 2, \dots , 10 mm, which are 6 different standard deviations. For statistically meaningful results, 50 different test clouds were generated with random rotations for each object, as some of “Clorox” scenes are shown in Figure 5.7. Thus the total number of tested point clouds for 10 test objects are $10 \times 6 \times 50 = 3000$.

Figure 5.9 presents recognition rates with respect to the six different Gaussian noise. As we would expect, the recognition rates of the five approaches decrease as the noise level σ increases. Our approach and Drost et al. (2010) report similar performance in “Clorox”, “Flash”, and “Starbucks Mug”, and Drost et al. (2010) even shows slightly better recognition rate in “Flash”. In “Kuka Mug” and “Starbucks Mug”, Hinterstoisser et al. (2012b) is better or at least comparable to both our approach and Drost et al. (2010). Except these cases, our approach reports the best recognition performance in general, and slightly worse performance is shown by Drost et al. (2010). Hinterstoisser et al. (2012b) shows moderate recognition rates, while Papazov et al. (2012) reports the worst recognition performance overall. According to the results of Hinterstoisser et al. (2012b) with and without the ICP, the ICP refinement helps to increase the recognition rate, but only for smaller Gaussian noise levels. As the standard deviation σ increases, the additional ICP process even worsens due mainly to the false point data association on the noisy point cloud. It is worth to notice that Hinterstoisser et al. (2012b) is relatively less affected by the Gaussian noise. It is because Hinterstoisser et al. (2012b) relies on template matching in which 2D templates are matched against the input scene image. Since the Gaussian noise is added only to the depth channel, the RGB channels are not affected at all, and thus the approach relatively less degenerates. However, our approach still shows better recognition rates than Hinterstoisser et al. (2012b) in most cases.

The results are also shown as precision-recall curves in Figure 5.8. The curves were generated by varying the threshold value on the score of the pose estimates: the number of votes for both our approach and Drost et al. (2010), the visibility term μ_V which is the ratio of the model surface area matched to the scene in Papazov et al. (2012), and the template matching score in Hinterstoisser et al. (2012b). According to the precision-recall graphs, our approach outperforms other approaches in most cases with some minor exceptions. In both mug objects, Hinterstoisser et al. (2012b) shows better or comparable precision and recall to our approach. In “Orange Juice”, Hinterstoisser et al. (2012b) with the ICP shows slightly better precision with yet less recall. In most cases, Papazov et al. (2012) shows poor recall, mostly lower than 30%, except “Milk” and “Tide”. Even in “Tide”, its recall is at best about 50%, while our approach at least reports about 80% in all cases. Drost et al. (2010) exhibits comparable performance to our approach in some objects, and yet our approach shows higher precision in general. From the results of Hinterstoisser et al. (2012b) with and without the ICP, we can notice that the ICP enhances both precision and recall in general, but it degrades in “Flash”, “Kuka Mug”, and “Starbucks Mug” objects which are relatively small objects.

The noise dataset only has one object per scene without any backgrounds, and thus all points are corresponding to the test object. In these relatively simple scenes, the performance difference between our approach and other

approaches is not so distinctive, but we will see the gap as we evaluate them in more challenging datasets in the following sections.

5.4.4 Synthetic Scenes: Multiple Objects Single Instance (MOSI)

As we mentioned in Section 5.4.3, when the scene is quite simple in the sense that only one object is exist in an uncluttered background, pose estimation is quite straightforward. However, as environments are more cluttered the number of possible pose hypotheses needed to hypothesize and to verify is exponentially increased with the number of points from cluttered backgrounds. To evaluate the performance of five approaches with respect to clutter, we generated 50 synthetic scenes where random selections of our test objects were placed in either the white laundry basket or the wire basket (Figure 5.10), whose 3D models were downloaded from Google 3D Warehouse (Google 2013). Since single instance of each object is considered, we call this setup as multiple objects single instance (MOSI).

Figure 5.10 presents selected pose estimation results from the 50 scenes. The first row shows the color images of the synthetic scenes, and the second to sixth rows present the detection results of Hinterstoisser et al. (2012b) without the ICP, Hinterstoisser et al. (2012b) with the ICP, Papazov et al. (2012), Drost et al. (2010), and our approach, respectively. For clear visualization, only correct pose estimates are displayed with color mesh models in the monochrome scene point clouds. Please note that except Hinterstoisser et al. (2012b) with the ICP any pose refinement processes were not performed for the rest of approaches, though the additional refinements would enhance the final pose accuracy. According to the qualitative results, it is clear that the recognition performance of our approach is superior to the other approaches. In the cluttered scenes, Hinterstoisser et al. (2012b) without the ICP does not detect any of objects. The main reason of the discouraging performance is due to the limitation of the template matching. Please recall that the templates were generated with the 15 deg rotational and 10 cm depth steps. Even though we employ the significant number of templates (44,928 in our experiments), the templates do not cover entire variations in the appearance of the object. So the detections from the template matching may not be accurate enough to be considered as true positives. The ICP refinement certainly helps in this case. Hinterstoisser et al. (2012b) with the ICP reports several detections in several scenes. Papazov et al. (2012) shows poor performance as it only recognizes one object per scene at best. The reason of the substandard performance may be due to the limited number of searches in the sampling approach. Note that we set the parameter P_M in Papazov et al. (2012) small enough to ensure a sufficiently large number of iterations. The number of iterations, 9208 in Equation (5.11), might be sufficient for simple scenes, such as the noise experiment dataset in Section 5.4.3, but it may not be sufficiently large enough for these complex scenes. Drost et al. (2010) seems more encouraging compared to Papazov et al. (2012), but it still fails to recognize some objects of which our approach with CPPF can successfully recognize and estimate the poses. Due mainly to more discriminative power of CPPF, our approach outperforms the other approaches.

Figure 5.11 presents precision-recall graphs of the MOSI experiment. Like Figure 5.8, the graphs were drawn by varying the threshold value on the score of the pose estimates. The precision-recall curves clearly show the distinguished performance of our approach, as it reports about 100% recall near 100% precision in some objects,



Figure 5.10. Selected pose estimation results of Hinterstoisser et al. (2012b) without the ICP (second row), Hinterstoisser et al. (2012b) with the ICP (third row), Papazov et al. (2012) (fourth row), Drost et al. (2010) (fifth row), and our approach (sixth row) in the MOSI dataset. The first row shows the color images of the synthetic scenes. Correct pose estimates are depicted as color mesh models in the second to sixth rows. In these cluttered scenes, Hinterstoisser et al. (2012b) without the ICP barely detects the target objects due to the coarse sampling of templates. From the additional ICP refinement, Hinterstoisser et al. (2012b) reports some detections, but it is much inferior to both Drost et al. (2010) and our approach. Papazov et al. (2012) also works poorly mainly due to the insufficient number of searches in the sampling approach. Drost et al. (2010) shows better results than Papazov et al. (2012) via the voting process that considers a much large number of pose hypotheses, but this approach still misses a number of true positive detections. Thanks to the additional color information encoded in CPPIs, our approach shows the best performance among the five approaches in these cluttered environments.

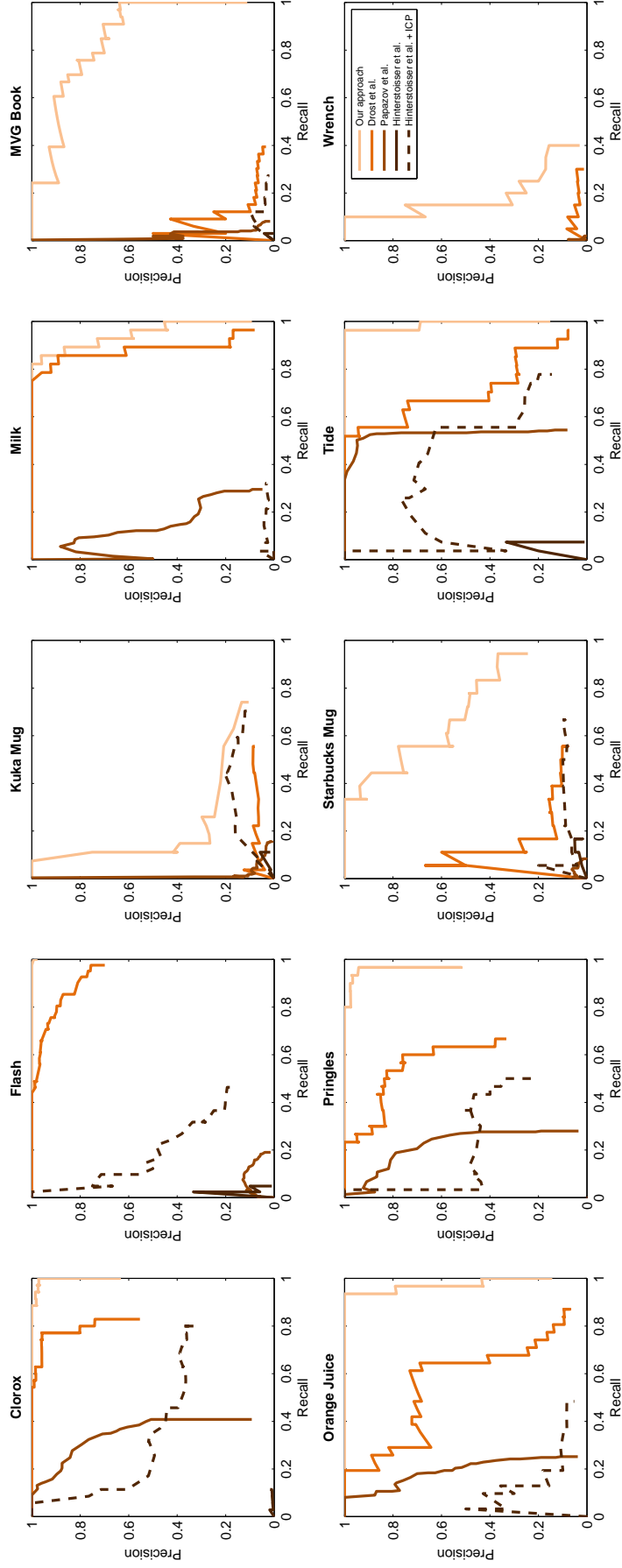


Figure 5.11. Precision-recall curves of the multiple objects single instance (MOSI) experiment. Our approach significantly outperforms the four other approaches in both precision and recall. [Hinterstoisser et al. \(2012b\)](#) without the ICP is the worst approach in this MOSI setting, but with the help of the ICP refinement its recognition performance is enhanced as good as the performance of [Papazov et al. \(2012\)](#).

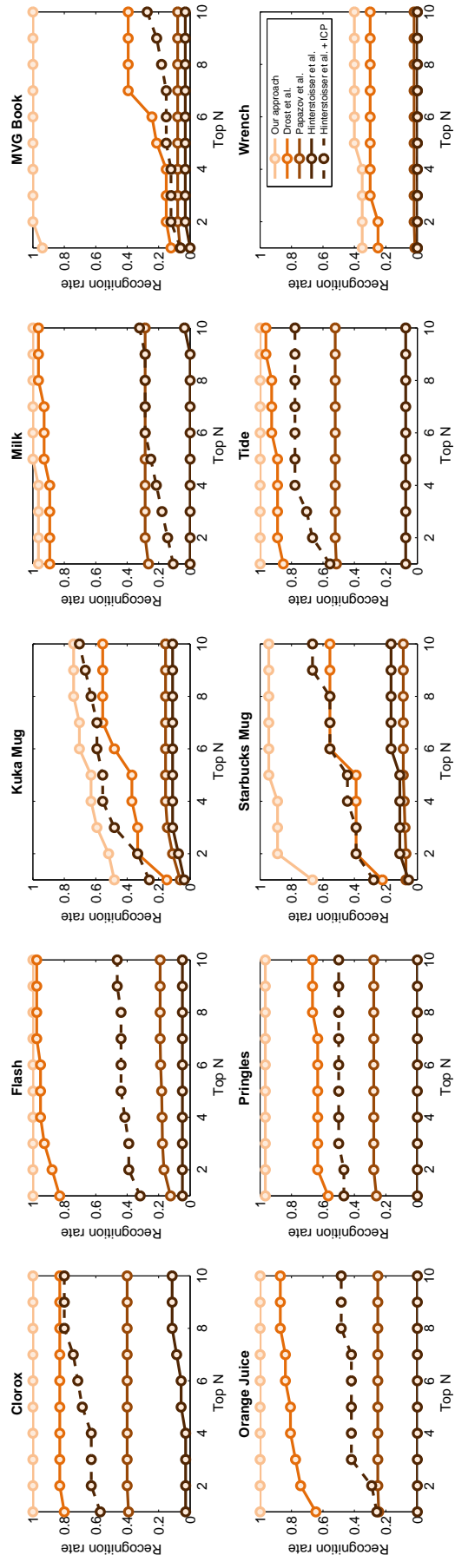


Figure 5.12. Recognition rates of top N pose results in the MOSI experiment. If the ground truth pose is within the top N poses, it is counted as a true positive. Our approach shows the best recognition rates in every object.



Figure 5.13. Selected pose estimation results of Hinterstoisser et al. (2012b) without the ICP (third row), Papazov et al. (2012) (fourth row), Drost et al. (2010) (fifth row), and our approach (sixth row) in the SOMI dataset. Our approach can handle the multiple instances scenario well, whereas Hinterstoisser et al. (2012b) and Papazov et al. (2012) fail to recognize multiple instances in most cases.

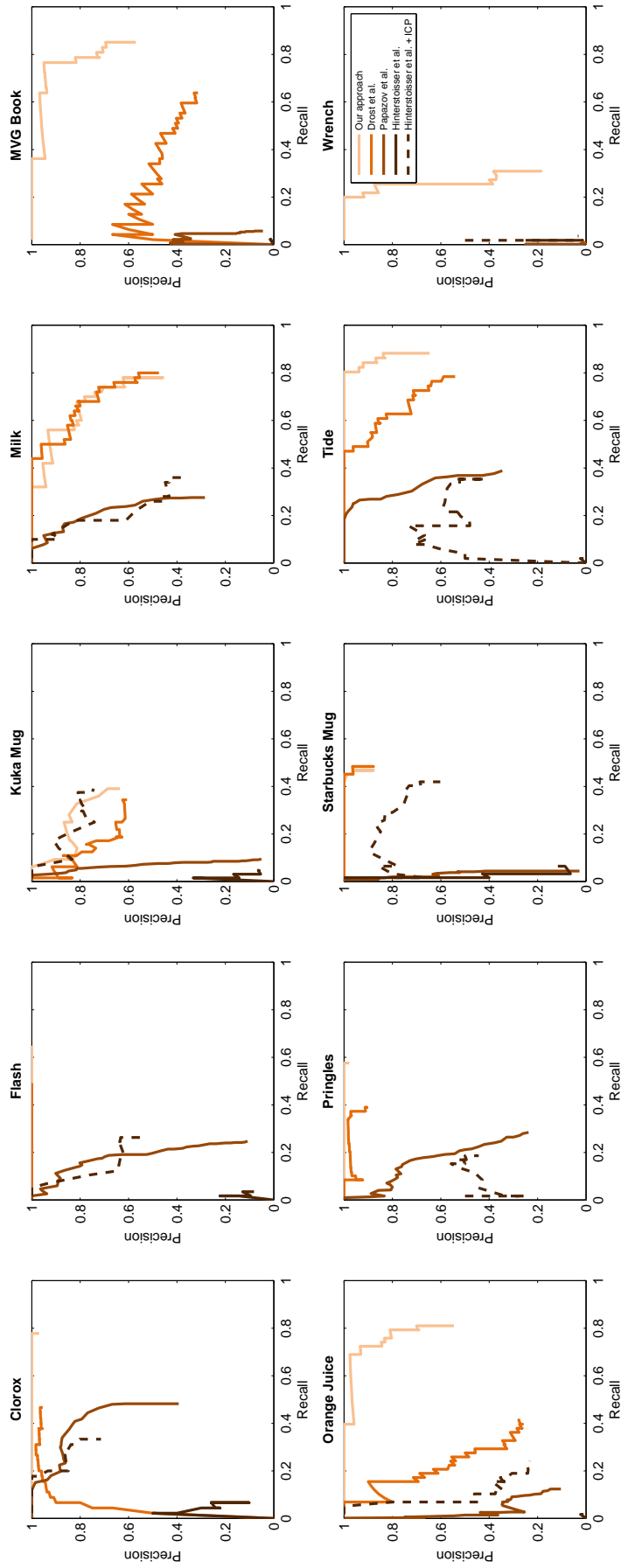


Figure 5.14. Precision-recall curves of the single object multiple instances (SOMI) experiment. It is clear that our approach is superior to both Papazov et al. (2012) and Hinterstoisser et al. (2012b) and still better than Drost et al. (2010) in general. The reason of the scarce difference between our approach and Drost et al. (2010) in “Milk” and “Starbucks Mug” cases is due to the majority color, which is white, of the two objects. Since the background color is also white, our approach does not take advantage of using color information in these two cases.

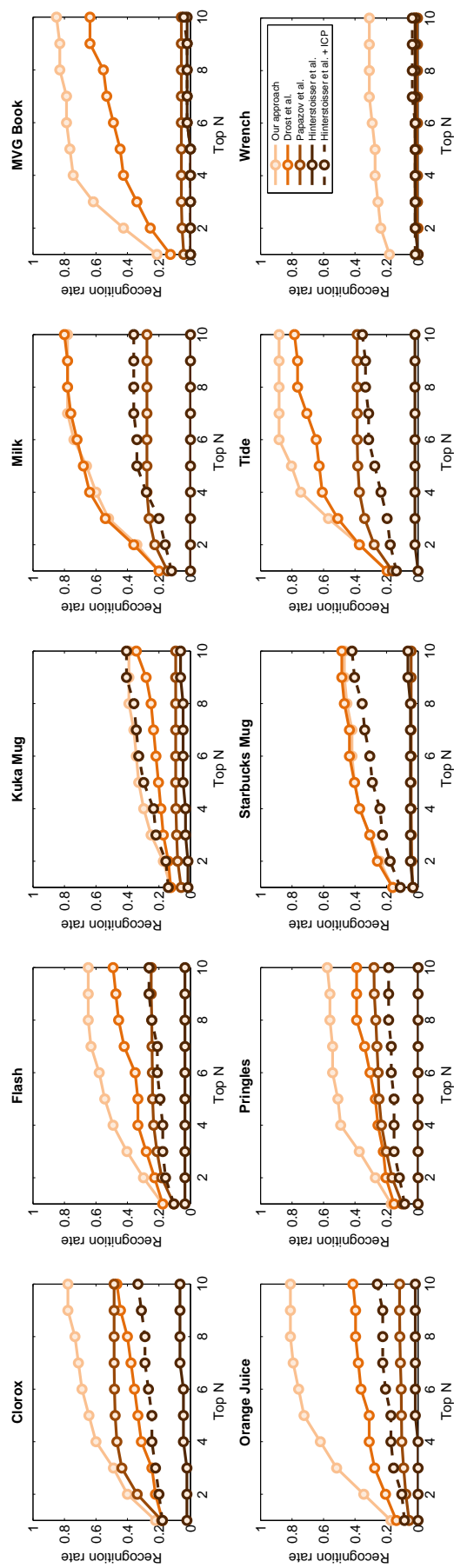


Figure 5.15. Recognition rates of top N pose results in the SOMI experiment. As N increases, the recognition rates have to monotonously increase, because each scene has up to 10 instances of each object. While our approach and Drost et al. (2010) perform as we expected, Hinterstoisser et al. (2012b) without the ICP and Papazov et al. (2012) fail to increase the recognition rates, implying that they are not suitable for this multiple instances setting.

such as “Clorox”, “Flash”, “Orange Juice”, “Pringles”, and “Tide”. [Drost et al. \(2010\)](#) also shows good performance especially in “Clorox”, “Flash”, and “Milk”, yet it is inferior compared to our approach in terms of both precision and recall. [Papazov et al. \(2012\)](#) exhibits moderate performance in some objects, such as “Clorox” or “Tide” which are relatively bigger size and having rich variations in surface normals. Whereas [Hinterstoisser et al. \(2012b\)](#) with the ICP is comparable to [Papazov et al. \(2012\)](#), the approach without the ICP is the worst approach in this dataset.

The five approaches return $N_c = 10$ pose results in maximum, and these multiple poses are sorted in decreasing order of the score. Thus it is of interest to examine the recognition rate with respect to the top N poses. Figure 5.12 presents the recognition rates of the top N pose estimates. The recognition rates are calculated by the ratio of true positives which are counted if the ground truth pose is within the top N poses. According to the plots, the same story is discovered; the recognition rates of our approach are highest among the five approaches, followed by [Drost et al. \(2010\)](#). In both Figure 5.11 and 5.12, it is clear that the ICP increases the recognition rates a lot for [Hinterstoisser et al. \(2012b\)](#).

5.4.5 Synthetic Scenes: Single Object Multiple Instances (SOMI)

Both experiments of Section 5.4.3 and 5.4.4 were designed to detect one instance of each object at a time. In many real scenarios, however, it happens to exist more than one instance of an object in an environment. One of the most popular scenarios especially in manufacturing is the bin-picking in which a bunch of identical objects, needed to be picked, are placed in a pile. Even in our daily lives, identical objects are often placed in environments, such as dishes and silverware in a shelf or identical milk bottles in a refrigerator. As such, it is of interest to evaluate the performance of five approaches in the multiple instances setting, which is called as single object multiple instances (SOMI). When there are multiple identical objects in a scene, it tends to have an amount of the similar local features, and hence grouping these features and verifying multiple hypotheses calculated from the features are generally required. The voting-based approaches are especially preferred in this setting, since it is designed to search all possible pose hypotheses and to group together in a set of consistent pose hypotheses via the voting process. For the evaluation in the SOMI setting, we generated 10 synthetic scenes for each object where randomly chosen numbers between 3 to 10 instances of the object were randomly placed in the white laundry basket. Like the previous experiment, $N_c = 10$ maximum number of clusters was used.

Some of the experimental results on the SOMI dataset are presented in Figure 5.13. Similar to Figure 5.10, the color image of the synthetic scenes are shown in the first row, and their corresponding pose estimation results by [Hinterstoisser et al. \(2012b\)](#) without the ICP, [Hinterstoisser et al. \(2012b\)](#) with the ICP, [Papazov et al. \(2012\)](#), [Drost et al. \(2010\)](#), and our approach are presented in the second to sixth rows respectively. Please note that our approach can handle this multiple instances setting very well. Our approach even reports three true positive detections in the “Pringles” scene, which is very challenging due to the occlusion of the grid in the laundry basket.

When we look at the precision-recall curves in Figure 5.14, it is quite clear that our approach significantly outperforms both [Hinterstoisser et al. \(2012b\)](#) and [Papazov et al. \(2012\)](#) and still better than [Drost et al. \(2010\)](#) in most

cases. The recognition performance of our approach and [Drost et al. \(2010\)](#) is similar in “Milk” and “Starbucks Mug”. It is because the majority of the colors from the two objects is white. Since the color of basket background is also white, our approach does not take advantage of using color information, and hence the performance of our approach is similar to the performance of [Drost et al. \(2010\)](#) in these cases. As we saw in the [MOSI](#) experiment, [Hinterstoisser et al. \(2012b\)](#) with the [ICP](#) in “Kuka Mug” object shows the recognition performance comparable to our approach. Other than that, the approach suffers from low precision and recall.

In the recognition rates of top N results (Figure 5.15), we can notice that the superior performance of our approach in most cases. Since each scene in the [SOMI](#) dataset contains up to 10 instances of the tested object, the recognition rates should monotonously increase as N increases. The recognition performance of both our approach and [Drost et al. \(2010\)](#) increase as we expect, whereas [Hinterstoisser et al. \(2012b\)](#) without the [ICP](#) and [Papazov et al. \(2012\)](#) do not. It implies that [Hinterstoisser et al. \(2012b\)](#) without the [ICP](#) and [Papazov et al. \(2012\)](#) are not encouraging in this multiple instances scenario. It is also worth to note that the performance of [Papazov et al. \(2012\)](#) in this [SOMI](#) experiment is comparable to the performance of [Hinterstoisser et al. \(2012b\)](#) with the [ICP](#), while it was not in the [MOSI](#) experiment. It is probably due to the multiple instances of the test object. As there are more than one instance, the sampling approach of [Papazov et al. \(2012\)](#) may have better chances to have true positive detections.

5.4.6 Real Cluttered Scenes

So far, we have evaluated the five pose estimation approaches in synthetic datasets which were generated by rendering 3D mesh models in the [OpenGL](#). However, it is necessary to compare the performance of the approaches in real [RGB-D](#) scenes which contain the real sensor noise as well as background clutter. For the more realistic test scenes, we put random subsets of our test objects in a paper box with random poses. All other objects not in our test objects were additionally placed as clutter to make more challenging dataset. We captured 31 test scenes, and 7 of them were captured by changing illumination with a non-white lamp; the scene of the right most column in Figure 5.16 is one example. This dataset is much more challenging than the synthetic datasets in previous sections, since additional clutter and the target objects are superimposed each other. For quantitative evaluation, the ground truth poses of the objects were carefully annotated. To annotate them, we first performed the five pose estimation approaches on the test scenes and ran the [ICP](#) ([Besl and McKay 1992](#)) algorithm starting from the estimated pose results. When none of the approaches recognized the object, we manually aligned the corresponding object model to the scene point cloud and then ran the [ICP](#) algorithm to obtain the ground truth pose. If the refined pose was close enough to the true pose, we saved it for quantitative analysis. We use the same criterion from the previous experiments for counting true positives.

Figure 5.16 shows selected pose estimation results from the 31 scenes. The images in the first row are test scene images captured from an [RGB-D](#) camera, and the second to sixth rows represent estimated poses of [Hinterstoisser et al. \(2012b\)](#) without the [ICP](#), [Hinterstoisser et al. \(2012b\)](#) with the [ICP](#), [Papazov et al. \(2012\)](#), [Drost et al. \(2010\)](#), and our approach in the scene point clouds respectively. While [Hinterstoisser et al. \(2012b\)](#), [Papazov et al. \(2012\)](#), and



Figure 5.16. Selected pose estimation results of Hinterstoisser et al. (2012b) without the ICP (second row), Hinterstoisser et al. (2012b) with the ICP (third row), Papazov et al. (2012) (fourth row), Drost et al. (2010) (fifth row), and our approach (sixth row) in the real cluttered dataset. The first row shows the color image of the scanned RGB-D scenes. In these cluttered scenes, neither of approaches can recall more than half of the objects except our approach. Hinterstoisser et al. (2012b) without the ICP does not detect any objects, whereas it can recall one or two objects with the help of the ICP algorithm. Papazov et al. (2012) is also suffered from low recall in the cluttered scenes, and hence it detects at best one object per scene. Drost et al. (2010) works poorly because the low dimensional PPF feature does not give good matches between the model and the scene. Using the color information, CPPF is more discriminative so that pose results from the voting scheme are more likely to be the true positive poses.

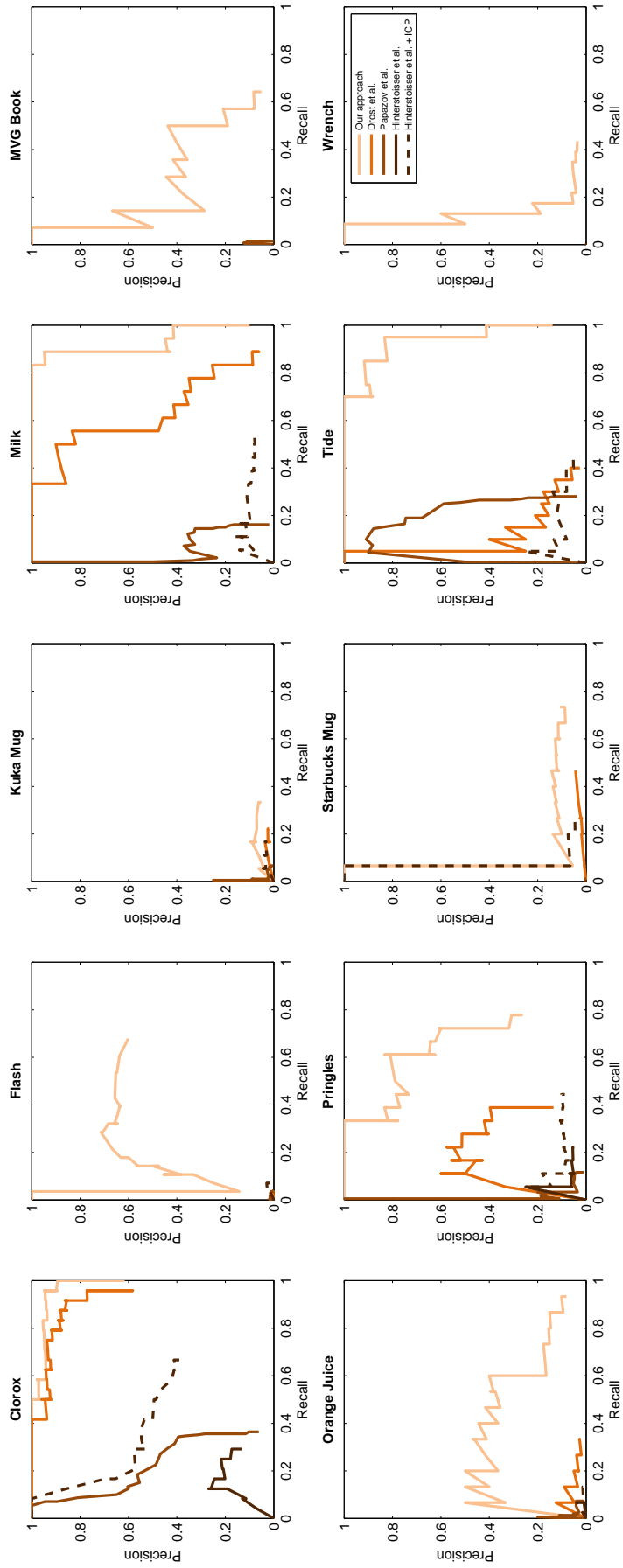


Figure 5.17. Precision-recall curves for the real cluttered scene experiments. While our approach reports good precision as well as high recall, other approaches report substandard results in the highly cluttered backgrounds.

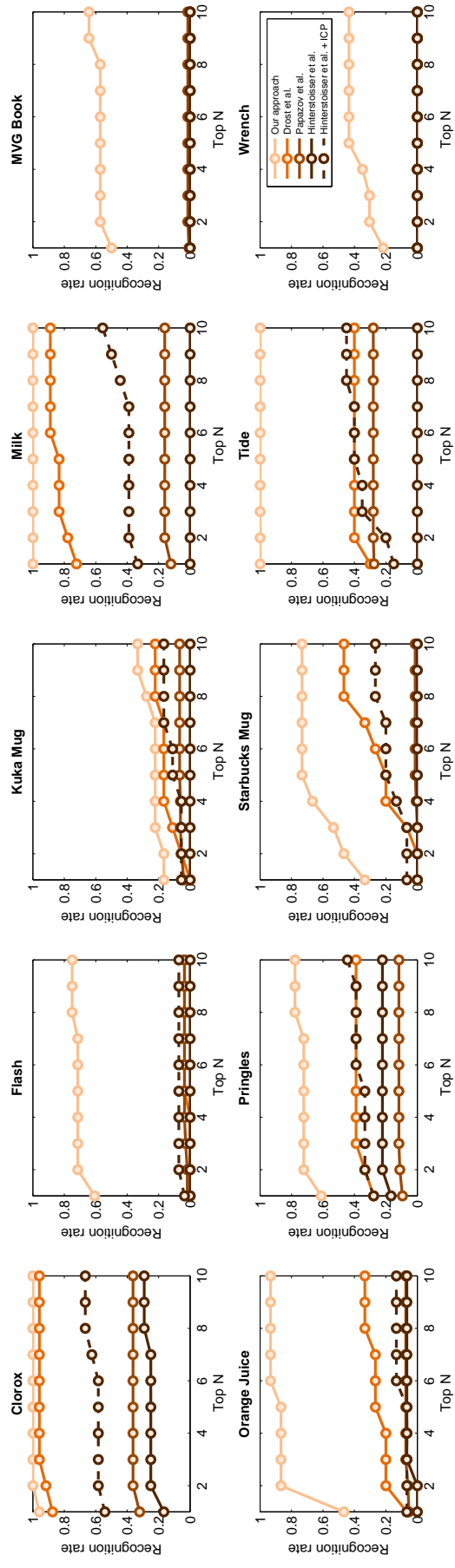


Figure 5.18. Recognition rates of top N pose results in the real cluttered scene experiments. Hinterstoisser et al. (2012b) and Papazov et al. (2012) report poor recognition rates, whereas our approach shows outstanding performance in most cases.

Drost et al. (2010) estimate only one or two object poses per scene, our approach recalls at least more than half of the test objects in the scenes. Precision-recall curves of these approaches are presented in Figure 5.17. Similar to the previous precision-recall curves in the previous experiments, they were drawn by varying the threshold value on the score of the pose estimates. Whereas the performance of our approach is promising, the performance of both Hinterstoisser et al. (2012b) and Papazov et al. (2012) is extremely not encouraging for these cluttered scenes. Especially, they barely report true positive poses in some objects such as, “Flash”, “Kuka Mug”, “MVG Book”, and “Wrench”. Drost et al. (2010) shows better performance than these approaches, yet it is not enough for this cluttered real dataset. In comparison with Figure 5.11 showing precision-recall curves for the synthetic MOSI experiment, the performance of five approaches in this dataset is lower than the performance on the synthetic experiment because of noise from the real RGB-D sensor and extra clutter. But the overall trend in the performance of the five approaches is quite similar in the sense that our approach exhibits the best performance and Hinterstoisser et al. (2012b) without the ICP results in the worst performance.

The recognition rates of top N poses on each object is presented in Figure 5.18. According to the plots, the recognition rates increase as the considered number of results N increases. In “Clorox”, “Milk”, and “Tide”, our approach shows nearly perfect performance as the first or second pose estimates are always true positives. Due to the difficulty in the scenes, the performance of all the approaches is degraded, but our approach still shows outstanding performance compared to the other approaches.

Table 5.1. Average computation time of the five approaches on the real dataset.

Object	Approaches	Prep. Model	Prep. Scene	Model2GPU§‡	Scene2GPU§	Main Comp.	ICP	Total†
Clorox	Ours	0.012 ± 0.001	0.086 ± 0.006	0.034 ± 0.005	0.006 ± 0.002	0.759 ± 0.179	—	0.897 ± 0.181 sec
	Drost et al.	0.013 ± 0.002	0.087 ± 0.006	0.029 ± 0.003	0.005 ± 0.001	1.479 ± 0.120	—	1.613 ± 0.124 sec
	Papazov et al.	0.012 ± 0.001	0.088 ± 0.005	—	—	7.369 ± 1.979	—	7.469 ± 1.981 sec
	Hinterstoisser et al.	—	—	—	—	14.290 ± 4.739	—	14.290 ± 4.739 sec
	Hinterstoisser et al. + ICP	0.012 ± 0.002	0.087 ± 0.005	—	—	14.818 ± 4.821	0.784 ± 0.692	15.701 ± 4.916 sec
Flash	Ours	0.008 ± 0.001	0.084 ± 0.004	0.018 ± 0.001	0.004 ± 0.001	0.260 ± 0.021	—	0.373 ± 0.025 sec
	Drost et al.	0.008 ± 0.001	0.084 ± 0.005	0.017 ± 0.001	0.004 ± 0.000	0.968 ± 0.016	—	1.080 ± 0.016 sec
	Papazov et al.	0.008 ± 0.001	0.086 ± 0.005	—	—	1.624 ± 0.256	—	1.718 ± 0.257 sec
	Hinterstoisser et al.	—	—	—	—	27.091 ± 5.831	—	27.091 ± 5.831 sec
	Hinterstoisser et al. + ICP	0.008 ± 0.001	0.086 ± 0.005	—	—	28.229 ± 6.109	0.291 ± 0.174	28.615 ± 6.153 sec
Kuka Mug	Ours	0.008 ± 0.001	0.084 ± 0.004	0.017 ± 0.001	0.004 ± 0.001	0.109 ± 0.014	—	0.222 ± 0.015 sec
	Drost et al.	0.008 ± 0.001	0.084 ± 0.005	0.017 ± 0.002	0.004 ± 0.001	0.262 ± 0.023	—	0.374 ± 0.027 sec
	Papazov et al.	0.008 ± 0.001	0.086 ± 0.005	—	—	2.653 ± 0.445	—	2.748 ± 0.445 sec
	Hinterstoisser et al.	—	—	—	—	27.478 ± 7.147	—	27.478 ± 7.147 sec
	Hinterstoisser et al. + ICP	0.008 ± 0.002	0.087 ± 0.004	—	—	28.572 ± 7.224	0.245 ± 0.129	28.912 ± 7.232 sec
Milk	Ours	0.013 ± 0.001	0.087 ± 0.006	0.034 ± 0.003	0.005 ± 0.001	0.346 ± 0.032	—	0.485 ± 0.035 sec
	Drost et al.	0.013 ± 0.004	0.086 ± 0.006	0.030 ± 0.004	0.005 ± 0.002	0.825 ± 0.075	—	0.960 ± 0.083 sec
	Papazov et al.	0.012 ± 0.001	0.086 ± 0.004	—	—	6.248 ± 1.380	—	6.346 ± 1.382 sec
	Hinterstoisser et al.	—	—	—	—	13.155 ± 4.102	—	13.155 ± 4.102 sec
	Hinterstoisser et al. + ICP	0.012 ± 0.002	0.086 ± 0.005	—	—	13.748 ± 4.149	1.130 ± 0.710	14.977 ± 4.253 sec
MVG Book	Ours	0.011 ± 0.001	0.084 ± 0.004	0.025 ± 0.002	0.004 ± 0.001	0.189 ± 0.018	—	0.312 ± 0.020 sec
	Drost et al.	0.012 ± 0.001	0.085 ± 0.004	0.024 ± 0.007	0.006 ± 0.014	1.001 ± 0.049	—	1.128 ± 0.057 sec
	Papazov et al.	0.011 ± 0.001	0.086 ± 0.005	—	—	2.890 ± 0.490	—	2.987 ± 0.492 sec
	Hinterstoisser et al.	—	—	—	—	13.205 ± 3.109	—	13.205 ± 3.109 sec
	Hinterstoisser et al. + ICP	0.011 ± 0.001	0.087 ± 0.006	—	—	13.744 ± 3.173	0.677 ± 0.508	14.519 ± 3.070 sec
Orange Juice	Ours	0.012 ± 0.002	0.087 ± 0.005	0.036 ± 0.006	0.006 ± 0.002	0.914 ± 0.167	—	1.055 ± 0.167 sec
	Drost et al.	0.012 ± 0.002	0.087 ± 0.004	0.030 ± 0.006	0.006 ± 0.002	1.715 ± 0.022	—	1.851 ± 0.023 sec
	Papazov et al.	0.011 ± 0.001	0.084 ± 0.004	—	—	4.856 ± 1.084	—	4.951 ± 1.085 sec
	Hinterstoisser et al.	—	—	—	—	13.272 ± 3.928	—	13.272 ± 3.928 sec
	Hinterstoisser et al. + ICP	0.011 ± 0.001	0.086 ± 0.005	—	—	13.892 ± 4.093	1.046 ± 0.585	15.035 ± 4.125 sec
Pringles	Ours	0.012 ± 0.003	0.090 ± 0.006	0.030 ± 0.003	0.006 ± 0.002	0.475 ± 0.060	—	0.613 ± 0.064 sec
	Drost et al.	0.011 ± 0.002	0.087 ± 0.005	0.028 ± 0.004	0.006 ± 0.002	1.004 ± 0.017	—	1.135 ± 0.020 sec
	Papazov et al.	0.010 ± 0.001	0.086 ± 0.006	—	—	2.588 ± 0.466	—	2.683 ± 0.468 sec
	Hinterstoisser et al.	—	—	—	—	15.478 ± 4.329	—	15.478 ± 4.329 sec
	Hinterstoisser et al. + ICP	0.010 ± 0.001	0.087 ± 0.006	—	—	16.198 ± 4.421	0.557 ± 0.347	16.852 ± 4.548 sec
Starbucks Mug	Ours	0.009 ± 0.001	0.086 ± 0.005	0.023 ± 0.005	0.005 ± 0.001	0.787 ± 0.029	—	0.910 ± 0.032 sec
	Drost et al.	0.010 ± 0.002	0.088 ± 0.005	0.020 ± 0.002	0.005 ± 0.001	1.297 ± 0.081	—	1.418 ± 0.084 sec
	Papazov et al.	0.008 ± 0.001	0.085 ± 0.004	—	—	2.230 ± 0.341	—	2.323 ± 0.341 sec
	Hinterstoisser et al.	—	—	—	—	23.733 ± 6.328	—	23.733 ± 6.328 sec
	Hinterstoisser et al. + ICP	0.009 ± 0.002	0.087 ± 0.005	—	—	24.676 ± 6.532	0.327 ± 0.154	25.099 ± 6.578 sec
Tide	Ours	0.014 ± 0.001	0.087 ± 0.005	0.033 ± 0.004	0.004 ± 0.001	0.373 ± 0.058	—	0.511 ± 0.058 sec

Table 5.1. Average computation time of the five approaches on the real dataset. (continued)

Object	Approaches	Prep. Model	Prep. Scene	Model2GPU [§]	Scene2GPU [§]	Main Comp.	ICP	Total [†]
Wrench	Drost et al.	0.014 ± 0.002	0.086 ± 0.004	0.027 ± 0.004	0.005 ± 0.002	0.885 ± 0.011	—	1.017 ± 0.016 sec
	Papazov et al.	0.012 ± 0.001	0.086 ± 0.004	—	—	6.432 ± 1.386	—	6.530 ± 1.388 sec
	Hinterstoisser et al.	—	—	—	—	13.072 ± 4.281	—	13.072 ± 4.281 sec
	Hinterstoisser et al. + ICP	0.013 ± 0.001	0.087 ± 0.005	—	—	13.714 ± 4.443	1.242 ± 0.676	15.056 ± 4.550 sec
	Ours	0.009 ± 0.001	0.087 ± 0.005	0.025 ± 0.004	0.006 ± 0.002	0.919 ± 0.050	—	1.045 ± 0.055 sec
	Drost et al.	0.009 ± 0.001	0.087 ± 0.005	0.023 ± 0.005	0.005 ± 0.001	1.721 ± 0.026	—	1.845 ± 0.028 sec
	Papazov et al.	0.008 ± 0.001	0.087 ± 0.005	—	—	1.070 ± 0.176	—	1.164 ± 0.178 sec
	Hinterstoisser et al.	—	—	—	—	30.508 ± 4.373	—	30.508 ± 4.373 sec
	Hinterstoisser et al. + ICP	0.008 ± 0.001	0.086 ± 0.005	—	—	31.525 ± 4.442	0.211 ± 0.131	31.830 ± 4.483 sec

[†] The most efficient computation time is indicated in **bold** type.

[§] As parallel algorithms on GPU require model and scene data in GPU memory, the transfer times from CPU memory to GPU memory are reported as well.

[‡] Transfer time of model points to the GPU memory also includes the running time of the object learning algorithm shown in Algorithm 5.2.

5.4.7 Computation Time

Our approach is not only more accurate but also more efficient. The average processing time of the five approaches, which is the amount of time required to estimate the pose of each object per scene, is shown in Table 5.4.6. The preprocessing in the third and fourth columns of the table includes subsampling and normal estimation. Given a model or scene point cloud, it subsamples the given cloud using a voxel grid that averages multiple points in a set of voxels to result in a set of subsampled points. Since we are using the color information, the RGB color attributes are also aggregated. As all approaches, except Hinterstoisser et al. (2012b) without the ICP, require oriented points, which have a normal vector for each point, normal estimation is then performed on both model and scene point clouds. The preprocessing is running in CPU, and thus the computation time of the preprocessing is nearly the same across the five approaches. For the subsampling, the same voxel size (10 mm) for the 10 test objects was used, so we can notice that the preprocessing time of the each object model varies depending on the size of the objects; the bigger objects, the more time to pre-process. Since our approach and the implementation of Drost et al. (2010) used in the experiments are parallelized on GPU, we also report the transfer time of model and scene point clouds from CPU memory to GPU memory. In model point clouds, the transfer time includes the running time of the object learning algorithm (Algorithm 5.2). Please note that the model transfer and the object learning can be run only one time for each object. The main computation time is for the rest of the pose estimation procedures. As Hinterstoisser et al. (2012b) with the ICP employs the ICP algorithm as a post-processing, the ICP time is also reported.

According to the table, the total computation time of our approach is the most efficient. Depending on the kind of objects, the total time varies. But the maximum runtime of our approach is within about 1 second, whereas the running time of Papazov et al. (2012) and Hinterstoisser et al. (2012b) takes up to about 7 and 30 seconds, respectively, which are not desirable for real robotic manipulation tasks.

It is also worth to compare the computation time between our approach and Drost et al. (2010). Although both were accelerated on GPU, our approach is about two times faster than Drost et al. (2010). Our algorithm is efficient due to the sparsity of correspondences between model and scene CPPFs. When the object model is learned, similar CPPFs are grouped together by the reduction parallel operation as explained in Section 5.3.3. Since our CPPF is more discriminative than PPF of Drost et al. (2010), our approach results in a smaller number of duplication for each key. The smaller number of duplication is directly related to the efficiency of the voting stage, as it determines the number of iterations of the for-loop in the 13-th line of Algorithm 5.3. The efficiency of our approach further comes from the fewer number of correspondences between model and scene CPPFs. Thanks to the color information, a set of mistakenly matched point pairs between model and scene can be ruled out if their color similarities are low, while the voting of Drost et al. (2010) cannot help skipping the false matches. This facts enable our approach to be more efficient.

5.5 Summary

We presented a voting-based pose estimation algorithm by combining geometric shape and color information from an RGB-D camera. Our color point pair feature (CPPF) was employed in the efficient voting process which can recognize learned objects in highly cluttered environments. As the voting algorithm is inherently possible to be parallelized, we devised a set of highly parallelized algorithms that perform a large amount of repetitive operations in the multi-processors of a GPU. For quantitative evaluations, we generated extensive synthetic and real datasets on which our approach was systematically compared with four other state-of-the-art approaches. Our approach turned out to be more efficient as well as more robust than the compared approaches.

5.6 Exploiting Object Boundary Information

In this section, we propose a voting-based pose estimation algorithm applicable to 3D sensors, which are fast replacing their 2D counterparts in many robotics, computer vision, and gaming applications. It was recently shown that a pair of oriented 3D points, which are points on the object surface with normals, in a voting framework enables fast and robust pose estimation. Although oriented surface points are discriminative for objects with sufficient curvature changes, they are not compact and discriminative enough for many industrial and real-world objects that are mostly planar. As edges play the key role in 2D registration, depth discontinuities are crucial in 3D. In this section, we investigate and develop a family of pose estimation algorithms that better exploit this boundary information. In addition to oriented surface points, we use two other primitives: boundary points with directions and boundary line segments. Our experiments show that these carefully chosen primitives encode more information compactly and thereby provide higher accuracy for a wide class of industrial parts and enable faster computation. We demonstrate a practical robotic bin-picking system using the proposed algorithm and a 3D sensor.

5.7 Contributions

Surface points with normals are good to register objects that have rich variations in surface normals. However, they are not very efficient in representing many industrial and real-world objects that are mostly planar. To this end, we propose several novel pair features that exploit the depth discontinuities in 3D data. The main contribution of this section is a comprehensive study and development of highly informative and compact features to model a 3D object by using its surface normals, boundaries, and their geometric relationships. We exploit a voting framework to efficiently compute poses with the compact features and apply the pose estimation algorithm to a practical bin-picking system using a 3D sensor.

5.8 Boundary-based Pose Estimation

5.8.1 System Overview

Fig. 5.19 (top-left) shows the setup of our bin-picking system. Our system uses a 3D sensor attached on a 6-axis industrial robot arm to estimate the poses of objects randomly placed in a bin. The 3D sensor is based on structured

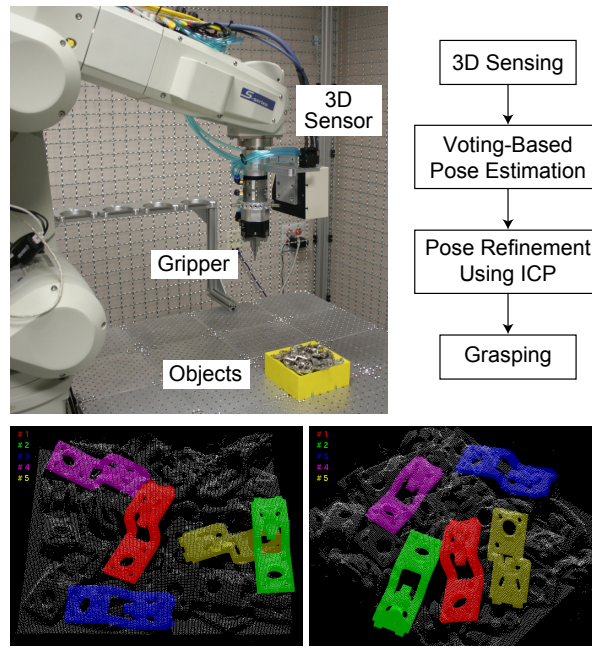


Figure 5.19. System overview. (Top-left) Setup of our bin-picking system. The system uses a 3D sensor attached on a robot arm to grasp an object randomly placed in a bin. (Top-right) Algorithm flowchart. (Bottom) Pose estimation results. Best five pose estimates are superimposed on the scanned 3D point cloud. Note that the scenario exhibits a lot of challenges such as noise, missing data, clutter, and occlusions.

light using an infrared laser and provides 3D data as depth maps of 640×480 pixels. The 3D sensor is calibrated with respect to the robot arm, thereby allowing grasping and picking of an object using the estimated pose.

Fig. 5.19 (top-right) shows the algorithm flowchart. Our system scans the bin of objects using the 3D sensor. Given a 3D CAD model of a target object, our voting-based algorithm (described in Section 5.1) performs detection and pose estimation of the target object using the scanned 3D point cloud. This provides multiple coarse pose hypotheses. The system selects several top pose hypotheses and individually refines them using a variant of ICP algorithm (Besl and McKay 1992). The refinement algorithm renders the CAD model using the current pose estimate and generates 3D points for the model by sampling the surface of the rendered model. It then computes the closest 3D point in the scanned point cloud for each 3D point in the model and updates the pose estimate using the 3D point correspondences.

After refinement, the registration error is given by the average distance between the corresponding scene and model points. The registration error could be high when the coarse pose computed by the voting algorithm is incorrect, or when a part of the object is missing due to occlusion from other objects. If the registration error is small and the estimated pose is safely reachable by the robot arm, the system grasps the object.

Please watch the accompanying video to see our bin-picking system in action.

While the CPPF is applicable to daily objects that contain various texture and color information, it may not be a good feature for industrial objects that lack texture and surface curvature changes. For estimating poses for such industrial parts, we present our effort to extend the voting-based pose estimation by exploiting boundary information

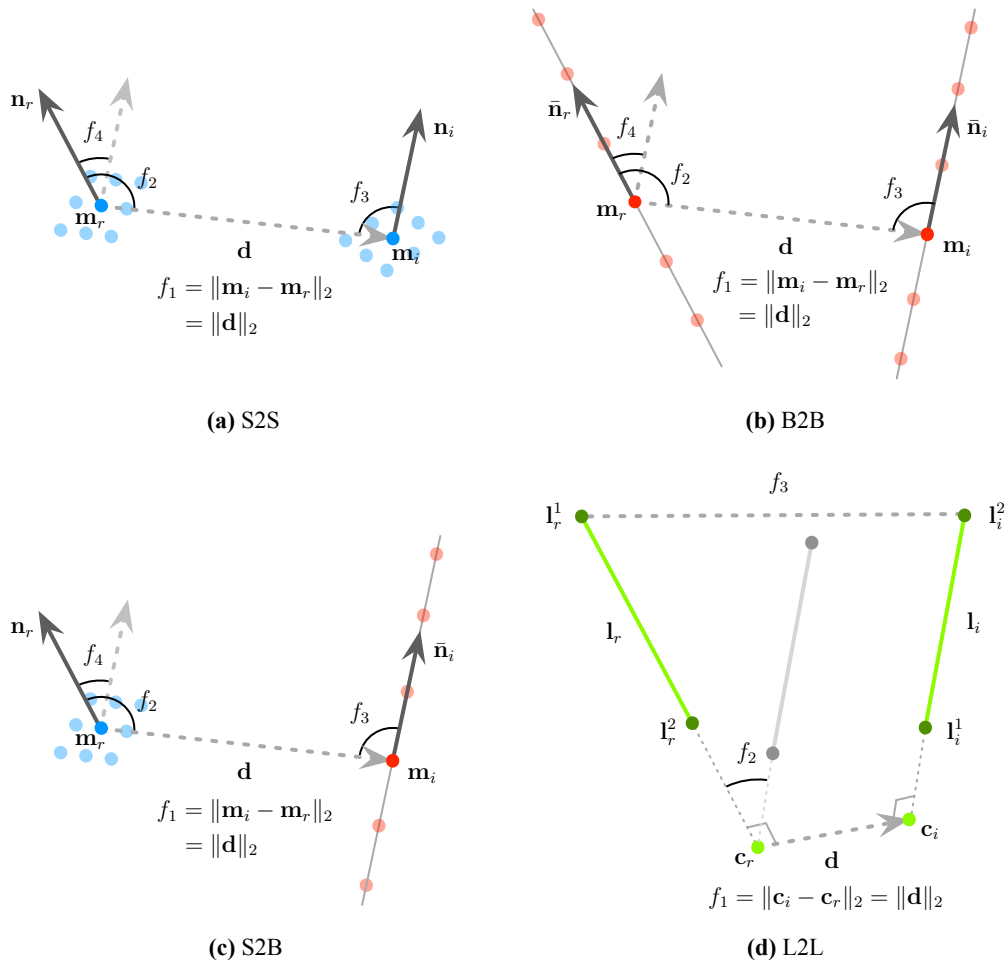


Figure 5.20. Pair features for voting-based pose estimation. (a-c) Point pair feature descriptors are defined by the relative position f_1 and orientations f_2 , f_3 , and f_4 of a pair of oriented points (\mathbf{m} , \mathbf{n}) where blue points indicate surface points with surface normal vectors and red points denote boundary points with directions. (d) The line pair feature descriptor is defined by the distance f_1 and the acute angle f_2 between two (infinite) lines, and the maximum distance between the two line segments f_3 .

of objects in this section.

5.8.2 Pair Features for Boundaries

As geometric primitives, we use oriented points and line segments. We denote a pair feature based on a pair of oriented points on the object surface (Drost et al. 2010) as *S2S*, which is equivalent to the *PPF* described in Section 5.3.1. Here, we propose three novel pair features: (1) a pair of oriented points on the object boundary (*B2B*), (2) a combination of an oriented point on the object surface with an oriented point on the object boundary (*S2B*), and (3) a pair of line segments on the object boundary (*L2L*).

B2B (Boundary-to-Boundary): We define *B2B*, a new point pair feature based on two points on the object boundary (depth edges). In contrast to surface points, boundary points do not have well defined normals. Therefore, we fit line segments to boundary points and use their directions as orientations.

We use a 3D extension of the 2D line fitting approach presented in Liu et al. (2010b). First we compute the edges

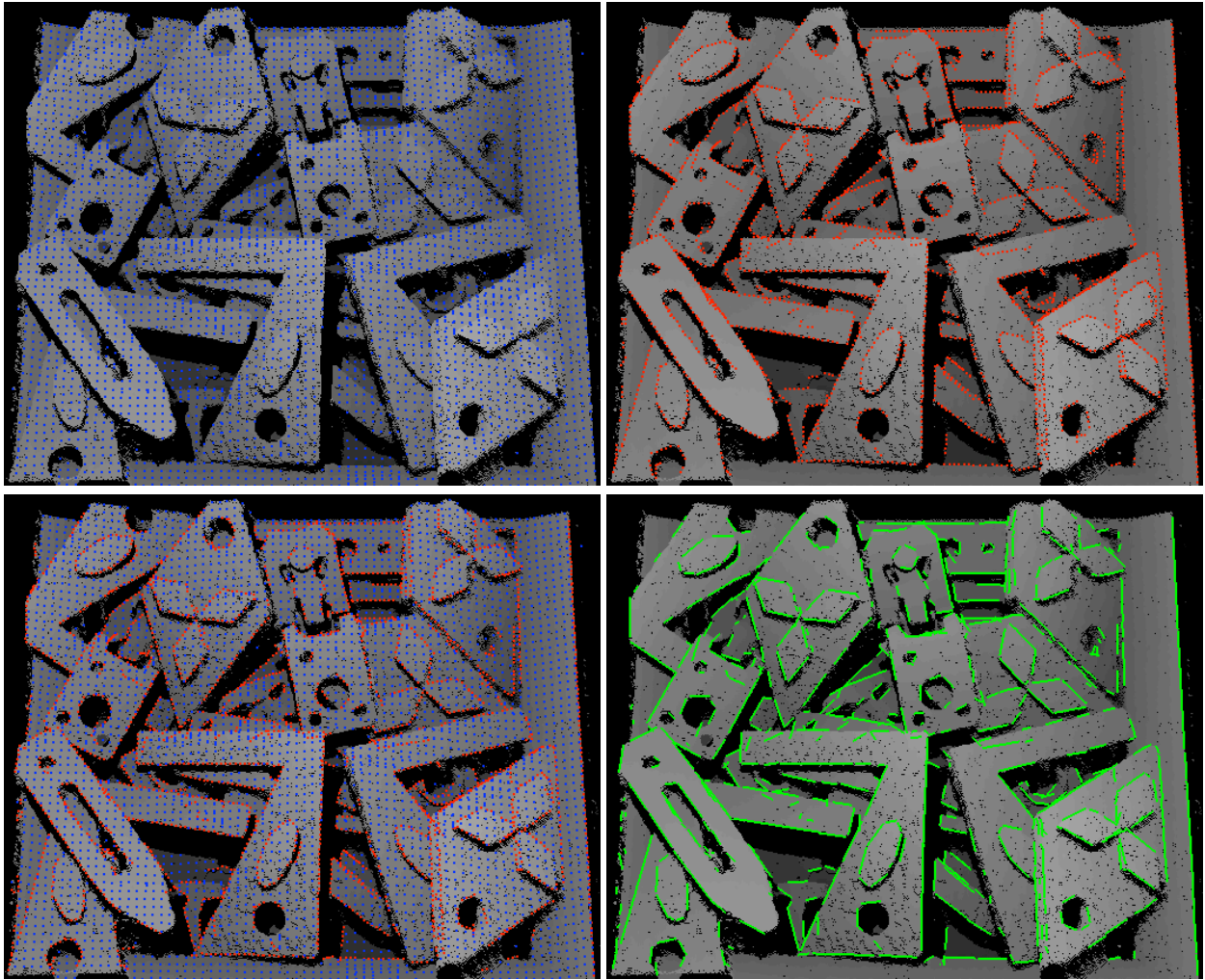


Figure 5.21. Geometric primitives \mathcal{M} for the pair features. From left to right: Surface points and normals for $S2S$, boundary points and directions for $B2B$, their combination for $S2B$, and 3D boundary line segments for $L2L$.

in range images using the Canny edge detector (Canny 1986). Points from the edge map are randomly sampled and 3D lines are fit locally using RANSAC. By iteratively finding and removing line segments with maximum inliers, we recover all line segments. These line segments are further refined using least squares.

After line fitting, we uniformly sample boundary points on the 3D line segments. In Fig. 5.20(b), the red points show the boundary points on two 3D line segments. We define $B2B$ feature descriptor $\mathbf{F}_{B2B} \in \mathbb{R}^4$ as

$$\mathbf{F}_{B2B} = \left(\|\mathbf{d}\|_2, \angle(\bar{\mathbf{n}}_r, \mathbf{d}), \angle(\bar{\mathbf{n}}_i, \mathbf{d}), \angle(\bar{\mathbf{n}}_r, \bar{\mathbf{n}}_i) \right)^\top. \quad (5.13)$$

This descriptor is equivalent to \mathbf{F}_{PPF} in 5.1 except that $\bar{\mathbf{n}}_r$ and $\bar{\mathbf{n}}_i$ are directions of the 3D lines. Note that the directions are not uniquely determined, therefore we consider two possible directions, $\bar{\mathbf{n}}$ and $-\bar{\mathbf{n}}$, when we use the $B2B$ feature. Object boundaries are highly informative. Compared to $S2S$, $B2B$ provides more concise modeling since there are fewer boundary points than surface points. Additionally, the orientations from local line segments are more robust to noise compared to surface normals.

S2B (Surface-to-Boundary): A pair feature only based on boundary points is not very reliable for objects with

high curvature. To jointly and efficiently model both planar and curved objects, we propose *S2B*, a heterogeneous pair feature using an oriented surface point and an oriented boundary point. As shown in Fig. 5.20(c), we define *S2B* feature descriptor $\mathbf{F}_{S2B} \in \mathbb{R}^4$ as

$$\mathbf{F}_{S2B} = \left(\|\mathbf{d}\|_2, \angle(\mathbf{n}_r, \mathbf{d}), \angle(\bar{\mathbf{n}}_i, \mathbf{d}), \angle(\mathbf{n}_r, \bar{\mathbf{n}}_i) \right)^\top. \quad (5.14)$$

L2L (Line-to-Line): We propose *L2L*, a pair feature using two 3D line segments. This pair feature is particularly efficient for polyhedral objects and objects having long boundary line segments, since the number of line segments is fewer than that of surface points or boundary points. Let \mathbf{c}_r and \mathbf{c}_i be the closest points on the infinite lines that contain the 3D line segments, and $\{\mathbf{l}_r^1, \mathbf{l}_r^2, \mathbf{l}_i^1, \mathbf{l}_i^2\}$ denote the end points of line segments, as shown in Fig. 5.20(d). We define *L2L* feature descriptor $\mathbf{F}_{L2L} \in \mathbb{R}^3$ as

$$\mathbf{F}_{L2L} = \left(\|\mathbf{c}_i - \mathbf{c}_r\|_2, \angle_a(\mathbf{l}_r^2 - \mathbf{l}_r^1, \mathbf{l}_i^2 - \mathbf{l}_i^1), d_{\max} \right)^\top, \quad (5.15)$$

where $\angle_a(\mathbf{v}_1, \mathbf{v}_2) \in [0; \frac{\pi}{2}]$ represents the acute angle between two vectors, and

$$d_{\max} = \max(\|\mathbf{l}_i^1 - \mathbf{l}_r^1\|_2, \|\mathbf{l}_i^1 - \mathbf{l}_r^2\|_2, \|\mathbf{l}_i^2 - \mathbf{l}_r^1\|_2, \|\mathbf{l}_i^2 - \mathbf{l}_r^2\|_2).$$

The first and second components are the distance and angle between the two infinite lines, while the last component represents the maximum distance between the two line segments. Using d_{\max} is helpful to prune false matches between two line segments having similar distance and angle. However, line segments usually break into several fragments during the line fitting procedure due to sensor noise, occlusion, etc. As a result, the end points of line segments are usually unstable. Thus we use a bigger quantization step for this component of the descriptor. Note that we discard pairs of parallel lines since the closest points cannot be uniquely determined.

5.8.3 Object Representation

As shown in Drost et al. (2010), we globally model an object using a set of all possible pair features computed from the object model. Once this set is determined, we calculate pair features in the scene point cloud and match them with the set of the model pair features.

The pair feature representation of a target object is constructed offline. We first obtain geometric primitives \mathcal{M} : surface points for *S2S*, boundary points for *B2B*, both surface and boundary points for *S2B*, and 3D lines for *L2L*. These primitives can be calculated from either 3D scanned data with known calibration between the sensor and the object, or synthetic depth data rendered from a known CAD model. With these primitives \mathcal{M} , all possible pair features, $(\mathbf{m}_r, \mathbf{m}_i) \in \mathcal{M}^2$ for *S2S*, *B2B*, or *S2B* and $(\mathbf{l}_r, \mathbf{l}_i) \in \mathcal{M}^2$ for *L2L*, are calculated.

For efficient feature matching, we store the set of pair features of the model in a hash table data structure \mathcal{H} , as in Drost et al. (2010). We quantize the pair feature descriptors and use them as the key for the hash table. Pair features that have similar descriptors are inserted together in the same bin and matching/voting can be done in

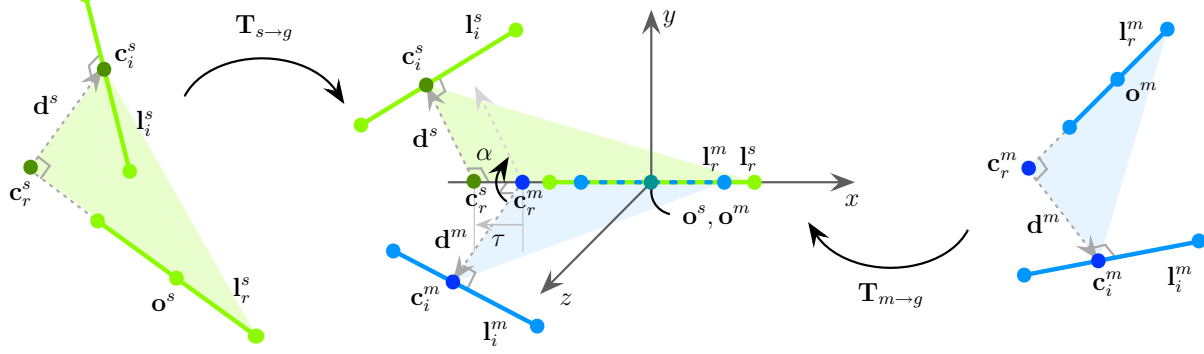


Figure 5.22. Aligning line pair features in the intermediate coordinate system. By $\mathbf{T}_{s \rightarrow g}$, the scene reference line \mathbf{l}_r^s is aligned to the \mathbf{x} -axis and its middle point \mathbf{o}^s is moved to the origin. The model reference line is similarly transformed by $\mathbf{T}_{m \rightarrow g}$ such that the reference lines are aligned. The referred lines \mathbf{l}_i^s and \mathbf{l}_i^m are then aligned by a rotation with angle α and a translation with τ along the \mathbf{x} -axis. Thus a 3D space $(\mathbf{o}^m, \alpha, \tau)$ is used for voting.

constant time. Note that it is important to define the quantization levels appropriately; using very large step sizes reduces discriminative power of the descriptors, whereas using very small step sizes makes the algorithm sensitive to noise.

5.8.4 Voting Scheme for *L2L* Feature

Since the point pair features—*S2S*, *B2B*, and *S2B*—are fundamentally equivalent, the voting scheme explained in Section 5.3.4 is applicable to these point pair features. However, the *L2L* feature, defined by a pair of line segments, requires a specialized voting scheme. Similar to the point pair features, the voting scheme for the *L2L* feature is based on aligning two pair features in an intermediate coordinate system. As illustrated in Fig. 5.22, the reference line \mathbf{l}_r^s and the referred line \mathbf{l}_i^s from the scene are transformed by $\mathbf{T}_{s \rightarrow g}$ in order to align \mathbf{l}_r^s to the \mathbf{x} -axis and to align the middle point \mathbf{o}^s to the origin. Similarly, \mathbf{l}_r^m and \mathbf{l}_i^m are transformed via $\mathbf{T}_{m \rightarrow g}$. Still there are two **DOF** to fully align the line pairs. As in the point pair features, the first one is the rotation around the \mathbf{x} -axis; this angle α is determined from the angle between \mathbf{d}^s and \mathbf{d}^m . The other **DOF** is the translation along the \mathbf{x} -axis; this corresponds to the displacement between the closest points \mathbf{c}_r^m to \mathbf{c}_r^s , denoted as τ . Therefore, we use a 3D space $(\mathbf{o}^m, \alpha, \tau)$ for voting using the *L2L* feature. The transformation from $(\mathbf{l}_r^m, \mathbf{l}_i^m)$ to $(\mathbf{l}_r^s, \mathbf{l}_i^s)$ can be computed as

$$\mathbf{l}_i^s = \mathbf{T}_{s \rightarrow g}^{-1} \mathbf{T}_{\mathbf{x}}(\tau) \mathbf{R}_{\mathbf{x}}(\alpha) \mathbf{T}_{m \rightarrow g} \mathbf{l}_i^m, \quad (5.16)$$

where $\mathbf{T}_{\mathbf{x}}(\tau) \in SE(3)$ is the translation along the \mathbf{x} -axis with τ .

5.8.5 Pose Clustering

In the voting scheme explained in the previous sections, raw pose hypotheses are obtained by thresholding in the voting space. Since an object is modeled by a large set of pair features, it is expected to have multiple pose hypotheses each for different reference primitives, points \mathbf{m}_r or lines \mathbf{l}_r^m , supporting the same pose. Thus, it is required to aggregate similar poses from different reference primitives (Drost et al. 2010). Although there are several methods for clustering in 3D rigid body transformation space $SE(3)$ such as mean shift on Lie groups (Tuzel et al. 2005), these

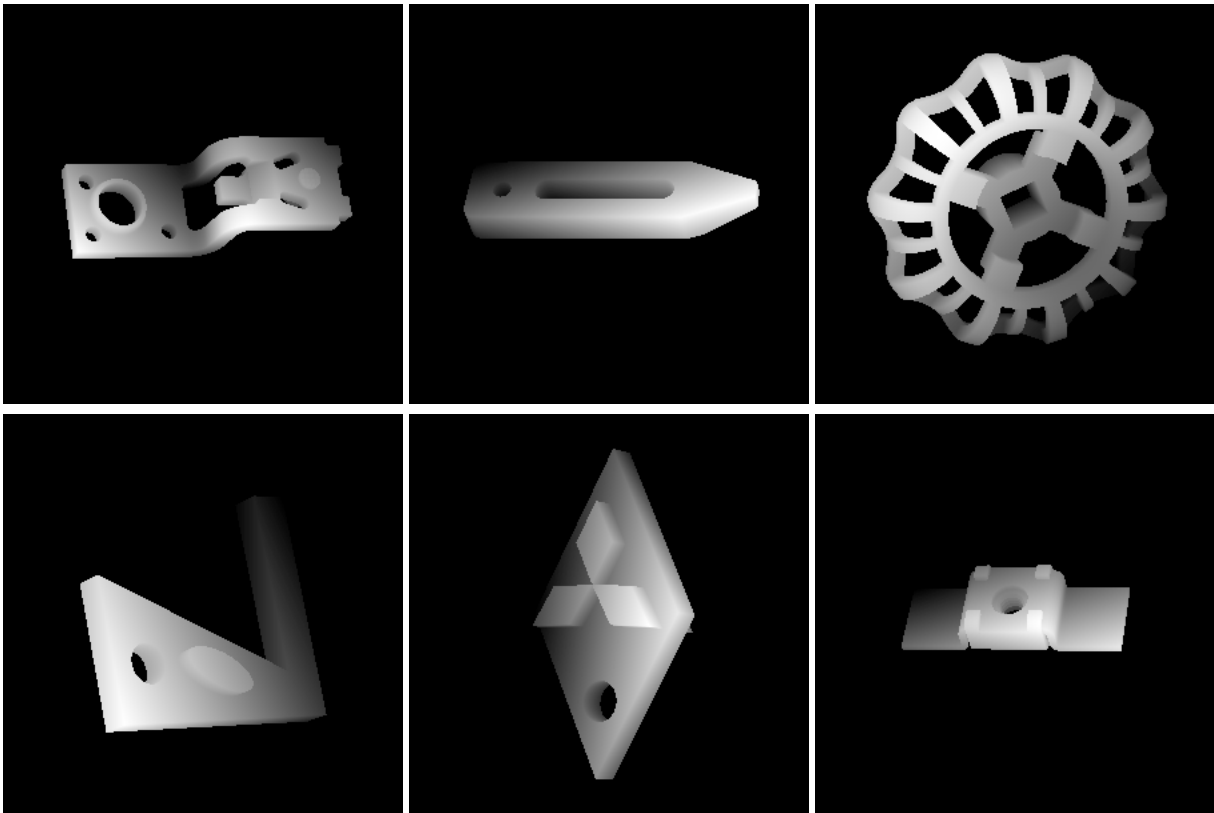


Figure 5.23. Test objects. The 3D CAD models of these test objects are used to create the model pair features for the voting algorithm and to generate synthetic dataset. From left to right: **Circuit Breaker**, **Clamp**, **Wheel**, **X-Shape**, **Logo**, and **Weld Nuts**.

methods are usually computationally prohibitive for time critical applications. Here we adopt an agglomerative clustering approach which is very efficient.

We first sort the raw pose hypotheses in decreasing order of the number of votes. From the highest vote, we create a new cluster. If the next pose hypothesis is close to one of the existing clusters, the hypothesis is added to the cluster and the cluster center is updated as the average of the pose hypotheses within the cluster. If the next hypothesis is not close to any of the clusters, it creates a new cluster. The proximity testing is done with fixed thresholds in translation and rotation. Distance computation and averaging for translation are performed in the 3D Euclidean space, while those for rotation are performed using quaternion representation. After clustering, the clusters are sorted in decreasing order of the total number of votes which determines confidence of the estimated poses.

5.9 Experimental Results

In this section, we present an extensive evaluation of the proposed methods on synthetic and real data. We also evaluate the performance of our bin-picking system described in Section 5.8.1.

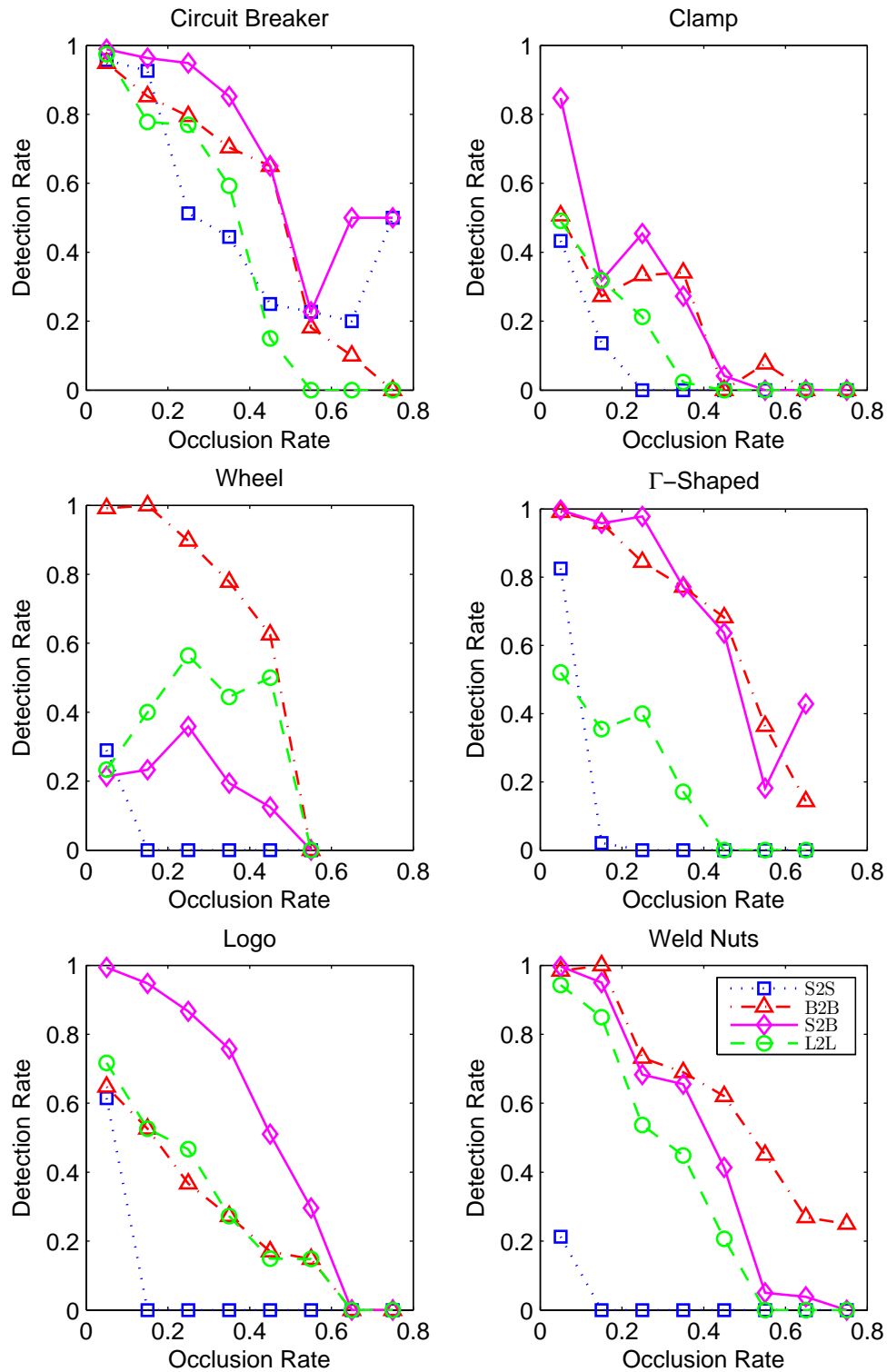


Figure 5.24. Detection rates against occlusion rates for the synthetic dataset. Performance of the four methods decreases as occlusion rate increases. Although the performance depends on objects, *B2B* and *S2B* features generally outperform the other features.

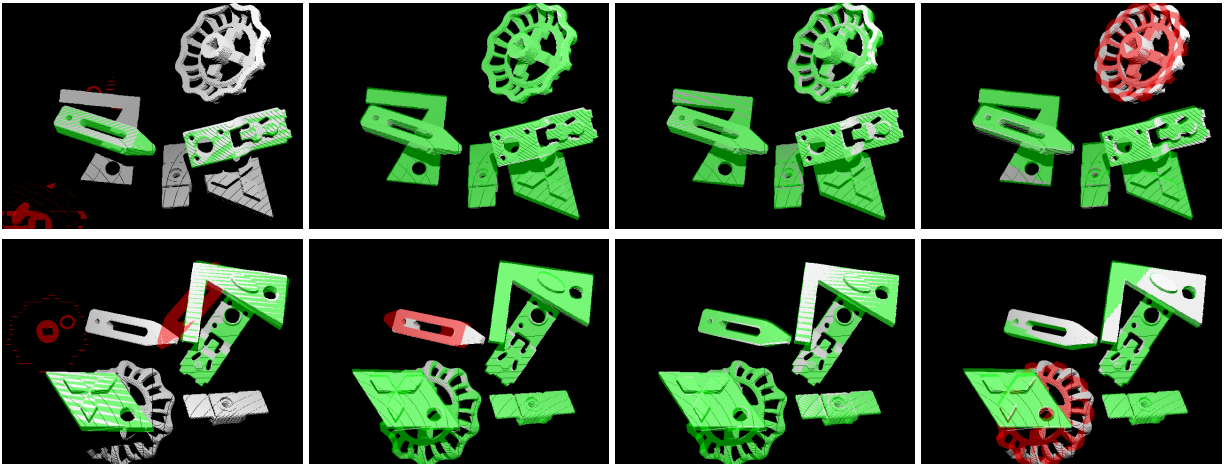


Figure 5.25. Two example scenes from the 500 synthetic scenes. From left to right: Results using *S2S*, *B2B*, *S2B*, and *L2L* features. Correct and incorrect poses are depicted as green and red renderings respectively.

5.9.1 Synthetic Data

To compare the performance of the four pair features, we generated 500 synthetic scenes in which six objects (Fig. 5.23) were drawn with randomly selected poses. We ensured that these random poses do not lead to physically infeasible overlapping objects by checking the intersection of their bounding boxes. We rendered the scenes with *OpenGL* by setting the parameters of the rendering camera based on the calibration parameters of our 3D sensor. For every object the correct pose is stored in a ground truth database for experimental validation. Note that we identify and account for the object symmetries during our experiments.

As shown in Fig. 5.25, objects in the synthetic scene severely occlude each other and the degree of occlusion is various over the 500 test scenes. We quantify the occlusion rate and study the detection performance for different occlusion rates. We follow the occlusion definition of [Johnson and Hebert \(1999\)](#):

$$\text{occlusion} = 1 - \frac{\text{model surface area in the scene}}{\text{total model surface area}}. \quad (5.17)$$

We performed the voting-based pose estimation using each pair feature and considered only the pose that got the maximum number of votes. The estimated pose was then compared with the ground truth. If the errors in translation and rotation were within 5 mm and 5° , we counted it as a true positive; otherwise it was regarded as a false positive.

Fig. 5.24 shows the detection rate at different occlusion rates for each of the six objects. For Wheel and Weld Nuts objects, the *B2B* feature outperforms the other pair features, while the *S2B* feature shows better results for other objects. Since each object possesses different geometric characteristics, the performance of the four pair features on different objects slightly varies; nevertheless, our boundary-based pair features (*B2B*, *S2B*, and *L2L*) show better performance than the *S2S* feature. The reason why the *S2S* feature reports inferior results is that pairs of surface points in the same planar region of the object can correspond to any planar region in the scene. As shown in the leftmost column of Fig. 5.25, planar surfaces of several objects are fitted to the background plane in the scene.

The boundary-based pair features are not only more discriminative, but also more efficient. Table 5.2 shows

Table 5.2. Average numbers of pair features in the synthetic scene dataset and relative processing time.

Feature	Number of Features	Relative Proc. Time [†]
<i>S2S</i> (Drost et al. 2010)	23040000 (= 4800 × 4800)	3.21
<i>B2B</i>	2616953 ($\approx 1618 \times 1618$)	1.00
<i>S2B</i>	7689280 ($\approx 4800 \times 1602$)	1.20
<i>L2L</i>	121058 ($\approx 348 \times 348$)	1.03

[†] The fastest method, *B2B*, is shown as one.

average numbers of pair features in the synthetic scenes and relative processing times where the time of the fastest method, *B2B*, is shown as one. Voting using the *S2S* feature requires a much larger number of pair features than voting using boundary-based pair features. Although the number of the *L2L* features is the smallest, average processing time per a line pair takes more because of the higher-dimensional voting space and more complex transformation via the intermediate coordinate system.

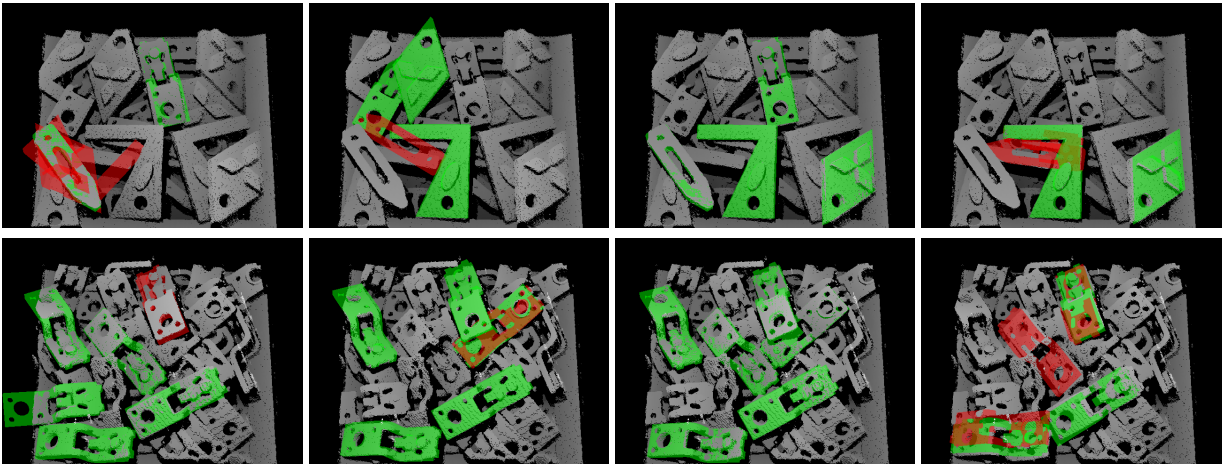


Figure 5.26. Two example scenes from the real scans. From left to right: Results using *S2S*, *B2B*, *S2B*, and *L2L* features. The scene on the upper row includes multiple instances of our test objects. The scene on the lower row contains multiple instances of Circuit Breaker object. Our algorithm can reliably estimate poses of the object even when there are multiple objects and the scene is highly cluttered.

5.9.2 Real Data

We tested the voting-based pose estimation for real 3D data scanned with our 3D sensor. The ground truth poses of the objects are manually identified. Fig. 5.26 shows results for each of the four pair features. The scene on the upper row contains multiple instances of four of our test objects. The objects occlude each other and make the scene highly cluttered. The displayed pose corresponds to the best pose hypothesis computed for each of the four objects. In the result of using the *S2S* feature, two estimated poses are false positives. Similar to the results for synthetic data, the planar area of Clamp object caused several false pose estimates. As shown in the lower row, we also tested the four pair features in the scene which has multiple instances of Circuit Breaker object. For comparison, we rendered top six pose hypotheses obtained using each pair feature. Although in general all pair features provide good performance for this object as shown in the synthetic experiments, the *L2L* feature has three false positives in this case, which are

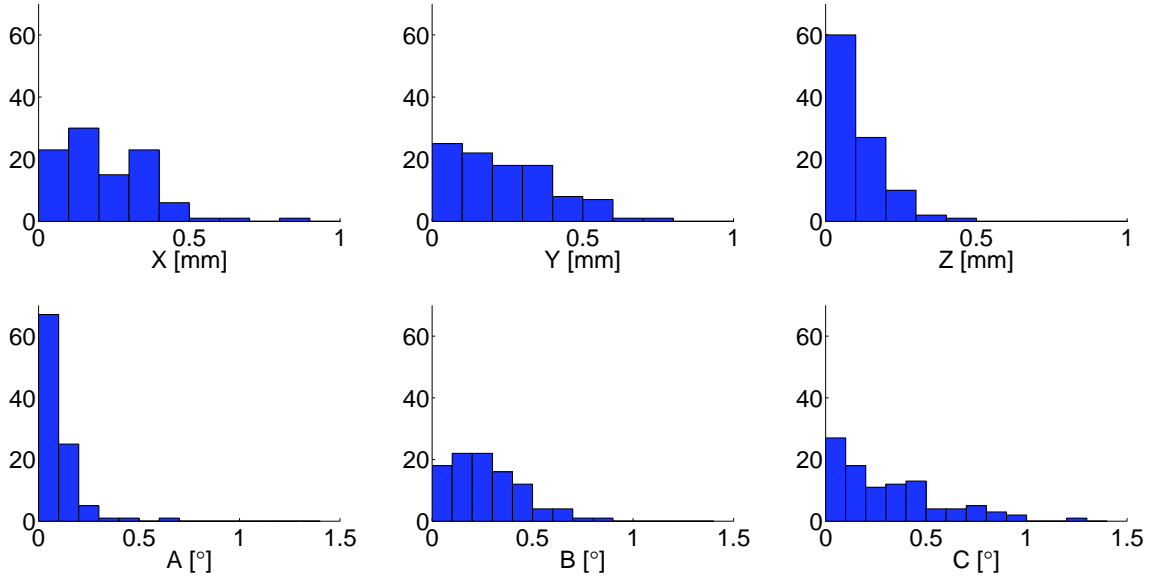


Figure 5.27. Histograms of pose estimation errors for each translation (X , Y , Z) and rotation around each axis (A , B , C). The errors are computed as absolute differences from their median.

the flipped poses of the ground truth poses. These poses have high similarities except the small differences inside the object. The $L2L$ feature is not very robust to such small differences, since the directions of line segments become unstable for short line segments.

5.9.3 Bin-Picking System Performance

Pose Estimation Accuracy: To quantitatively estimate the accuracy of our bin-picking system, we used a single Circuit Breaker object placed on a plane. We scanned it from different locations by moving the robot arm, estimated at each location the pose of the object in the robot coordinate system, and computed pose estimation errors as absolute differences from their median. We selected 100 random sensor locations such that the object is centered in the field of view of the 3D sensor. The locations were within 20° from the z -axis of the robot coordinate system with a distance to the object of 330 mm. For pose estimation, we used the voting algorithm with the $L2L$ feature followed by ICP-based pose refinement.

Fig. 5.27 shows the histograms of pose estimation errors. Table 5.3 describes their average for each translation and rotation around each axis. They demonstrate the consistent pose estimation results of our system with average absolute errors of less than 0.3 mm for all (X , Y , Z) translations and less than 0.3° for all (A , B , C) rotations.

Pickup Success Rate: We measured the pickup success rate of our system by placing 36 Circuit Breaker objects randomly in a bin as shown in Fig. 5.19. We used the $B2B$ feature and used 20% of the total number of boundary points for voting (each 3D scan included ~ 3000 boundary points). The system performed ICP-based pose refinement for the best 5 poses computed with the voting algorithm, and picked up a single object as described in Section 5.8.1 for each cycle. We refilled the bin when the system detected no pickable objects or the system continuously picked up a predefined number (15) of objects. The system picked up 10.2 objects on average in a continuous process.

Table 5.3. Average absolute pose estimation errors.

X [mm]	Y [mm]	Z [mm]	A [°]	B [°]	C [°]
0.22	0.24	0.09	0.09	0.27	0.30

Table 5.4. Pickup success rate.

Total Trial	Success	Failure	Success Rate
344	338	6	98.3%

As shown in Table 5.4, our system achieved a success rate of more than 98% over 344 trials. All 6 failures were due to occlusion of the gripping location of the object. The estimated object poses were correct even in these failure cases.

Processing Time: In the above bin-picking experiments, the voting-based pose estimation algorithm using the *B2B* feature (including 3D line fitting, voting, and pose clustering) took around 500 msec. The refinement process using ICP required around 100 msec for each pose hypothesis. The system was implemented on an Intel Core i7-2600 PC with C++. As shown in the accompanying video, the entire pose estimation process can be performed during robot motion, avoiding the wait time for the robot.

5.10 Summary

We developed a family of pair features using oriented surface points, oriented boundary points, and boundary line segments to model a wide variety of objects. We used the pair features in a voting framework for robust and efficient pose estimation. We showed that the pair features based on the object boundary are more compact and informative, thereby leading to higher accuracy and faster computation. We demonstrated a bin-picking system with pickup success rate of more than 98% and pose estimation error less than 0.3 mm and 0.3° for translations and rotations.

CHAPTER VI

OBJECT POSE TRACKING (3D)

This chapter presents a particle filtering approach for 6-DOF object pose tracking using an RGB-D camera. Our particle filter is massively parallelized in a modern GPU so that it exhibits real-time performance even with several thousand particles. Given an *a priori* 3D mesh model, the proposed approach renders the object model onto texture buffers in the GPU, and the rendered results are directly used by our parallelized likelihood evaluation. Both photometric (colors) and geometric (3D points and surface normals) features are employed to determine the likelihood of each particle with respect to a given RGB-D scene. Our approach is compared with a tracker in the PCL both quantitatively and qualitatively in synthetic and real RGB-D sequences, respectively.

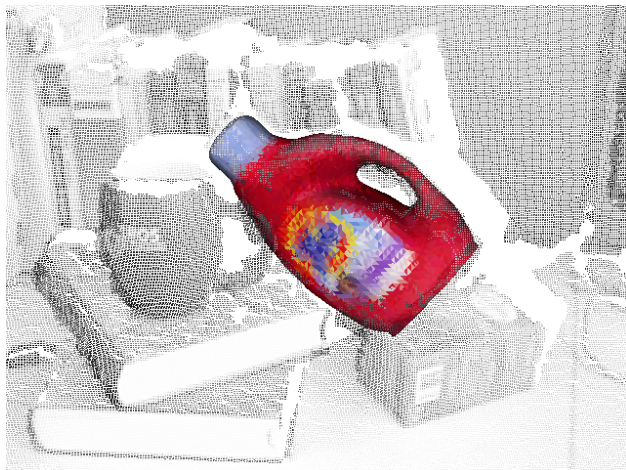


Figure 6.1. A tracking example. An object is tracked via our particle filter parallelized on a GPU. The object rendering represents the mean of the particles.

6.1 Contributions

We propose a robust particle filter parallelized on a GPU, which can track a known 3D object model over a sequence of RGB-D images. Unlike the PCL tracking (Bersch et al. 2012), we render the 3D object model to be used in the likelihood evaluation so that our approach can track the object despite significant pose variations. Our key contributions are as follows:

- We employ extended features to evaluate the likelihood of each particle state. While most of the previous work has mainly relied on 2D edges (Klein and Murray 2006, Azad et al. 2011) or intensity differences (Montemayor et al. 2004, Mateo Lozano and Otsuka 2009) to calculate the importance weights of particles, we use

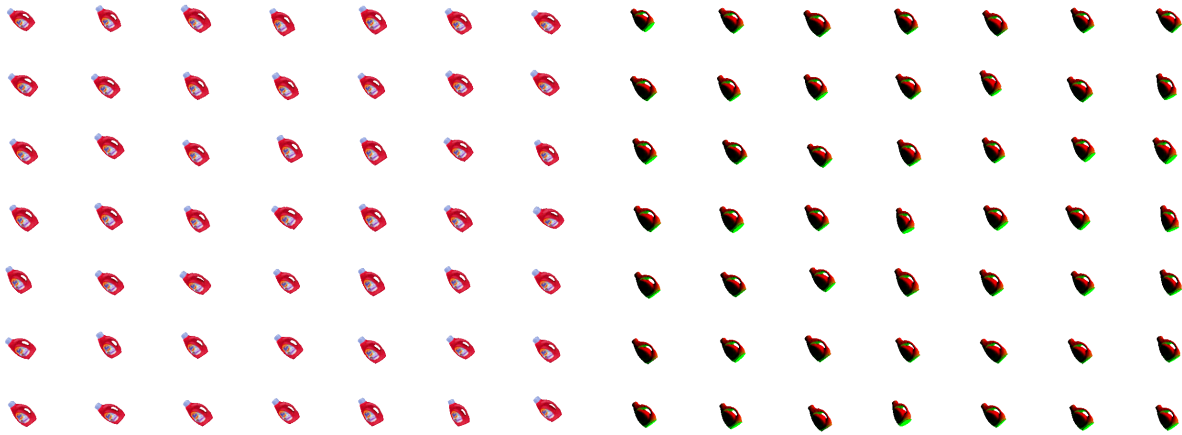


Figure 6.2. Multiple renderings for the likelihood evaluation. The object of interest is rendered with the first V particle states and the likelihoods of the V particles are evaluated from the rendering results. For the rest particles, each particle finds the closest rendering and use the rendering result to evaluate its likelihood. Left and right images represent the color and normal renderings in the GPU. (Best viewed in color)

both photometric (colors) and geometric features (3D points and surface normals) available from both RGB-D images and OpenGL rendering.

- We use the framebuffer object extension (FBO) in OpenGL and the CUDA OpenGL interoperability to reduce the mapping time of the rendering result to CUDA's memory space. As Azad et al. (2011) mentioned, the mapping between OpenGL framebuffer and the memory space of CUDA takes as much as copying the rendered result to the memory space of CPU. To avoid this problem, our rendering is performed in the FBO so that the mapping time is nearly negligible.
- We devised a hierarchical approach to consider multiple renderings of the object. While the PCL tracking (Rusu and Cousins 2011) maintains only one reference object cloud, we render the object to multiple viewports with different poses. It would be ideal if we draw all particle poses to the render buffers in the GPU, but it is not possible due to the memory limitation of the buffers. Instead, our approach renders the object to V viewports, and each particle searches the closest rendering from V viewports so that each likelihood evaluation is performed by transforming the closest rendered result with the current particle state (see Figure 6.2).

To the best of our knowledge, our proposed solution is the first real-time particle filter for 6-DOF object pose tracking using rich visual features from the RGB-D sensor. Figure 6.1 shows an example frame of our tracking where a target object is tracked in background clutter. The rendered 3D mesh model represents the mean of the particles for visualization purpose.

This chapter is organized as follows. A particle filter for 6-DOF object pose tracking is briefly mentioned in Section 3.3.2, and the likelihood function employing points, colors, and normals is introduced in Section 6.2. After the further explanation on the OpenGL and the CUDA implementation in Section 6.3, our approach is compared with a baseline in both synthetic and real RGB-D image sequences in Section 6.4.

6.2 Likelihood Evaluation

Designing an efficient and robust likelihood function is crucial, since it directly determines the overall performance of the particle filtering in terms of both time and accuracy. When an RGB-D camera is considered, there are various measurements we can employ: 3D point coordinates, colors of points, surface normals, curvature, edges from depth discontinuities or surface textures, etc. In this work, we choose the point coordinates $\mathbf{x} \in \mathbb{R}^3$ and their associated colors $\mathbf{c} \in \mathbb{R}^3$ and normals $\mathbf{n} \in \mathbb{R}^3$. Thus, a measurement point \mathbf{p} is defined as

$$\mathbf{p} = (\mathbf{x}^\top, \mathbf{n}^\top, \mathbf{c}^\top)^\top \in \mathbb{R}^9. \quad (6.1)$$

For clear notation, let us define accessing operators for \mathbf{p} such that

$$\mathbf{x}(\mathbf{p}) = (\mathbf{x}^\top \ 1)^\top \in \mathbb{R}^4 \quad (6.2)$$

$$\mathbf{n}(\mathbf{p}) = (\mathbf{n}^\top \ 1)^\top \in \mathbb{R}^4 \quad (6.3)$$

$$\mathbf{c}(\mathbf{p}) = \mathbf{c} \in \mathbb{R}^3. \quad (6.4)$$

The reason we choose the three measurements is that this combination allows us to perform direct comparisons between the given RGB-D scene and the rendering results from the computer graphics pipeline. Hence, we can efficiently calculate the likelihood for a large number of particles.

Given the current pose hypothesis $\mathbf{X}_t^{(n)}$ and the rendered object model \mathbf{M}_t , the likelihood of the scene \mathbf{Z}_t is defined as

$$p(\mathbf{Z}_t | \mathbf{X}_t^{(n)}, \mathbf{M}_t) = \prod_{(i,j) \in \mathcal{A}_p} p(\mathbf{z}_t^{(i)} | \mathbf{X}_t^{(n)}, \mathbf{m}_t^{(j)}) \quad (6.5)$$

where $\mathcal{A}_p = \{(i, j) | \text{proj}(\mathbf{x}(\mathbf{z}_t^{(i)})) = \text{proj}(\mathbf{X}_t^{(n)} \cdot \mathbf{x}(\mathbf{m}_t^{(j)}))\}$ is the set of point associations between the scene \mathbf{Z}_t and the object model \mathbf{M}_t , and $\mathbf{z}_t^{(i)}, \mathbf{m}_t^{(j)} \in \mathbb{R}^9$ are corresponding points in the scene and model, respectively. The operator $\text{proj}(\cdot)$ calculates 2D image coordinates of given 3D homogeneous point coordinates by projecting the point with the known camera intrinsic parameters $\mathbf{K} \in \mathbb{R}^{3 \times 3}$. With the proj operator, the point associations \mathcal{A}_p can be efficiently determined. The likelihood of each association (i, j) is then defined as

$$\begin{aligned} p(\mathbf{z}_t^{(i)} | \mathbf{X}_t^{(n)}, \mathbf{m}_t^{(j)}) &= \exp^{-\lambda_e \cdot d_e(\mathbf{x}(\mathbf{z}_t^{(i)}), \mathbf{X}_t^{(n)} \cdot \mathbf{x}(\mathbf{m}_t^{(j)}))} \\ &\quad \cdot \exp^{-\lambda_n \cdot d_n(\mathbf{n}(\mathbf{z}_t^{(i)}), \mathbf{X}_t^{(n)} \cdot \mathbf{n}(\mathbf{m}_t^{(j)}))} \\ &\quad \cdot \exp^{-\lambda_c \cdot d_c(\mathbf{c}(\mathbf{z}_t^{(i)}), \mathbf{c}(\mathbf{m}_t^{(j)}))} \end{aligned} \quad (6.6)$$

where $d_e(\mathbf{x}_1, \mathbf{x}_2)$, $d_n(\mathbf{n}_1, \mathbf{n}_2)$, and $d_c(\mathbf{c}_1, \mathbf{c}_2)$ are Euclidean, normal, and color distances as shown below

$$d_e(\mathbf{x}_1, \mathbf{x}_2) = \begin{cases} \|\mathbf{x}_1 - \mathbf{x}_2\| & \text{if } \|\mathbf{x}_1 - \mathbf{x}_2\| \leq \tau \\ 1 & \text{otherwise} \end{cases} \quad (6.7)$$

$$d_n(\mathbf{n}_1, \mathbf{n}_2) = \frac{\cos^{-1}(\mathbf{n}_1^\top \mathbf{n}_2 - 1)}{\pi} \quad (6.8)$$

$$d_c(\mathbf{c}_1, \mathbf{c}_2) = \|\mathbf{c}_1 - \mathbf{c}_2\| \quad (6.9)$$

and $\lambda_e, \lambda_n, \lambda_c$ are the parameters that determine the sensitivity of the distances to the likelihood. The τ in (6.7) is a threshold value for the Euclidean distance between the two points. Note that $\mathbf{n}_1, \mathbf{n}_2 \in \mathbb{R}^4$ in (6.8) are homogeneous point coordinates, so 1 need to be subtracted from the inner product. For the color distance in (6.9), any kind of color space can be considered as long as $0 \leq d_c(\mathbf{c}_1, \mathbf{c}_2) \leq 1$, but we adopted the **HSV** color space due mainly to its invariance to illumination changes. Please note that the point and normal coordinates of object model point $\mathbf{m}_t^{(j)}$ are in the object coordinate frame, so the transformed point by the current pose $\mathbf{X}_t^{(n)}$ should be considered to calculate the distances.

6.3 Implementation Details

As we already mentioned in Section 6.1, we render our object of interest onto V viewports in the render buffers with the first V particle poses. For the rest of the particles, each particle finds a closest rendering with respect to the pose hypothesis and transforms the closet rendering result with the current pose. For this calculation, we need to access the color, vertex (*i.e.* 3D point), and normal information for all visible points in the rendered result. It is relatively straightforward to get color information via accessing the render buffer, but it is tricky to access 3D point and 3D normal data from the buffer. To tackle this problem, we employ the **OpenGL** Shading Language (**GLSL**) to directly access the point and normal data in the middle of the graphics pipeline. For point information, we designed a set of simple vertex and fragment shaders so that the 3D coordinates of visible points `gl_Vertex` are saved to color texture points `gl_FragColor`. Similarly, for normal data another set of vertex and fragment shaders is used so that the surface normals of the object `gl_Normal` are saved in color texture points. Note that these points and normals are in the object coordinate frame, so we do not need to perform an inverse transform on the rendering result before transforming them with respect to the current particle pose.

The main purpose of the multiple viewports rendering is not for visualization but for the likelihood evaluation of each particle. Thus using the Frame Buffer Object (**FBO**, `GL_ARB_framebuffer_object`) is a good choice for our rendering purpose. The **FBO** is an **OpenGL** extension for off-screen framebuffers. Unlike the default framebuffer of **OpenGL** provided by window systems, the **FBO** is more flexible and efficient since all resources bound in the **FBO** are shared within the same context. The **FBO** allows users to attach multiple texture images to color attachments. For our purpose, we attach three texture images to the three color attachments: `GL_COLOR_ATTACHMENT0` for color data, `GL_COLOR_ATTACHMENT1` for point data, and `GL_COLOR_ATTACHMENT2` for normal data. In the rendering phase, our object of interest is drawn onto each color attachment. While the color texture is drawn using the usual **OpenGL** rendering, the point and normal textures are rendered via the aforementioned shader programs. For each rendering, the object is rendered to V viewports by calling `glViewport()`.

After rendering the object with shader programs, we evaluate the likelihood function on the **GPU**. To utilize the texture images attached to the **FBO** in our likelihood evaluation kernel, we use the **CUDA OpenGL** interoperability



Figure 6.3. Mesh models for objects and kitchen. Object models were generated by fusing multiple RGB-D views, followed by running the Poisson reconstruction algorithm (Kazhdan et al. 2006). To generate a set of synthetic sequences, a kitchen model was downloaded from the Google 3D warehouse. From left to right, “Tide”, “Milk”, “Orange Juice”, “Kinect Box”, and “Kitchen”.

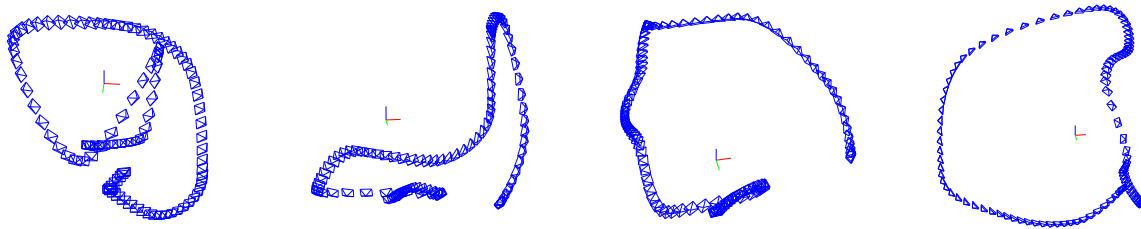


Figure 6.4. Camera trajectories in synthetic sequences. Synthetic RGB-D sequences were generated with their corresponding ground truth trajectories of the objects. Please note the significant variations in translation, rotation, and velocity. From left to right, trajectories of “Tide”, “Milk”, “Orange Juice”, and “Kinect Box”.

that allows to map/unmap OpenGL buffers to CUDA’s memory space. So our CUDA kernel can access the rendered buffers very efficiently.

6.4 Experiments

In this section, we present a set of comparative experiments between the PCL tracking and our proposed approach. The performance of the two approaches is quantitatively evaluated using a set of synthetic RGB-D sequences with the ground truth object trajectories in Section 6.4.2, followed by the qualitative evaluation using real RGB-D sequences in Section 6.4.3. For the evaluations, both trackers are initialized with the known ground truth in synthetic sequences and with the converged pose estimates after running our tracker from a sufficiently close initial pose. The PCL tracking provides an option for adaptive particle size based on Fox (2003), but here the fixed particle size is considered for fair comparisons with our approach and for the performance evaluation with respect to different particle sizes. All experiments were performed using a standard desktop computer (Intel Core2 Quad CPU Q9300, 8G RAM) with an off-the-shelf GPU (NVIDIA GeForce GTX 590, CUDA 4.1) and an RGB-D camera (ASUS Xtion Pro Live).

6.4.1 Object Models

For the experiments, four objects were chosen, and the 3D mesh models of the objects (Figure 6.3) were generated by using an RGB-D sensor. To generate the mesh model, we first obtained multiple RGB-D views and registered them. We could use one of the RGB-D SLAM approaches (Henry et al. 2010, Newcombe et al. 2011, Sturm et al. 2012) to

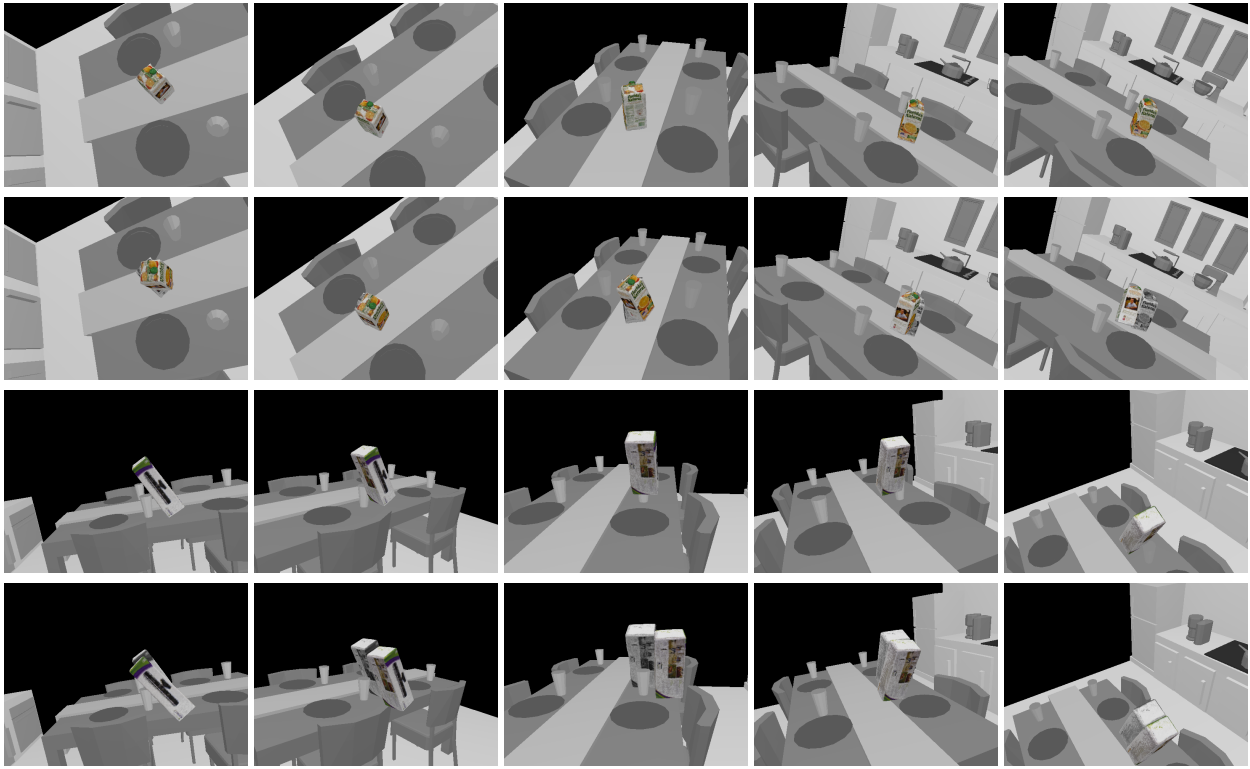


Figure 6.5. Tracking results on the “Orange Juice” and the “Kinect Box” synthetic sequences. For both sequences, the upper rows show the tracking results of our approach, while the lower rows present the results of the PCL tracking ($N = 6400$ for the “Orange Juice” sequence and $N = 12800$ for the “Kinect Box” sequence). In both sequences, our approach tracks the true object trajectories well, but the PCL tracking is often lost due to the limitation of the object model. (Best viewed in color)

register the multiple views, but we employed several ARTags for simplicity. Once the multiple views were fused, the object cloud was segmented from the background clouds and was reconstructed to result in a mesh model via the Poisson reconstruction algorithm (Kazhdan et al. 2006). As the PCL tracking can not use the 3D mesh models, the object point clouds which were obtained by rendering with the initial poses were fed to be used as reference models.

6.4.2 Synthetic Sequences

For a quantitative analyses, we generated a set of synthetic RGB-D sequences. To simulate a realistic environment, a virtual kitchen model (see Figure 6.3) was downloaded from the Google 3D warehouse (Google 2013)¹. After placing each object model in the kitchen model, a set of synthetic sequences was generated by moving the virtual camera around the object. Figure 6.4 shows the camera trajectories of the synthetic data which exhibit high variations in translation, rotation, and velocity of the camera. The object trajectories were saved to be used as ground truth poses of the objects with respect to the camera coordinate frame.

To compare our approach with the PCL tracking, we calculated the root mean square (RMS) errors and average computation time per frame over the four synthetic RGB-D sequences as shown in Table 6.1. For the sake of comparison, better results are indicated in bold type. The RMS errors vary depending on both the object type and

¹Kitchen: <http://sketchup.google.com/3dwarehouse/details?mid=25e8368f5b81a1312ed4225c1c283c73>

Table 6.1. RMS errors and computation time in synthetic RGB-D sequences (PCL vs. Our tracking).

Objects	Tracker	N	RMS Errors [†]						Time (ms) [†]
			X (mm)	Y (mm)	Z (mm)	Roll (deg)	Pitch (deg)	Yaw (deg)	
Tide	PCL	100	3.43	4.96	3.09	9.10	4.05	5.85	81.46
		200	2.45	3.66	2.38	6.96	2.88	4.34	106.50
		400	2.13	3.00	1.89	5.85	2.53	3.50	152.57
		800	1.74	2.79	1.59	5.43	2.35	3.20	253.33
		1600	1.69	2.61	1.51	5.58	2.23	3.39	430.15
		3200	1.49	2.50	1.11	5.26	2.17	3.00	744.50
		6400	1.34	2.11	0.95	5.02	2.15	2.93	1376.92
	12800	1.46	2.25	0.92	5.15	2.13	2.98	2762.73	
	Ours ($\tau = 0.01$)	100	3.47	6.55	4.30	7.58	5.14	4.96	41.48
		200	3.51	4.16	2.71	8.17	3.12	4.55	41.55
		400	2.33	3.25	2.53	5.19	2.26	2.86	42.00
		800	1.88	2.96	2.24	3.79	1.93	2.32	42.37
		1600	1.66	2.42	1.91	3.48	1.71	2.21	44.60
		3200	1.27	1.87	1.54	2.43	1.36	1.58	48.93
6400		1.14	1.54	1.42	2.25	1.13	1.39	77.88	
12800	0.83	1.37	1.20	1.78	1.09	1.13	111.48		
Milk	PCL	100	3.04	7.91	3.60	62.43	37.62	50.76	74.09
		200	2.48	6.65	3.33	62.37	37.76	50.82	91.37
		400	2.37	4.89	2.35	50.49	33.71	41.33	130.47
		800	2.36	4.81	2.03	52.23	33.78	42.89	208.83
		1600	1.78	3.99	1.69	49.27	33.65	40.22	351.83
		3200	13.68	43.72	24.92	64.45	21.52	74.78	585.79
		6400	2.03	4.24	1.57	55.55	34.51	45.67	1126.65
	12800	13.38	31.45	26.09	59.37	19.58	75.03	2205.18	
	Ours ($\tau = 0.01$)	100	3.51	5.96	2.95	12.18	3.36	11.19	49.14
		200	2.42	4.79	2.52	14.06	2.97	12.55	50.21
		400	1.99	4.00	2.21	10.50	2.57	9.29	50.28
		800	1.79	3.16	1.97	7.77	2.03	6.96	51.32
		1600	1.28	2.55	1.74	4.35	1.68	3.90	52.95
		3200	1.20	2.37	1.41	6.22	1.74	5.49	60.55
6400		1.05	2.07	1.21	4.00	1.44	3.47	94.55	
12800	0.93	1.94	1.09	3.83	1.41	3.26	133.95		
Orange Juice	PCL	100	4.83	4.95	3.18	84.38	41.50	46.51	74.15
		200	3.58	3.47	2.99	84.12	44.11	44.76	88.67
		400	3.21	2.83	2.49	86.48	44.67	45.26	114.36
		800	3.06	2.54	2.44	84.76	42.36	45.87	182.19
		1600	2.61	2.39	2.11	84.42	41.65	46.37	295.26
		3200	26.76	5.18	10.83	97.34	51.81	50.67	487.15
		6400	26.86	5.30	11.19	92.67	53.43	65.90	896.85
	12800	2.53	2.20	1.91	85.81	42.12	46.37	1637.13	
	Ours ($\tau = 0.01$)	100	3.49	4.19	2.70	6.82	2.86	6.55	50.39
		200	3.39	4.05	2.41	5.02	2.23	5.81	50.19
		400	2.86	3.64	2.14	3.88	1.77	4.68	50.54
		800	2.18	2.53	1.94	2.55	1.44	2.80	50.64
		1600	1.86	2.39	1.79	2.36	1.29	2.76	52.58
		3200	1.56	2.17	1.50	1.54	1.09	2.09	56.74
6400		1.12	1.61	1.45	1.54	0.84	1.61	80.22	
12800	0.96	1.44	1.17	1.32	0.75	1.39	117.08		
Kinect Box	PCL	100	50.02	48.08	54.28	12.78	2.98	19.77	96.23
		200	44.45	71.22	53.61	86.93	34.92	64.54	146.30
		400	32.62	51.28	51.01	12.46	2.43	12.56	238.51
		800	44.83	43.50	56.51	10.23	2.31	10.63	395.86
		1600	43.93	42.70	55.70	9.80	2.33	10.58	666.85
		3200	44.59	42.93	55.78	11.82	1.94	11.33	1218.44
		6400	43.43	41.92	55.78	7.21	2.08	7.74	2377.42
	12800	43.99	42.51	55.89	7.62	1.87	8.31	4539.42	
	Ours ($\tau = 0.01$)	100	11.26	30.02	19.66	15.86	3.75	14.51	50.07
		200	7.67	18.51	12.02	11.27	2.18	11.15	49.74
		400	5.61	13.86	9.24	10.12	1.71	10.16	52.21
		800	3.67	5.28	2.60	8.40	1.58	8.43	51.87
		1600	3.37	4.13	2.15	7.86	1.17	7.86	56.33
		3200	6.11	9.16	7.51	7.63	1.03	7.51	64.00
6400		2.38	3.28	1.52	8.14	1.42	7.52	116.17	
12800	1.84	2.23	1.36	6.41	0.76	6.32	166.14		

[†] For the sake of comparison, better results are indicated in bold type.

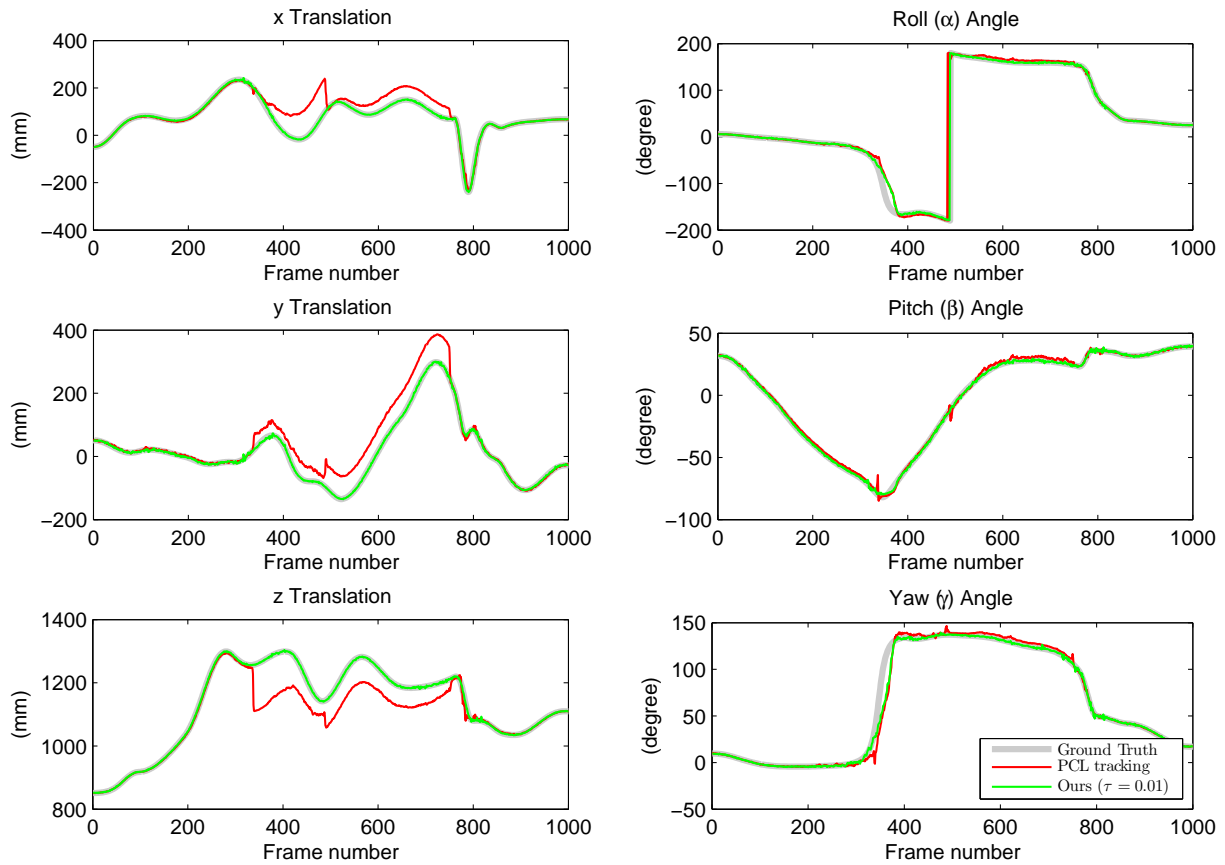


Figure 6.6. The 6-DOF pose plots of the “Kinect Box” results in Figure 6.5. While our approach well follows the ground truth, the PCL tracking suffers from wrong tracking due to the limitation of the object model.

the difficulty of the sequences. But the overall trend here is that as the number of the particles N increases the translational and rotational errors are decreased. In the “Tide” sequence which is rather simple compared to the other sequences, for example, the PCL tracking reports slightly better results when $N \leq 800$. But as N increases, our tracker shows more accurate results. An interesting fact in the “Tide” sequence is that the PCL tracking shows slightly better result in z translation. This may be due to the fact that the PCL tracking has only one reference object point cloud as an object model so that it does result in smaller error in that direction. However, the limited number of the reference cloud is getting problematic when it runs on more challenging sequences. Please note that big errors in both translation and rotations in the “Milk”, “Orange Juice”, and “Kinect Box” sequences. Since the objects are self-symmetric themselves, the one reference view of each object is not enough to track the objects over the entire sequences. As shown in Figure 6.5 and Figure 6.6, the PCL tracking is often stuck in local minima during the tracking, while our approach robustly tracks the objects in the sequences.

For robotic applications, the computation time is really important since it directly determines the performance and the reliability of the robotic systems. As we can see in both Table 6.1 and Figure 6.8, the computation time of both approaches increases linearly as N increases. However, our tracking only takes about 50 ms per frame (*i.e.* 20 frames per second) with several thousands particles, whereas the PCL tracking suffers from low frame rates. Although the PCL tracking shows comparable performance in the “Tide” sequence, if we consider the real frame rates the PCL



Figure 6.7. Tracking results on the “Tide” and “Milk” real sequences. For both sequences, the upper rows show the tracking results of our approach, while the lower rows present the results of the **PCL** tracking ($N = 1600$). As the objects undergo significant variations in rotations, the **PCL** tracking suffered from false pose estimates. Thanks to multiple model renderings every time, our tracking reliably tracks the true poses of the objects. (Best viewed in color)

tracking would have much higher errors due to the lost frames.

6.4.3 Real Sequences

We ran both tracking approaches in a set of real image sequences which exhibits significant noise and more variable object motions compared to the synthetic sequences. Figure 6.7 shows the pose tracking results on the “Tide” and “Milk” sequences. For clear visualization, the **RGB** channels from the **RGB-D** sequences were converted to gray scale, and each rendered object model was drawn on top of them. Invalid depth points (also known as Nan points) were shown as black points. For both sequences, $N = 1600$ particles were employed. As the objects experience significant variations in rotations, the **PCL** tracking (lower rows in each sequence) often loses its tracking. Thanks to employing the 3D object model and rendering it in multiple views, our tracking dependably tracks the object in spite of the challenging rotational motions.

6.5 Summary

We presented an approach for **RGB-D** object tracking which is based on a particle filter on a **GPU**. Rich visual features were employed to evaluate the likelihood of each particle, and this process, which is a typical bottleneck in most particle filters, was parallelized on the **GPU** so that our proposed solution achieves real-time performance. Through

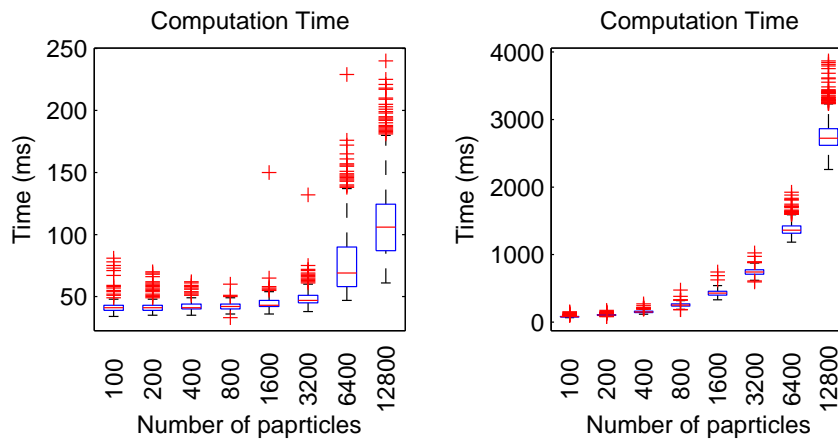


Figure 6.8. Boxplots showing computation time of our tracking (left) and the PCL tracking (right) on the “Tide” synthetic sequence. For both approaches, time linearly increases as the number of the particles N increases, but our increasing rate is much smaller. Note that the computation time of our approach is less than 50 ms until $N \leq 3200$.

a set of extensive experiments with both synthetic and real RGB-D sequences, we verified that our approach is not only faster but also more accurate than the PCL tracking.

CHAPTER VII

CONCLUSIONS

In this chapter, we first summarize the objectives and contributions of this thesis in Section 7.1, and then important conclusions are highlighted in detail in Section 7.2. To guide readers to possible future directions, we discuss possible extensions of the presented approaches as well as several future directions in Section 7.3. Finally, we finish this thesis with concluding remarks in Section 7.4.

7.1 Summary

In this thesis, we have contributed toward robust object perception capable of running in unstructured environments. More specifically, four critical challenges in visual object perception have been explicitly addressed. To perceive objects regardless of the degree of texture, we combined photometric and geometric features available from 3D object prior knowledge, and this combination also makes our approach more effective by reducing search space. In addition, object perception in highly cluttered backgrounds was addressed by employing robust multiple hypotheses frameworks: a particle filtering framework for robust object pose tracking and a voting framework using pair features for effective pose estimation. Several object discontinuities during tracking were taken into account to enable our tracking approach to be fully automatic in the presence of the discontinuities. Lastly, we accelerated repetitive computations of our frameworks in a modern parallel computation architecture so that our pose estimation and tracking approaches comply with real-time constraints. Our approaches were compared with other state-of-the-art work in various synthetic and real datasets, and advantages and limitations of the proposed systems were discussed both qualitatively and quantitatively.

7.2 Conclusions

Important conclusions drawn from the thesis are highlighted as follows:

- **Combining photometric and geometric features enables object perception to be more efficient and to increase the scope of objects.** While some photometric features representing texture of objects are well suited for well-textured objects, geometric features are more effective for textureless objects. As these two sets of features complementary each other, we considered several combinations of those and verified that fusing photometric and geometric features enables perception system to handle both well-textured and textureless objects. In addition, we further learned that a well-designed combination of these features helps to reduce search space of perception problems. In Section 3, a photometric feature (SURF keypoints) was employed to hypothesize initial particles instead of searching for the initial pose from scratch, and a geometric feature

(edges from geometric shapes) was considered to calculate inter-frame motions as a local search scheme. In Section 5.1, the color point pair feature (CPPF) was defined by combining photometric (colors) and geometric (3D points associated with 3D normals) features. Since the more descriptive CPPF helps the voting framework to prune out possibly wrong feature matchings, it reported much better pose estimation results than the one without color information in terms of both precision and recall. These features were further utilized in our object pose tracking approach using an RGB-D sensor in Section 6, where 3D points along with their colors and surface normals were used to calculate the likelihood of each particle pose given an RGB-D scene. We noticed that these three features (3D points, normals, and colors) are general attributes in the sense that they can be applied to either well-textured or textureless objects.

- **Multiple hypotheses frameworks are effective in heavy background clutter.** The multiple pose hypotheses frameworks for pose estimation in Section 5 and tracking in Section 3, 4, and 6 show that maintaining multiple pose hypotheses makes object perception more robust in significant clutter. As shown in Section 3.4.2, the particle filter tracker reports reliable tracking despite cluttered backgrounds, while the single hypothesis tracker gets stuck in local minima led by false edge associations from clutter. For pose estimation, the voting framework is indeed robust in cluttered environments by aggregating pose hypotheses determined from a large set of pair feature matching. In Section 5.1, we showed that the voting approach reports better recognition performance than the template matching and the sampling approach especially in cluttered backgrounds.
- **Integrating pose estimation and tracking drives a fully automatic pose tracking system capable of handling object discontinuous cases.** Most of the existing tracking approaches did not address object discontinuities, such as significant occlusions or disappearing. As a way of addressing these cases, we presented a systematic combination of pose estimation and tracking for which a re-initialization method using the effective number of particles as a likelihood of sounding tracking was proposed (Section 3 and 4). This scheme turned out to be an effective way to overcome the tracking discontinuities.
- **Object perception algorithms can be beneficiaries of modern parallel computing architecture.** As explained in both Section 5.1 and 6, the proposed frameworks in this thesis are composed of a significant amount of repetitive computations. Thanks to the independence of each computation, it is possible to perform the computation in parallel. By parallelizing on GPU, we achieved significant computational performance enhancement so that our framework can comply with the constraints of time. So one conclusion is that heavy computations in perception algorithms can take advantage of parallel computing power. Computer vision and robotic perception require a large number of computations, and we believe that designing parallel algorithms will be an important effort.
- **Depth information enables us to consider many rich visual features.** The RGB-D sensor is promising

in that reasonable depth information is available with reasonable price. Reliable depth information provides various geometric features, such as depth points, surface normal, object boundaries, 3D line segments, and principal curvatures, which are beyond currently dominant photometric features in computer vision. It also makes object segmentation trivial by looking at depth discontinuities. Due to these advantages, active adoption of geometric features along with photometric features is highly anticipated in the future robotics and perception research.

7.3 *Future Directions*

Despite our endeavors toward robust object perception, numerous challenges are still remained. This section describes several of these to guide readers to possible future directions.

Object model adaptation. As more 3D models are available on web databases, exploring how to take the discrepancy between the 3D models and the real objects into account will be an interesting topic. It would be an ideal case if a perfect 3D model of an object of interest is available, but it is neither practically possible nor seemingly necessary. Though an exact model may not always be available, we can still access to various shape prior and utilize the prior knowledge to generate a model matched to our object via a geometric deformation or adaptation. [Klank et al. \(2009\)](#) showed an initial effort morphing exemplar models into more accurate models, but it only focused on geometric adaptation. Updating photometric attributes as well as geometric deformation would be beneficial to lots of appearance-based approaches.

Active object modeling. As an alternative to the model adaptation, 3D object modeling from an **RGB-D** sensor would be a good research direction. Autonomous object modeling ([Krainin et al. 2011](#)) is preferred in service robotics where the number of objects robots needed to handle is reasonably limited. Dense environment reconstruction using the **RGB-D** sensor has been shown ([Newcombe et al. 2011](#)), and hence it would be straightforward to extend to object modeling. Whereas these approaches rely only on visual sensor modalities, robots can exploit other sensing modalities as they are embodied systems. Tactile sensing provides another sensory information which complements the visual sensory data. Thus, it is of interest to model 3D shape with a manipulation interaction ([Allen 1985, 1988](#), [Ilonen et al. 2013](#)), and it would be possible to model very challenging objects, such as articulated objects ([Katz et al. 2013](#)) or transparent objects (Section 4) that are very hard to build 3D models only with visual sensory input.

Multi-object tracking. In visual object tracking, it would be of interest to work on tracking multiple objects. As robotic tasks are getting complex in various robotic applications, such as flexible factory assembly and human-robot collaboration, robots are required to reason beyond a single object. In this thesis, we have addressed single object instance tracking where any visual regions other than the target object's region are regarded as background clutter or occlusions. If there are multiple objects and they interact each other, we may take advantage of important information from the interaction. More importantly, estimating multiple poses would help occlusion reasoning. When robots perform manipulation tasks, it is quite often to face occlusions by other entities, such as robots themselves, objects, or humans. Multi-object tracking might be helpful in this case since we can rule out some possibly

occluded regions in the stage of measurement data association. A preliminary idea was shown in the 2D people detection problem (Shu et al. 2012), and applying this to 3D object tracking would be highly expected. This explicit occlusion reasoning may significantly increase the accuracy of pose estimate for each object and consequently result in a robust tracking solution even with severe occlusions.

Scalable object perception. As robotic systems are deployed in real environments for practical tasks, scalable object perception will be an important topic. The time complexity of most approaches in object perception linearly increases as the number of learned objects increases. Since robots are expected to deal with hundreds or thousands of objects within a time constraint, research efforts on devising scalable algorithms would be highly anticipated. Computer vision literature has started to address the scalable recognition (Nister and Stewenius 2006, Deng et al. 2009), but scalable object perception on 6-DOF space is still little explored.

3D object categorization. Lastly, utilizing 3D prior knowledge of objects for object category recognition would be a promising topic. In this thesis, we have focused on object instance recognition in which the primary goal is to find the exact object rather than generalizing objects at categorical levels. However, inferring categories of unknown objects may be an interesting research topic as functional prediction of the objects is often associated with the object categories. In computer vision community, preliminary results have been shown (Savarese and Fei-Fei 2007, Stark et al. 2010, Liebelt and Schmid 2010, Pepik et al. 2012). Dominant philosophy is to adopt the well-known deformable part model (Felzenszwalb et al. 2009) along with the histogram of oriented gradients (HOG) feature (Dalal et al. 2005) for addressing intra-class variations and to learn a model from multiple 3D CAD models. s Though these approaches have shown great potential, recognition rate is far less than our expectation and employed features are still limited to photometric features. Given the prevalence of depth sensors, it is of interest to exploit geometric features for predicting categories of 3D objects.

7.4 Concluding Remarks

This thesis has explored how to enhance object perception algorithms so that they can dependably execute in highly cluttered and dynamic environments. Among many remained challenges in object perception, the four challenges in unstructured scenes were addressed in order for robotic systems to efficiently recognize objects and estimate their poses regardless of the degree of object texture, background clutter, and object discontinuous cases.

Despite the prolonged efforts, object perception is certainly not solved yet, and significant amount of work should thus follow in the future. We believe that exploiting 3D prior of objects would be crucial for visual object perception. The prevalent RGB-D cameras and various depth sensors might enable to revisit the geometric era of the early machine perception and to combined geometric information with photometric observations. Imagine that future perception systems are actively learning new objects along with their strategies for behavior in the lifelong scale and that the revolutionized robotic systems will perform tasks that are seemingly impossible missions for now. We hope that our efforts in this thesis would be a step forward toward reliable and dependable object perception.

APPENDIX A

IRLS AND ITS JACOBIAN DERIVATION

Given the current pose hypothesis \mathbf{X}_t , the residual vector $\hat{\mathbf{r}} \in \mathbb{R}^{N_z}$ which represents Euclidean distances between the 2D projected sampled points \mathbf{p} and their corresponding nearest edge points \mathbf{q} , we would like to find $\hat{\boldsymbol{\mu}} \in \mathbb{R}^6$ which minimizes the residual $\hat{\mathbf{r}}$ as follows:

$$\hat{\boldsymbol{\mu}} = \arg \min_{\boldsymbol{\mu}} \sum_{i=1}^{N_z} \|\hat{r}_i\|^2 \quad (\text{A.1})$$

$$= \arg \min_{\boldsymbol{\mu}} \sum_{i=1}^{N_z} \|\mathbf{q}_i - \mathbf{p}_i\|^2. \quad (\text{A.2})$$

Note that each element of $\boldsymbol{\mu}$ is the coefficient of each basis of the Lie algebra $\mathfrak{se}(3)$ in (3.6). Thus, the optimized pose hypothesis $\hat{\mathbf{X}}_t$ can be presented as:

$$\hat{\mathbf{X}}_t = \mathbf{X}_t \cdot \exp\left(\sum_{i=1}^6 \hat{\mu}_i \mathbf{E}_i\right) \quad (\text{A.3})$$

where $\exp : \mathfrak{se}(3) \mapsto SE(3)$ is the exponential map. From the camera model in (3.1), the projected point \mathbf{p}_i in (A.2) of the i^{th} 3D sampled point in object coordinates $\mathbf{P}_i^{\mathcal{O}} = (x_i^{\mathcal{O}}, y_i^{\mathcal{O}}, z_i^{\mathcal{O}}, 1)^{\top}$ is presented by

$$\mathbf{p}_i = \begin{pmatrix} u_i \\ v_i \end{pmatrix} = \text{Project}(\mathbf{K}, \hat{\mathbf{X}}_t, \mathbf{P}_i^{\mathcal{O}}) \quad (\text{A.4})$$

where \mathbf{K} is the intrinsic parameter matrix (3.3).

Given \mathbf{p}_i and its unit normal vector \mathbf{n}_i , we can build a Jacobian matrix $\mathbf{J} \in \mathbb{R}^{N_z \times 6}$ by computing the partial derivative of $\mathbf{n}_i^{\top} \mathbf{p}_i$ in the direction μ_j at $\boldsymbol{\mu} = \mathbf{0}$:

$$J_{ij} = \left. \frac{\partial \mathbf{n}_i^{\top} \mathbf{p}_i}{\partial \mu_j} \right|_{\boldsymbol{\mu}=\mathbf{0}} \quad (\text{A.5})$$

$$= \mathbf{n}_i^{\top} \left. \frac{\partial}{\partial \mu_j} \begin{pmatrix} u_i \\ v_i \end{pmatrix} \right|_{\boldsymbol{\mu}=\mathbf{0}} \quad (\text{A.6})$$

$$= \mathbf{n}_i^{\top} \left. \frac{\partial}{\partial \mu_j} \left(\text{Project}(\mathbf{K}, \hat{\mathbf{X}}_t, \mathbf{P}_i^{\mathcal{O}}) \right) \right|_{\boldsymbol{\mu}=\mathbf{0}} \quad (\text{A.7})$$

in order to find $\hat{\boldsymbol{\mu}}$ which minimizes the residual $\hat{\mathbf{r}}$ by solving the following equation:

$$\mathbf{J} \hat{\boldsymbol{\mu}} = \hat{\mathbf{r}}. \quad (\text{A.8})$$

For convenience, we can split $\text{Project}()$ in the camera model (3.1) into two parts as follows:

$$\begin{pmatrix} u_i \\ v_i \end{pmatrix} = \begin{pmatrix} f_u & 0 & u_0 \\ 0 & f_v & v_0 \end{pmatrix} \begin{pmatrix} \tilde{u}_i \\ \tilde{v}_i \\ 1 \end{pmatrix} \quad (\text{A.9})$$

$$\begin{pmatrix} \tilde{u}_i \\ \tilde{v}_i \end{pmatrix} = \begin{pmatrix} \frac{x_i^c}{z_i^c} \\ \frac{y_i^c}{z_i^c} \end{pmatrix}. \quad (\text{A.10})$$

Their corresponding Jacobian matrices are then obtained as:

$$\mathbf{J}_K = \begin{pmatrix} \frac{\partial u_i}{\partial \tilde{u}_i} & \frac{\partial u_i}{\partial \tilde{v}_i} \\ \frac{\partial v_i}{\partial \tilde{u}_i} & \frac{\partial v_i}{\partial \tilde{v}_i} \end{pmatrix} = \begin{pmatrix} f_u & 0 \\ 0 & f_v \end{pmatrix} \quad (\text{A.11})$$

$$\mathbf{J}_P = \begin{pmatrix} \frac{\partial \tilde{u}_i}{\partial x_i^c} & \frac{\partial \tilde{u}_i}{\partial y_i^c} & \frac{\partial \tilde{u}_i}{\partial z_i^c} \\ \frac{\partial \tilde{v}_i}{\partial x_i^c} & \frac{\partial \tilde{v}_i}{\partial y_i^c} & \frac{\partial \tilde{v}_i}{\partial z_i^c} \end{pmatrix} = \begin{pmatrix} \frac{1}{z_i^c} & 0 & -\frac{x_i^c}{(z_i^c)^2} \\ 0 & \frac{1}{z_i^c} & -\frac{y_i^c}{(z_i^c)^2} \end{pmatrix}. \quad (\text{A.12})$$

If we calculate the partial derivative of the $\dot{\mathbf{X}}_t$ in the direction μ_j at $\boldsymbol{\mu} = \mathbf{0}$, we have the following from the equation of optimized pose in (A.3):

$$\left. \frac{\partial \dot{\mathbf{X}}_t}{\partial \mu_j} \right|_{\boldsymbol{\mu}=\mathbf{0}} = \mathbf{X}_t \left. \frac{\partial}{\partial \mu_j} \exp(\sum_{i=1}^6 \mu_i \mathbf{E}_i) \right|_{\boldsymbol{\mu}=\mathbf{0}} \quad (\text{A.13})$$

$$= \mathbf{X}_t \mathbf{E}_j. \quad (\text{A.14})$$

We then obtain the partial derivative of the \mathbf{P}_i^c in the direction μ_j at $\boldsymbol{\mu} = \mathbf{0}$ via the transformation (3.2) from object to camera coordinate frames:

$$\begin{aligned} \left. \frac{\partial \mathbf{P}_i^c}{\partial \mu_j} \right|_{\boldsymbol{\mu}=\mathbf{0}} &= \left. \frac{\partial \dot{\mathbf{X}}_t \mathbf{P}_i^{\mathcal{O}}}{\partial \mu_j} \right|_{\boldsymbol{\mu}=\mathbf{0}} \\ &= \left. \frac{\partial \dot{\mathbf{X}}_t}{\partial \mu_j} \right|_{\boldsymbol{\mu}=\mathbf{0}} \mathbf{P}_i^{\mathcal{O}} \\ &= \mathbf{X}_t \mathbf{E}_j \mathbf{P}_i^{\mathcal{O}}. \end{aligned} \quad (\text{A.15})$$

With the two Jacobian matrices in (A.11), (A.12) and the partial derivative of the \mathbf{P}_i^c in (A.15), we finally obtain the Jacobian matrix \mathbf{J} from the intermediate result in (A.7) as

$$J_{ij} = \mathbf{n}_i^\top \left. \frac{\partial}{\partial \mu_j} \left(\text{Project}(\mathbf{K}, \dot{\mathbf{X}}_t, \mathbf{P}_i^{\mathcal{O}}) \right) \right|_{\boldsymbol{\mu}=\mathbf{0}} \quad (\text{A.16})$$

$$= \mathbf{n}_i^\top \mathbf{J}_K \begin{pmatrix} \mathbf{J}_P & 0 \\ 0 & 0 \end{pmatrix} \left. \frac{\partial \mathbf{P}_i^c}{\partial \mu_j} \right|_{\boldsymbol{\mu}=\mathbf{0}} \quad (\text{A.17})$$

$$= \mathbf{n}_i^\top \mathbf{J}_K \begin{pmatrix} \mathbf{J}_P & 0 \\ 0 & 0 \end{pmatrix} \mathbf{X}_t \mathbf{E}_j \mathbf{P}_i^{\mathcal{O}}. \quad (\text{A.18})$$

We could solve the equation (A.8) using the general pseudo-inverse of \mathbf{J} :

$$\hat{\boldsymbol{\mu}} = (\mathbf{J}^T \mathbf{J})^{-1} \mathbf{J}^T \hat{\mathbf{r}}. \quad (\text{A.19})$$

Above equation, however, tends to be sensitive to erroneous measurements. As alternatives, M-estimators (Huber 1981) or RANSAC (Fischler and Bolles 1981) are preferable. Especially, M-estimator, in which a robust penalty function to the residuals is applied, is preferred to RANSAC when the majority of measurements is inliers. For a M-estimator, it is common to define a function $\Psi(r) = \frac{1}{\lambda_w + |r|}$ which penalizes for the residual errors r and λ_w is a parameter which is usually set to one standard deviation of the inliers. With this function, we can have a diagonal matrix $\mathbf{W} = \text{diag}[\Psi(r_1), \Psi(r_2), \dots, \Psi(r_{N_z})] \in \mathbb{R}^{N_z \times N_z}$. Then $\hat{\boldsymbol{\mu}}$ can be estimated via weighted least squares as:

$$\hat{\boldsymbol{\mu}} = (\mathbf{J}^T \mathbf{W} \mathbf{J})^{-1} \mathbf{J}^T \mathbf{W} \hat{\mathbf{r}}. \quad (\text{A.20})$$

Drummond and Cipolla (2002) introduced this as Iteratively Reweighted Least Squares (IRLS) in which the penalty function is recursively updated by computing the weighted least squares problem. They mentioned performing a single iteration for each image frame was enough to converge to the true pose.

BIBLIOGRAPHY

- Agin, G. J. (1972). Representation and description of curved objects. Technical report, DTIC Document.
- Agrawal, A., Yu, S., Barnwell, J., and Raskar, R. (2010). Vision-guided robot system for picking objects by casting shadows. *International Journal of Robotics Research (IJRR)*, 29(2–3):155–173.
- Aldoma, A., Vincze, M., Blodow, N., Gossow, D., Gedikli, S., Rusu, R., and Bradski, G. (2011). CAD-model recognition and 6DOF pose estimation using 3D cues. In *IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, pages 585–592.
- Allen, P. K. (1985). *Object recognition using vision and touch*. PhD thesis, University of Pennsylvania.
- Allen, P. K. (1988). Integrating vision and touch for object recognition tasks. *International Journal of Robotics Research (IJRR)*, 7(6):15–33.
- AMD (2012). Bolt C++ template library. <http://developer.amd.com/tools-and-sdks/heterogeneous-computing/bolt-c-template-library/>.
- Amit, Y. and Geman, D. (1997). Shape quantization and recognition with randomized trees. *Neural computation*, 9(7):1545–1588.
- Armstrong, M. and Zisserman, A. (1995). Robust object tracking. In *Proceedings of Asian Conference on Computer Vision (ACCV)*, volume 1, pages 58–61.
- Azad, P., Munch, D., Asfour, T., and Dillmann, R. (2011). 6-DoF model-based tracking of arbitrarily shaped 3D objects. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, pages 5204–5209.
- Ballard, D. (1981). Generalizing the hough transform to detect arbitrary shapes. *Pattern Recognition*, 13(2):111–122.
- Barrow, H. G., Tenenbaum, J. M., Bolles, R. C., and Wolf, H. C. (1977). Parametric correspondence and chamfer matching: Two new techniques for image matching. In *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI)*, volume 2, pages 659–663.
- Bay, H., Ess, A., Tuytelaars, T., and Gool, L. V. (2008). Speeded-up robust features (SURF). *Computer Vision and Image Understanding (CVIU)*, 110(3):346–359.
- Beis, J. and Lowe, D. (1997). Shape indexing using approximate nearest-neighbour search in high-dimensional spaces. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1000–1006.
- Bersch, C., Pangercic, D., Osentoski, S., Hausman, K., Marton, Z.-C., Ueda, R., Okada, K., and Beetz, M. (2012). Segmentation of textured and textureless objects through interactive perception. In *RSS Workshop on Robots in Clutter: Manipulation, Perception and Navigation in Human Environments*.

- Besl, P. J. and McKay, N. D. (1992). A method for registration of 3-D shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, pages 239–256.
- Bibby, C. and Reid, I. (2008). Robust real-time visual tracking using pixel-wise posteriors. In *Proceedings of European Conference on Computer Vision (ECCV)*, pages 831–844. Springer.
- Biederman, I. (1985). Human image understanding: Recent research and a theory. *Computer Vision, Graphics, and Image Processing*, 32(1):29–73.
- Binford, T. O. (1971). Visual perception by computer. In *IEEE conference on Systems and Control*, volume 261, page 262.
- Bradski, G. and Kaehler, A. (2008). *Learning OpenCV: Computer vision with the OpenCV library*. O’Reilly Media.
- Bray, M., Koller-Meier, E., and Gool, L. V. (2004). Smart particle filtering for 3D hand tracking. In *Sixth IEEE International Conference on Automatic Face and Gesture Recognition, 2004. Proceedings*, pages 675–680.
- Breiman, L. (2001). Random forests. *Machine learning*, 45(1):5–32.
- Brooks, R. (1982). Symbolic reasoning among 3D models and 2D images. *Artificial Intelligence*, 17:285–348.
- Canny, J. (1986). A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 8(6):679–698.
- Chen, B. X. (2014). For hints at Apple’s plans, read its shopping list. *The New York Times*. <http://nyti.ms/1pgvPRT>.
- Chiuso, A. and Soatto, S. (2000). Monte Carlo filtering on Lie groups. In *IEEE Conference on Decision and Control*, volume 1, pages 304–309.
- Choi, C. and Christensen, H. (2012a). 3D pose estimation of daily objects using an RGB-D camera. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.
- Choi, C. and Christensen, H. (2012b). 3D textureless object detection and tracking: An edge-based approach. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.
- Choi, C. and Christensen, H. I. (2010). Real-time 3D model-based tracking using edge and keypoint features for robotic manipulation. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, pages 4048–4055.
- Choi, C. and Christensen, H. I. (2011). Robust 3D visual tracking using particle filtering on the SE(3) group. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*.
- Choi, C. and Christensen, H. I. (2012c). Robust 3D visual tracking using particle filtering on the special Euclidean group: A combined approach of keypoint and edge features. *International Journal of Robotics Research (IJRR)*, 31(4):498–519.

- Choi, C. and Christensen, H. I. (2013). RGB-D object tracking: A particle filter approach on GPU. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1084–1091.
- Choi, C. and Christensen, H. I. (2014). RGB-D object pose estimation in unstructured environments. *International Journal of Robotics Research (IJRR)*, (Under review).
- Choi, C., Taguchi, Y., Tuzel, O., Liu, M., and Ramalingam, S. (2012). Voting-Based pose estimation for robotic assembly using a 3D sensor. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*.
- Choi, W., Pantofaru, C., and Savarese, S. (2011). Detecting and tracking people using an RGB-D camera via multiple detector fusion. In *IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, pages 1076–1083.
- Collet, A., Berenson, D., Srinivasa, S. S., and Ferguson, D. (2009). Object recognition and full pose registration from a single image for robotic manipulation. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, pages 48–55.
- Collet, A., Martinez, M., and Srinivasa, S. (2011). The MOPED framework: Object recognition and pose estimation for manipulation. *International Journal of Robotics Research (IJRR)*, 30(10):1284–1306.
- Comport, A. I., Marchand, E., and Chaumette, F. (2004). Robust model-based tracking for robot vision. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, volume 1.
- Criminisi, A. and Shotton, J. (2013). *Decision Forests for Computer Vision and Medical Image Analysis*. Springer Publishing Company, Incorporated.
- Dalal, N., Triggs, B., Rhone-Alps, I., and Montbonnot, F. (2005). Histograms of oriented gradients for human detection. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1.
- Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 248–255. IEEE.
- Deutscher, J., Blake, A., and Reid, I. (2000). Articulated body motion capture by annealed particle filtering. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 126–133 vol.2.
- Dickinson, S. (2009). The evolution of object categorization and the challenge of image abstraction. *Object Categorization: Computer and Human Vision Perspectives*, 7.
- Dickinson, S. J., Pentland, A., and Rosenfeld, A. (1992). 3-d shape recovery using distributed aspect matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 14(2):174–198.

- Dorai, C. and Jain, A. (1997). COSMOS-A representation scheme for 3D free-form objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 19(10):1115–1130.
- Doucet, A., De Freitas, N., and Gordon, N. (2001). *Sequential Monte Carlo methods in practice*, volume 1. Springer New York.
- Doucet, A., Godsill, S., and Andrieu, C. (2000). On sequential Monte Carlo sampling methods for Bayesian filtering. *Statistics and computing*, 10(3):197–208.
- Drost, B., Ulrich, M., Navab, N., and Ilic, S. (2010). Model globally, match locally: Efficient and robust 3D object recognition. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.
- Drummond, T. and Cipolla, R. (2002). Real-time visual tracking of complex structures. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 24(7):932–946.
- Epstein, Z. (2013). Microsoft says Xbox 360 sales have surpassed 76 million units, Kinect sales top 24 million. *BGR*. <http://bgr.com/2013/02/12/microsoft-xbox-360-sales-2013-325481/>.
- Fanelli, G., Gall, J., and Van Gool, L. (2011). Real time head pose estimation with random regression forests. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 617–624.
- Felzenszwalb, P. F., Girshick, R. B., McAllester, D., and Ramanan, D. (2009). Object detection with discriminatively trained part based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*.
- Fiala, M. (2005). ARTag, a fiducial marker system using digital techniques. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 590–596.
- Fischler, M. A. and Bolles, R. C. (1981). Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395.
- Fox, D. (2003). Adapting the sample size in particle filters through KLD-sampling. *International Journal of Robotics Research (IJRR)*, 22(12):985–1003.
- Google (2013). Google 3D warehouse. <http://sketchup.google.com/3dwarehouse/>.
- Google (2014). Project Tango. <https://www.google.com/atap/projecttango/>.
- Gordon, N. J., Salmond, D. J., and Smith, A. F. (1993). Novel approach to nonlinear/non-Gaussian Bayesian state estimation. In *IEE Proceedings F Radar and Signal Processing*, volume 140, pages 107–113.
- Guzman, A. (1971). Analysis of curved line drawings using context and global information. *Machine intelligence*, 6:325–375.
- Harris, C. (1992). *Tracking with Rigid Objects*. MIT Press.

- Henry, P., Krainin, M., Herbst, E., Ren, X., and Fox, D. (2010). RGB-D mapping: Using depth cameras for dense 3D modeling of indoor environments. In *Proceedings of International Symposium on Experimental Robotics (ISER)*.
- Herbst, E., Henry, P., Ren, X., and Fox, D. (2011). Toward object discovery and modeling via 3-d scene comparison. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, pages 2623–2629.
- Hinterstoisser, S., Cagniart, C., Ilic, S., Sturm, P., Navab, N., Fua, P., and Lepetit, V. (2012a). Gradient response maps for Real-Time detection of Texture-Less objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 34(5):876–888.
- Hinterstoisser, S., Lepetit, V., Ilic, S., Holzer, S., Bradski, G., Konolige, K., and Navab, N. (2012b). Model based training, detection and pose estimation of texture-less 3D objects in heavily cluttered scenes. In *Proceedings of Asian Conference on Computer Vision (ACCV)*.
- Hoberock, J. and Bell, N. (2010). Thrust: A parallel template library. <http://thrust.github.io/>.
- Huber, P. J. (1981). *Robust Statistics*. Wiley-Interscience.
- Huttenlocher, D. P., Klanderman, G. A., and Rucklidge, W. A. (1993). Comparing images using the Hausdorff distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, pages 850–863.
- Huttenlocher, D. P. and Ullman, S. (1987). Object recognition using alignment. In *Proceedings of IEEE International Conference on Computer Vision (ICCV)*, pages 102–111.
- Ikeuchi, K. and Kanade, T. (1988). Applying sensor models to automatic generation of object recognition programs. In *Proceedings of IEEE International Conference on Computer Vision (ICCV)*, volume 88, pages 228–237.
- Ilonen, J., Bohg, J., and Kyrki, V. (2013). Fusing visual and tactile sensing for 3-D object reconstruction while grasping. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, pages 3547–3554.
- Internet.org (2013). A focus on efficiency: A whitepaper from Facebook, Ericsson and Qualcomm. <http://internet.org/efficiencypaper>.
- Isard, M. and Blake, A. (1998). Condensation—conditional density propagation for visual tracking. *International Journal of Computer Vision (IJCV)*, 29(1):5–28.
- Johnson, A. E. and Hebert, M. (1999). Using spin images for efficient object recognition in cluttered 3D scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 21(5):433–449.
- Kasper, A., Xue, Z., and Dillmann, R. (2012). The KIT object models database: An object model database for object recognition, localization and manipulation in service robotics. *International Journal of Robotics Research (IJRR)*, 31(8):927–934.

- Katz, D., Kazemi, M., Begnell, J. A., and Stentz, A. (2013). Interactive segmentation, tracking, and kinematic modeling of unknown 3D articulated objects. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*.
- Kazhdan, M., Bolitho, M., and Hoppe, H. (2006). Poisson surface reconstruction. In *Proceedings of Eurographics Symposium on Geometry processing*.
- Kemp, C. and Drummond, T. (2005). Dynamic measurement clustering to aid real time tracking. In *Proceedings of IEEE International Conference on Computer Vision (ICCV)*, pages 1500–1507.
- Kim, E. and Medioni, G. (2011). 3D object recognition in range images using visibility context. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3800–3807.
- Klank, U., Zia, M. Z., and Beetz, M. (2009). 3D model selection from an internet database for robotic vision. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*.
- Klein, G. and Drummond, T. (2004). Tightly integrated sensor fusion for robust visual tracking. *Image and Vision Computing*, 22(10):769–776.
- Klein, G. and Murray, D. (2006). Full-3d edge tracking with a particle filter. *Proceedings of British Machine Vision Conference (BMVC)*.
- Krainin, M., Henry, P., Ren, X., and Fox, D. (2011). Manipulator and object tracking for in-hand 3D object modeling. *International Journal of Robotics Research (IJRR)*, 30(11):1311–1327.
- Krainin, M., Konolige, K., and Fox, D. (2012). Exploiting segmentation for robust 3D object matching. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*.
- Kratochvil, B. E., Dong, L., and Nelson, B. J. (2009). Real-time rigid-body visual tracking in a scanning electron microscope. *International Journal of Robotics Research (IJRR)*, 28(4):498–511.
- Kwon, J., Choi, M., Park, F. C., and Chun, C. (2007). Particle filtering on the Euclidean group: framework and applications. *Robotica*, 25(06):725–737.
- Kwon, J. and Park, F. C. (2010). Visual tracking via particle filtering on the affine group. *International Journal of Robotics Research (IJRR)*, 29(2-3):198.
- Kyrki, V. and Kragic, D. (2005). Integration of model-based and model-free cues for visual object tracking in 3d. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, volume 2, pages 1566–1572.
- Lai, K., Bo, L., Ren, X., and Fox, D. (2011). A scalable tree-based approach for joint object and pose recognition. In *AAAI Conference on Artificial Intelligence*.

- Lamdan, Y. and Wolfson, H. J. (1988). Geometric hashing: A general and efficient model-based recognition scheme. In *Proceedings of IEEE International Conference on Computer Vision (ICCV)*, pages 238–249.
- Lepetit, V. and Fua, P. (2006). Keypoint recognition using randomized trees. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 28(9):1465–1479.
- Lepetit, V., Moreno-Noguer, F., and Fua, P. (2009). EPnP: An accurate $O(n)$ solution to the PnP problem. *International Journal of Computer Vision (IJCV)*, 81(2):155–166.
- Liebelt, J. and Schmid, C. (2010). Multi-view object class detection with a 3d geometric model. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1688–1695. IEEE.
- Liu, M.-Y., Tuzel, O., Veeraraghavan, A., and Chellappa, R. (2010a). Fast directional chamfer matching. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1696–1703.
- Liu, M.-Y., Tuzel, O., Veeraraghavan, A., Chellappa, R., Agrawal, A., and Okuda, H. (2010b). Pose estimation in heavy clutter using a multi-flash camera. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, pages 2028–2035.
- Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision (IJCV)*, 60(2):91–110.
- Ma, Y., Soatto, S., Kosecká, J., and Sastry, S. S. (2004). *An invitation to 3-D vision: From images to geometric models*. Springer, New York.
- Mateo Lozano, O. and Otsuka, K. (2009). Real-time visual tracker by stream processing. *Journal of Signal Processing Systems*, 57(2):285–295.
- Mian, A. S., Bennamoun, M., and Owens, R. (2006). Three-dimensional model-based object recognition and segmentation in cluttered scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, pages 1584–1601.
- Moakher, M. (2003). Means and averaging in the group of rotations. *SIAM Journal on Matrix Analysis and Applications*, 24(1):1–16.
- Montemayor, A. S., Pantrigo, J. J., Sánchez, A., and Fernández, F. (2004). Particle filter on GPUs for real-time tracking. In *ACM SIGGRAPH 2004 Posters*, page 94.
- Mörwald, T., Prankl, J., Richtsfeld, A., Zillich, M., and Vincze, M. (2010). BLORT - the blocks world robotic vision toolbox. In *In Best Practice in 3D Perception and Modeling for Mobile Manipulation (in conjunction with ICRA 2010)*.
- Mörwald, T., Zillich, M., and Vincze, M. (2009). Edge tracking of textured objects with a recursive particle filter. In *Proceedings of International Conference on Computer Graphics and Vision (Graphicon)*, pages 96–103.

- Mundy, J. (2006). Object recognition in the geometric era: A retrospective. *Toward category-level object recognition*, pages 3–28.
- Murase, H. and Nayar, S. K. (1995). Visual learning and recognition of 3-d objects from appearance. *International Journal of Computer Vision (IJCV)*, 14(1):5–24.
- Nevatia, R. and Binford, T. O. (1973). Structured descriptions of complex objects. In *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI)*, pages 641–647.
- Nevatia, R. and Binford, T. O. (1977). Description and recognition of curved objects. *Artificial Intelligence*, 8(1):77–98.
- Newcombe, R., Izadi, S., Hilliges, O., Molyneaux, D., Kim, D., Davison, A., Kohli, P., Shotton, J., Hodges, S., and Fitzgibbon, A. (2011). KinectFusion: real-time dense surface mapping and tracking. In *Proceedings of International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 127–136.
- Nister, D. and Stewenius, H. (2006). Scalable recognition with a vocabulary tree. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 2161–2168. IEEE.
- Olson, C. and Huttenlocher, D. (1997). Automatic target recognition by matching oriented edge pixels. *IEEE Transactions on Image Processing*, 6(1):103–113.
- Papazov, C., Haddadin, S., Parusel, S., Krieger, K., and Burschka, D. (2012). Rigid 3D geometry matching for grasping of known objects in cluttered scenes. *International Journal of Robotics Research (IJRR)*, 31(4):538–553.
- Park, I. K., Germann, M., Breitenstein, M. D., and Pfister, H. (2010). Fast and automatic object pose estimation for range images on the GPU. *Machine Vision and Applications*, 21(5):749–766.
- Park, W., Liu, Y., Zhou, Y., Moses, M., and Chirikjian, G. S. (2008). Kinematic state estimation and motion planning for stochastic nonholonomic systems using the exponential map. *Robotica*, 26(4):419–434.
- Pauwels, K., Rubio, L., Diaz, J., and Ros, E. (2013). Real-time model-based rigid object pose estimation and tracking combining dense and sparse visual cues. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 2347–2354.
- Pepik, B., Gehler, P., Stark, M., and Schiele, B. (2012). 3D2PM–3D deformable part models. In *Proceedings of European Conference on Computer Vision (ECCV)*, pages 356–370. Springer.
- Perkins, W. A. (1978). A model-based vision system for industrial parts. *IEEE Transactions on Computers*, C-27(2):126–143.
- Pham, M., Woodford, O., Perbet, F., Maki, A., Stenger, B., and Cipolla, R. (2011). A new distance for scale-invariant 3D shape recognition and registration. In *Proceedings of IEEE International Conference on Computer Vision (ICCV)*.

- Pinz, A. (2005). Object categorization. *Foundations and Trends® in Computer Graphics and Vision*, 1(4):255–353.
- Pressigout, M. and Marchand, E. (2006). Real-time 3d model-based tracking: Combining edge and texture information. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, pages 2726–2731.
- Prisacariu, V. A. and Reid, I. D. (2012). PWP3D: real-time segmentation and tracking of 3D objects. *International Journal of Computer Vision (IJCV)*, 98(3):335–354.
- Pupilli, M. and Calway, A. (2006). Real-time camera tracking using known 3D models and a particle filter. In *Proceedings of IEEE International Conference on Pattern Recognition (ICPR)*, volume 1.
- Raskar, R., Tan, K., Feris, R., Yu, J., and Turk, M. (2004). Non-photorealistic camera: Depth edge detection and stylized rendering using multi-flash imaging. *ACM Transactions on Graphics*, 23:679–688.
- Roberts, L. G. (1965). Machine perception of three-dimensional solids. In *Optical and Electrooptical Information Processing*. MIT Press.
- Rodrigues, J. J., Kim, J.-S., Furukawa, M., Xavier, J., Aguiar, P., and Kanade, T. (2012). 6D pose estimation of textureless shiny objects using random ferns for bin-picking. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3334–3341. IEEE.
- Ross, D. A., Lim, J., Lin, R. S., and Yang, M. H. (2008). Incremental learning for robust visual tracking. *International Journal of Computer Vision (IJCV)*, 77(1):125–141.
- Rosten, E. and Drummond, T. (2005). Fusing points and lines for high performance tracking. In *Proceedings of IEEE International Conference on Computer Vision (ICCV)*, volume 2.
- Rothganger, F., Lazebnik, S., Schmid, C., and Ponce, J. (2006). 3d object modeling and recognition using local affine-invariant image descriptors and multi-view spatial constraints. *International Journal of Computer Vision (IJCV)*, 66(3):231–259.
- Rusu, R. B., Blodow, N., and Beetz, M. (2009). Fast point feature histograms (FPFH) for 3D registration. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, pages 3212–3217.
- Rusu, R. B., Bradski, G., Thibaux, R., and Hsu, J. (2010). Fast 3D recognition and pose using the viewpoint feature histogram. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.
- Rusu, R. B. and Cousins, S. (2011). 3D is here: Point Cloud Library (PCL). In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–4.
- Salas-Moreno, R., Newcombe, R., Strasdat, H., Kelly, P., and Davison, A. (2013). SLAM++: simultaneous localisation and mapping at the level of objects. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

- Savarese, S. and Fei-Fei, L. (2007). 3D generic object categorization, localization and pose estimation. In *Proceedings of IEEE International Conference on Computer Vision (ICCV)*, pages 1–8. IEEE.
- Shotton, J., Fitzgibbon, A., Cook, M., Sharp, T., Finocchio, M., Moore, R., Kipman, A., and Blake, A. (2011). Real-time human pose recognition in parts from single depth images. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.
- Shu, G., Dehghan, A., Oreifej, O., Hand, E., and Shah, M. (2012). Part-based multiple-person tracking with partial occlusion handling. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1815–1821. IEEE.
- Stark, M., Goesele, M., and Schiele, B. (2010). Back to the future: Learning shape models from 3D CAD data. In *Proceedings of British Machine Vision Conference (BMVC)*, volume 2, page 5.
- Steder, B., Rusu, R., Konolige, K., and Burgard, W. (2011). Point feature extraction on 3d range scans taking into account object boundaries. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, pages 2601–2608.
- Stein, F. and Medioni, G. (1992). Structural indexing: Efficient 3-D object recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 14(2):125–145.
- Sturm, J., Engelhard, N., Endres, F., Burgard, W., and Cremers, D. (2012). A benchmark for the evaluation of RGB-D SLAM systems. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.
- Tamadazte, B., Marchand, E., Dembélé, S., and Fort-Piat, N. L. (2010). CAD model-based tracking and 3D visual-based control for MEMS microassembly. *International Journal of Robotics Research (IJRR)*, 29(11):1416–1434.
- Teulière, C., Marchand, E., and Eck, L. (2010). Using multiple hypothesis in model-based tracking. In *ICRA*.
- Thrun, S., Burgard, W., and Fox, D. (2005). *Probabilistic Robotics*. The MIT Press.
- TurboSquid (2014). Turbosquid. <http://www.turbosquid.com/>.
- Tuzel, O., Subbarao, R., and Meer, P. (2005). Simultaneous multiple 3D motion estimation via mode finding on lie groups. In *Proceedings of IEEE International Conference on Computer Vision (ICCV)*, pages 18–25.
- Underwood, S. A. and Coates Jr, C. L. (1975). Visual learning from multiple views. *IEEE Transactions on Computers*, 100(6):651–661.
- Vacchetti, L., Lepetit, V., and Fua, P. (2004). Combining edge and texture information for real-time accurate 3d camera tracking. In *Proceedings of International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 48–56.

- Wahl, E., Hillenbrand, U., and Hirzinger, G. (2003). Surflet-pair-relation histograms: A statistical 3D-shape representation for rapid classification. In *Proceedings of International Conference on 3-D Digital Imaging and Modeling (3DIM)*, pages 474–481.
- Wang, Y. and Chirikjian, G. S. (2006). Error propagation on the euclidean group with applications to manipulator kinematics. *IEEE Transactions on Robotics*, 22(4):591–602.
- Wang, Y. and Chirikjian, G. S. (2008). Nonparametric second-order theory of error propagation on motion groups. *International Journal of Robotics Research (IJRR)*, 27(11-12):1258–1273.
- Woodford, O., Pham, M., Maki, A., Perbet, F., and Stenger, B. (2011). Demisting the hough transform for 3D shape recognition and registration. In *Proceedings of British Machine Vision Conference (BMVC)*.
- Woodham, R. J. (1980). Photometric method for determining surface orientation from multiple images. *Optical engineering*, 19(1).
- Xavier, J. and Manton, J. H. (2006). On the generalization of AR processes to Riemannian manifolds. *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing*.
- Yesin, K. B. and Nelson, B. J. (2005). A CAD model based tracking system for visually guided microassembly. *Robotica*, 23(04):409–418.