UNIVERSITY OF SOUTHAMPTON

FACULTY OF PHYSICAL AND APPLIED SCIENCES

Electronics and Computer Science

On Evidence Gathering in 3D Point Clouds of Static and Moving Objects

by

Anas Abuzaina

Thesis for the degree of Doctor of Philosophy

June 2015

UNIVERSITY OF SOUTHAMPTON

FACULTY OF PHYSICAL AND APPLIED SCIENCES
Electronics and Computer Science

On Evidence Gathering in 3D Point Clouds of Static and Moving Objects

by

Anas Abuzaina

Jury:	**Supervisors:**
Prof. Robert B. Fisher	Prof. Mark S. Nixon
Dr. Enrico Costanza	Dr. John N. Carter

Thesis for the degree of Doctor of Philosophy

June 2015

UNIVERSITY OF SOUTHAMPTON

ABSTRACT

FACULTY OF PHYSICAL AND APPLIED SCIENCES
Electronics and Computer Science

Doctor of Philosophy

ON EVIDENCE GATHERING IN 3D POINT CLOUDS OF STATIC AND MOVING OBJECTS

by Anas Abuzaina

The recent and considerable progress in 3D sensing technologies mandates the development of efficient algorithms to process the sensed data. Many of these algorithms are based on computing and matching of 3D feature descriptors in order to estimate point correspondences between 3D datasets.

The dependency on 3D feature description and computation can be a significant limitation to many 3D perception tasks; the fact that there are a variety of criteria used to describe 3D features, such as surface normals and curvature, makes feature-based approaches sensitive to noise and occlusion. In many cases, such as smooth surfaces, computation of feature descriptors can be non-informative. Moreover, the process of computing and matching features requires more computational overhead than using points directly.

On the other hand, there has not been much focus on employing evidence gathering frameworks to obtain solutions for 3D perception problems. Evidence gathering approaches, which use data directly, have proved to provide robust performance against noise and occlusion. More importantly, evidence gathering approaches do not require initialisation or training, and avoid the need to solve the correspondence problem.

The capability to detect, extract and reconstruct 3D objects without relying on feature matching and estimating correspondences between 3D datasets has not been thoroughly investigated, and is certainly desirable and has many practical applications.

In this thesis we present theoretical formulations and practical solutions to 3D perceptual tasks, that are based on evidence gathering. We propose a new 3D reconstruction algorithm for rotating objects that is based on motion-compensated temporal accumulation. We also propose two fast and robust Hough Transform based algorithms for 3D static parametric object detection and 3D moving parametric object extraction.

Furthermore, we introduce two algorithms for 3D motion parameter estimation that are based on Reuleaux's and Chasles' kinematic theorems. The proposed algorithms estimate 3D motion parameters directly from the data by exploiting the geometry of rigid transformation. Moreover, they provide an alternative to the both local and global feature description and matching pipelines commonly used by numerous 3D object recognition and registration algorithms.

Our objective is to provide new means for understanding static and dynamic scenes, captured by new 3D sensing technologies as we believe that these technologies will be dominant in the perception field as they are going under rapid development. We provide alternative solutions to commonly used feature based approaches by using new evidence gathering based methods for the processing of 3D range data.

Contents

List of Figures

List of Tables

Declaration of Authorship

I, Anas Abuzaina , declare that the thesis entitled *On Evidence Gathering in 3D Point Clouds of Static and Moving Objects* and the work presented in the thesis are both my own, and have been generated by me as the result of my own original research. I confirm that:

- this work was done wholly or mainly while in candidature for a research degree at this University;
- where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated;
- where I have consulted the published work of others, this is always clearly attributed;
- where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work;
- I have acknowledged all main sources of help;
- where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself;
- parts of this work have been published as: [1], [2] and [3].

Signed:...

Date:...

Acknowledgements

First of all, I would like to express my sincere gratitude to my supervisors, Prof. Mark Nixon and Dr. John Carter for their constant support, guidance and inspiration throughout the three years of this research, and throughout the three years before in my undergraduate degree.

I am also grateful for the tenure of a studentship from the Engineering and Physical Sciences Research Council (EPSRC).

I would also like to thank all my colleagues in the computer vision group for their support. In particular, I would like to thank John D. and Tayba for their invaluable advice and helpful chats.

Thanks also go to Prof. Robert Fisher and Dr. Enrico Costanza who provided me with precise feedback on examining my thesis for which I am grateful, those remarks led to a more in-depth analysis and evaluation of the techniques I demonstrate.

I would also like to thank all the staff of the School of Electronics and Computer Science at the University of Southampton for the wonderful six years I spent there. In particular I would like to thank Joyce for the many graphic design projects she involved me with, they have certainly been of great fun and generous financial reward, and Reuben for the ventures and business projects we have been through.

Thanks are also due to my friends outside work for the diversions and amusements that make life more interesting.

Finally, I want to thank my sisters, grandparents, all my aunts and uncles, and most of all my parents, for all their encouragement and support, and I would like to dedicate this thesis to them.

In loving memory of my uncle, Mohammad Abuzaina (1952 - 2015).

Nomenclature

R	3×3 Rotation matrix
$\mathbf{t}$	3D translation vector
p	A 3D point with (p_x, p_y, p_z) geomteric coordinates
T	4×4 Homogeneous transformation matrix
P	Point cloud, a set of M 3D points
θ	Rotation angle
w	Angular velocity
$\dot{w}$	Estimated angular velocity
ξ	Accumulator angular resolution
W	Range of angular velocities
V	Velocity point cloud
A	Temporal accumulation
D	Subsampled velocity point cloud
S	Stack of subsampled velocity point clouds
O	Octree representation of V
σ	Noise standard deviation
ρ	Error metric
c_x	x-coordinate of sphere centre
c_y	y-coordinate of sphere centre
c_z	z-coordinate of sphere centre
r	Sphere radius
υ	Root mean square error metric
$\mathbf{d}$	Displacement vector
$\mathbf{m}$	Midpoint of displacement vector
B	Orthogonal plane through displacement midpoint
$\mathbf{r}$	Axis of rotation
$\dot{\mathbf{r}}$	Estimated axis of rotation
$\mathbf{r_g}$	Ground truth axis of rotation
$\mathbf{v}$	Direction of axis of rotation

q	Point on axis of rotation
K	Max number of possible estimated axes
L	Number of computed axes
α	Geometric consistency tolerance threshold
t_{pos}	Euclidean tolerance threshold
t_{dir}	Cosine similarity threshold
U	Voting algorithm iterations
$\mathbf{n}$	Normal vector
$\mathbf{h_c}$	Screw axis
$\dot{\mathbf{h}}_\mathbf{c}$	Estimated screw axis
$\mathbf{h_g}$	Ground truth screw axis
$\mathbf{h}$	Direction of screw axis
$\mathbf{c}$	Critical point
η	Screw axis pitch
$\underline{p}$	Projected point
e_{pos}	Positional offset
e_{dir}	Directional offset
e_{θ}	Angle offset
β	Subsampling voxel size

Chapter 1

Introduction

1.1 Context

Much progress has been made in the development of 3D acquisition methods and technologies, especially recently when new high-resolution 3D sensing technologies became available to the public such as the Microsoft Kinect. The research community quickly employed such consumer depth sensors to solve many computer vision problems, and to develop numerous 3D perception applications.

By providing a third dimension of the spatial coordinates of their output data such as point clouds, the Kinect and similar depth sensors provide practical and accurate means for overcoming traditional problems in computer vision applications. Such problems are mainly related to background and foreground segmentation, object occlusion, unknown object scaling and changing lighting conditions. As a result, computer vision is rapidly becoming more sophisticated; scenes are understood and objects are recognised.

The processing of 3D imagery plays a crucial role in state of the art 3D computer vision applications such as 3D object detection and tracking, 3D geometry computation, object recognition, surface registration, structure perception and production of 3D models. Introducing robust, realistic and fast solutions for such problems, although extensively studied, is still a challenging research task.

The significant perception improvement in 3D sensing technologies lead towards the development of numerous algorithms designed specifically to work with 3D imagery; many of these algorithms are based on the computing and matching of *3D feature descriptors* and estimating point correspondences to solve 3D perception problems such as recognising objects and registering 3D surfaces. Even though these algorithms provide robust and accurate performance in some cases, the fact that there are a variety of criteria used to describe 3D features, such as surface normals and curvature, makes feature-based 3D perception algorithms more sensitive to noise and occlusion. Moreover, the process of computing and matching features requires more computational overhead than using points directly.

On the other hand, there has not been much focus on employing *evidence gathering* frameworks to obtain solutions for these problems. Having been employed in many 2D applications, evidence gathering is well-known for its ability to handle noise and occlusion; which are significant challenges to many computer vision algorithms. Moreover, evidence gathering approaches do not require initialisation or training, avoid the need to solve the correspondence problem, and use the data (3D points) directly; hence bypassing the need for computing and matching 3D feature descriptors.

By the nature of evidence gathering, the best correspondences of the investigated feature produce the highest accumulator peaks. The *evidence* is the votes cast in an *accumulator* array, which is simply an N-dimensional array where each cell corresponds to certain parameter.

In this research we present theoretical formulations and practical solutions to 3D perceptual tasks, that are based on evidence gathering. We propose an algorithm for 3D reconstruction of rotating objects that is based on motion-compensated temporal accumulation, two methods for 3D static parametric object detection and 3D moving parametric object extraction, based on the Hough transform, and a new method for 3D motion estimation based on Reuleaux's and Chasles' kinematic theorems employing invariant properties that can be inferred from the motion.

The developed methods must be robust in terms of noisy data and occlusions, and to handle errors and false positives. Also these methods have to be efficient in terms of computational power and memory requirements, and to work within available computational constraints. Moreover, they have to be generic in terms of the shape or model of the object to be detected and constructed, and the quality and source of the input data.

Our objective is to provide new means for understanding static and dynamic scenes, captured by new 3D sensing technologies as we believe that these technologies will be dominant in the perception field as they are going under rapid development, and being employed widely in research as well as in practical fields. We provide alternative means to commonly used feature based approaches by introducing employment of evidence gathering based methods to the processing of 3D range data.

1.2 Contributions

The contribution of this research is of four parts:

- We present an algorithm for 3D reconstruction of point clouds that is based on motion-compensated temporal accumulation. This algorithm accumulates surface information over the sequence after applying a series of rigid transformations on the point clouds of the input sequence, and then resamples the accumulated point clouds based on a computed space partitioning and votes for the best reconstruction based on the number of points in each resampled point cloud.

- We introduce a fast and robust Hough Transform (HT) based algorithm for detecting spherical structures in 3D point clouds. To the best of our knowledge, our algorithm is the first HT based implementation that detects spherical structures in typical 3D point clouds generated by consumer depth sensors. Our approach has been designed to be computationally efficient; reducing an established limitation of HT based approaches.

- We present the new 3D velocity Hough Transform (3DVHT) which incorporates motion parameters in addition to structural parameters in an evidence gathering process to accurately detect moving spheres at any given point cloud from the sequence. The 3DVHT globally integrates information in all frames (point clouds), so information missed in one frame, yet present in another, still contributes to the final evidence gathered. We demonstrate its capability to predict the occurrence of spheres which are obscured within the sequence of point clouds, even in the case of full occlusion.

- We introduce two algorithms for 3D motion parameter estimation that are based on two kinematics theorems. The proposed algorithms estimate 3D motion parameters directly from the data by exploiting the geometry of rigid transformation using an evidence gathering technique in a Hough-voting-like approach. The two proposed algorithms provide an alternative to the both local and global feature description and matching pipelines commonly used by numerous 3D object recognition and registration algorithms, as they do not rely on keypoint detection and feature descriptor computation and matching.

  Moreover, we propose a method for voting for 3D motion parameters using a one-dimensional accumulator space, which enables voting for motion parameters more efficiently than other methods that use up to 7-dimensional accumulator spaces.

1.3 Publications

- The technique for object reconstruction by temporal accumulation was presented at the 2014 International Conference on Pattern Recognition [1].
- The static sphere detection technique was presented at the 2013 conference of Computer Analysis of Images and Patterns [2].
- The moving sphere detection technique was presented at the 2013 IEEE Image, Video and Multidimensional Signal Processing Workshop on 3D Image/Video Technologies and Applications [3].
- The new method for motion estimation will be submitted to IEEE Transactions on Pattern Analysis and Machine Intelligence (in preparation).

1.4 Thesis Outline

This thesis is organised as follows as follows:

Chapter 2 reviews the background on 3D motion estimation, gives a detailed overview of both local and global surface matching methods and discusses the main issues driving the different research strands in the area of surface matching and motion estimation. This is followed by a list of the most recent advancements in available 3D sensing technology and applications and description of the properties of the output data these sensors provide. In addition, properties of point clouds and subsampling methods are discussed, and a variation to current techniques is introduced. The next four chapters will each have a section on related work discussing methods and algorithms specifically related to the content of each chapter.

Chapter 3 describes an algorithm for 3D reconstruction of point clouds that is based on motion-compensated temporal accumulation. The chapter first discusses the related work to our algorithm in terms of registration and temporal accumulation, followed by a detailed overview of the presented algorithm. After that, experimental results on synthetic and real data are demonstrated, and finally the performance of the algorithm evaluated in terms of noise and processing time.

Chapter 4 describes the 3D Hough transform for sphere detection in static point clouds. This chapter begins by describing parametric 3D shape detection and listing related work. After that, the theory of the 3D Hough transform for sphere detection is explained. Next, details of the implementation of the algorithm are described. Finally, evaluation and performance analysis in terms of occlusion, noise and bin size, are provided.

Chapter 5 describes the 3D velocity Hough transform (3DVHT). The chapter begins by describing moving object detection and listing related work. After that, it describes the algorithm for detecting moving parametric objects via the 3DVHT. Next, details of the implementation of the algorithm are described. After that, experimental analysis demonstration examples of synthetic and real point cloud sequences are provided.

Chapter 6 describes a new method for 3D motion estimation. The chapter begins by discussing related work, in terms of motion estimation using geometrical analysis and Hough voting. The next section describes the proposed method for rotation estimation. The third section describes the second proposed method for general motion estimation. After that, demonstration examples to validate and verify the theory of the two algorithms are given. Then analysis, experimental evaluation and results of implementing the algorithms on real data are provided followed by a discussion of the properties of the proposed algorithms.

Chapter 7 outlines the further work of this research and concludes this thesis.

Chapter 2

State of the Art

Analysing multiview 3D image data largely concerns deploying processing pipelines that can find the relative motion between the data sets by assessing similarity between surfaces in the underlying 3D data. Determining similarities between datasets, or *surface matching*, leads to finding the relative positions and orientations between datasets, and hence finding the transformation consisting of translation and rotation which transforms one data set to the other, which is referred in the literature as *motion estimation* or *transformation estimation.*

In other words, surface matching concerns with finding correct correspondences between 3D datasets; as finding transformation between datasets is easily calculated if correct point correspondences are known.

Object recognition and registration are of the most fundamental topics that have been focused on by 3D computer vision researchers; especially in the last few years when low cost sensors, such as the Microsoft Kinect, became popular within the research community. Object recognition aims to identify and detect the presence of objects in a given scene, and determine their poses (positions and orientations), while registration concerns integrating separately acquired datasets into a consistent global model.

Given a model of the object, recognition systems typically identify and classify objects present in a single viewpoint and estimate their pose in the real world by matching surfaces on the given model to points in the scene. On the other hand, registration is achieved by finding the poses of multiple partial datasets acquired from different viewpoints, and aligning them into a complete model in such a manner that intersecting regions overlap perfectly. This also requires matching surfaces and estimating the motion between multiple partial datasets.

Intensive research has been done on developing methods for finding, matching and refining point correspondences between 3D datasets, and to estimate the resulting transformation. In the following sections, we explain 3D motion estimation and give a detailed

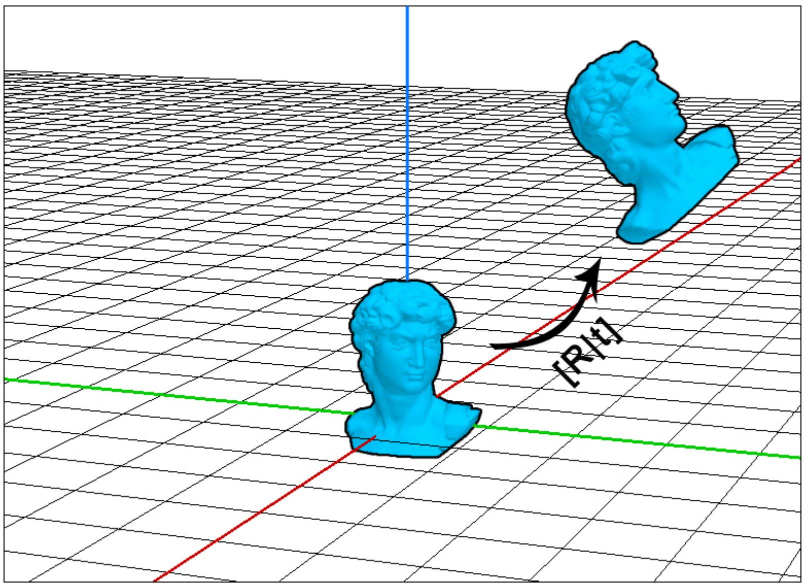


Figure 2.1: 3D rigid transformation.

overview of both local and global surface matching methods; we discuss the procedural pipelines of both categories, explain related concepts, and list state of the art methods and algorithms and how they are employed to estimate the motion between 3D datasets. We also discuss the main issues driving the different research strands in the area of surface matching and motion estimation.

Additionally, we list the most recent advancements in available 3D sensing technology and describe the properties of the output data these sensors provide. We also discuss properties of point clouds and subsampling methods, introducing a variation to current techniques that provides a better representation of the subsampled point cloud.

2.1 Motion Estimation

The process of computing the relative position and orientation between 3D datasets from predetermined corresponding pairs is called motion estimation, also referred to in the literature as rigid transformation estimation and the absolute orientation problem.

A rigid (Euclidean) transformation, shown in Figure 2.1, preserves the distance between any two points on the surface of the object for all motions; it consists of a rotation component represented by a 3×3 matrix (R) and a translation component represented by a 3D vector ($\mathbf{t}$). If p is any point on the 3D object undergoing the rigid transformation and p' is the corresponding point resulting the transformation, then the following relation holds:

$$p' = Rp + \mathbf{t} \tag{2.1}$$

which can be expanded to

$$\begin{bmatrix} p'_x \\ p'_y \\ p'_z \end{bmatrix} = \begin{bmatrix} R_{11} & R_{12} & R_{13} \\ R_{21} & R_{22} & R_{23} \\ R_{31} & R_{32} & R_{33} \end{bmatrix} \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix} \tag{2.2}$$

The rigid transformation can also be represented by a single 4×4 matrix ($T = [R|t]$) in homogeneous coordinates, Equation 2.3.

$$T = \begin{bmatrix} R_{11} & R_{12} & R_{13} & t_x \\ R_{21} & R_{22} & R_{23} & t_y \\ R_{31} & R_{32} & R_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.3}$$

If only the 3D coordinates of the correspondences are used, three correspondence pairs are required to deduce a transformation between the surfaces to be matched. Some other methods which use surface normals at keypoints in addition to their 3D coordinates, require two points for motion estimation, as in [4].

Many algorithms have been developed to compute the parameters of the 3D rigid transformation from established 3D point correspondences between a pair of 3D datasets. Comprehensive surveys of 3D motion estimation methods are in [5, 6].

[5] Presented a qualitative summary and comparison of iterative as well as closed form motion estimation methods categorised by the type of feature used in the motion estimation task.

[6] Presented a comparative analysis of four popular and efficient algorithms, based on accuracy, stability and efficiency. Each algorithm computes the translational and rotational components of the transformation in closed form, given the point correspondences between the two 3D point sets. These algorithms differ in terms of the transformation representation used and the mathematical derivation of the solution; namely singular value decomposition (SVD), orthonormal matrices, unit quaternions and dual quaternions. The results of this analysis indicated that under typical, real-world noise levels, there is no difference in the robustness of the final solutions of those four algorithms.

Other methods exploit the geometrical properties of a rigid transformation to compute the 3D motion parameters such as in [7, 8] in which the vector distance between feature points and angular information are used as constraints to analyse transformation parameters. [9] Determine the transformation in the form of screw displacement parameters using line geometry and the Reuleaux method. Methods that use geometric properties for motion estimation are discussed in detail in Chapter 6.

2.2 3D Surface Matching

There are two main categories for 3D surface matching methods; matching by global descriptors and matching by local feature correspondences. Global surface matching methods rely on global feature descriptors that describe the entire 3D object, while local methods use local feature descriptors that are computed locally around keypoints on the surface of the objects. Figure 2.2 shows the pipelines of global and local surface matching methods.

A 3D feature descriptor, also called geometric feature, shape descriptor, or just simply 3D feature, is a compact high-dimensional representation of 3D data. 3D feature descriptors are vector functions that describe the geometry of an object (global) or a local neighbourhood (local) around a keypoint on the surface of an object, by projecting the neighbourhood into a proper feature space. Thus, comparing two different objects, or two different points on two different surfaces, results in comparing the difference in some space between their feature vectors. A 3D feature descriptor is invariant to a specific class of transformations and/or set of disturbances.

Unlike 2D feature description, 3D descriptor computation is based on surface geometry rather than the attributes like colour and texture used in 2D image recognition and retrieval. This is due to the fact that the similarity between 3D objects is congregated by their shapes. Furthermore, colour and texture information are not guaranteed to be provided in 3D data.

2.2.1 Global Surface Matching

Global surface matching methods characterise the object as one entity and define a set of global features that describe the entire 3D object concisely. Usually, global feature descriptors are computed for 3D data that is likely to be objects, and they usually deploy a specific preprocessing step based on segmentation.

The key idea of global matching methods is to transform an arbitrary 3D model into a parameterized function that can easily be compared with other functions, which is a relatively simple problem when compared to surface matching methods that use feature correspondences, model fitting and parameterisation.

The global surface description pipeline starts with a segmentation step, and then the segmented objects are described with a global feature descriptor in order to be matched in the next step. After matching, a transformation that aligns the datasets is computed and finally a hypothesis verification step implemented to improve the estimated transformation accuracy.

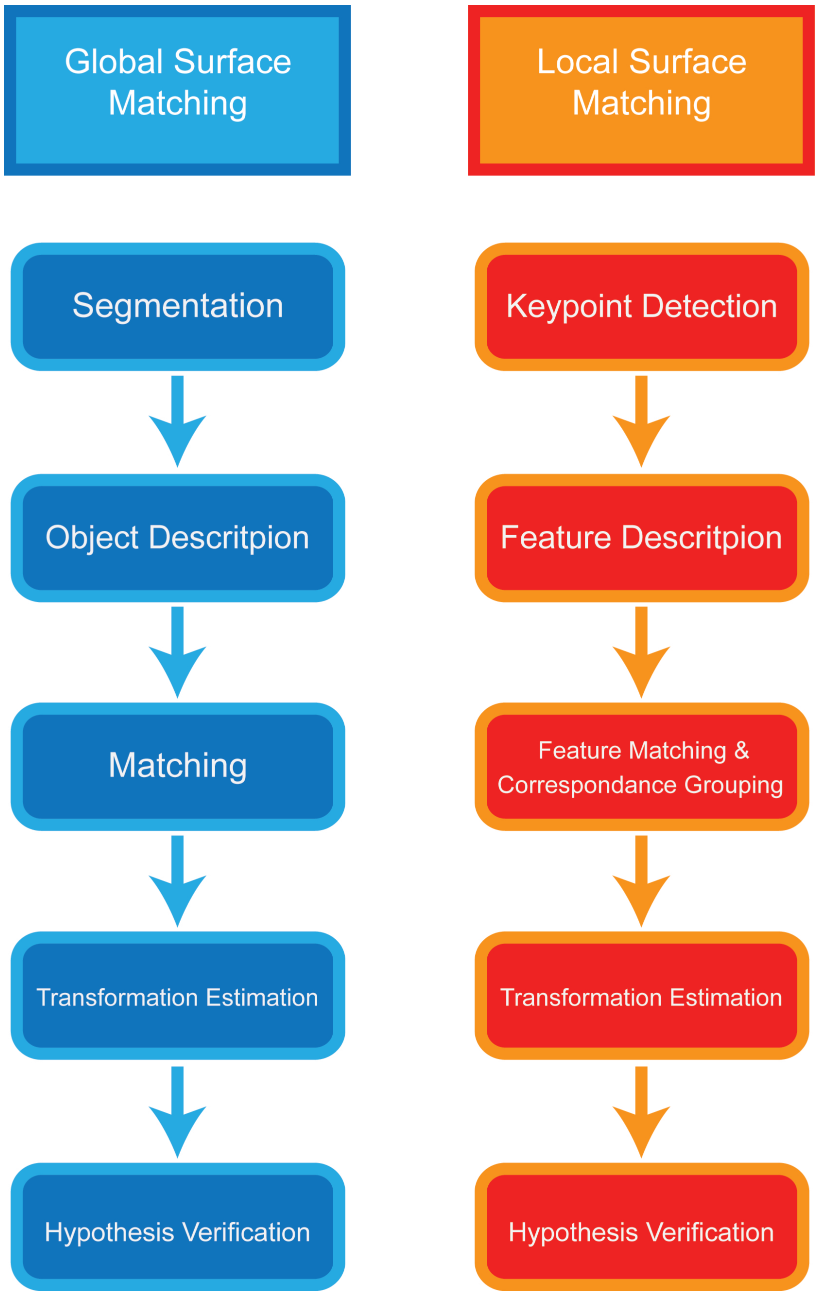


Figure 2.2: Surface matching pipelines.

2.2.1.1 Segmentation

Since global feature descriptors require the notion of objects, segmentation is required to extract objects from the scene. Several methods have been developed for the specific task of segmentation, such as in [10, 11]. These methods are usually based on the extraction of a dominant plane in the scene and clustering remaining points.

2.2.1.2 Global Object Description

Global object descriptors are high-dimensional representations of object geometry and are mainly employed for object recognition, pose estimation and object classification. They can be considered as a mapping from the space of 3D objects to some finite-dimensional vector space. Such mappings are designed to preserve as much information as possible and to build the resulting representing vector in a possibly low-dimension to allow fast and reliable similarity searches. The viewpoint feature histogram (VFH) [12], 3D moments [13], Eigen shapes [14], shape distributions [15], and extended Gaussian images [16] are amongst the most widely used global descriptors in recognition and registration pipelines.

2.2.1.3 Global Matching

In the global feature framework, the shape and geometry of segmented objects is described by means of a single global descriptor. As mentioned earlier, this descriptor can be a represented as a histogram or another high dimensional mapping in a different vector space. This descriptor is typically compared against those obtained of a given model in the training stage, or other data sets and the best K Nearest neighbours (NNs) are retrieved. The K NNs represent the K most similar datasets according to the description obtained by means of the desired global feature and the metric used to compare the feature vectors [17]. To improve matching results, a number of nearest neighbours instead of the single nearest neigbour are usually retrieved. For histogram based global feature matching methods, typically a brute-force matching algorithm is used for fast and reliable similarity searches, such as in [18]. Other methods find similarities by means of correlation between mapped feature vectors [16, 19, 20].

2.2.1.4 Transformation estimation

Once global descriptors are matched, a six degrees of freedom transformation consisting of rotation and translation is computed to perform alignment between the 3D datasets, also referred to as 6 DoF pose estimation. This is typically achieved by an optimisation function that correlates the generated global descriptor vectors to obtain the transformation that best aligns the surfaces of the input 3D datasets.

2.2.1.5 Hypothesis Verification

Typically, the last step of both local and the global surface matching pipelines is hypotheses verification, which aims to improve the outcome of the surface matching by reducing the number of false positives while retaining correct correspondences [17]. This is achieved by aligning matched surfaces by the transformation computed in previous steps and then refining the alignment with methods such as the iterative closest point algorithm (ICP) and its variants [21]. [22] Introduced a hypothesis verification method for global surface matching pipelines that is based on the number of inliers between datasets.

2.2.1.6 Limitations

Generally, global feature descriptors suffer from a number of limitations; since these descriptors process the object as one complete entity they ignore the specific details of the object. Also it is difficult to handle partial data using global descriptors. Moreover, they require a priori segmentation of the object from the scene, therefore are not suitable for the recognition of a partially visible object from cluttered scenes, as the global descriptor may include points from the clutter around the object. Furthermore, mapping to other spaces can be non-informative and ambiguous in some cases.

Additionally, some of these global methods are limited to special cases; [20] depends heavily on point cloud resolution and its consistency and can be computationally complex. [16] Relies on the calculation of normals which do not provide enough information about the geometry in some point clouds and does not give accurate results when handling arbitrarily curved objects. [19] Does not produce a single solution and therefore requires methods to evaluate a set of potential solutions. According to [23], most focus has been on surface feature based methods as they are more robust to occlusion and clutter which are frequently present in a real-world scene.

2.2.2 Local Surface Matching

Local surface matching methods depend on the detection of local distinguishable points (keypoints) and estimating the correspondences between point sets by computing and comparing feature vector functions (descriptors) describing the geometry in the neighbourhood of the keypoint, thus finding correspondences is achieved by comparing differences between these descriptors. In the last few years, numerous keypoint detection as well as feature description and matching methods, aimed to solve recognition and registration paradigms, have been introduced and implemented widely within research as well as industry.

Typically, a local feature-based matching pipeline which aims to find the transformation between 3D datasets, such as registration and object recognition systems, is implemented in the following five steps:

1. Keypoint detection: The first step is to identify keypoints on the surface of the datasets to be matched, these keypoints are identified by a 3D detector.

2. Feature description: At each keypoint, a vector describing the local neighbourhood of the keypoint is computed; this vector is called feature descriptor.
3. Feature matching: Computed feature descriptors are compared and matched, and correspondences are estimated from the most reliable matches. An additional procedure is included in this step to eliminate wrong correspondences, usually referred as correspondence grouping.
4. Transformation estimation: Remaining correspondences are used to compute the transformation between the 3D datasets.
5. Hypothesis Verification: Similar to the global surface matching pipeline, this step is implemented to improve the accuracy of the estimated transformation.

In the following sections we present a detailed overview these five steps of the local surface matching pipeline.

2.2.2.1 Keypoint Detection

3D datasets typically have large number of points; point clouds generated by the Kinect sensor for example have 300K points on average, also laser range finders can produce point clouds with multi million points. Extracting local features from such a great number of points can be computationally inefficient. Instead, working with a subset of interest points with rich information content from the 3D dataset significantly reduces time and computational cost. The interest points are called keypoints, Figure 2.3.

Keypoint detection is the first step of any feature-based surface matching; detecting keypoints significantly reduce the computational burden of the subsequent feature extraction and matching phases, as feature descriptors are computed only for detected keypoints instead of every point in the 3D dataset.

Keypoints can be defined as points of the shape that are prominent according to a particular definition of interestingness or saliency [24]. The position and the neighbourhood size of the keypoint define a local surface which is used in the subsequent feature description phase. According to [25], a 3D keypoint on a 3D model or a 2.5D surface must satisfy three constrains: the detected keypoints must have high repeatability between different 2.5D views and 3D model of the same object. Second, a unique 3D coordinate basis can be defined from the neighbourhood surface to extract invariant local features. And third, the neighbourhood surface of the keypoint must contain sufficient descriptive information which uniquely characterizes that point.

3D keypoints are extracted from surfaces by a 3D detector. 3D keypoint detectors analyse local neighbourhoods around the elements of the given surface in order to identify

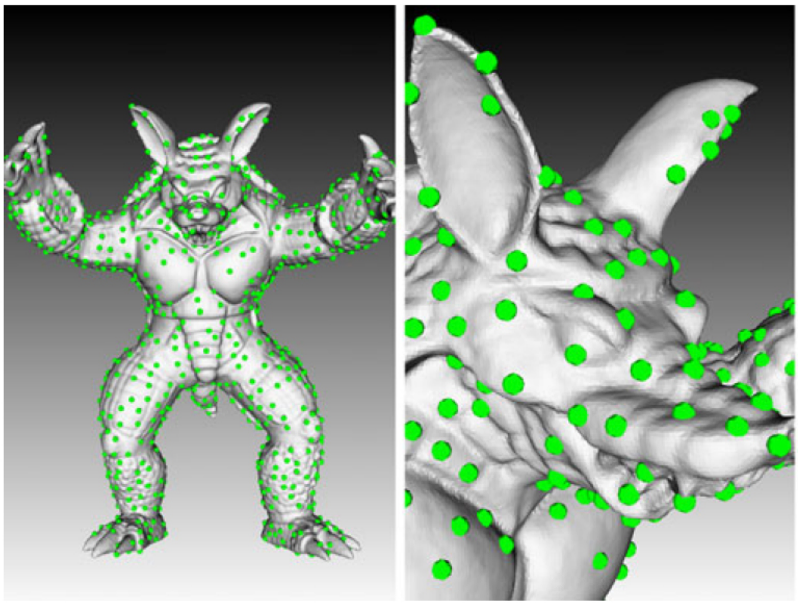


Figure 2.3: Detected keypoints on 3D object, source [24].

such interest points. A good keypoint detector should have two main traits; distinctiveness and repeatability [24]. Distinctiveness is the ability to detect keypoints that can be effectively described and matched, which subsequently will provide correct correspondences. Repeatability in this context means that the detector should have the capability to detect the same keypoints accurately under various nuisances affecting the data such as viewpoint changes, missing parts, point density or topology variations, clutter, sensor noise, etc.

In the past few years several 3D keypoint detectors have been introduced, detailed comprehensive surveys and evaluations of 3D keypoint detectors can be found in [23, 24, 25]. In summary, 3D keypoint detection methods (keypoint detectors) can be categorised into three groups: sparse sampling methods, fixed-scale detectors and adaptive-scale detectors.

Sparse sampling and mesh decimation methods are the simplest keypoint detectors. As their name suggests, these methods subsample the 3D data into a reduced number of sparse points, this is achieved by specially partitioning the 3D space into small volumes using a data structure such as octree [26] or Kd-tree [27], overviews of these methods can be found in [28, 29].

The problem with sparse sampling methods is that they do not result in qualified keypoints in terms of repeatability and amount of information, and thus their resulting features are not sufficiently descriptive; as they do not consider the richness of discriminative information of the keypoints [23]. We discuss point cloud subsampling in detail in Sub-section 2.5.2.

Other keypoint detection methods are categorised based on the scale (neighbourhood size) of the key point; whether it is predetermined (fixed-scale) or adaptively detected (adaptive scale).

Fixed-scale keypoint detection methods find distinctive key points at a specific, constant neighbourhood size or scale. This neighbourhood size is determined by the scale, which is an input parameter to the keypoint detection algorithm [24]. Although these methods are easy to implement, using fixed scale might cause the detection of few keypoints on the surface, and neglecting useful scale information encoded in local structures. Examples of fixed-scale keypoint detectors include local surface patches (LSP) [30], rotational projections [31] and intrinsic shape signatures [32].

A lot of recent research has been focused on adaptive-scale keypoint detection methods, which build a scale-space for the 3D dataset and pick the points with extreme distinctiveness measures in both the spatial and scale neighbourhoods as keypoints [23]. These methods provide the ability of associating a characteristic scale with each detected keypoint, which is selected as the maximum along the scale dimension of a suitable function, generally coinciding with the saliency [24]. This defines the support for the subsequent feature description stage. Examples of adaptive-scale keypoint detectors include Laplace-Beltrami Scale-space [33], 3D SURF [34] and 3D SIFT [35].

According to [23], these methods can be divided into four categories: coordinate smoothing, geometric attribute smoothing, surface variation and transform based methods.

2.2.2.2 Feature Description

According to [36], the importance of 3D local features comes from the fact that they provide a more robust alternative to representing points with only their 3D Cartesian coordinates; as the latter is an ill-posed problem, because even though 3D points are equal with respect to some distance measure (e.g. Euclidean metric), they could be sampled on completely different surfaces, and thus represent totally different information when taken together with the other surrounding points in their vicinity 3D feature descriptors provide applications which need to compare points with better characteristics and metrics to be able to distinguish between geometric surfaces.

Once keypoints are extracted, they are associated to local feature descriptors; this is done by determining a subset of neighbouring points around extracted keypoints and using it to compute a description of the local geometry for each keypoint. Generally, 3D local descriptors are developed for specific applications such as registration, object recognition, and local surface categorization. Typically keypoint detection is combined with a local feature descriptor to achieve an optimal performance, which is also dependent on the size of the neighbourhood around the keypoint; as large neighbourhood enables a descriptor to encode more information while being more sensitive to occlusion and clutter.

According to [23], local feature descriptors can be categorised according to the approaches employed to construct the feature descriptors; these sub-categories include signature based, histogram based, and transform based methods:

- Signature based Methods: these methods describe the local neighbourhood of a keypoint by encoding one or more geometric measures computed individually at each point of a subset of the neighbourhood. Examples of this sub-category are the normal aligned radial feature (NARF) [37] and the Binary Robust Appearance and Normal Descriptor (BRAND) [38].
- Histogram based Methods: these methods describe the local neighbourhood of a keypoint by accumulating geometric or topological measurements (e.g., point numbers, mesh areas) into histograms according to a specific domain (e.g., point coordinates, geometric attributes). These methods can further be divided into spatial distribution histogram such as pairwise geometric histograms [39], Rotational Projection Statistics (RoPS) [31], geometric attribute histogram such as Histogram of Oriented Normal Vectors HONV [40], Point feature histogram (PFH) [41], Signature of Histograms of OrienTations SHOT [42], and oriented gradient histogram based methods such as MeshHOG [43] and Local Depth SIFT (LD-SIFT) [35].
- Transform based Methods: these methods first transform a range image from the spatial domain to another domain (e.g., spectral domain), and then describe the 3D surface neighbourhood of a given point by encoding the information in that transformed domain, such as 3D SURF [34] and Heat Kernel Signature HKS [44].

Finally, it is worth to mention that many 3D keypoint detection methods, as well as 3D feature descriptors are inspired by their successful 2D ancestors. For example, LD-SIFT and 3D SURF feature detectors are respectively extensions of SIFT [45], SURF [46] detectors.

2.2.2.3 Local Matching

Once local feature descriptors are computed for the 3D datasets, they need to be matched to yield point-to-point correspondences. Local feature descriptors are compared using the Euclidean distance, and correspondences are set between each descriptor from one 3D dataset and its nearest neighbour (NN) in the other dataset according to a matching threshold. According to [47] there are three popular strategies for feature matching, i.e., threshold based, Nearest Neighbour (NN) based and Nearest Neighbour Distance Ratio (NNDR) based strategies.

Once point-to-point correspondences are computed, they will contain true and false matches, hence additional stages need to be implemented to select the best correspondences and discard false ones in order to obtain the correct transformation. These additional stages include techniques such as correspondence grouping, RANSAC, game theory and Hough transform based methods.

In correspondence grouping, correspondences are discarded by enforcing geometrical consistency between them. More specifically, by assuming that the transformation between the model and its instance in the current scene is rigid, the set of correspondences related to each model is grouped into subsets, each one holding the consensus for a specific rotation and translation of that model in the scene. Subsets whose consensus is too small are discarded, simply by thresholding based on their cardinality [17].

Even with enforcing geometric consistency, it is not guaranteed that all correspondences are consistent with a unique 3-D rigid rotation and translation. Hence, additional steps can be included to the local feature matching process to reduce the number of the remaining correspondences such as RANSAC. Random sample consensus (RANSAC) eliminates correspondences which are not consistent with the same six degrees of freedom transformation (translation and rotation), such as in [4].

In addition to geometric consistency and RANSAC, researchers have introduced other methods for eliminating false correspondences, such as game theory based methods [48] and Hough transform based methods [34, 49]. Game theory based methods perform a selection process to select feature correspondences which satisfy a global rigidity constraint while Hough transform based methods perform a voting process in a parameter space, the peaks of the parameter space are considered to be the transformation hypothesis. A detailed discussion on Hough based methods is in Chapter 6.

2.2.2.4 Transformation Estimation

Once correspondences are estimated by the surface matching step, and then wrong correspondences are eliminated by correspondence grouping, a rigid transformation that aligns the matched 3D datasets is computed from remaining correspondences, as explained in section 2.1.

2.2.2.5 Hypothesis Verification

Similar to global surface methods, a final hypotheses verification step is required to guarantee that any correspondence is consistent with the estimated transformation. The ICP algorithm is employed with some other metrics usually used with local surface matching methods to perform verification. Other methods such as in [50] minimise cost functions of geometrical cues to achieve optimal hypotheses.

2.2.2.6 Computational Complexity

Searching for matches between feature descriptors can be implemented by a brute-force search, which compares a scene feature with all model features. Assuming there is one model, the computational complexity of a direct brute-force search is $O(N_f)$, where N_f is the number of model features [23]. Faster alternatives have been introduced in which special data structures employed for indexing. For example, [51] used hash tables, and [48] and [31] used a kd-tree based method to perform feature matching.

Other than the number of features, the dimensionality D of the feature vector affects the computational complexity. Generally, the number of bins required for a particular division of the features space into div divisions for each dimension is div^D, [17]. The dimensionality of the feature space can be significantly large, for example the 3D SIFT descriptor proposed by [52] has 2048 dimensions. 3D feature descriptors vary in dimensionality and in general require spaces with many dimensions.

2.3 Issues

There are many issues that are driving the different research strands in the area of surface matching and motion estimation and subsequently 3D applications such as surface registration and object recognition; in [53], the authors highlight the main issues of registration and fusion of 2.5D range images into full 3D data sets, here we briefly describe the most significant of these issues.

1. Initial registration and range of convergence. Most registration algorithms use an iterative algorithm that ideally converges to the best multiple data set registration. Many algorithms only converge to a good solution if the initial relative pose estimate is sufficiently close to the optimal registration. So, one issue is how to find a good initial relative pose estimate. It is interesting to note that even with good initialisation, algorithms may not find an optimal registration or even may not be able to refine the initial coarse registration. This is due to a large number of local minima due to noise, occlusion, appearance and disappearance of points, and general lack of knowledge about the distribution of points.

2. Inexact correspondences. Two or more 3D data sets are unlikely to be acquired such that the 3D data points exactly correspond. For example, the sampling density might be different due to scanner settings, scanner distance or surface slopes. Thus, a point or feature in one data set will not have an exact correspondence in another data set, yet some sort of alignment between the elements in the two data sets will be needed. However, it is observed that registration errors are a function of inter-point distances and thus, the resolution of the scanning. The smaller such

distances, the larger the potential for accurate registration. This suggests that registration accuracy may be improved from fitted surfaces.

3. Outliers/partial overlap. When registering two (or more) data sets, there are usually 3D data points or other features that do not have any correspondence between the data sets. The two main causes of this are: 1) regions of the data where there is no overlap (a natural consequence of extending the data description by incrementally fusing partially overlapping data) and 2) noise outliers. Hence, great effort has been made to identify outliers and partial overlap based on techniques. It has been shown that the identification of outliers and partial overlap are essential steps to the estimation of motion parameters. Moreover, these two steps are often interweaved and affect each other especially when no exact information is available about the distribution of points, occlusion, appearance and disappearance of points.

4. Pose versus correspondence search. Most registration algorithms align data sets by finding approximately corresponding data features and then estimating the pose that aligns these. A problem that arises from this approach is the convergence to significantly misaligned local minima, which can happen when the data sets are initially far from correct alignment or slight misalignment when near to the global optimum. More recent research has started search in the pose space instead of the correspondence space and seem to be finding a broader range of initial poses that still lead to convergence near the correct alignment.

5. Registerable feature type. When searching for corresponding features in the two data sets, there are a variety of criteria that one can use for classifying points as being similar or interesting or key points, such as colour, local surface normals, local curvature shape, edgeness, texture, etc. All such features are to varying degrees sensitive to noise and other conditions such as occlusion and thus the extraction of such features is also a challenging task. This contrasts with using the points directly with the shortcoming that such algorithms often require good initialisation.

6. Performance acceleration. The correspondence search algorithms generally have a large potential space to search through in order to find corresponding features. This problem is usually solved by including processes to reduce the computational complexity of this search, including coarse-to-fine strategies.

Figure 2.4: Consumer depth sensors. Microsoft Kinect (top left), Microsoft Kinect v2 (top right), Asus Xion Pro (bottom left) and Leap Motion (bottom right).

2.4 Depth Sensors

In recent years, a number of consumer depth sensors have been introduced to the public, such as the Microsoft Kinect[1], Microsoft Kinect v2[2], Asus Xion Pro [3] and Leap Motion[4], Figure 2.4. The low cost, reliability and real-time 3D measurement not only made these sensors very popular within computer gamers, but also within the computer vision research community. Moreover, numerous applications in computer vision, robotics, human-computer interaction, health care, industrial design, 3D printing, gaming, even artistic projects and fashion have employed such devices.

More recently, there has been a focus to port 3D sensing technology in mobile devices; such as Google Project Tango[5] devices which contain customized hardware and software designed to track the full 3D motion of the device, and the Structure Sensor[6], that can be attached to mobile devices and use their hardware for computation, Figure 2.5. The introduction of 3D sensing to mobile devices will open up new avenues for indoor navigation and immersive gaming, among many other things.

Out of these sensors, the Microsoft Kinect is the most widely employed in research as well as in other applications. The Kinect can capture 3D measurement within an approximate volume of $(7.5 \times 4.5 \times 9)$ meters.

Microsoft Kinect and other similar low-cost range sensors provide an attractive alternative to other expensive 3D sensors such as time-of-flight cameras (ToF), laser scanners and structured light sensors. However they have several drawbacks compared to their expensive counterparts. These drawbacks are mainly related to the quality and accuracy

[1]http://msdn.microsoft.com/en-us/library/hh855355.aspx
[2]http://www.microsoft.com/en-us/kinectforwindows
[3]http://www.asus.com/uk/Multimedia/Xtion_PRO
[4]http://www.leapmotion.com
[5]http://www.google.com/atap/projecttango
[6]http://www.structure.io

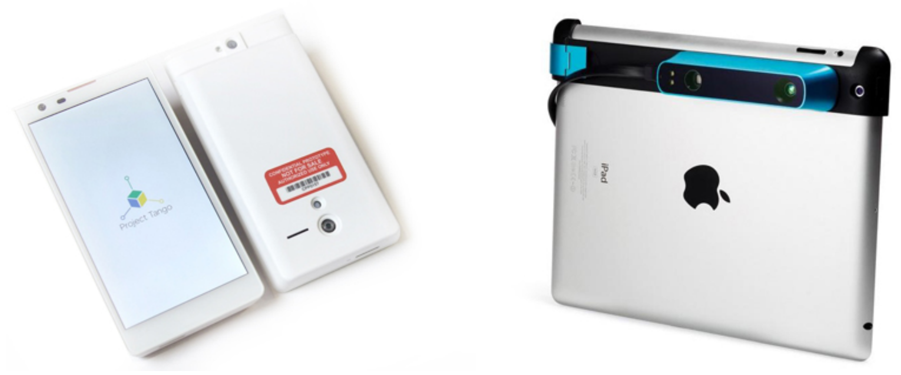


Figure 2.5: Mobile depth sensors. Google Project Tango (left), Structure (right).

of the 3D data, as the Kinect and similar low cost sensors have higher level of noise and quantisation errors.

The Kinect sensor was developed especially for use with the Microsoft Xbox console for gesture-based interaction. Since its release in late 2010, it has been employed in numerous vision applications. Such applications include human gesture recognition [54], human body analysis [55], virtual reality [56], robotics [57] and other applications. The Kinect has an RGB camera, an infrared sensor and a laser emitter; the latter two parts form the depth sensor. The Kinect captures a 640×480 RGB image and a 3D point cloud simultaneously at a frame rate of up to 30 fps. The RGB camera and the depth sensors are pre-calibrated and hence each point in the point cloud is assigned to the RGB colour captured by the camera. Kinect Point clouds typically contain about 300K points. The Kinect works by projecting a pattern of infrared dots over the scene in front of it; this pattern is captured by the infrared camera. The captured pattern is correlated to a reference pattern and the XYZ distances are calculated of all the dots forming the pattern from the disparity triangulation between the reference pattern and the pattern projected on the scene.

The Kinect sensor has inherent drawbacks such as temporal flickering artefacts, missing depth information, misaligned pixels, which all lead to unstable points which degrade the quality of 3D reconstruction. According to [58] experimental results show that the random error of depth measurement increases with increasing distance to the sensor, and ranges from a few millimetres up to about 4 cm at the maximum range of the sensor.[59] and [58] present a detailed analysis on the accuracy, depth measurement resolution and error properties of the Kinect.

Microsoft has recently (2014) released a new version of the Kinect that is based on a time of flight camera. The new version has better output than its predecessor in terms of 3D resolution, precision and noise elimination. This research has been carried out on data generated from a Kinect of the first version before the release of the second.

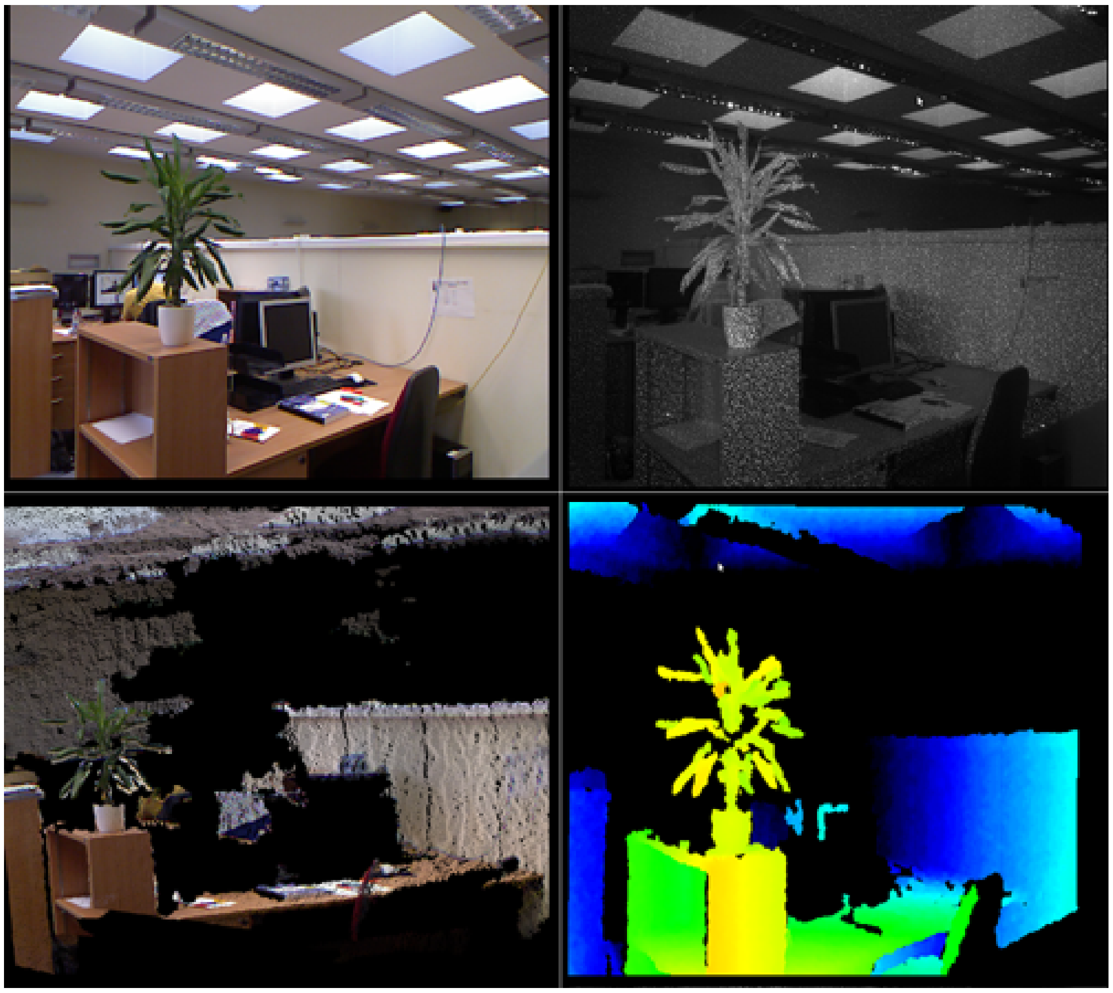


Figure 2.6: Kinect output. RGB camera image (top left), infrared camera image of projected pattern (top right), 3D point cloud (bottom left) and depth map (bottom right).

2.5 3D Imagery

The output of 3D sensors is a dataset that contain 3D measurements of the observed scene, which is usually referred to in the literature as *range images*. Range images can have different representations, depth maps and point clouds are the most used representations for 3D sensor output.

Note that the output of the Kinect and similar depth sensors is 2.5D, i.e. only points on surfaces facing the sensor are acquired, Figure 2.7.

Range images generated by depth sensors provide direct measure of the 3D Cartesian coordinates of the points in object space. Whereas monocular computer vision applications working on 2D camera images are limited by the fact that images are only projections of the 3D world to the 2D image plane; the 3D geometry of any objects in the image is lost. In other words, 2D images capture how objects look rather than how they are.

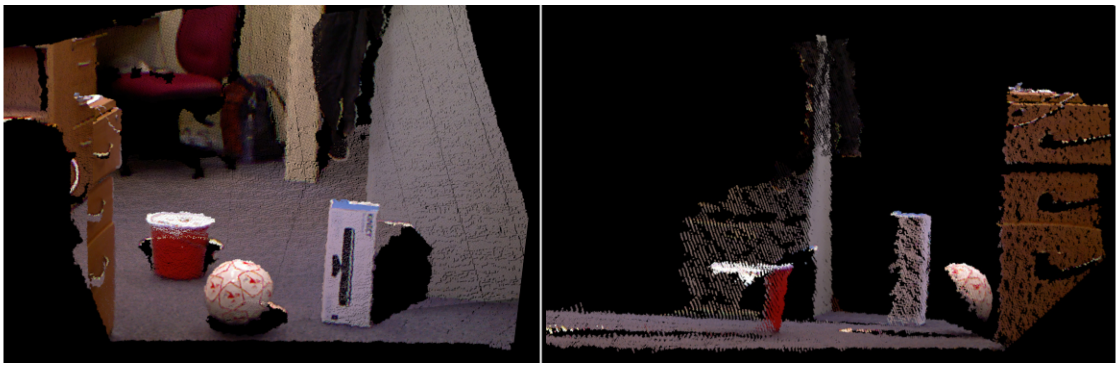


Figure 2.7: 2.5D Kinect output. Front view (left) and side view (right) of a point cloud.

Even with multi-camera configurations and stereo rigs, the geometry of an object is acquired from projections of the real world and a number of processing steps, such as camera calibration, feature matching and optimisation, are required to produce 3D geometry of objects in the scene. Moreover, points in range images define the surfaces of the objects in the scene, meaning there is no need for any edge detection procedures required in 2D images. The advantage of point clouds is that 3D information is available directly, eliminating any occlusion or scaling issues common in 2D images.

Depth maps, are 2D images that have pixel values corresponding to the distance measured from the sensor (note that in some literature depth maps are referred to as range images). If the sensor is properly calibrated the pixel values can be given directly in physical units, such as meters. On the other hand, point clouds are sets of data points defined in a typically 3D coordinate system; each point in the set has 3D coordinates, Figure 2.6. Generally, depth maps and point clouds are not directly usable in many high-end applications, such as mobile manipulation and collision detection, therefore point clouds are usually converted to polygon or triangle meshes, which approximate surfaces in more detail, via a process commonly referred to as surface reconstruction [60].

In this research we will be working with 3D point clouds; switching between point clouds and depth maps representations is relatively simple, and can be achieved by projection and back-projection [61], hence all algorithms presented by this research can be applied to depth maps.

2.5.1 3D Point Clouds

A point cloud is a collection of 3D points (or vertices) in a fixed 3D coordinate system. With analogy to images generated from conventional cameras, point clouds represent the output format for 3D perception systems, such as laser scanners and consumer depth sensors, and provide sampled (discrete) representations of the scene. Every point (vertex) in a point cloud is given by a vector (x, y, x) representing the points position in space with respect to the origin of the coordinate system of the point cloud. Unless otherwise specified, the origin of the point cloud is at the 3D sensor and hence each point in the cloud represents the 3D distance between the sensor and the surface on which the point is sampled on. Moreover, a point may have other properties in addition to the 3D coordinates, such as colour information and surface normals.

2.5.2 Point Cloud Subsampling

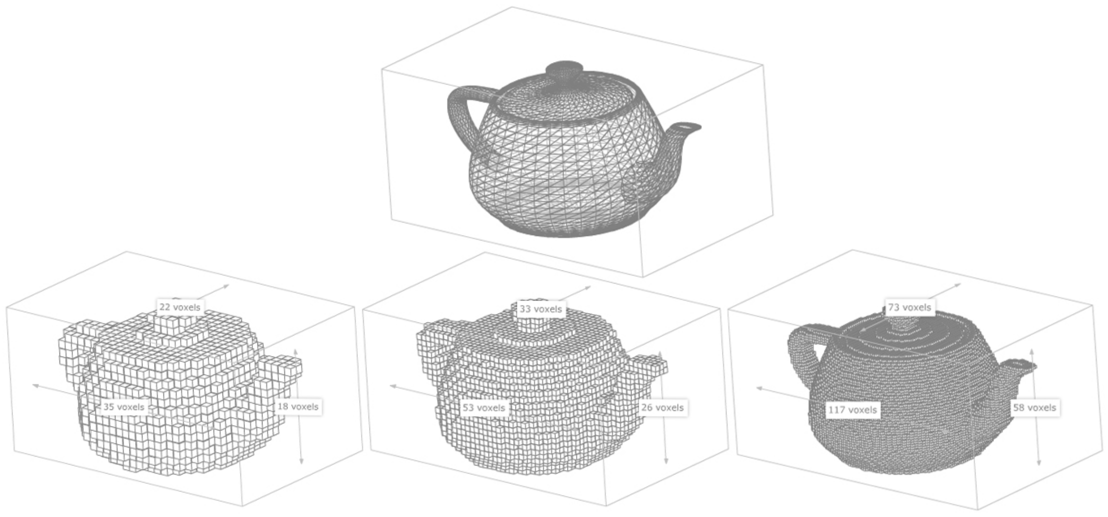


Figure 2.8: Voxelisation at three resolutions.

As previously mentioned, point clouds generated from 3D sensors usually contain large number of points. Moreover, the Kinect for example generates point clouds at a rate of 30 Hz, which can lead to the requirement for extremely fast processors to apply any 3D vision process, and large memory allocation. To overcome memory and processing speed limitations, researches have introduced spatial decomposition techniques to partition point clouds to chunks, such that neighbouring points are grouped in one volume of specified dimensions. This will allow queries and processing based on point locations and neighbouring points to be performed in significantly sorter time.

As mentioned in Subsection 2.2.2, there are different sparse sampling and mesh decimation methods which subsample the 3D data into a reduced number of points. Many of these methods are based on voxelisation; which is the process of converting a geometric representation of a 3D object into a set of voxels that "best" represents that synthetic model within the discrete voxel space [62], Figure 2.8.

A voxel is an enclosed three-dimensional shape, mostly a cube, although it can be any regular, more complicated shape. Voxels are positioned in a symmetrical/regular three-dimensional continuous grid, usually stored in a data structure such as octree [26] or Kd-tree [27]. Figure 2.9 shows a voxel grid applied on a point cloud.

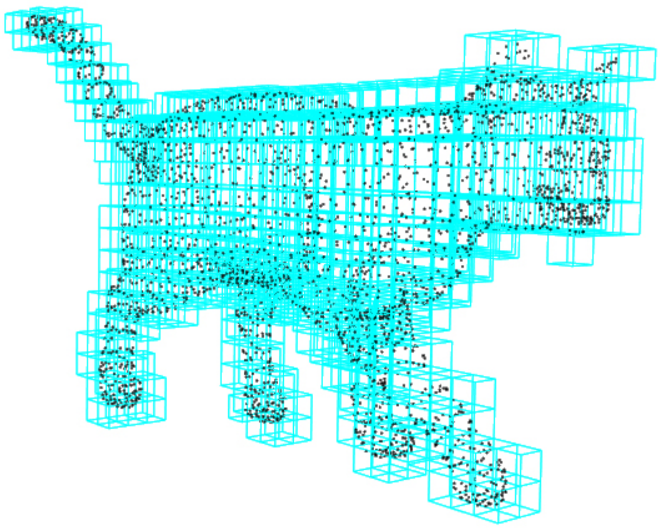


Figure 2.9: Voxel grid applied on a point cloud. Source of input point cloud [64].

Since we are working with 3D point clouds, in this research we employ an octree-based sparse voxelisation technique to subsample the input point clouds. The root node of an octree represents the entire scene; each of its children represents an eighth of its volume and so forth for every node; this structure allows querying for voxel data at any resolution and with increasing detail by descending the tree hierarchy, Figure 2.10. A detailed overview of octree-based sparse voxelisation and can be found in [63].

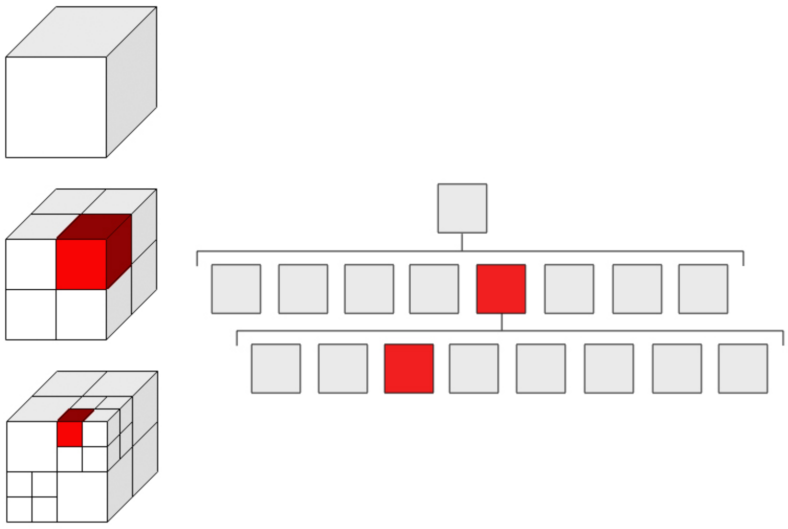


Figure 2.10: Octree visualisation. Every interior node in the tree has 8 children. The 8 children are determined by the physical center point of their parent node.

We create a voxel grid implemented by an octree, initialised by a certain resolution. The resolution parameter is the smallest voxel size at the lowest octree level. After applying the voxel grid on the input point cloud, subsampling is performed such that all points within each of the voxels in the octree are approximated with only one point; this

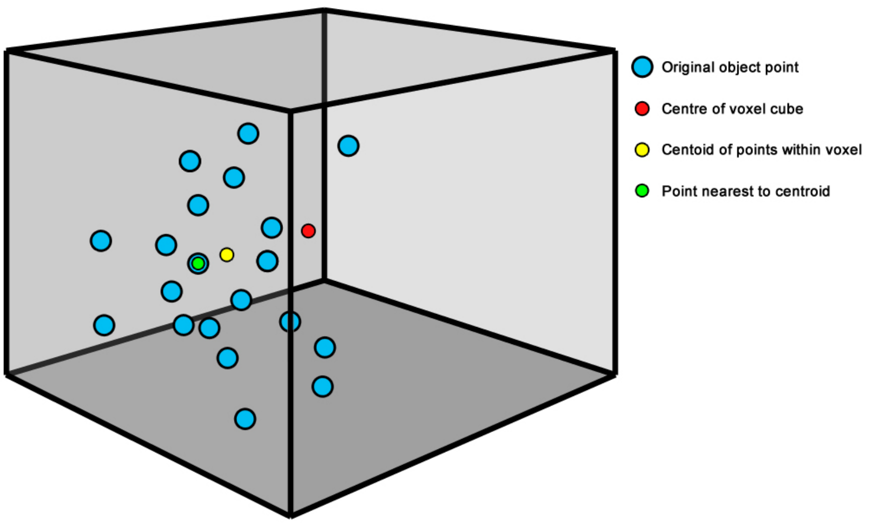


Figure 2.11: Options for approximation of points within a voxel for subsampling. Note that, unlike the other two methods, the proposed method (point nearest to centroid) does not introduce a new point, hence more accurate subsampling.

point is typically the centre of the voxel or the centroid of the points within the voxel. Here we introduce a third approximation that preserves the surface, and is the most true subsampled representation of the input point cloud; that is by taking the original surface point nearest to the centroid, Figure 2.11.

Although approximating points within each voxel with its centre is very fast, approximating the points with their centroid represents the underlying surface more accurately. However it is not the most accurate representation of the surface as the centroid might not be a point on the actual surface. Taking one original point that is nearest to the calculated centroid of points within the voxel achieves subsampling, and guaranties the most true representation of the surface of the object as it does not introduce any new points. This point is found by a simple neighbours within voxel search operation. This search method assigns the search point to the corresponding leaf node voxel and returns the index of point nearest to the computed centroid.

After voxelisation, the resultant subsampled point cloud will have an approximately uniform point density; the density is dependent on the voxel size (resolution) used in the octree. Figure 2.12 shows the the results of the three subsampling approximation methods.

2.6 Conclusions

In this chapter we gave a detailed overview of both local and global 3D motion estimation and surface matching methods; we discussed the procedural pipelines of both categories, explained related concepts, and listed state of the art methods and algorithms and how

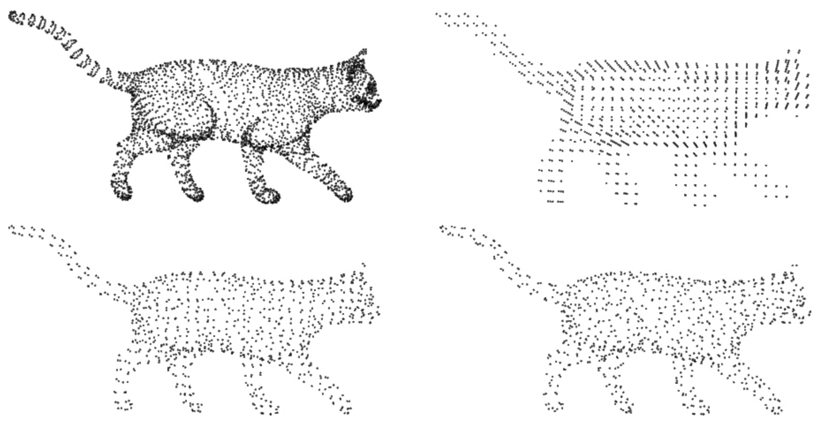


Figure 2.12: Subsampling approximation methods results. Input point cloud (top left), centre of voxel cube (top right), centroid of points within voxel (bottom left) and original point nearest to centroid (bottom right). Original input point cloud has 3400 points, the three subsampled point clouds have 780 points.

they are employed to estimate the motion between 3D datasets. We also discussed the main issues driving the different research strands in the area of surface matching and motion estimation.

Additionally, we listed the most recent advancements in available 3D sensing technology and described the properties of the output data these sensors provide. We also discussed properties of point clouds and subsampling methods, introducing a variation to current techniques that provides a better representation of the original subsampled point cloud.

In the following chapters we present novel evidence gathering algorithms that provide practical solutions to 3D perceptual tasks, without relying on feature computing and matching. We propose an algorithm for 3D reconstruction of rotating objects that is based on motion-compensated temporal accumulation, two methods for 3D static parametric object detection and 3D moving parametric object extraction, based on the Hough transform, and a new method for 3D motion estimation based on Reuleaux's and Chasles' kinematic theorems employing invariant properties that can be inferred from the motion.

Chapter 3

3D Moving Object Reconstruction by Temporal Accumulation

3.1 Introduction

The reconstruction of 3D models of rigid objects is generally achieved by the following steps: First the data acquisition step where point clouds or range images (depth maps) are generated by the 3D sensor. This data is 2.5D where only the surfaces facing the sensor are captured. Secondly, an optional segmentation and filtering step is applied to separate the observed object from its background. Thirdly scans from different viewpoints are aligned together in one coordinate frame (registration). Then the aligned scans are typically resampled and merged (integrated) by surface reconstruction techniques into a seamless 3D surface and rendered for display.

We present an algorithm for 3D reconstruction of point clouds that is based on motion-compensated temporal accumulation. Given a fixed centre or axis of rotation, the algorithm estimates the best rigid transformation parameters for registration, and reconstructs the full geometry of rotating 3D objects from 2.5D point clouds. This algorithm accumulates surface information over the sequence after applying a series of rigid transformations on the point clouds of the input sequence, and then resamples the accumulated point clouds based on a computed space partitioning and votes for the best reconstruction based on the number of points in each resampled point cloud.

The novelty of the algorithm relies in the fact that it does not not require key-point detection, feature description, correspondence matching, any subsequent ICP refinement steps, provided object models or any geometric information about the object. Moreover, the algorithm performs surface resampling, noise reduction and estimates the optimum angular velocity of the moving object integrally.

For this research, we assume that the object is already segmented, the motion is purely rotational with a given axis or centre of rotation, and the angular velocity is approximately constant subject to monotonic change. However the algorithm can be generalised to any rigid motion.

This chapter is structured as follows: first we will discuss the related work to our algorithm in terms of registration and temporal accumulation. Second, we give a detailed overview of our algorithm with establishing the foundations for understanding its methodology. Then we validate and verify the theory and give experimental analysis. After that we show experimental results on synthetic and real data and finally we evaluate the performance of the algorithm in terms of noise and processing time and conclude this chapter.

3.2 Related Work

3.2.1 Registration

As described in Chapter 2, registration concerns finding the transformation (rotation and translation) that aligns data sets into one global coordinate system. The goal is to find transformations that aligns point clouds acquired from different views to one consistent 3D point cloud model correctly representing the object.

There are many methods for 3D data registration which can be classified in a number of ways. Generally, registration methods can be divided to rough (global) and fine registration (local), a detailed review of methods for 3D registration is in [65]. The dominant and most widely used method for local registration of three dimensional data is the ICP (Iterative Closest Point) algorithm [66] and its variants [21]. The high speed of the ICP algorithm and its variants resulted in development of many real time, high quality dense 3D model acquisition methods using structured light [67, 68, 69], and more recently using 3D sensors, such as the Microsoft Kinect [70]. ICP based methods require an initial alignment between 3D point sets to be known, as it depends on local optimisation and not guaranteed to the global optimal alignment, and require significant overlap between point clouds to be registered.

3.2.2 Temporal Accumulation

A temporal accumulation algorithm was used in [71] to determine bulk motion of walking people in 2D image sequences for gait analysis purposes. [72] used temporal accumulation of Fourier descriptors to extract arbitrarily moving arbitrary shapes in 2D images, by tracing a locus of votes in the form of the template shape, adjusted for the estimated motion of the object relative to the time reference of each frame. [73] perform temporal evidence accumulation on stereo image sequences for extraction of specified objects undergoing linear motion.

A colour-augmented search algorithm is used in [74] to accumulate coloured point clouds from successive time frames for a moving vehicle, to track the vehicle and build a 3D model of it. Their algorithm requires 3D point clouds and a colour image to align successive frames. The alignment is found by a pre-filtered iterative coarse to fine optimisation by first aligning the centroids of each colour-interpolated point cloud and then projecting them to a 2D occupancy grid and then by voting based on Euclidean as well as colour difference between point pairs.

This approach is similar to ours in terms of using the best alignment of the point clouds of the object to estimate velocity. However it is very different in its methodology for finding the alignment and input data. Their algorithm assumes a small motion between frames, and hence alignment based on centroid locations is most likely to be a "good enough" rough initial alignment. If the motion is large, which leads to no significant overlap between frames, alignment would fail and hence the registration would be false.

Our algorithm does not require any colour information, computation of object centroids, projecting to a 2D plane or significant overlap between frames.

3.3 Algorithm

The proposed algorithm finds the registration of a series of point clouds based on motion-compensated temporal accumulation. Motion compensation is back-projection along the expected line of motion to convert the coordinates to the same temporal frame of reference as the initial frame. The algorithm takes each cloud in the sequence of a rotating 3D object and applies a series of rigid transformations based on velocities given in a predefined range. Transformed point clouds are accumulated together based on their velocity, each velocity in the range will have a corresponding "velocity cloud" that is the accumulation of point clouds transformed by a transformation corresponding to that velocity. An octree-based [26] space partitioning algorithm is applied to downsample each velocity cloud, and the resulting downsampled velocity cloud with the minimum number of points is taken as the truest representation of the object.

3.3.1 Rotation in 3D

By definition, a rotation R about an origin is a transformation that preserves the origin; so rotations centre on the coordinate system, as matrix or vector multiplication has no

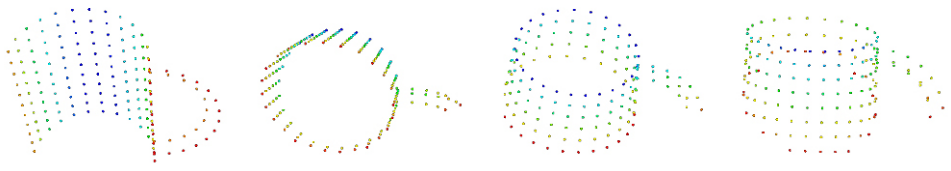


Figure 3.1: Four frames from a 2.5D sequence of rotating object.

effect on the zero vector (the coordinates of the origin). To rotate an object P around any fixed point $a = (a_x, a_y, a_z)$ in 3D space, we take the fixed point as the origin of a Cartesian coordinate system, then apply the rotation. This is done by applying a translation vector $\mathbf{t} = -[a_x, a_y, a_z]^\top$, then rotation, then translating back to the point of rotation, given by:

$$P' = \mathbf{t}R\mathbf{t}^{-1}P \tag{3.1}$$

Alternatively, a rotation around an axis in the direction of unit vector u and passing through the point $a = (a_x, a_y, a_z)$ is given by the following transformation matrix:

$$T(\theta, u, a)| = \begin{bmatrix} u_x^2 + (u_y^2 + u_z^2)\cos\theta & u_xu_yc - u_z\sin\theta & u_xu_zc + u_y\sin\theta & \mathbf{t_x} \\ u_xu_yc + u_z\sin\theta & u_y^2 + (u_x^2 + u_z^2)\cos\theta & u_yu_zc - u_x\sin\theta & \mathbf{t_y} \\ u_xu_zc - u_y\sin\theta & u_yu_zc + u_x\sin\theta & u_z^2 + (u_x^2 + u_y^2)\cos\theta & \mathbf{t_z} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.2}$$

$$\mathbf{t} = \begin{bmatrix} (a_x(u_y^2 + u_z^2) - u_x(a_yu_y + a_zu_z))c + (a_yu_z - a_zu_y)\sin\theta \\ (a_y(u_x^2 + u_z^2) - u_y(a_xu_x + a_zu_z))c + (a_zu_x - a_xu_z)\sin\theta \\ (a_z(u_x^2 + u_y^2) - u_z(a_xu_x + a_yu_y))c + (a_xu_y - a_yu_x)\sin\theta \end{bmatrix} \tag{3.3}$$

where $c = (1 - \cos\theta)$.

Rotations in 3D can be defined by the axis-angle representation in which a 3D unit vector represents an axis and a scalar angle describes the magnitude of the rotation about that axis. Six degrees of freedom are needed to describe a rigid transformation in 3D; three to determine position of the axis or point of rotation, two for the normalized direction vector of the axis and one for the scalar angle.

3.3.2 Angular Velocity

Having a sequence $C = \{P_0, P_1, \ldots, P_{N-1}\}$ of N point clouds, where each point cloud P_t represents a set M of 3D points distributed in $\mathbb{R}^3$, $P = \{p_0, p_1, \ldots, p_{M-1}\}$, $p_i = \{x_i, y_i, z_i\}$ of a rigid object rotating around a fixed axis of a direction given by the unit vector $u = [u_x, u_y, u_z]$, and passing through the point $a = (a_x, a_y, a_z)$, with angular velocity w, given by the equation:

$$w(\theta, u) = \frac{d\theta}{dt}u \tag{3.4}$$

```
for every point cloud $P_t$ $(t = 0; t < N; t{+}{+})$ do
    for every angular velocity $w_j$ $(w_j = w_{min}; w_j = w_{max}; w{+}{+})$ do
        Rotate $P_t$ by $R(w_j, t)$
        Add to velocity cloud $V_j$
    end
    Save $V_j$ in $A$
end
for every $V_j$ in $A$ do
    Build an octree representation $O_j$ for $V_j$
    for every occupied voxel in $O_j$ do
        Find centroid of points within voxel
        Add centroid point to downsampled cloud $D_j$
    end
    Save $D_j$ in $S$
end
Search $S$ for $D_j$ with minimum number of points.
```

Algorithm 1: 3D object reconstruction by temporal accumulation.

which is defined as the rate of change of angular displacement θ measured by degrees per unit time t, which with a time reference relative to the arbitrary start point (e.g. point cloud number versus point cloud 0). The angular velocity describes the speed of rotation and the orientation of the instantaneous axis about which the rotation occurs.

The first step of the algorithm is to rotate each point cloud in the sequence by a series of rotations; the instantaneous rotation angle θ is given by multiplying the angular velocity w by time t:

$$\theta = wt \tag{3.5}$$

where $w \in W = [w_{min}, w_{max}]$. At any angular velocity, according to [75], the instantaneous rotation by an angle of θ about an axis in the direction of unit vector u is given by:

$$R(\theta, u)|_{w,t} = \begin{bmatrix} \cos\theta + u_x^2 c & u_x u_y c - u_z \sin\theta & u_x u_z c + u_y \sin\theta \\ u_y u_x c + u_z \sin\theta & \cos\theta + u_y^2 c & u_y u_z c - u_x \sin\theta \\ u_z u_x c - u_y \sin\theta & u_z u_y c + u_x \sin\theta & \cos\theta + u_z^2 c \end{bmatrix} \tag{3.6}$$

where $c = (1 - \cos\theta)$. If the position and direction of the axis of rotation are given, to find the rotation we need to solve for only one degree of freedom, that is the angle of rotation θ.

However if only a point of rotation is known, i.e. the direction of the axis of rotations remains unknown, the dimensionality of the problem increases, as there are three degrees of freedom to be solved. This can be solved by applying Equation 3.6 three consecutive times; by taking three angles around the three main axis. Thus the angular velocity will

	P_0		P_1		$\cdots$		P_{N-1}		$\sum$
w_{min}	$w_{min}P_0$	$+$	$w_{min}P_1$	$+$	$\cdots$	$+$	$w_{min}P_{N-1}$	$=$	V_0
w_1	w_1P_0	$+$	w_1P_1	$+$	$\cdots$	$+$	w_1P_{N-1}	$=$	V_1
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
w_{max}	$w_{max}P_0$	$+$	$w_{max}P_1$	$+$	$\cdots$	$+$	$w_{max}P_{N-1}$	$=$	V_{W-1}

Table 3.1: Temporal accumulation process.

be presented by three angles instead of an angle and an axis, as given by:

$$w = \left(\frac{d\gamma}{dt}\hat{x}, \frac{d\phi}{dt}\hat{y}, \frac{d\psi}{dt}\hat{z}\right) \tag{3.7}$$

where γ, ϕ, ψ are angles of consecutive rotations about the X, Y, Z axis respectively, and $\hat{x}, \hat{y}, \hat{z}$ are the unit vectors of these axes.

Similar to the axis-angle configuration, the consecutive instantaneous rotation angles γ, ϕ, ψ are given by multiplying the corresponding component of the angular velocity w by time t. The instantaneous rotation R is given by:

$$R(\gamma, \phi, \psi)|_{w,t} = \begin{bmatrix} \cos\phi\cos\psi & \cos\psi\sin\phi\sin\gamma - \cos\gamma\sin\psi & \sin\gamma\sin\psi + \cos\gamma\sin\phi\cos\psi \\ \cos\phi\sin\psi & \cos\gamma\cos\psi + \sin\gamma\sin\phi\sin\psi & \cos\gamma\sin\phi\sin\psi - \sin\gamma\cos\psi \\ -\sin\phi & sin\gamma\cos\phi & \cos\gamma\cos\phi \end{bmatrix} \tag{3.8}$$

The final rotated point cloud P' is given by Equation 3.1.

The reason of using this representation in this case is that voting for parameters in the temporal accumulation process is faster than applying Equation 3.6 three consecutive times, when these parameters are angles rather than 3D vectors. As the degree of freedom to solve for is one scalar angle in the case of known axis, when we use Euler angles we retain the consistency of the algorithm by simply increasing the voting process dimensionality to three scalar angles.

3.3.3 Temporal Accumulation

In the second stage of the algorithm, all point clouds rotated by the rotation corresponding to the same angular velocity are accumulated together; each angular velocity $w_j \in W$ will be represented as a point cloud V_j which is the sum of all object points of all point clouds rotated by this velocity.

$$V_j = \sum_{t=0}^{N-1} P'_{j,t} = \sum_{t=0}^{N-1} R|_{w_j,t} P_t \tag{3.9}$$

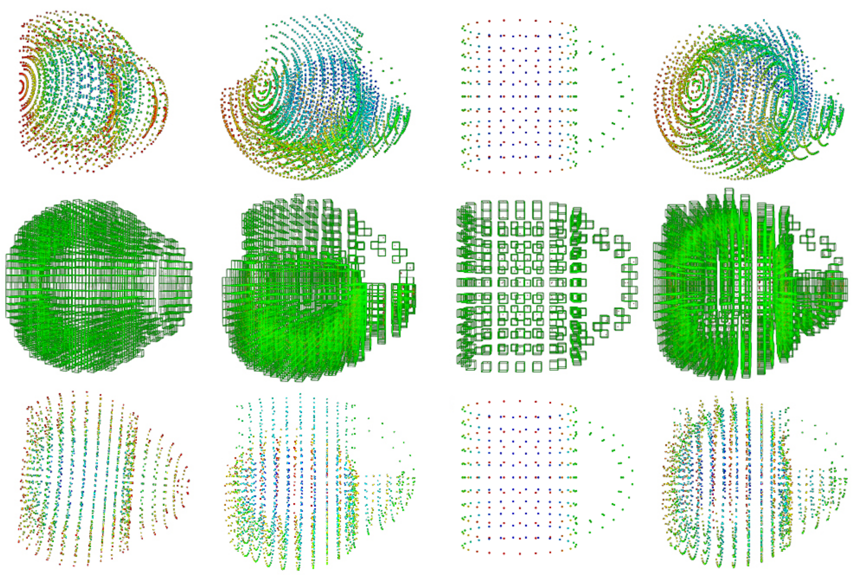


Figure 3.2: Steps of proposed algorithm. Top row shows velocity clouds (0-3) accumulated from the 2.5D sequence according to angular velocities (0-30) dt^{-1} respectively. Middle row shows the octree space partitioning of the velocity clouds into voxels. Bottom row shows the downsampled velocity clouds. The correct representation of the object is the downsampled point cloud with minimum number of points.

The total temporal accumulation process (A) of the complete sequence can be summarised by the equation:

$$A = \bigcup_{j=w_{min}}^{w_{max}} V_j \tag{3.10}$$

For each accumulated "velocity point cloud" V_j, the 3D space is spatially partitioned into fixed-sized volumes (voxels) using an octree data structure, such that all points $p_i \in V_j$ are associated with a voxel $v_k \in O_j$, where O_j is the octree representation of V_j.

Then a downsampling process is performed such that all points within each of the voxels in the octree are approximated with only one point which is their centroid. All velocity point clouds will have the same number of points before downsampling. However after downsampling, points from different point clouds transformed by a rotation corresponding to the correct angular velocity will theoretically coincide, or near enough to be in the same voxel, requiring less voxels in the voxel grid, and subsequently less points in the down-sampled velocity point cloud D_j, Figure 3.2. Hence the downsampled velocity point cloud with the least number of points (D^*) is the truest alignment of the unknown 3D object:

$$D^* = \underset{M}{\operatorname{argmin}}(S) \tag{3.11}$$

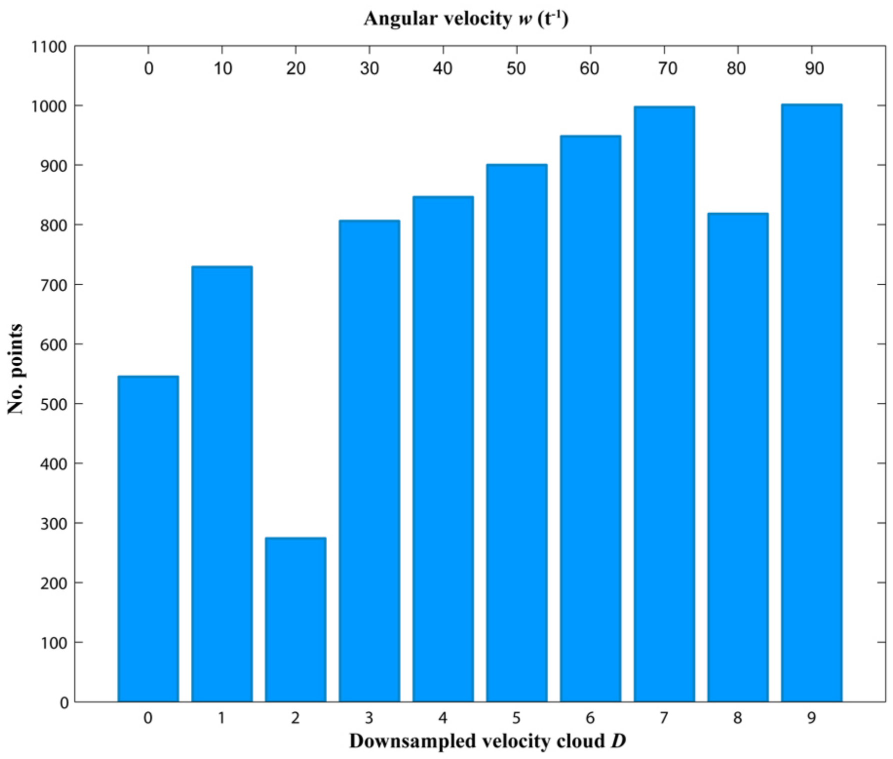


Figure 3.3: Histogram (1D accumulator space) for the Mug sequence using parameters described in Table 3.2 and Table 3.3. The downsampled velocity point with the least number of points (D_2) is the truest alignment of the 3D object.

where S is the stack of the downsampled velocity point clouds and M is the number of points in each cloud.

$$S = \bigcup_{j=w_{min}}^{w_{max}} D_j \tag{3.12}$$

The accumulator space used to vote for the best representation of the object is one-dimensional. It is a histogram of the number of points in the downsampled velocity clouds, Figure 3.3. Note that the bin size used in the accumulator is the rotational velocity step (angular resolution) ξ which is equal to 10 dt^{-1} for the Mug sequence.

We can also find the best estimate for the object's angular velocity ($\dot{w}$) by this equation:

$$\dot{w} = \arg\min_{w \in W}(S) \tag{3.13}$$

3.4 Analysis

The algorithm is verified by implementing it on a 2.5D synthetic sequence of a rotating object. Figure 3.1 shows four synthetic point clouds from a sequence of 31 point clouds of an object rotating about the horizontal X axis of the origin $(0, 0, 0)$. Only the front

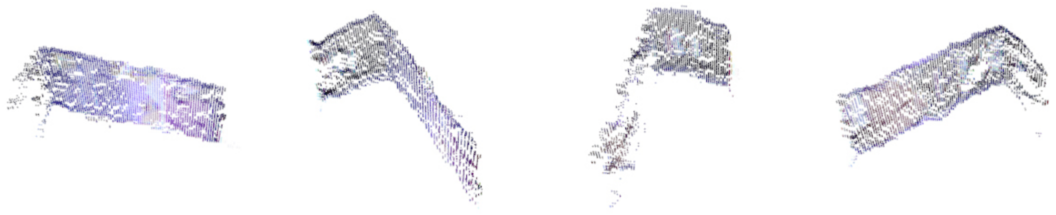


Figure 3.4: Four frames from a Kinect sequence of a rotating object.

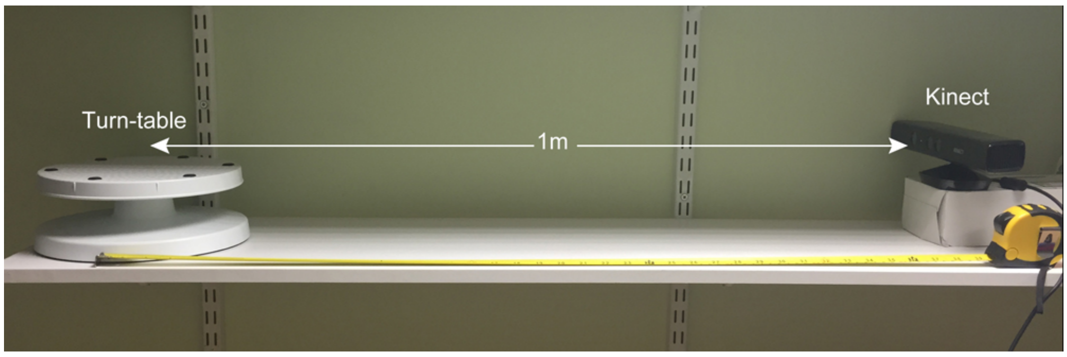


Figure 3.5: Configuration used to capture rotating point cloud sequences.

surface of the object is available to resemble real data generated by depth sensors. The object has an angular velocity of 10 degrees per unit time (dt^{-1}). Given a range of velocities $[0, 100]$ with a step of 10 dt^{-1}, we apply the algorithm; the point clouds are transformed and accumulated into the velocity clouds, the 3D space of each velocity point cloud is partitioned into voxels by an octree structure, and downsampled, Figure 3.2. Note that all velocity clouds have the same number of points before downsampling.

Similarly, the algorithm is implemented on a sequences of real point clouds placed on a turntable, Figure 3.4 shows four real Kinect point clouds from a sequence of eight point clouds in which the object was rotated around the vertical Y axis positioned at $(0, 20, 100)cm$. The object has an angular velocity of 45 degrees per unit time (dt^{-1}). Figure 3.6 shows the steps of the algorithm implemented on the real sequence. Full details of the experiments are in Table 3.2.

Note that all real point cloud sequences used in this chapter are generated by a Kinect positioned exactly $1m$ from the turntable, Figure 3.5. All objects are placed on the axis of rotation to show the performance of the algorithm in the case of self-occlusion, which is shown in Figure 3.6 (top left) where the accumulated frames have zero rotational velocity, i.e. no rotation.

If we take the histogram of the number of points in the voxels, the velocity cloud corresponding to the correct velocity should have a histogram with a peak value of the

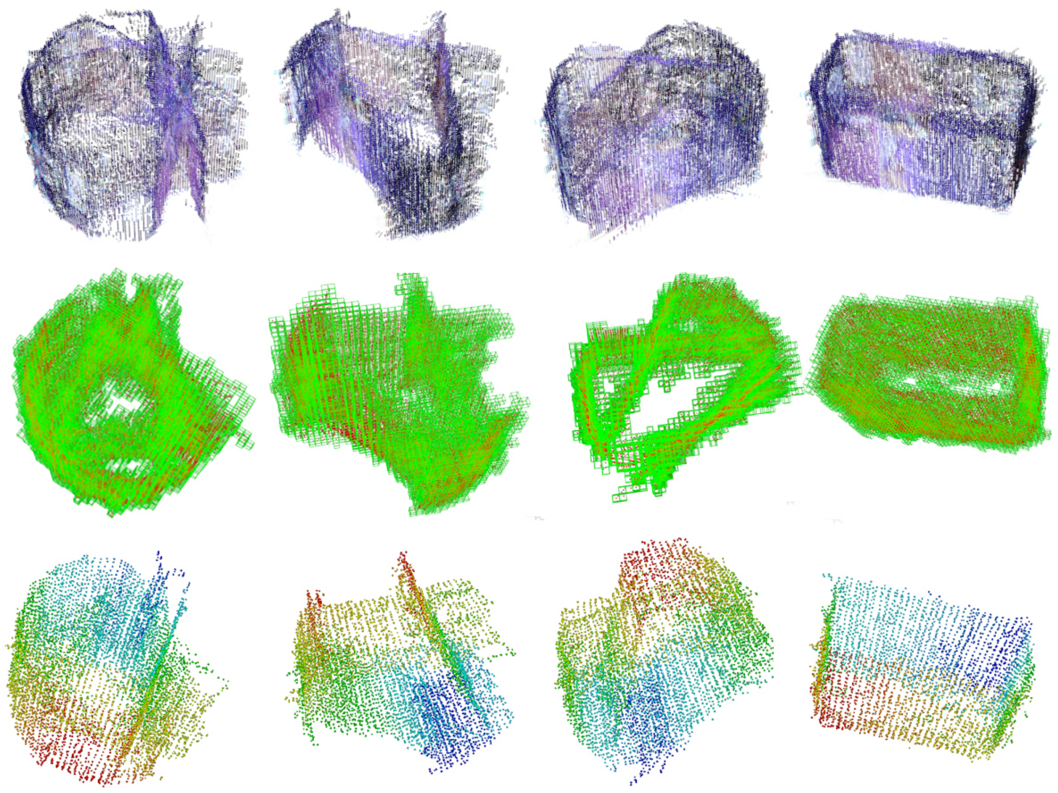


Figure 3.6: Steps of proposed algorithm implemented on a real point cloud sequence, similar to Figure 3.2.

number of point clouds in the sequence, 31 the case of the synthetic sequence shown in Figure 3.1, as each correctly rotated point cloud accumulates one point. However, because the sequence is 2.5D, as with real data generated by depth sensors, the histogram should be centred at approximately half the number of point clouds. Histograms of the rest of the accumulated velocity clouds will be centred towards the left at low number of points in the voxel and typically have multiple peaks with higher values for numbers of voxels, Figure 3.7.

3.4.1 Octree-based Voxelisation

This observation is based on the choice of voxel size relative to the density of the point clouds; in this case the voxel size was chosen to be small enough to allow only coincident or very close points to exist within the voxel. Larger voxels will still verify this observation, but will allow more points to be within them resulting in a higher number of peaks. After parameter tuning we set the voxel size for real and synthetic point clouds to $(10 \times 10 \times 10)mm^3$. The point cloud dimensions and the voxel size used in the octree determine the number of levels of the octree, Table 3.3.

The angular velocity also has an effect on the shape of the histogram; the larger it is, the smaller the overlap area between point clouds is and hence more voxels containing less

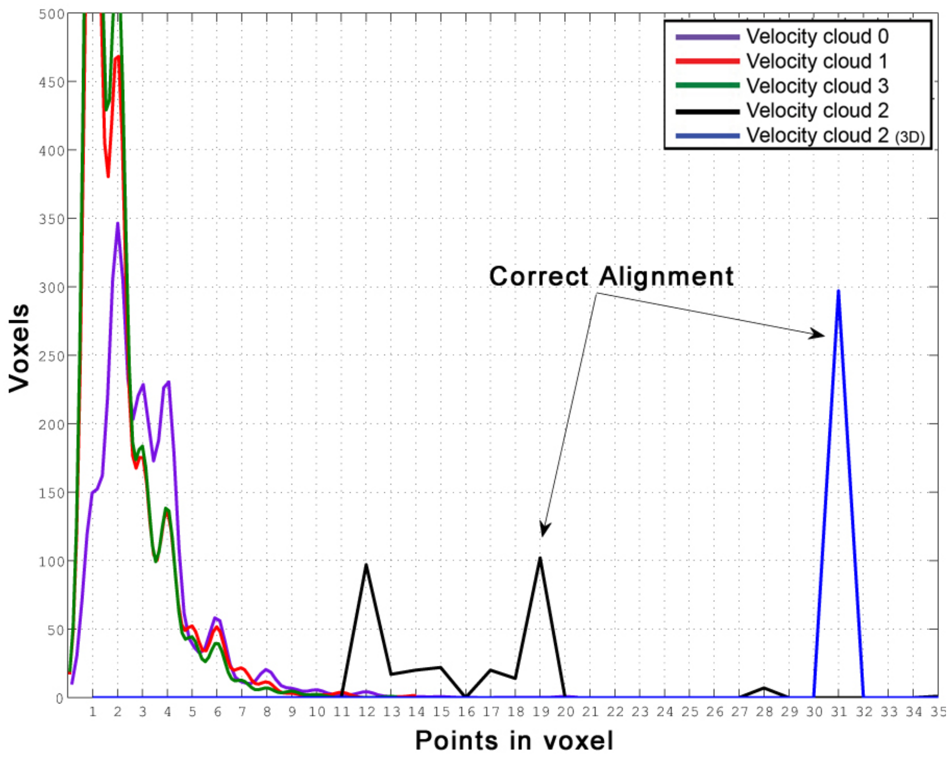


Figure 3.7: Histograms of accumulated velocity clouds. It is clear that the histogram of velocity cloud 2 is centered on higher number of points, thus it is considered the most correct alignment. The blue histogram is generated from the same velocity cloud but from a 3D sequence instead of 2.5D. Note that the peak is centered at the number of point clouds in the sequence (31), which verifies the methodology of the algorithm.

points. Additionally, the cloud's point density and the isometry of the object also affect the shape of the histogram. Nevertheless the significant property of the accumulated velocity clouds is the distribution of the histogram rather than the number of peaks or their amplitude.

3.5 Results

Figure 3.8 show 2.5D point clouds from four sequences, one synthetic and three real. And Figure 3.9 shows the reconstructed full 3D point clouds generated using the proposed algorithm. Details of the obtained results are in Table 3.2.

3.5.1 Surface Resampling and Reconstruction

The final step of every 3D model acquisition method is to reconstruct the surface of the object after estimating the alignment between multiple scans. When point clouds are registered to one model, overlapping areas of different point clouds coincide, resulting

Sequence	N	$N \times M$	W	ξ (°)	Time (s)
Mug	31	4896	10	10	0.16
Cat	18	30600	150	1	13.01
Box	8	33453	10	15	1.46
Teddy	8	58458	30	10	6.65
Bird	8	18464	20	5	3.25

Table 3.2: Object reconstruction by temporal accumulation algorithm performance. Columns left to right: sequence name, number of point clouds, total number of points, number of velocities, rotational velocity step (bin size) and total processing time. (running on 2.4 GHz Intel Core i7 and 4GB RAM).

Sequence	dms (mm^3)	vox (mm^3)	No. vox	O levels
Mug	$119 \times 95 \times 80$	10	274	6
Cat	$350 \times 378 \times 193$	10	4233	8
Box	$184 \times 107 \times 100$	10	953	8
Teddy	$218 \times 254 \times 197$	10	3162	7
Bird	$145 \times 92 \times 140$	10	598	7

Table 3.3: Octree voxelisation parameters. Columns left to right: sequence name, optimum velocity cloud bounding box dimensions, voxel size, number of voxels in the octree and number of octree levels.

for the density of points to be multiplied. Moreover, typical quantisation errors and missing data from 3D sensors can generate empty areas in the point cloud. This variety of point density on the registered model negatively affects any subsequent rendering or recognition processes. This problem is solved by resampling. Whether it is up-sampling or downsampling, resampling methods aim to smooth surfaces to have an approximately constant point density, a robust resampling algorithm is presented in [76].

In our algorithm, resampling is integral; it is achieved in the stage before last of our algorithm when the points in octree voxels are approximated by their centroid. The resultant downsampled point cloud will have an approximately uniform point density; the density is dependent on the voxel size used in the octree.

Detailed surface reconstruction is beyond the scope of this research, here the Poisson surface reconstruction algorithm [77] is applied on the resultant cloud that is the output of our algorithm to reconstruct the 3D surface of the rotating object, Figure 3.10. For more accurate resampling that will produce smoother reconstructed surfaces, an upsampling and hole filling algorithm such as in [76] can be applied after using our algorithm to obtain the correct object velocity, and transform and accumulate the original point clouds of the sequence.

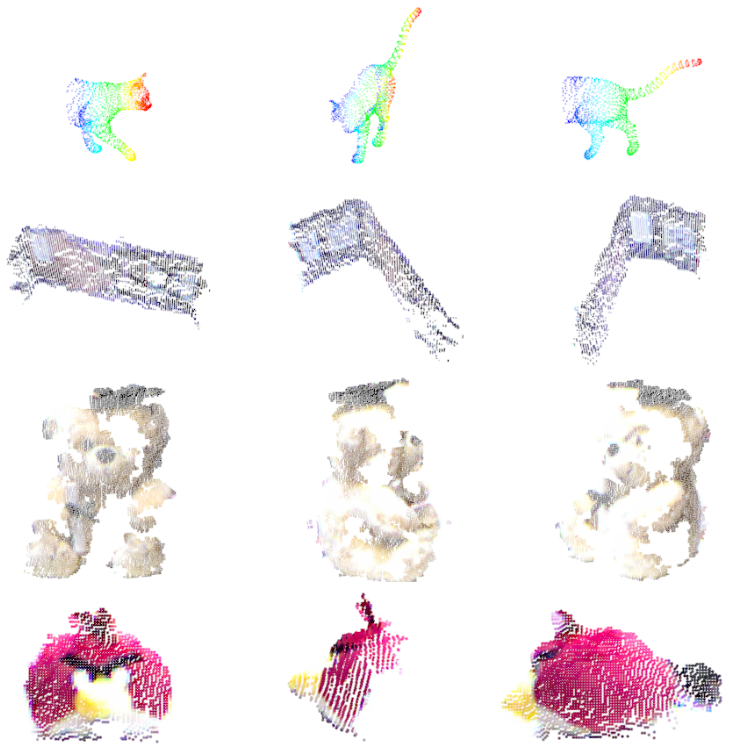


Figure 3.8: Three point clouds from four 2.5D sequences. Top row is of a synthetic sequence of 18 frames. Bottom three rows are of real objects placed on a turn-table, each sequences has 8 frames.

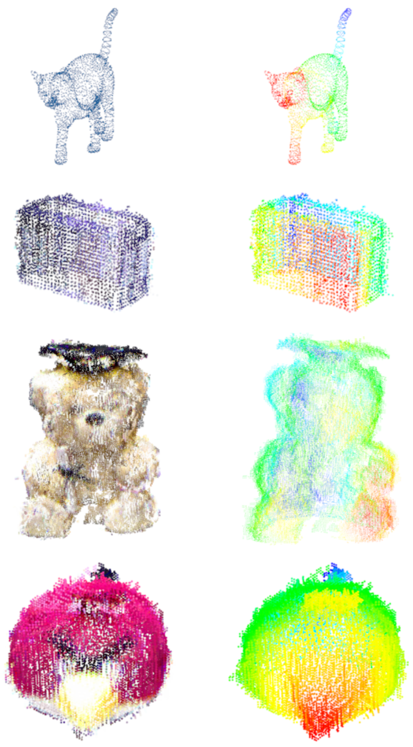


Figure 3.9: Reconstructed objects by proposed algorithm.

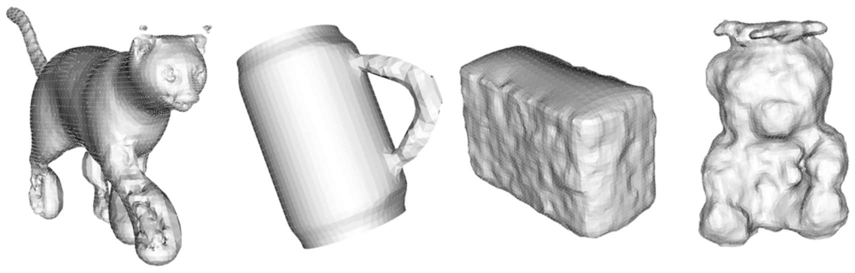


Figure 3.10: Poisson surface reconstruction.

3.6 Evaluation

3.6.1 Noise

Even with significantly corrupted input data, our algorithm was still able to detect the correct angular velocity and reconstruct the object. Moreover, the noise is integrally reduced in the reconstructed point cloud as a result of space partitioning and resampling stages of our algorithm, Figure 3.11. The performance of the algorithm is quantified against noise by conducting an actual implementation on data corrupted with noise.

Figure 3.13 shows the performance of the algorithm applied on a synthetic object with the presence of noise. The noise is modeled by adding a normally distributed random number (rnd) to each of the three coordinates of each point in the point cloud, Equation 3.14.

$$\begin{aligned} \dot{p}_x &= p_x + rnd \\ \dot{p}_y &= p_y + rnd \\ \dot{p}_z &= p_z + rnd \end{aligned} \tag{3.14}$$

Where $\dot{p}_i$ represent a 3D point with added Gaussian noise. The Gaussian (normal) distribution model G used to generate the random number has an increasing standard deviation (σ) and zero mean (μ), given by Equation 3.15.

$$G(x, \mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(x-\mu)^2/2\sigma^2} \tag{3.15}$$

The point clouds illustrated in Appendix A show the effect of adding Gaussian noise with increasing standard deviation. Note that the noise is added to all the point clouds in the sequence.

To give a useful meaning to the noise readings, we introduce the error metric ρ, which is given by the ratio of mean cloud-to-cloud distance to the width of the object, Equation 3.16.

$$\rho = \left(\frac{1}{N} \sum_{c \in C}^{N} \min_{g \in G} \|c - g\| \right) / width \times 100\% \tag{3.16}$$

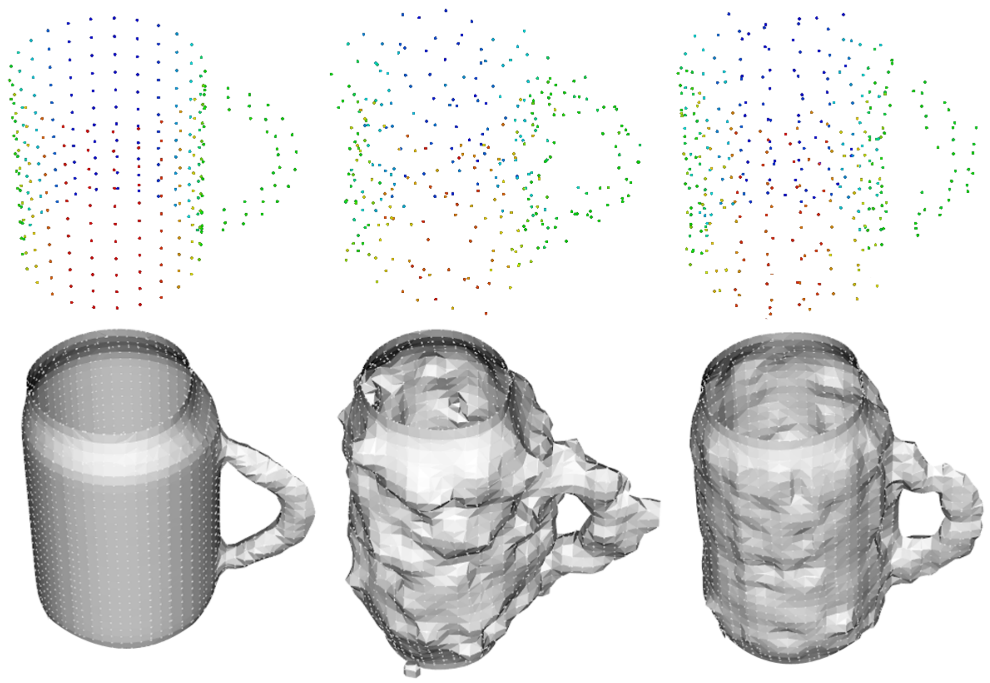


Figure 3.11: Noise reduction. Top row: Original point cloud from sequence of 30 clouds (left), added Gaussian noise ($\sigma = 4mm$) (middle), and result of implementing the proposed algorithm on noisy sequence (right). Bottom row shows the surface reconstruction of each point cloud.

Where N is the number of points in the reconstructed cloud C, and G is the ground truth model. The mean cloud-to-cloud (C2C) distance is the mean of all distances from each point in the reconstructed point cloud to its closest point in the original ground truth model.

3.6.1.1 Angular Resolution

As mentioned earlier, the bin size used in the accumulator is the rotational velocity step, or in other words the angular resolution (ξ). The value of the angular resolution relative to the object's rotational velocity affects the performance of the algorithm. For example, in the Mug sequence the object has a rotational velocity of $10dt^{-1}$, if the accumulator's angular resolution was chosen so that the value of the velocity occurs at an integer multiple of the bin size, such as $\xi = 2\ dt^{-1}$ or $\xi = 5\ dt^{-1}$ then the algorithm will give a better reconstruction of the object than other bin size values that are not an integer multiple of velocity such as $\xi = 3\ dt^{-1}$ or $\xi = 4\ dt^{-1}$.

Nevertheless, the algorithm will still estimate that "nearest" velocity value to the ground truth value and reconstruct the object based on that velocity. For example if the angular resolution was chosen to be $\xi = 3\ dt^{-1}$, the algorithm will estimate the object's velocity to be $9\ dt^{-1}$ as it is the nearest velocity value to the correct velocity which falls at a bin.

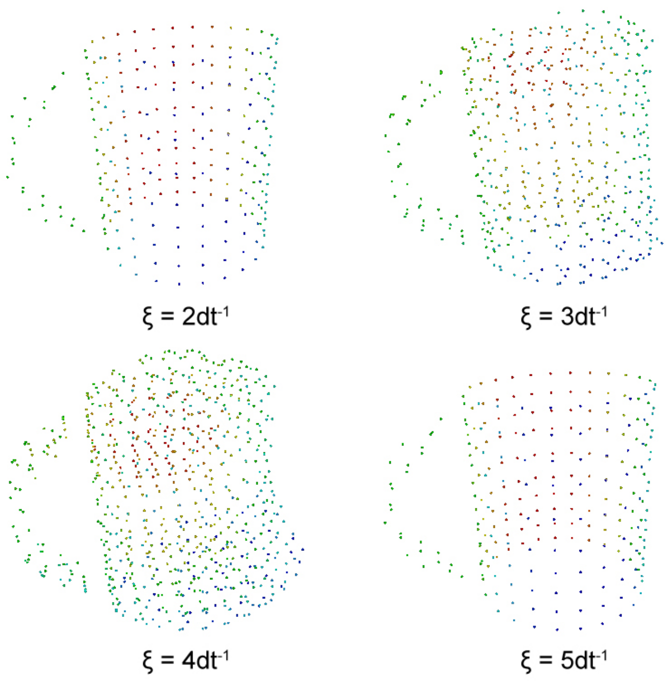


Figure 3.12: Reconstruction at different accumulator angular resolutions. Note that at angular resolutions that are divisors of the correct rotational velocity, the object is reconstructed with minimal error.

Figure 3.12 shows the reconstructed object at different values accumulator angular resolutions, the estimated rotational velocities for angular resolutions 2, 3, 4, 5, 10 $dt^{-1}$are 10, 9, 8, 10, 10 dt^{-1} respectively. Figures 3.14 to 3.17 show the performance of the algorithm against noise at different noise levels similar to Figure 3.13. The five figures 3.13 to 3.17 are summarised in figures 3.18 and 3.19. The figures clearly show that at angular resolutions (2, 5 and 10 dt^{-1}) which are divisors of the correct rotational velocity, the algorithm gives that same performance.

Similarly, if the angular velocity has changed relative to a fixed size angular resolution, the velocity will be estimated to the nearest resolution. The smaller the angular resolution the more accurate the reconstruction will be.

We showed that the algorithm integrally reduces the noise in the reconstructed point cloud as a result of space partitioning and resampling stages, Figure 3.11. This gives it the advantage to produce better reconstructions of objects than methods that use the frames directly without performing temporal accumulation or resampling, such as the ICP algorithm. Naturally any least square minimisation algorithm or the ICP algorithm can be used as an extra stage within the pipeline of the algorithm to better align the accumulated point clouds in each velocity cloud just before the downsampling stage.

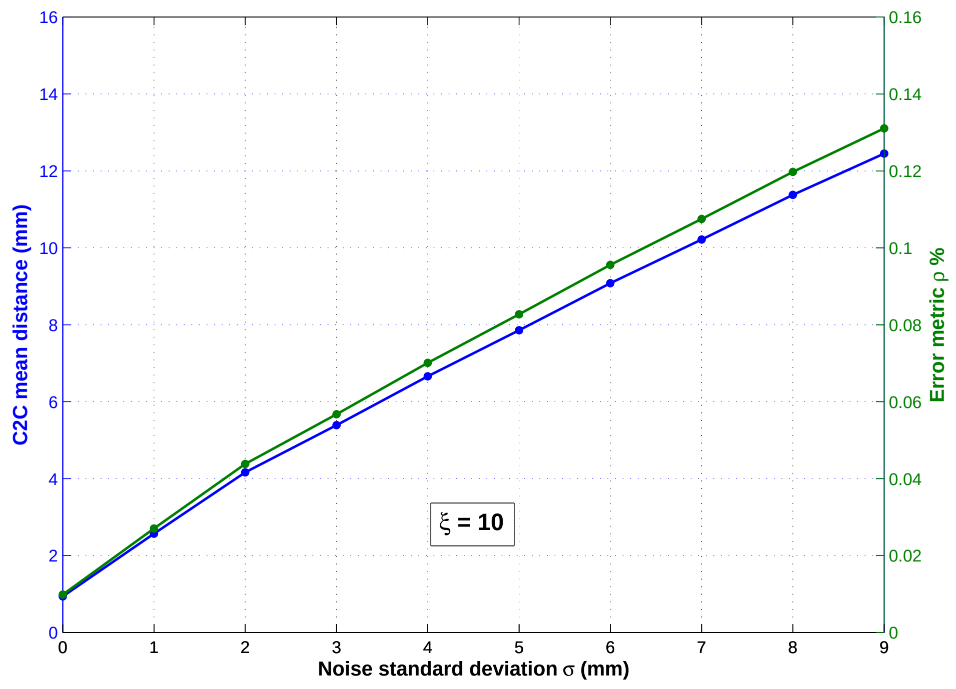


Figure 3.13: Performance of object reconstruction algorithm against noise at $\xi = 10dt^{-1}$. The mean C2C distance at no added noise is 0.93 mm, and the error metric $\rho = 0.010\%$, which assures the accuracy of the algorithm. This plot is generated by applying the algorithm on the Mug sequence using parameters described in Table 3.2 and Table 3.3.

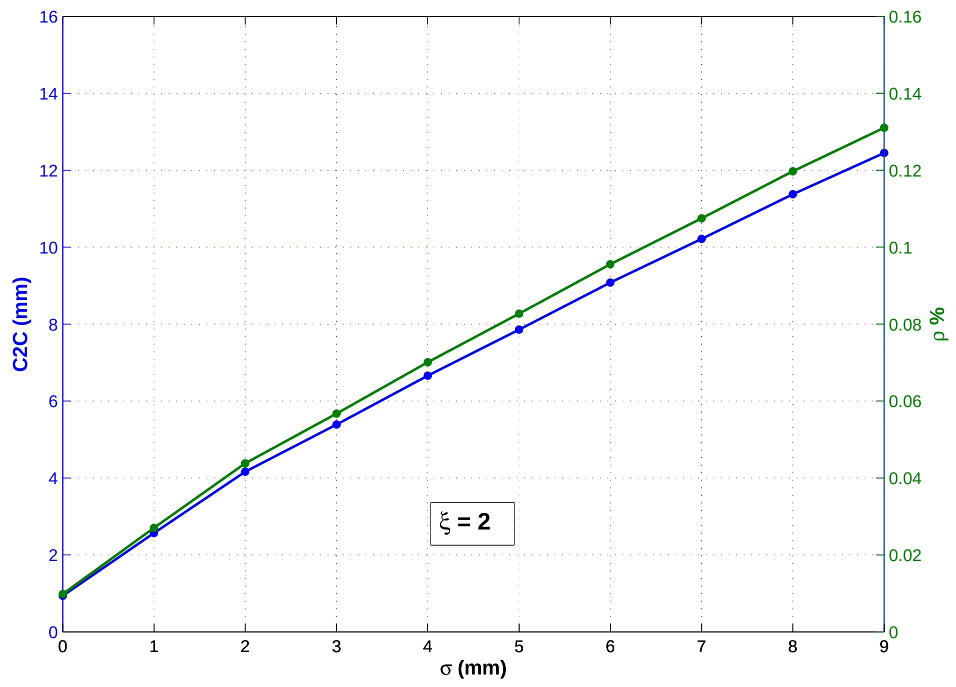


Figure 3.14: Performance against noise at $\xi = 2dt^{-1}$.

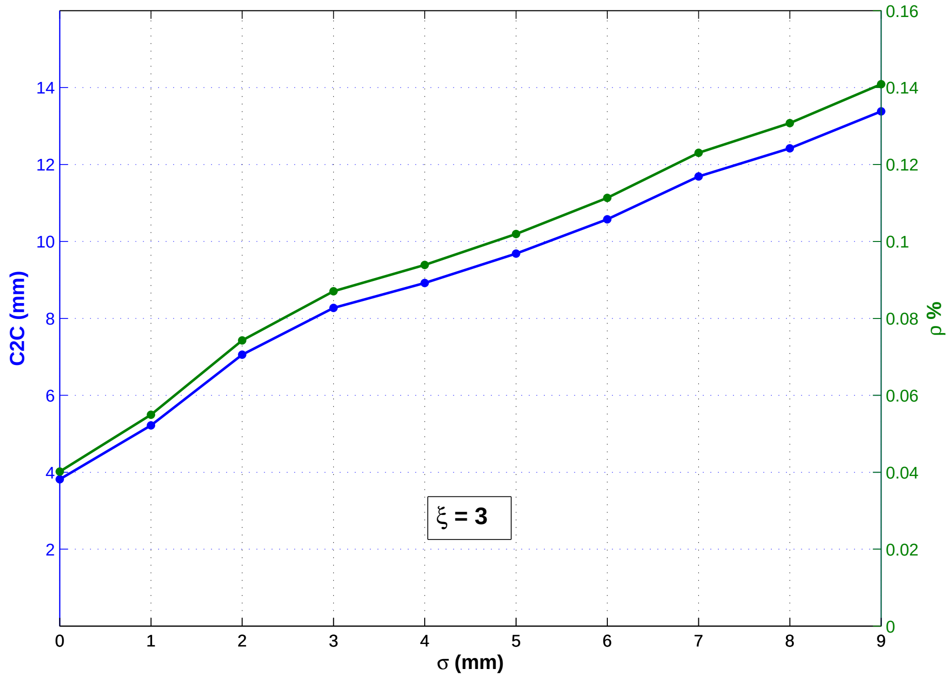


Figure 3.15: Performance against noise at $\xi = 3dt^{-1}$.

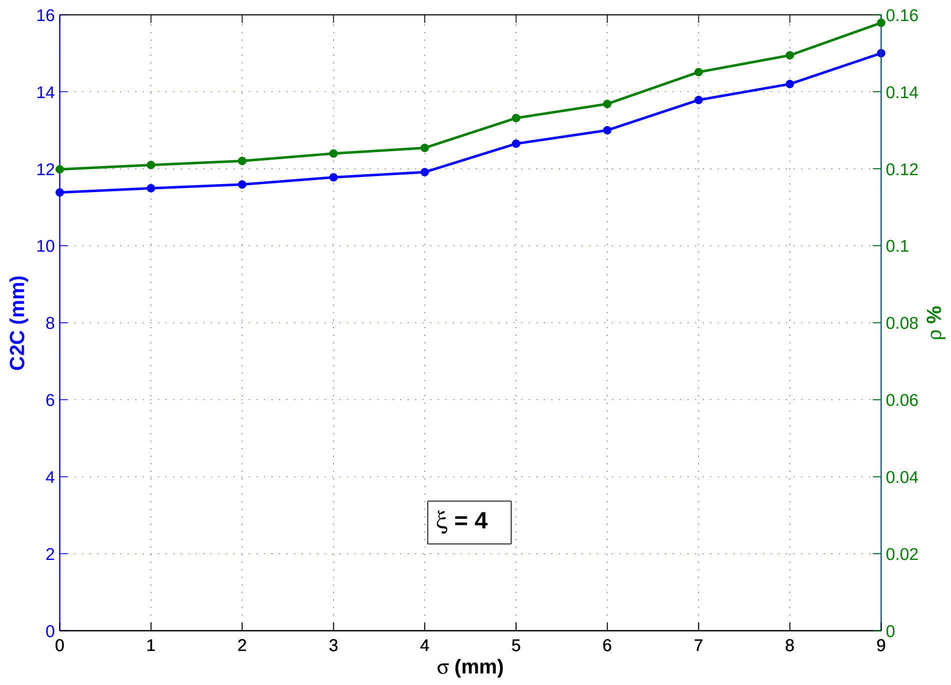


Figure 3.16: Performance against noise at $\xi = 4dt^{-1}$.

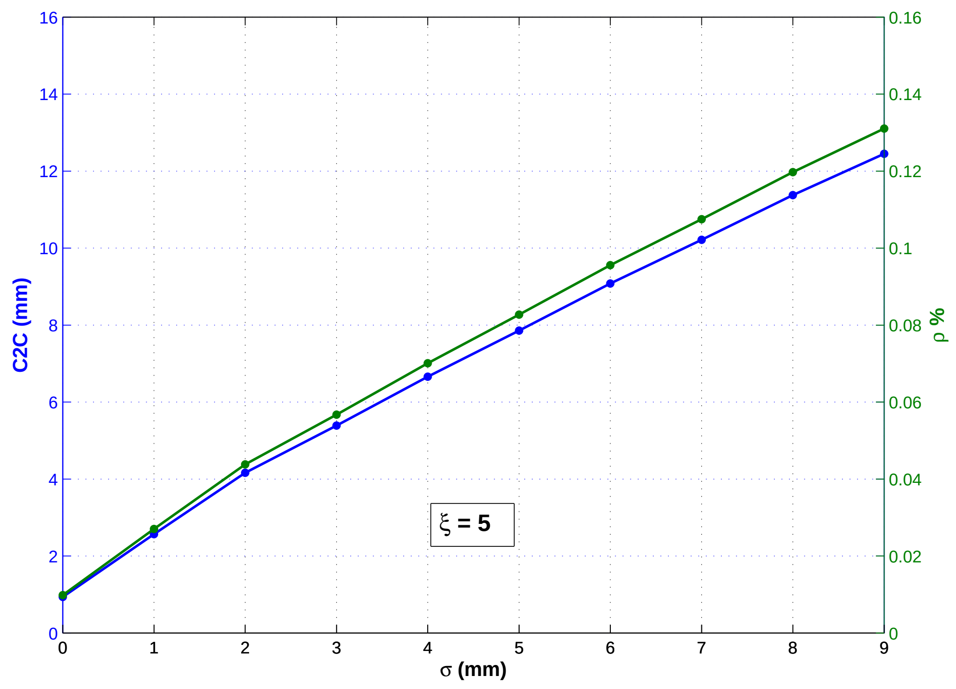


Figure 3.17: Performance against noise at $\xi = 5dt^{-1}$.

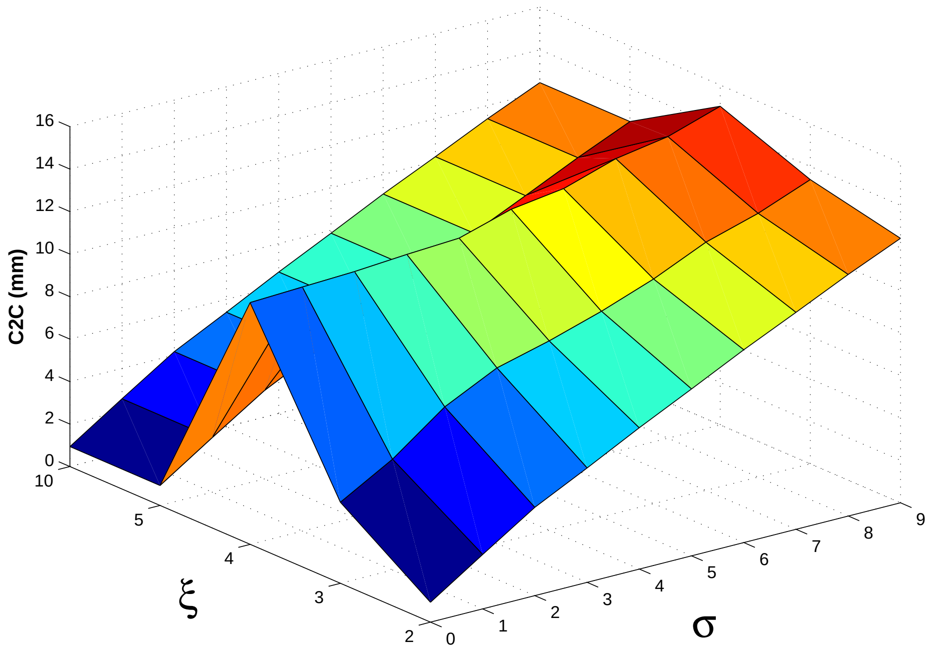


Figure 3.18: C2C (cloud to cloud) mean distance with varying angular resolution (ξ) and noise (σ).

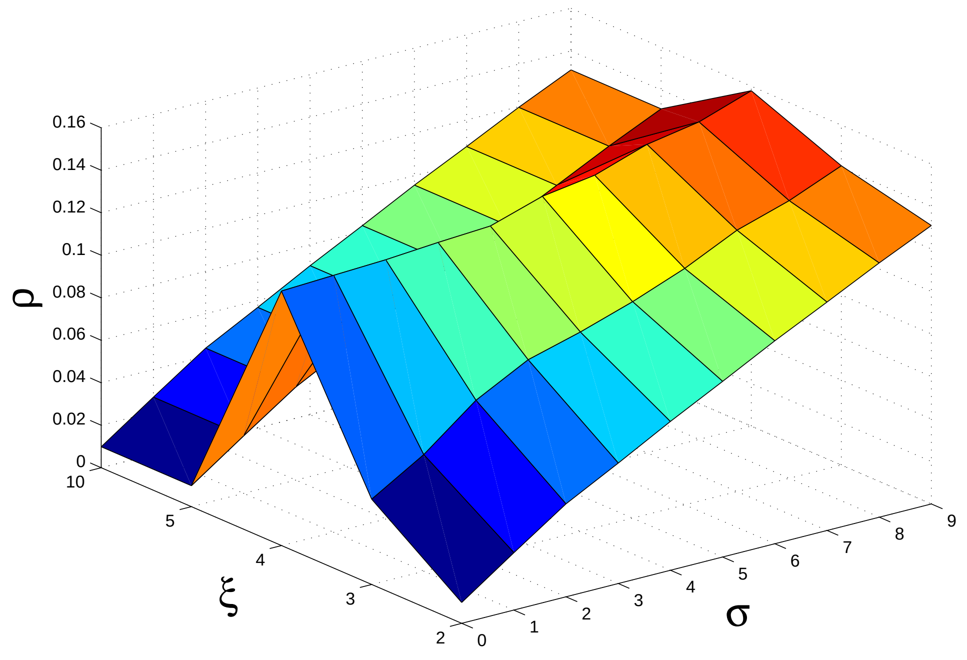


Figure 3.19: Error metric (ρ) with varying angular resolution (ξ) and noise (σ).

3.6.2 Time

The performance of the algorithm in the case of given axis of rotation is $O(N \cdot M \cdot W)$ while in the case of a point of rotation is $O(N \cdot M \cdot W^3)$. N is the number of point clouds, M is the average number of points in each point cloud, and W is the number of velocities in the given range. Table 3.2 shows the algorithm's processing time.

3.7 Conclusion

We presented a novel algorithm for full 3D reconstruction of unknown rotating objects in 2.5D point cloud sequences. Our algorithm incorporates structural and temporal motion information to build 3D models of rotating objects that is based on motion compensated temporal accumulation. We verified the theory of the algorithm, provided experimental analysis, showed experimental results on synthetic and real data which show that successful reconstruction can be achieved. A limitation of the proposed algorithm is the requirement of known point or axis of rotation; this problem is solved in Chapter 6.

The next two chapters present evidence gathering methods to detect and extract static and moving parametric objects such as spheres in 3D point clouds.

Chapter 4

The 3D Hough Transform for Sphere Detection

Object detection in 3D point clouds is gaining more interest due to the increasing popularity of 3D sensors within the research community. Even in complex scenes, many real objects can be approximated by parametric 3D shapes such as planes, spheres and cylinders.

Although for some 3D vision applications, direct measurements in point clouds may already suffice, most applications require an automatic processing of point clouds to extract information on the shape of the recorded objects. This processing often involves the recognition of specific geometric shapes or more general smooth surfaces [78].

In this chapter we introduce a fast and robust and Hough Transform (HT) based algorithm for detecting spherical structures in 3D point clouds. The use of a Hough transform provides resilience to noise and occlusion, and it is easily implemented in 3D. To our knowledge, our algorithm is the first HT based implementation that detects spherical structures in typical in 3D point clouds generated by consumer depth sensors such as the Microsoft Kinect. Our approach has been designed to be computationally efficient; reducing an established limitation of HT based approaches.

We are interested in detecting spheres in real data, which is point clouds acquired from the Kinect. Point clouds generated by the Kinect and similar sensors are 2.5D, i.e. only points on surfaces facing the sensor are acquired, which are about 50% of the total number of points on the object's surface if the object is symmetrical, such as a sphere, hence a complete geometry of the object cannot be acquired. Our algorithm is able to robustly detect partial spheres such those acquired by depth sensors, without this characteristic it would not be applicable on real data.

We begin this chapter by describing parametric 3D shape detection and listing related work. After that the theory of the 3D Hough transform for sphere detection is explained.

Next, details of the implementation of the algorithm are described and results of applying the algorithm on synthetic and real data are shown. Finally, we provide evaluation and performance analysis based on a number of elements.

4.1 Parametric Shape Detection in 3D

3D parametric object detection in 3D point clouds is fundamental to 3D computer vision. Parametric shapes are those shapes that can be represented by parametric equations, unlike arbitrary shapes which require more complex representations. In 3D, parametric shapes such as planes, spheres, cylinders and cones can be considered as geometric primitives that are used to explain parts of the 3D point cloud. According to [79], such geometric primitives enable the system to add surface information for parts of the object that were not covered by sensor data. Moreover, exploiting the knowledge about the shape enables to eliminate inaccuracies caused by sensor noise as it determines the best shape fit for the sensor data.

Detecting parametric shapes approaches in 3D data can be catagorised to two groups: model-fitting approaches that are based on RANSAC [80] and evidence gathering approaches which are besed on Hough transform (HT) [81].

RANSAC (RANdom SAmple Consensus) based approaches decompose the point cloud into a concise structure of inherent shapes and a set of remaining points using random sampling, each detected shape serves as a proxy for a set of corresponding points. These approaches have been proven to be efficient for detecting parametric 3D shapes (planes, spheres, cylinders, etc.), such as in [82].

The other direction of methods for detecting parametric shapes is to use evidence gathering techniques such as the Hough transform (HT), in which a voting procedure is carried in the parameter space from which object candidates are obtained as local maxima. Hough transform based methods have been successfully implemented to detect planes [83, 84], spheres [85] and cylinders [86] in three-dimensional data. A well known characteristic of the Hough transform is its resilience to noise and occlusion. Moreover, HT based methods in general have faster runtime than RANSAC. Moreover HT approaches can detect multiple instances of the shape while RANSAC methods are limited to a single instance.

In this research we will develop an HT based approach to detect spheres, not only for the mentioned reasons, but also for the ability to be extended to include motion parameters, discussed in the following chapter. Detailed comparisons between RANSAC and HT approaches for detecting parametric shapes in 3D data are in [87] and in [88].

4.2 Related Work

Detecting spheres in 3D data using the Hough transform is a topic that has not been investigated enough; one of the earliest approaches was by [89], who proposed a pipelined multi-window surface parameter estimation approach, this approach detects spheres and cylinders using a multi-window parameter estimation technique, multi-resolution k-tree parameter space searching and voting, and a conflict resolution process that eliminates invalid parameter hypotheses. This algorithm splits the parameter set in into several windows and uses a near neighbour voting strategy to reduce problems of clusters splitting across bins. This algorithms shows good results for low resolution range images, however no occlusion, noise, scene complexity were considered by these method.

[90] proposed the partitioned Hough transform for ellipsoid detection , multiple ellipsoids with the same rotational axis in low resolution range images can be detected by this method. The voting in this method is performed for seven parameters (three for centre position, three for deformation coefficients and one for rotation angle). The limitation of this method is that it only applicable for small datasets, and computationally unfeasible for large data sets. Moreover it requires gradient information to detect the orientation of the ellipse border at each step.

[91] proposed a technique to automatically determine the sub-voxel position and size of a sphere in unsegmented 3D CT and MRI images. This method detects the sphere centre using gradient vectors and voting in a 3D accumulator space. Then the sphere radius is determined by mapping gradient voxel values on a histogram and finding the peak. The limitations of this method that it requires gradient information and limited to a small radius range.

[92] proposed the hierarchical Hough transform to detect spheres in s stack of CT slices. This method takes all slices into consideration sequentially and a 2D accumulator array is used to obtain the projection coordinates of the sphere centre into every XY-plane. Then a 2D Hough transform for circle or circular detection is implemented to detect circle parameters in each CT slice. The limitation of this method is that it is not general to any 3D data type.

[93] presented a method for detecting and tracking a spherical shape canister in the space. This method tracks and detects spheres using an enhanced Hough Transform technique and H_∞ filter is proposed. Similar to [92], a 2D Hough transform is employed to detect circles assuming that a projected sphere on a pixel matrix always results in a circular shape.

[85] proposed a method for sphere detection in point clouds which employs an optimized accumulator design; the method is fast and accurate, however it is demonstrated on a high resolution point cloud of calibration object of a number of “perfect” spheres

Method	Source	G	M	R	O	N
Hsu and Huang [90]	4096 px RI	No	Yes	Yes	Weak	Weak
Taylor [89]	4096 px RI	No	Yes	Yes	Weak	Robust
Van der Glas [91]	MR stack	Yes	No	yes	Weak	Robust
Cao et al. [92]	CT stack	Yes	No	Yes	Weak	Robust
Kharbat et al. [93]	2D	No	No	No	Weak	Robust
Ogundana et al. [85]	>0.3M PC	No	No	No	Weak	Robust
Our method [2]	>0.3M PC	No	No	Yes	Robust	Robust
Camurri et al. [94]	>0.3M PC	No	Yes	Yes	Robust	Robust

Table 4.1: Comparison between different sphere detection methods. From left to right: method authors, source type and size (Range Images, MR and CT stacks, 2D images, Point Clouds), whether the method uses gradient information (G), whether it supports multiple sphere detection (M), whether it supports variable radius sphere detection (R), robustness level to outlier (O) and robustness level to noise (N). Source [94].

acquired by an optical shape measurement system (SMS). Moreover this method assumes that the number of spheres is fixed and known, which limits the practicality.

A very recent work [94] (published after this research was complete) proposed a novel combined HT-based method (CMPHT) for detecting spheres in point clouds, this method scans the point cloud in two steps: firstly with a single-point algorithm to identify a region of interest, and secondly, a multi-point algorithm refines the final detection. This method combines the advantages of both single point and multi-point approaches and shows robust results for synthetic data, however it is less stable with real datasets. [94] also included a comparison between established techniques summarised in Table 4.1 which shows how our method presented in this chapter and their method have been designed with the same objectives and both have less performance restriction and are intrinsically more robust than previous approaches.

4.3 Static Sphere Detection via The 3D Hough Transform

The Hough Transform (HT) [81] is a well-known technique originally developed to extract straight lines, since then it has been extended to extract parametric shapes (such as conic sections) and non-parametric shapes in 2D images. The Hough transform is recognized as a powerful evidence gathering tool in shape analysis that gives good results even in the presence of noise and occlusions. The implementation of Hough transform based methods defines a mapping from the image (or point cloud in case of 3D) to an accumulator space; the evidence is the votes which original image points cast in the accumulator space. In three dimensions, a sphere has a polar parametric representation given by the following equations:

$$\begin{aligned} x &= c_x + r\cos\theta\sin\varphi \\ y &= c_y + r\sin\theta\sin\varphi \\ z &= c_z + r\cos\varphi \\ (0 \leq \theta &\leq 2\pi, 0 \leq \varphi \leq \pi) \end{aligned} \tag{4.1}$$

Where x, y and z are the Cartesian coordinates of the surface points,, r is the radius expressed in the same units as the centre and surface points and θ and φ are the azimuthal and polar angles respectively, Figure 4.1.

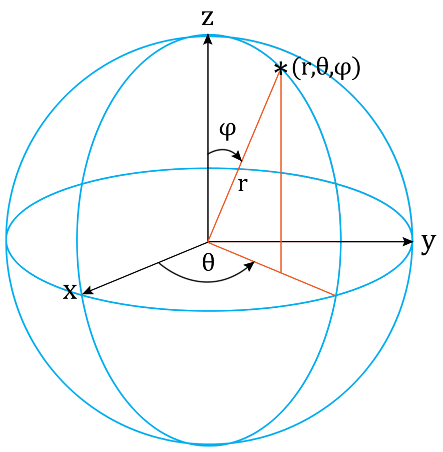


Figure 4.1: Spherical polar parameters in 3D.

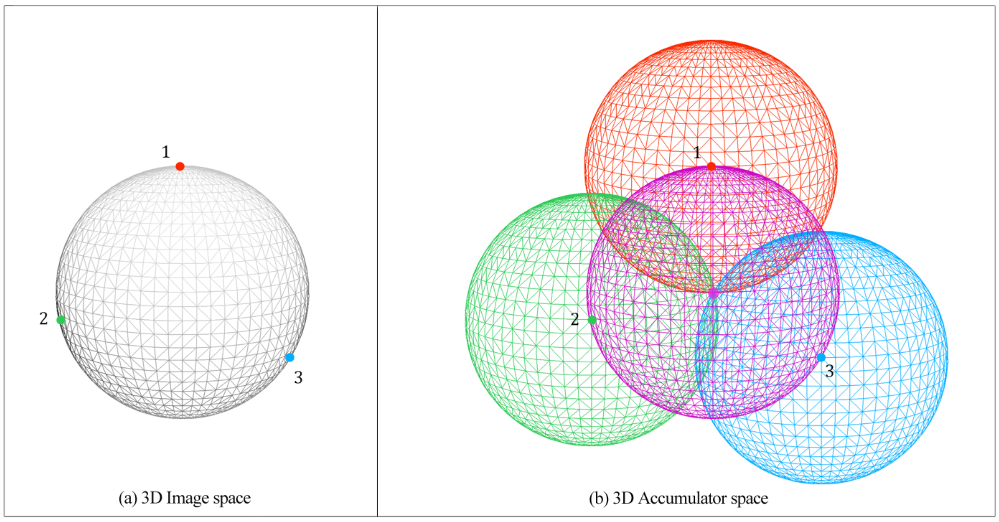


Figure 4.2: 3D accumulator space for a sphere with a predefined radius. Each point in (a) defines a sphere in the accumulator space (b). The intersection point of the spheres defined by points 1, 2 and 3, represents the parameters (centre coordinates) of the original sphere (a).

```
for $everypoint(x, y, z)$ do
    for $(r = r_{min};\ r \leq r_{max};\ r{+}{+})$ do
        for $(\theta = 0;\ \theta \leq 2\pi;\ \theta{+}{+})$ do
            for $(\varphi = 0;\ \varphi \leq \pi;\ \varphi{+}{+})$ do
                $c_x = x - r\cos\theta\sin\varphi$
                $c_y = y - r\sin\theta\sin\varphi$
                $c_z = z - r\cos\varphi$
                $\text{Accumulator}[r][c_x][c_y][c_z]++$
            end
        end
    end
end
Search Accumulator for peak.
```

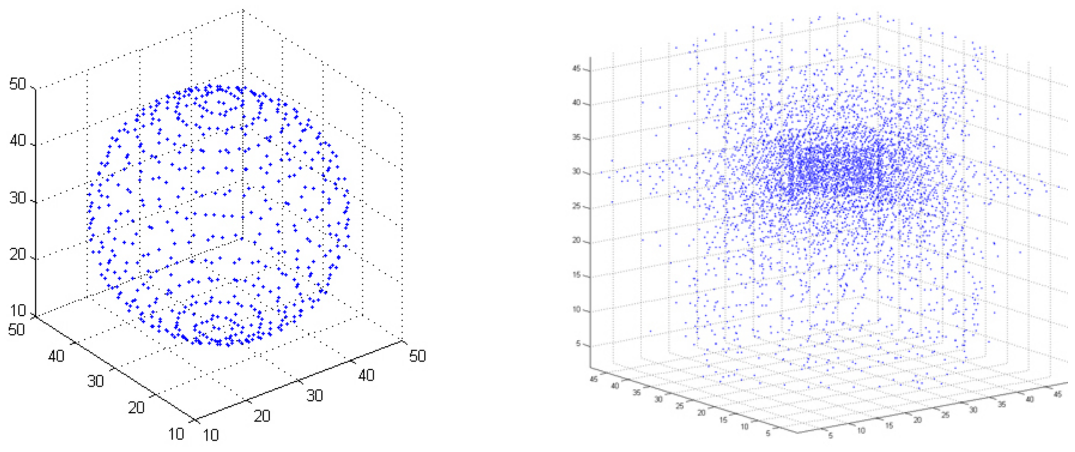
Algorithm 2: 3D Hough transform for sphere detection.

The algorithm of the 3D HT to detect spheres, Algorithm 2, is as follows: each point on the sphere surface defines a set of spheres in the accumulator space; these spheres are defined by all possible values of the radius, and they are centered at the coordinates of the point. For a given radius value, each point on the sphere defines only one sphere in the accumulator space. After gathering all evidence of the points on the sphere using the Hough transform mapping (Equation 4.2) applied for each point on the sphere's surface, the maximum of the accumulator space corresponds to the parameters of the original sphere.

Since we are dealing with 3D point clouds, there will be four parameters to vote for (cx, cy, cz and r), hence the accumulator space is 4-dimensional. Figure 4.2 shows a 3D accumulator space for a sphere with a predefined radius.

$$\begin{aligned} c_x &= x - r\cos\theta\sin\varphi \\ c_y &= y - r\sin\theta\ sin\varphi \\ c_z &= z - r\cos\varphi \end{aligned} \tag{4.2}$$

Figure 4.3(a) shows a point cloud in 3D space of a sphere with a predefined radius of 20 cm centred at $(30, 30, 30)$. The 3D accumulator space generated from voting for all points in the point cloud using Algorithm 2 is shown in Figure 4.3(b). Note that a threshold is applied to the accumulator space for visualisation purposes. Figure 4.4 shows three slices of the accumulator space, it can be easily seen the slice of the highest peak has the correct sphere parameter values.



(a) Sphere point cloud

(b) 3D accumulator space

Figure 4.3: (a) 3D point cloud of a sphere and (b) 3D plot of (thresholded) accumulator space generated by Algorithm 2.

4.4 Implementation

To reduce memory requirements, and for fast accessing of elements, we use a multi-dimensional sparse matrix representation for the accumulator space. Sparse matrices store only non-zero elements, and provide direct mapping between indices and their corresponding values. The non-zero elements are stored in a hash table that grows when it is filled so that the search time is $O(1)$ on average regardless of whether the element is present or not, and regardless of the size of the matrix. Sparse matrices solve the problem of parameter space dimensionality explosion very efficiently, which is a vital characteristic when dealing with multi-dimensional data. Moreover, for fast mapping to the Hough space, our algorithm uses sine and cosine values stored in look-up tables, instead of calculating the sine and cosine values at each iteration of the algorithm, which gives significant improvement on the processing time.

4.4.1 Peak Detection

Our algorithm determines the correct parameters of the sphere by a search operation on the 4D accumulator space, which finds the accumulator bin with the maximum number of votes (global maximum), Figure 4.4. As the novelty here resides more in the development of evidence-gathering techniques to 3D surface data, the techniques have used basic HT approaches such as global maximum detection, rather than techniques that filter the accumulator space (though these apply equally to the new algorithms described here).

A detailed survey and comparison of Hough accumulator peak detection algorithms is presented in [95]. [96] proposed a high accuracy 2D straight line parameters estimation

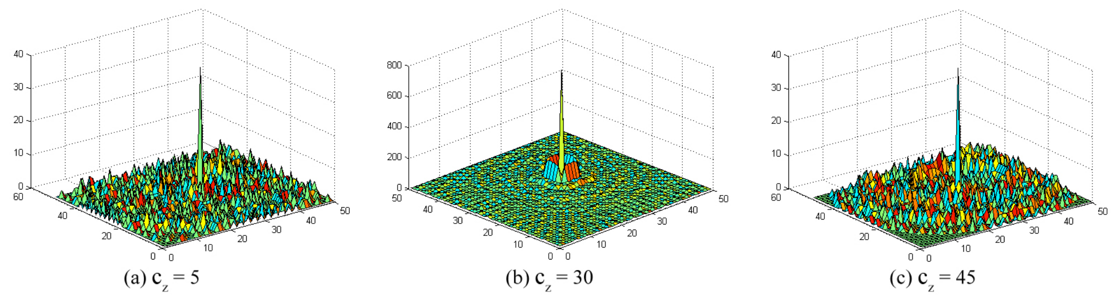


Figure 4.4: Distribution of votes in the three slices of the accumulator space (c_z = 5, c_z = 30 and c_z = 45). The highest peak is at the true centre of the sphere (b).

which consists of smoothing the Hough accumulator prior to finding its peak location, and interpolating about this peak to find a final sub-bucket peak.

[97] presented an optimization algorithm for locating peaks in the Hough accumulator with a voting kernel for detection of 2D lines. The authors presented a detailed discussion of the accuracy that can be achieved by locating these peaks in the accumulator, and show that the error bounds on the estimates of line parameters are always within those based upon least squares. The authors have also presented post-processing algorithms to remove outliers among the list of edge pixels associated with each line segment and to merge line pigments that have been split due to discretisation of the Hough parameter space.

The choice for an "obvious" direct search peak detection was due to its simplicity and because of the fact that it did provide satisfactory initial results. The results obtained on real and synthetic data, with the optimal choice of accumulator resolution, were with error values of less than 1 *cm*. Hence this research has been focused on the actual Hough mapping processes for static sphere detection and moving sphere extraction (next chapter) rather than developing sophisticated peak detection methods.

However, post-voting operations can be applied to the accumulator space to improve the accuracy of the peak detection. For example, smoothing the accumulator space using convolution before searching for the peaks, will allow parameters that lie close to the quantised parameter space and have been spread over multiple bins, to cast votes in the same bin, and hence better detection of the peak. Also, averaging the bin indices can provide better performance especially if the bin size is large.

4.5 Results

Figure 4.5 shows the results of applying Algorithm 2 on a number of point clouds of the same synthetic scene having spheres of different radii and locations in 3D space.

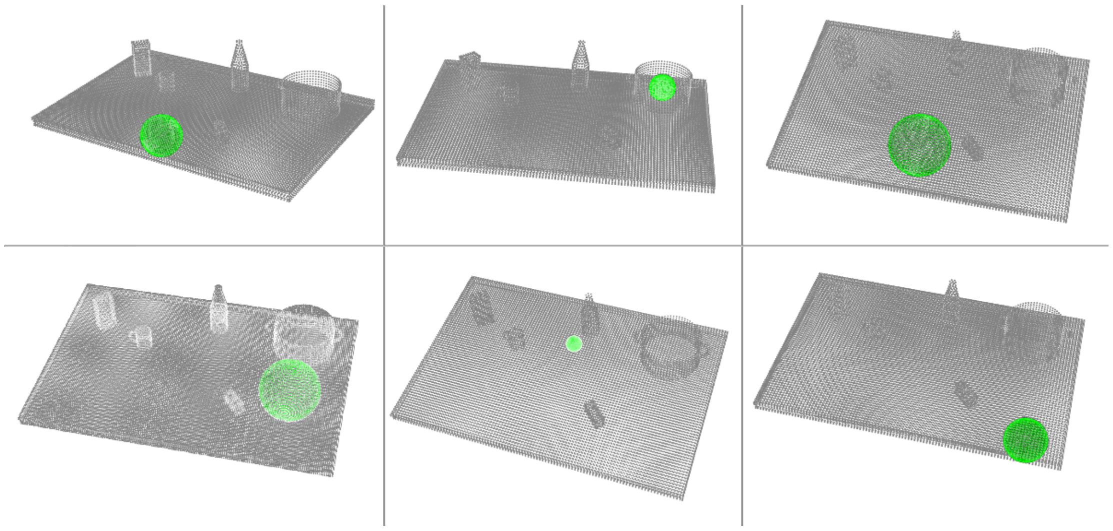


Figure 4.5: Robust detection of spheres using 3D HT.

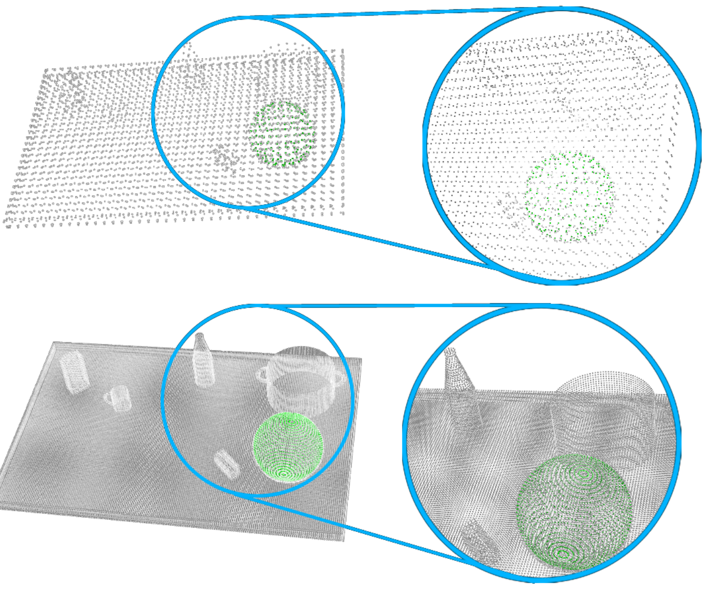


Figure 4.6: 3D HT performance on point clouds with different point densities. Low point density cloud (top) has 4.5k points, high point density point cloud (bottom) has 88k points.

Additionally, the 3D HT shows a robust performance with different point densities (resolutions) of the point cloud. The 3D HT gave the same robust performance across a range of point densities of the same scene; the lowest point density having 4.5k points and the highest point density with 88k points, Figure 4.6.

Point clouds generated by the Kinect have on average 300k points, each point cloud have been cropped to a suitable volume before applying the 3D HT for detecting spheres to discard redundant points, i.e. points that are far away from the interest area in the scene. Figure 4.7 shows the robust performance of the 3D HT applied on Kinect point clouds of different scenes including a football and an orange. The row on the left shows original textured point clouds, the middle row shows detected spheres superimposed on original clouds and the row on the right row shows the untextured point clouds with detected spheres. Figure 4.8 demonstrates the capability of the 3D HT to detect semi-spherical shape, the 3D HT shows ability to detect a bowl and an apple accurately. More results of the 3D HT applied on Kinect point clouds are in Appendix B.

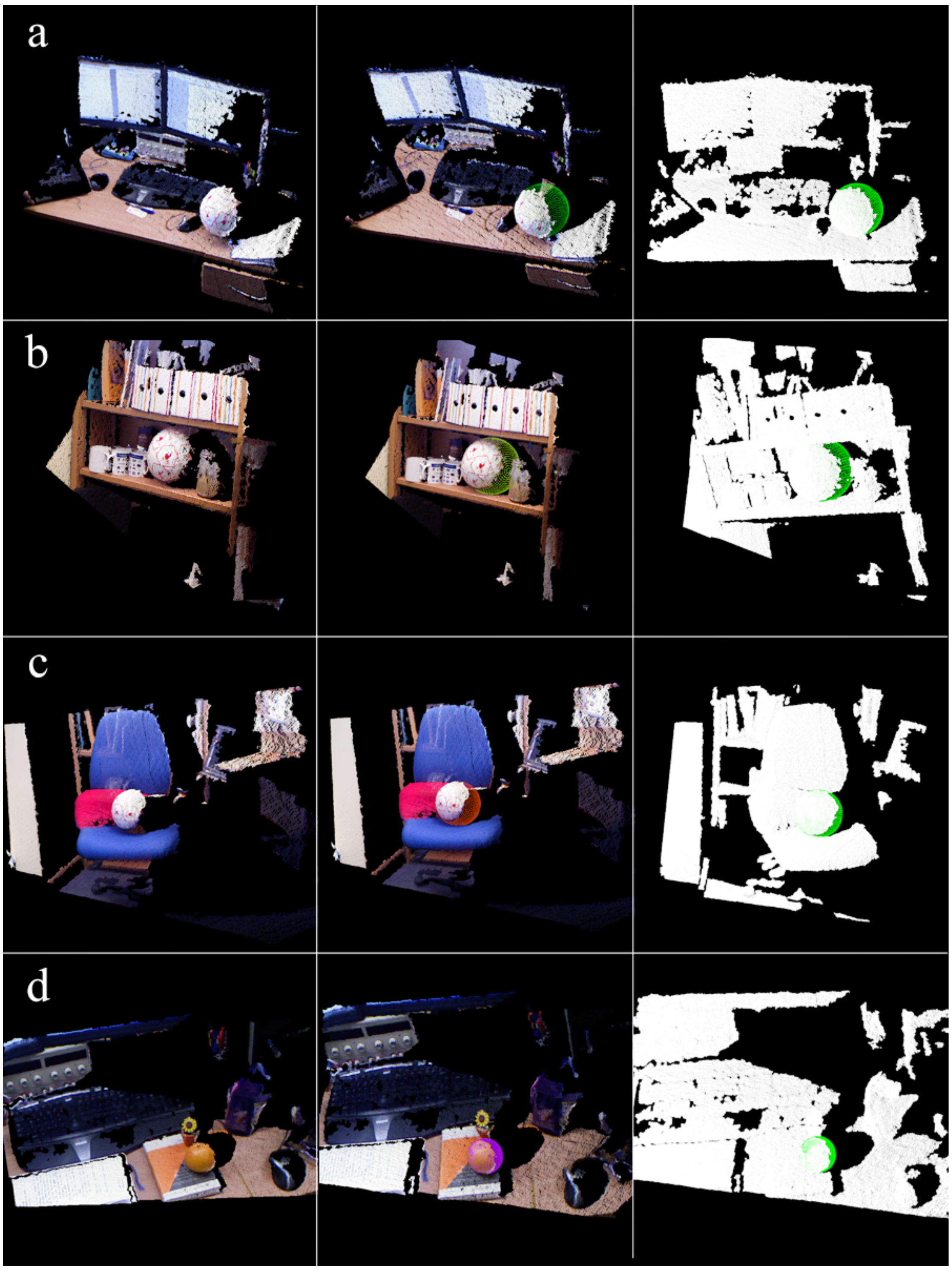


Figure 4.7: Sphere detection in real point clouds. (a) 146k points, (b) 193k points, (c) 174k points, (d) 120k points.

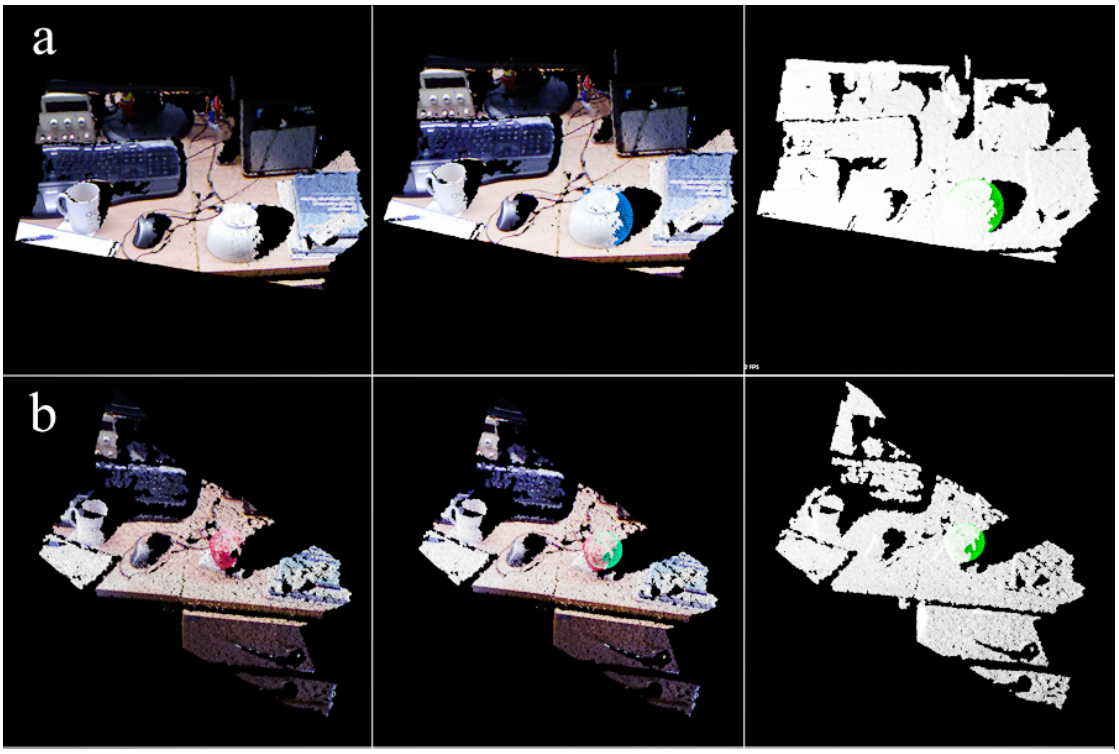


Figure 4.8: Detection of semi-spherical objects. (a) 136k points, (b) 64k points.

Point cloud	No. points	Time(s), i=1	Time(s), i=10	Time(s), i=20	No. Its.
Deskball	146k	5.30	0.76	0.44	94.6M
ShelfBall	193k	7.11	0.91	0.55	125M
Chairball	174k	6.31	0.89	0.52	112.8M
Bowl	136k	4.08	0.46	0.24	88M
Orange	120k	3.42	0.38	0.20	77.8M
Apple	64k	2.06	0.25	0.14	41.5M

Table 4.2: Total processing time of Algorithm 2 at point steps of 1,10 and 20 (fixed radius) and number of iterations. (running on 2.4GHz Intel Core i7 and 4GB RAM).

4.6 Evaluation and Analysis

Our experimental analysis of Kinect clouds shows that the *volume point density*, i.e. the number of points on a 3D surface, is ~ 25 points per cube centimetre (taken at distance of 1m). We are interested in coordinates values to the nearest centimetre, hence it is not necessary to include all points in voting process.

Taking every *ith* point instead of every point in the cloud, reduces the processing time significantly. However if the point step (i) is too large, false detection will occur. Based on our experimental analysis, a point step value of ($i = 20$) is a suitable value for typical Kinect clouds. Table 4.2 shows the processing time of Algorithm 2 for detecting spheres

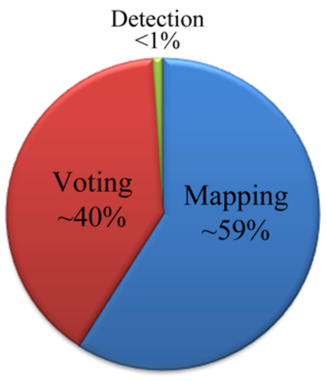


Figure 4.9: Approximate computational time percentages of the mapping, voting and detection processes of Algorithm 2.

with pre-defined radius, and the related parameters. For undefined radii, the processing time is simply multiplied by the number of the radii in the given range $[r_{min}, r_{max}]$.

The computation of Algorithm 2 is divided to three processes:

1. Mapping: This is the process of transforming of points from the point cloud 3D space to the 3D Hough (parameter) space, applied by Equation 4.2. This process is the most time consuming, it takes roughly 59.4% of the total processing time when Algorithm 2 is applied on a typical Kinect point cloud.
2. Voting: In this process, casting of votes occurs by accessing the elements of the sparse matrix and incrementing their values according to the Hough mapping. This process takes roughly 40.2% of total processing time.
3. Detection: A search algorithm is performed in this process to find the peak of the accumulator. This process is extremely fast and it is less than 0.4% of total processing time. This is due the implementation of a sparse matrix for the accumulator in which the search time is always $O(1)$.

The processing time shown in Table 4.2 for each cloud is the total computation times (sum of the three processes) for Algorithm 2.

Even though our developed algorithm is similar to [85] in terms of spherical equations used for voting and sparse matrix use, in our research we are the first to apply HT based method to detect spheres in point clouds in real life scenarios of typical household and office environments generated by consumer depth sensors such as the Kinect. Moreover we demonstrate the capability of detecting semi-spherical objects such as apples an bowls. Additionally, we provide performance analysis based on a number of different elements. Finally and most importantly, our algorithm detects spheres correctly in significantly less time. It can be seen from Table 4.2 that our algorithm is approximately 4 times faster than their achieved result, that is if *every* point has been taken in the

Bin size (cm^3)	No. bins	Time (s)	Matrix size (MB)
1	62380	1.765	0.12
5	1039	1.731	0.002
10	210	1.724	0.0004

Table 4.3: Processing time is irrelevant to number of accumulator bins (accumulator space resolution). Tested on a point cloud of 73k points.

voting process. However our algorithm can correctly detect spheres up to 57 times faster if the point step (i) is increased to a suitable value.

We are interested in providing a performance measure in terms of accuracy of detection and computational time; analysing the performance of the 3D HT for detecting spheres is not a simple task as there are a number of elements that affect the algorithm.

4.6.1 Accumulator Space Resolution (Bin Size)

The resolution of the accumulator space is the bin size of the cells in which the votes are cast. The larger the bin size is, the smaller the dimensionality of the accumulator, as multiple votes will be accumulated in the same bin.

Our experimental analysis shows that bin size is a trade-off between memory storage and accuracy, and it does not affect processing time. The reason that bin size does not affect processing time is due to the implementation of a sparse matrix for the accumulator, in which the search time is always $O(1)$, Table 4.3.

Rounding to the nearest bin size not only leads to reduced memory storage requirements for the accumulator, but also overcomes problems of noise and measurement errors of 3D sensors as points on the sphere's surface vote around the true sphere centre coordinates in the parameter space and not on the exact true coordinates. However it comes at the expense of reduced accuracy in detecting the location in 3D space. For example if the bin size was chosen to be 5 cm^3, then the estimated centre coordinates of the sphere will be rounded to the nearest 5 cm^3. And similarly for the the radius; if the bin size was 5 cm the estimated sphere radius will be to the nearest 5 cm.

In all our experiments on real and synthetic data we have used a step size of 1 cm for each of the coordinates of the sphere and for the radius, and hence accumulator (resolution) bin size of 1 cm^4.

Figures 4.10 and 4.11 show the radius and centre position estimation errors when the radius and the centre position vary relative to the accumulator resolution. In Figure 4.10, the accumulator resolution is fixed at 1 cm, and the 3D HT algorithm is applied on a series of synthetic spheres with 1 mm difference in their radii. The figure shows the estimated radius and the difference error between the estimated radius and the ground truth radius. It can be seen that until 0.7 *cm* the algorithm will round the radius to the lowest centimeter. Then the radius will be rounded to the highest centimeter. Note that the rounding is at 0.7 *cm* rather than 0.5 *cm*, this is related to the implementation of the Hough space and the compiler used, an accumulator smoothing algorithm can be used to reduce this quantisation error, as explained in Subsection 4.4.1.

Figure 4.11 shows the RMS error υ when the centre of the sphere is moved by 1 *mm* in the three dimensions (for example at 50.1 the centre coordinates are $(50.1, 50.1, 50.1)$). As mentioned earlier, since the accumulator resolution is fixed to 1 *cm*, the coordinates will be estimated to the nearest centimeter. It can be seen from the figure that the RMS error υ increases and then at 50.5 *cm* starts to decrease; this is because the first coordinates are being rounded to the nearest centimeter. For example centre coordinates (50.3,50.3,50.3) are rounded to (50,50,50), centre coordinates (50.6,50.6,50.6) are rounded to (51,51,50) and centre coordinates (50.8,50.8,50.8) are rounded to (51,51,51). While it is expected that the error plot will be symmetrical at 50.5 *cm*, the slight error is also related to the implementation of the Hough space and the compiler used (GNU GCC 4.3).

4.6.2 Occlusion Ratio

Correct detection of spheres depends on the number of points on the surface of the sphere which cast votes. These points should be enough to vote for the correct peak in the parameter space, if only few points are present on the sphere there will be no notable peak and hence the detection will fail.

For example, in Figure 4.4, the slice at $c_z = 30$ contains a much higher peak (600 votes) compared to other peaks which have around 30 votes. Moreover, if the scene is cluttered with objects; points on the surfaces of the objects can cast false votes in the accumulator space. Points on the sphere's surface can also cast false votes due to imperfections and noise, which can effect the saliency of the peak corresponding to the correct parameters.

The number of "true" sphere points relative to the total number of points that a "complete" sphere would have, can be defined as the *occlusion ratio*. For example, any sphere in a 2.5D point cloud will have an occlusion ratio of 50% as only half the sphere is present in the point cloud. However, since we are interested in real point clouds, we define the occlusion ratio by dividing the number of points on the occluded sphere by the number of points on the original non-occluded 2.5D sphere. Hence a full 2.5D sphere has an occlusion ratio of 0% and a sphere completely not visible in the scene has an occlusion ratio of 100%.

Figure 4.12 shows the Euclidean distance error for the coordinates of the spheres at increasing occlusion ratios. The RMS (root mean square) error metric υ is quantified by calculating the distance between the extracted centre parameters and those of the

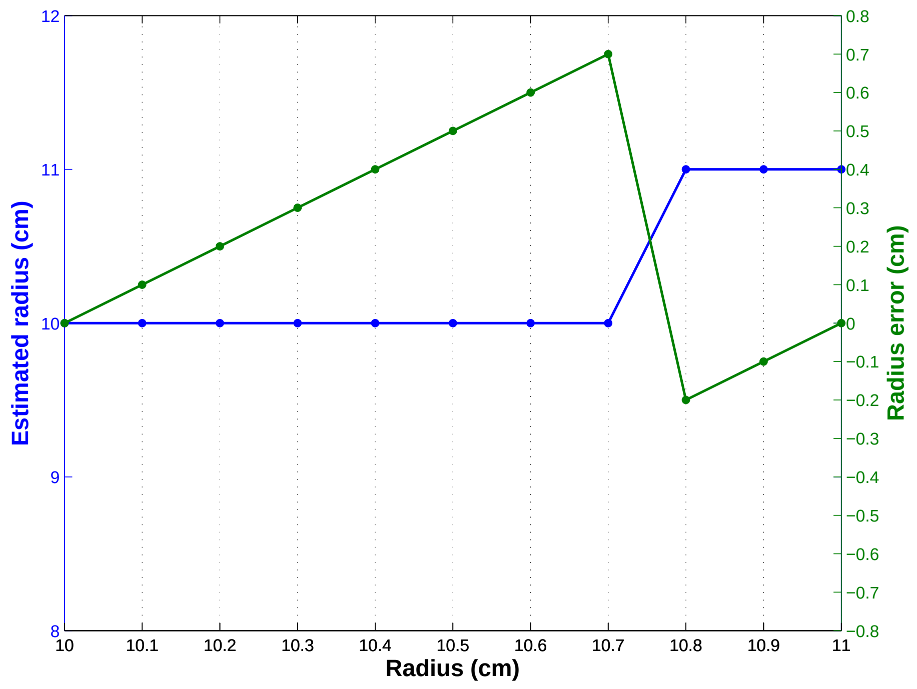


Figure 4.10: Radius estimation performance at 1 cm^4 accumulator resolution. The radius error is defined as (ground truth radius − estimated radius).

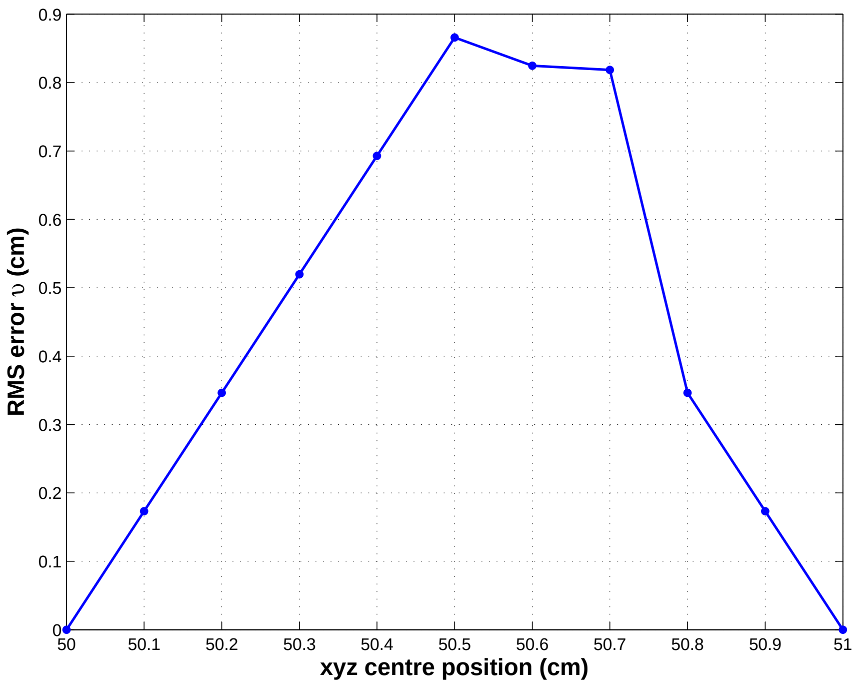


Figure 4.11: Centre position estimation performance at 1 cm^4 accumulator resolution.

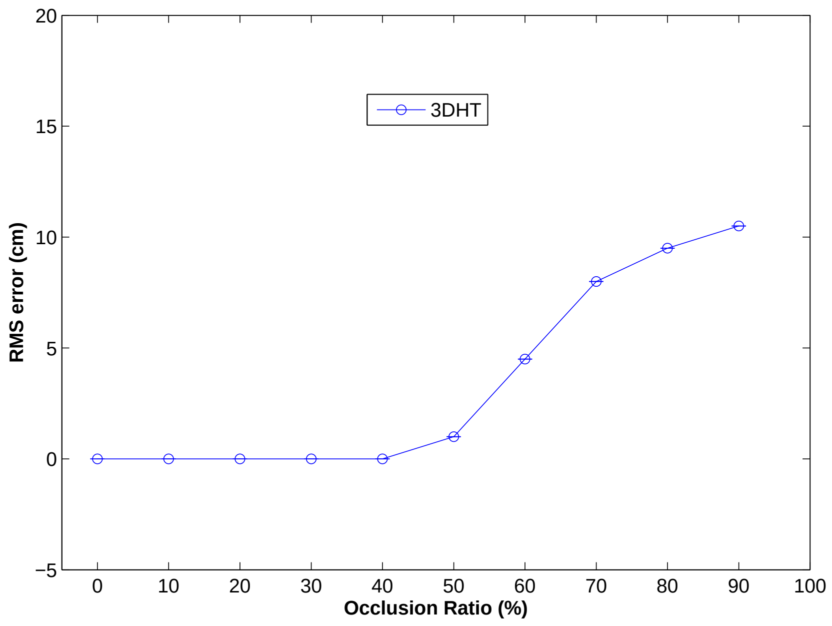


Figure 4.12: RMS error metric υ of the 3D HT at increasing occlusion ratio.

actual sphere (ground truth), measured by equation:

$$\upsilon = \sqrt{(x_G - x)^2 + (y_G - y)^2 + (z_G - z)^2} \tag{4.3}$$

where (x_G, y_G, z_G) are the ground truth coordinates of the centre. The occlusion performance analysis was carried out on four point clouds of a synthetic sphere of a predefined radius of 10 cm with no other objects in the cloud. These point clouds have different point densities and the error values are the average from the four clouds. An accumulator resolution of 1 cm is used for each of the coordinates of the sphere and for the radius, and hence of accumulator bin size of 1 cm^4.

It can be seen from Figure 4.12 that the algorithm starts breaking at a 50% occlusion ratio; this is because at this occlusion ratio the number of points on the surface of the sphere is not sufficiently high to cast enough votes, and hence the peak cannot be distinguished from its neighbouring cells in the accumulator.

4.6.3 Noise

[98] extends the original work of [99] and employs a statistical performance measure based on signal detection theory to determine the probability of the Hough transform correctly estimating the parameters of a circle in an image with the presence of noise. The kernel of this measure is determining the probability that the number of votes in accumulator cell corresponding to the feature's parameters is greater than the number of votes in any other accumulator cell, for a given parametric feature, parameter range,

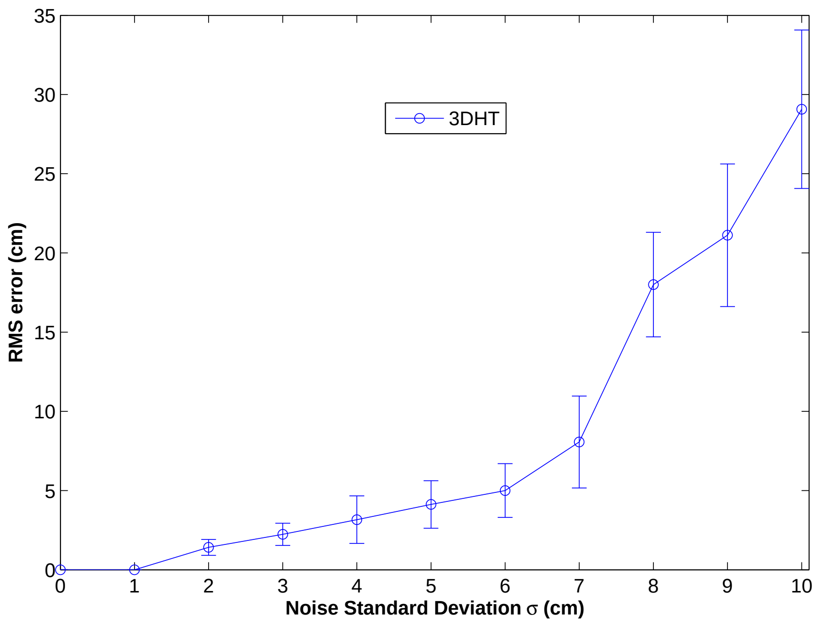


Figure 4.13: 3DHT performance against noise with increasing standard deviation and zero mean.

image size and variance of added Gaussian noise. The principles of this statistical measure remain constant irrespective of the parametric feature used or the dimensionality of the accumulator space. Hence the proposed theory can be directly extended to the 3D HT for detecting spheres. The reader is referred to [98] for a detailed study of the probability of detecting parametric shapes with the presence of noise.

In our research we quantify the performance of the 3D HT for sphere detection against noise by implementing the algorithm to detect a synthesised sphere. The noise performance analysis was carried out on four point clouds of a synthetic sphere of a predefined radius of 10 cm with no other objects in the cloud. A bin size 1 cm is used for each of the coordinates of the sphere and for the radius, and hence of accumulator (resolution) bin size of 1 cm^3. The root mean squared error is computed 10 times for the same sphere at each standard deviation step.

Figure 4.13 shows the performance of the Algorithm 2 applied on a synthetic sphere with the presence of noise. The noise is added by Gaussian distribution model given by Equation 3.15, explained in detail in Subsection 3.6.1. The effect of added noise is shown in Figure 4.14 and also in Appendix A.

It can been seen from Figure 4.13 that the RMS error v increases as the noise level increases. When the level of noise increases, the response of the voting process to noise will also increase until a point is reached where the response due to noise dominates the response due to the presence of the sphere within the point cloud. Once this point is

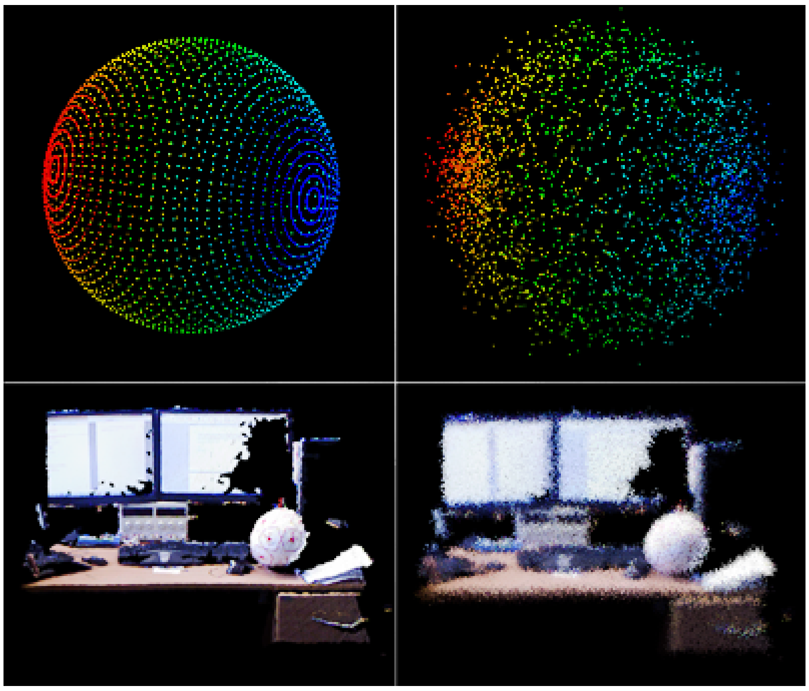


Figure 4.14: Effect of additive Gaussian noise on synthetic (top) and real (bottom) point clouds ($\sigma = 2cm$).

reached, the coordinates of the extracted accumulator peak essentially become random, and hence estimates of the sphere parameters cease to be of value.

This is because points on the sphere's surface to do not cast votes on the same bin due to noise, instead they will cast votes in bins around the bin corresponding to the correct sphere parameters, and hence the "correct" bin will not have a peak as high as it would be without noise. The larger the amount of noise, the further the votes will be from the correct bin in the accumulator, and hence the larger the error υ.

4.7 Conclusion

In this chapter we extended the Hough transform to three dimensions and showed its robust ability to detect spheres in synthetic and real point clouds. Our approach has been designed to be computationally efficient, reducing an established limitation of HT based approaches. To our knowledge, our research is the first Hough transform based implementation to detect spheres in real point clouds acquired by depth sensors such as the Kinect. We provided experimental analysis of the achieved results, and we showed robust performance for detecting spheres. This chapter established the foundation for our novel 3D velocity Hough transform, which detects spheres undergoing linear motion even with full occlusion, discussed in detail in the next chapter.

Chapter 5

The 3D Velocity Hough Transform

In the previous chapter we discribed the detection of static spheres in 3D point clouds. In this chapter we consider the global analysis of point cloud sequences and the extraction of spheres undergoing linear motion in full 3D. The new 3D Velocity Hough Transform (3DVHT), which can be considered as a 3D extension to the velocity Hough transform originally introduced by [98], parameterises shapes together with their motion, allowing a collective description of the motion between point clouds. More specifically, the algorithm incorporates motion parameters in addition to structural parameters in an evidence gathering process to accurately detect moving spheres at any given point cloud from the sequence. The thrust of this work is the *extraction* of moving spheres in a point cloud sequence rather than tracking them. Extraction can be distinguished from tracking by a more holistic view of the sequence, and frequent use of prior knowledge and an off-line approach [72].

The 3DVHT globally integrates information in all frames (point clouds), so information missed in one frame, yet present in another, still contributes to the final evidence gathered. We demonstrate its capability to predict the occurrence of spheres which are obscured within the sequence of point clouds, which conventional approaches cannot achieve. We apply our algorithm on real and synthetic data and demonstrate the ability of detecting moving spheres by exploiting inter-frame correlation within the 3D point cloud sequence.

The 3DVHT algorithm does not require initialisation or training and avoids the need to solve the correspondence problem, inheriting these characteristics from the VHT.

We begin this chapter by describing moving object detection and listing related work. After that the theory of the 3D velocity Hough transform is explained. Next, details of the implementation of the algorithm are described. Finally, we provide demonstration examples and experimental analysis.

5.1 Related Work

Whether in 2D or in 3D, moving object detection in sequences of 2D images or in 3D point clouds concerns locating moving objects, separating them from their background and describing their motion trajectories. According to [100], for 2D image sequences, most methods of moving object detection can be categorised in two groups: extraction methods and tracking methods. The strategy of extraction methods is to determine the moving object, by background removal, and then to track points in the moving object. The tracking strategy determines interest points, such as corners, and to then link the appearance of these points in successive frames.

For 3D point clouds, researchers have developed methods for moving object detection based on the same strategies; there have been methods for extracting 3D objects based on segmenting the object from its background and looking for inconsistencies between point clouds, such as in [101] and [102]. And tracking-by-detection methods, which imply that the object is detected by looking for objects matching a model on each frame of the sequence. Tracking-by-detection framework is limited to object categories for which pre-trained detector models are available, such as in [103] and [104].
Both strategies consider information present in a single point cloud and work on a frame-to-frame basis by performing spatial alignment between consecutive frames based on locating feature descriptors or other attributes. Moving object detection methods are usually accompanied with probabilistic mechanisms such as extended Kalman and particle filters or other Bayesian frameworks to establish temporal dependence between frames, and for optimization and expectation maximization, especially for real-time applications.

Very few methods have achieved object tracking in an *evidence gathering* context; in this research we are interested in this group of methods.

5.1.1 Moving Object Detection by Evidence Gathering

In 2D, one of the first methods to address the problem of tracking in an evidence gathering framework is the Velocity Hough Transform (VHT) [98]. The VHT was originally proposed in the context of extracting circles, moving with constant linear velocity, from a sequence of images. The VHT simply extends the standard HT to include linear motion in the voting process. By including motion parameters in the evidence gathering process, the HT is extended from the spatial to the temporal domain.

[105] proposed the dynamic velocity Hough transform (DVHT) which extends the VHT to track parametric shapes undergoing arbitary motion. Like the VHT, this technique processes the whole image sequence, gathering global evidence of motion and structure. The method tries to find an optimal, smooth trajectory in the parameter space with maximum energy. The constrained optimisation problem is solved using a temporal (time-delay) dynamic programming algorithm.

[72] proposed a continuous-template variant of the VHT (CVHT) for extracting moving arbitrary shapes, which has been created by fusing two evidence-gathering techniques, the VHT and Fourier-descriptor template representation. The Fourier descriptors provide a continuous template representation, minimising discretisation error in the algorithm, and the VHT component exploits the temporal correlation across a sequence, mitigating the effects of noise and occlusion.

[106] proposed a method for extracting moving articulated objects from a temporal sequence of images for the perpose of automated determination of parameters pertaining to human gait. This line feature extraction technique, uses a genetic algorithm (GA) based implementation of the Velocity Hough Transform (VHT).

[107] proposed a combined a 2D feature tracking method using the Kalman filter and the Hough transform. An extended Kalman filter is used to model the parameters and motion of a set of lines detected in a Hough space.

[108] proposed a derivation of the Hough transform for extracting pixel velocities from binary image sequences. The temporal spatio-velocity (TSV) transform represents the intensity at each pixel as a measure of the likelihood of occurrence of a pixel with instantaneous velocity in the current position. Binarisation of the TSV image extracts blobs based on the similarity of velocity and position.

[73] proposed the the 3D Stereo Velocity Hough Transform, an HT based algorithm that exploits both stereo geometry constraints and the invariance properties of the cross-ratio to accumulate evidence for a specified shape undergoing 3D linear motion (constant velocity or otherwise). The method extends ideas originally developed in the VHT to stereo images.

[109] proposed the Bounded Hough Transform to track objects in a sequence of sparse range images. This method is based upon a variation of the Generalised Hough Transform (GHT) [110] that exploits the coherence across image frames that results from the relationship between known bounds on the object's velocity and the sensor frame rate. The inter-frame motion bounds allows the transformation space to be reduced to a small size. In this method, tracking is related to pose determination, as long as the frame rate is fast enough with respect to the object's velocity, then the pose estimate can be propagated to each subsequent frame. This method is typically followed by the ICP algorithm to increase the accuracy of tracking. A limitation of this method is that

it requires the initial transformation of the object to be known a priori, which can be difficult in many applications. Another limitation is that this method is dependent on the frame rate of the sensor. This method works on a frame by frame basis, and hence it does not include any prediction capabilities for objects in missing frames.

5.2 Moving Object Detection Via The 3D Velocity Hough Transform

As discussed in the previous chapter, Hough transform methods conventionally concern the extraction of structural parameters of stationary features, within single frames. In order to detect spheres moving with linear velocity, we extend the VHT from optimum shape and motion parameters of conic sections moving in parametrically described fashion in a given 2D image sequence, to 3D point clouds, introducing the 3D Velocity Hough Transform (3DVHT).

Considering a sequence of point clouds enables any evidence gathering technique to combine feature extraction and motion analysis concurrently in order to extract optimal structural and motion parameters exhibited by objects in the *global* evidence gathering process. The inclusion of motion in the parametric representation of the feature allows info in one cloud of the sequence to be related to other info in other clouds of the same sequence. This employs motion as inter-frame mapping which gives the 3DVHT its global nature. By this global process, evidence from the whole sequence is gathered in a single accumulator space and optimal structural and motion parameters are extracted concurrently. This gives improved performance over frame-by-frame tracking methods especially in the case of missing or occluded structural information, as any missing structural data in one point cloud is compensated from other point clouds.

The other significant advantage of the 3DVHT, which is inherited from the VHT, is that the correspondence problem is avoided; there is no need for feature detection and correspondence analysis between consecutive point clouds, which many 3D tracking methods rely on, and is implicit in this new technique. Moreover, due to the global scope of the 3DVHT, there is no need to initialize the evidence gathering algorithm to a specific volume.

The 3DVHT detects spherical structures undergoing linear motion in point cloud sequences by combining concepts from the VHT and the 3D HT for sphere detection, in a generic evidence gathering process. The 3DVHT extends the spherical Hough mapping equations (4.2) by adding velocity terms representing constant linear motion:

$$\begin{aligned} c_x &= x - v_x t - r\cos\theta\sin\varphi \\ c_y &= y - v_y t - r\sin\theta\ sin\varphi \\ c_z &= z - v_z t - r\cos\varphi \end{aligned} \tag{5.1}$$

where v_x, v_y and v_z are the velocity of the sphere along the x, y and z axis respectively, measured by cm per time unit t, where t is the time reference value relative to the initial point cloud, i.e. for cloud 1, $t = 0$ and, cloud 2, $t = 1$ and so on. The evidence gathering process of the 3DVHT is similar to the 3D HT, however it employs an extended multi-dimensional accumulator space to estimate the motion parameters (v_x, v_y, v_z) in addition to the sphere's structural parameters $(r, c_x,\ c_y, c_z)$, described in Algorithm 3.

```
for (No. of point clouds) do
    for every point (x, y, z) do
        for ($r = r_{min};\ r \leq r_{max};\ r{+}{+}$) do
            for ($v_x = v_{xmin};\ v_x \leq v_{xmax};\ v_x{+}{+}$) do
                for ($v_y = v_{ymin};\ v_y \leq v_{ymax};\ v_y{+}{+}$) do
                    for ($v_z = v_{zmin};\ v_z \leq v_{zmax};\ v_z{+}{+}$) do
                        for ($\theta = 0;\ \theta \leq 2\pi;\ \theta{+}{+}$) do
                            for ($\varphi = 0;\ \varphi \leq \pi;\ \varphi{+}{+}$) do
                                $c_x = x - v_x t - r\cos\theta\sin\varphi$
                                $c_y = y - v_y t - r\sin\theta\sin\varphi$
                                $c_z = z - v_z t - r\cos\varphi$
                                Accumulator$[r][c_x][c_y][c_z][v_x][v_y][v_z]++$
                            end
                        end
                    end
                end
            end
        end
    end
end
Search Accumulator for peak.
```

Algorithm 3: The 3D Velocity Hough Transform (3DVHT).

For each point cloud in the sequence, at each 3D point in the cloud, equations (5.1) are employed to obtain the possible centre of the sphere in a given range of radii and, x, y and z velocities, provided the time reference t of the point cloud. Points corresponding to each parameter combination are incremented in the accumulator space. Note that all point clouds are accumulating in the same accumulator, the peak of this global accumulator is the best estimate of the sphere's structural parameters (centre and radius), and motion parameters (x, y and z velocities).

To detect the sphere at a particular point cloud from the sequence, we apply Equation 5.2.

$$\begin{aligned} c_x|_t &= c_x|_{t=0} + v_x t \\ c_y|_t &= c_y|_{t=0} + v_y t \\ c_z|_t &= c_z|_{t=0} + v_z t \end{aligned} \tag{5.2}$$

Similar to the 3DHT algorithm presented in the previous chapter, a basic peak detection method is used to estimate the moving sphere parameters. However additional processes can be applied to the accumulator to refine the parameter estimates, as explained in Subsection 4.4.1.

5.3 Implementation

Similar to the 3DHT presented in the previous chapter, we also use a multi-dimensional sparse matrix representation for the accumulator space of the 3DVHT. As mentioned earlier, the 3DVHT accumulates the evidence in one accumulator space for the whole sequence. This accumulator is seven-dimensional as it accumulates votes for the three motion parameters (v_x, v_y, v_z) in addition to the four spherical structure parameters of the 3DHT. Even with seven dimensions, sparse matrices store and provide fast access to the accumulator vote data very efficiently. Look-up tables for trigonometric functions values are also used by the 3DVHT for speed optimisation.

5.4 Demonstration

To analyse the performance of the 3DVHT, we apply Algorithm 3 on a number of point cloud sequences of synthetic and real scenes with different configurations and complexities, providing the properties of each sequence and the details of the 3DVHT application and also showing the achieved results.

In all following experiments an accumulator resolution (step size) of 1 cm is used for each of the coordinates of the sphere and for the radius, and a 5 cm/t step size for the three velocity parameters.

5.4.1 Basic Synthetic Sphere Sequence

We apply the 3DVHT algorithm on a sequence of a synthetic sphere moving in 3D. Figure 5.1 shows five consecutive point clouds (merged) of a sphere with specified radius $r = 10$ cm moving in three dimensions with constant $3D$ velocity, $v_x = 10$ cm/t, $v_y =$

10 cm/t and $v_z = 10\ cm/t$ starting at 3D centre location of $c_{x0} = 19\ cm$, $c_{y0} = 19\ cm$ and $c_{z0} = 19\ cm$. Each sphere is made of 1.5k points. The cm/t unit is the centimeter per unit time, in our case is centimeter per point cloud.

Algorithm 3 is applied on this sequence, mapping these points to a six-dimensional accumulator stored in a sparse matrix. The accumulator has 6 dimensions instead of 7 because we use a fixed size radius known a priori. A range of 125 values ($5 \times 5 \times 5$) for xyz velocities with step size of 5 cm/t is used for all velocities. To find the sphere's centres and velocities, the six-dimensional accumulator sparse matrix is searched for its maximum value. After applying our algorithm a peak is detected, and the sphere's initial position and velocity are estimated with high accuracy, with RMS error v less than 1 cm. The size of the 6D accumulator sparse matrix of the 3DVHT applied on this sequence is 9.5 MB. Table 5.1 summarises the details of the application of the 3DVHT on this sequence.

Clouds	No. points	Step	Velocities	Acc. size (MB)	No. iterations	Time(s)
5	7.5k	1	125	9.5	607.5M	46.57

Table 5.1: Basic synthetic sphere sequence. Columns left: Number of point clouds in the sequence, total number of points in the sequence, point step, number of velocities within the range, accumulator size, number of iterations and total processing time.

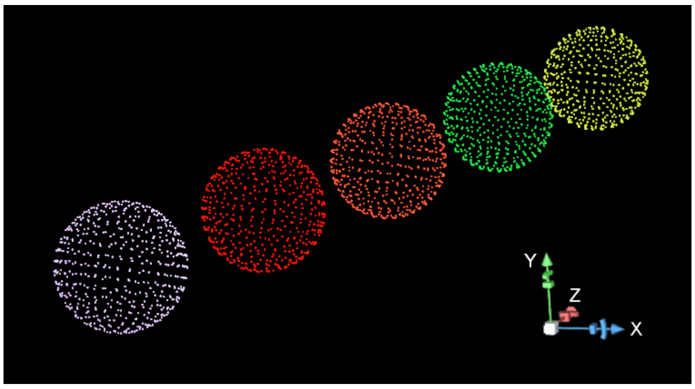


Figure 5.1: Synthetic sphere sequence moving with constant 3D velocity.

5.4.2 Synthetic Scene Sequence

Figure 5.1 represents a perfect scenario; only sphere points are present in the cloud sequence. To test the robustness of the 3DVHT, we test it on a more complex sequence. Figures 5.2 and 5.3 show a sequence of 10 point clouds of a synthetic scene including a moving sphere. In this sequence, there are 27.5k points forming the scene in each point cloud. A sphere of radius $r = 5\ cm$ and 3D velocity $(10, 5, 6)\ cm/t$ starting at centre location $(c_{x0}, c_{y0}, c_{z0}) = (-30, 10, 25)$. A range of 125 values $(5 \times 5 \times 5)$ for xyz velocities

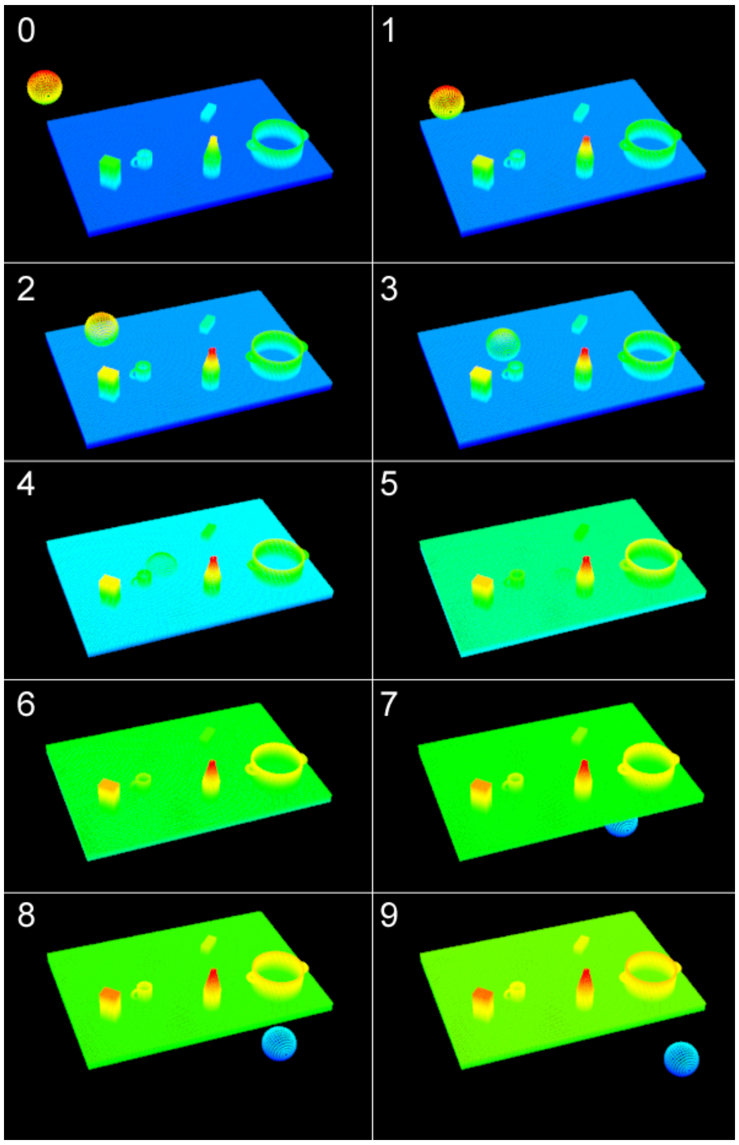


Figure 5.2: Synthetic scene sequence of a sphere moving with constant 3D velocity.

with step size of 5 cm/t is used for all velocities. A range of 10 values for the radius $[1, 10]$ cm with a step of 1 cm is used. Again, the 3DVHT gives a robust performance and finds the exact centre and velocity values of the moving sphere with RMS error v less than 1 cm. In the basic synthetic sphere sequence (5.4.1), voting was carried out for every point of every cloud in the sequence, for this sequence we take every $10th$ point to reduce the processing time. Figure 5.4 shows the computed trajectory of the sphere using 3DVHT.

Clouds	No. points	Step	Velocities	Acc. size (MB)	No. iterations	Time(s)
10	275k	10	125	24.3	2.2B	116.42

Table 5.2: Synthetic scene sequence.

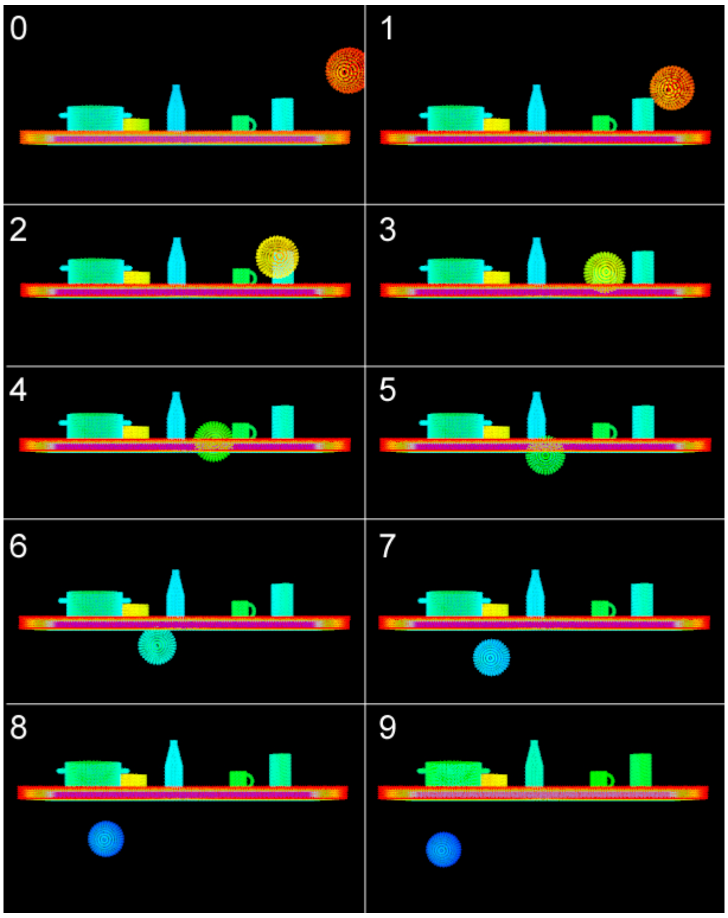


Figure 5.3: Side view of synthetic sphere sequence.

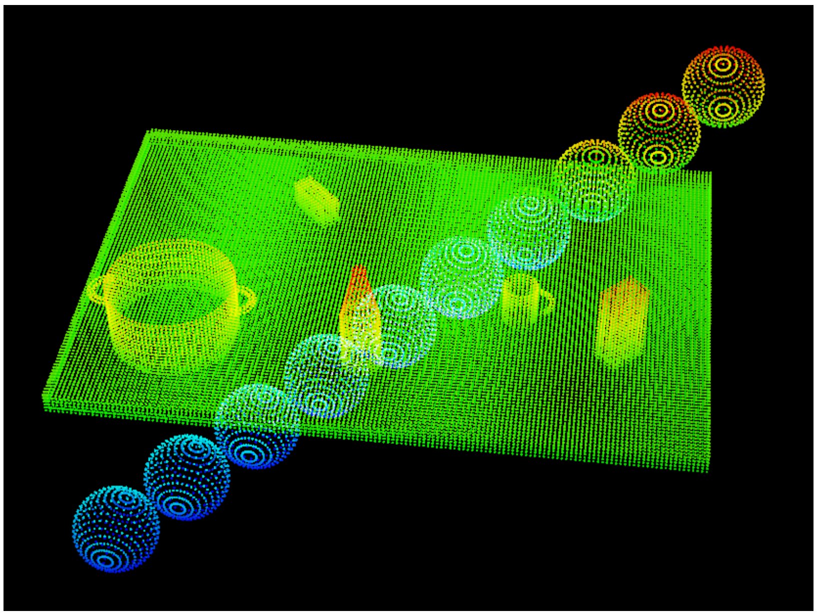


Figure 5.4: Computed trajectory of a sphere moving with 3D velocity.

5.4.3 Real Football Sequence

Moving on to real data, we will first apply the 3DVHT on a simple sequence of Kinect point clouds. Figure 5.5 shows a sequence of five Kinect point clouds of a football moving in only one direction. At each cloud, the football has been exactly placed with a 30 cm displacement on the x direction from the previous cloud, thus the football has a constant velocity of 30 cm per point cloud in the positive x direction, ($v_x = 30cm/t, v_y = v_z = 0$). Figure 5.6 shows the five point clouds merged giving the ground-truth trajectory of the football. Since there are no velocity components in the y and z directions, and the radius of the football is known a priori, the accumulator space is 4-dimensional. Each of the point clouds has 105k points on average. A range of 50 velocities in the x direction with a step size of 5 cm/t is used.

The 3DVHT was able to estimate the first football centre and the x velocity with RMS error υ less than 1 cm centre coordinates and velocity, Figure 5.7. This result was verified by detecting all spheres using the 3DHT for sphere detection on a frame-by-frame bases. We carried out more experimental analysis on this sequence by increasing and decreasing the velocity step; finer velocity steps, specifically of 1 cm/t and 2 cm/t, gave the correct centre but with an error of 1 cm/t for the velocity, i.e. the detected velocity was 29 cm/t.

Clouds	No. points	Step	Velocities	Acc. size (MB)	No. iterations	Time(s)
5	525k	10	50	12.4	1.7B	72.15

Table 5.3: Real football sequence.

5.4.4 Real Orange Sequence

We increase the complexity of the problem; the scene is cluttered with objects, the sphere to be detected is smaller and the motion is in two directions. This sequence is made of six Kinect clouds of an orange moving in the x and z directions. At each cloud, the orange has been exactly placed with a 5 cm displacement on the x and z directions from the previous cloud, thus the orange has a constant velocity of 5 cm per point cloud in the positive x direction, and 5 cm per point cloud in the positive z direction ($v_x = v_z = 5cm/t,\ v_y = 0$). Figure 5.8 shows the trajectory of the orange.

The accumulator matrix for this sequence is 5-dimensional, as the radius is known a priori and there is no velocity component in the y direction. A range of 100 velocities (10×10) the x and z directions with a step size of 5 cm/t is used. The 3DVHT was able to estimate the first orange centre and the x and y velocities with RMS error υ less than 1 cm for centre coordinates and velocity. Same as with sequence 5.4.3, the maximum velocity error was only 1 cm for x and y components ($\upsilon = 1.41$ cm).

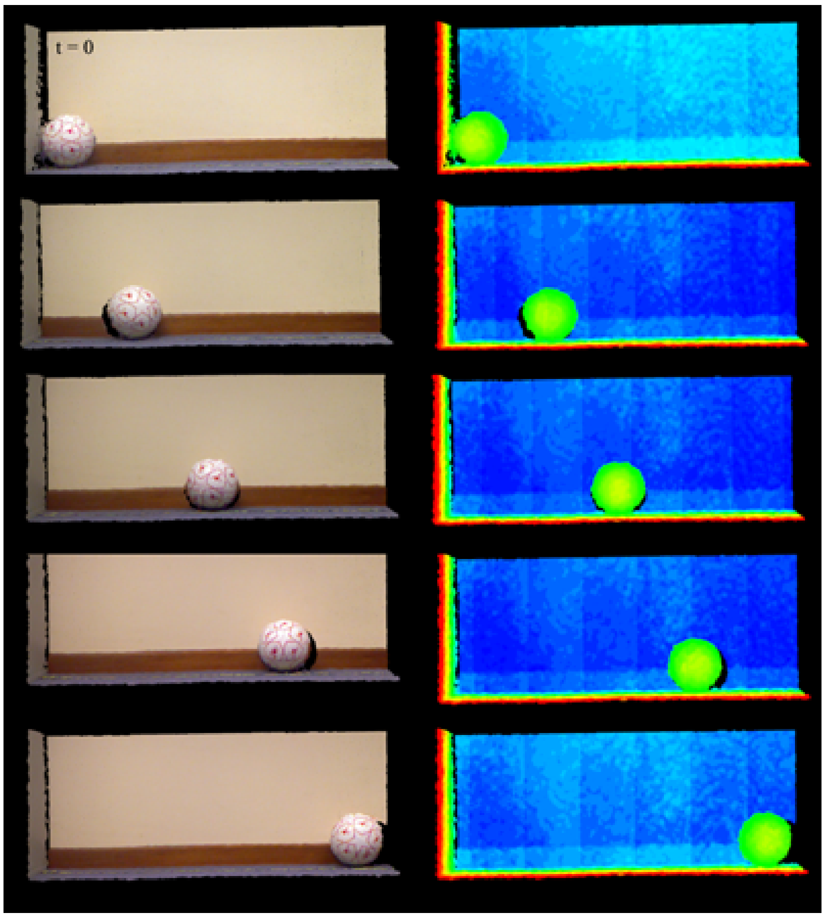


Figure 5.5: Sequence of real football with x velocity $= 30cm/t$. (textured left, depth map right).

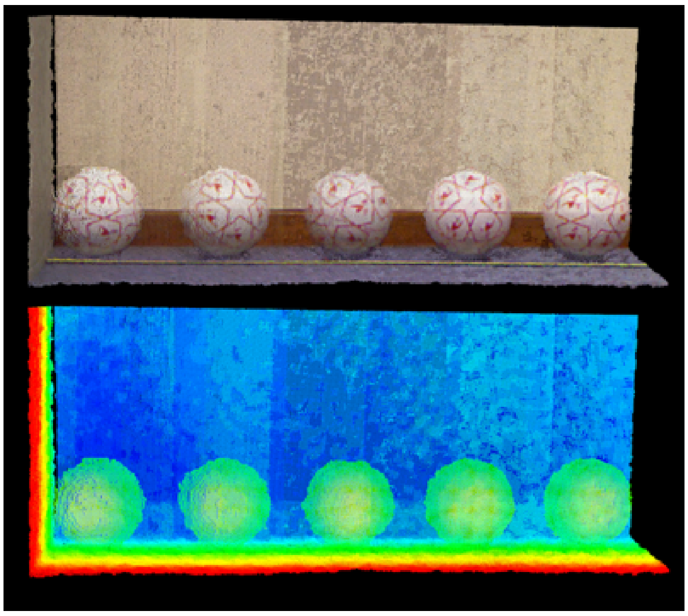


Figure 5.6: Merged sequence of football with x velocity $= 30cm/t$. (textured top, depth map bottom).

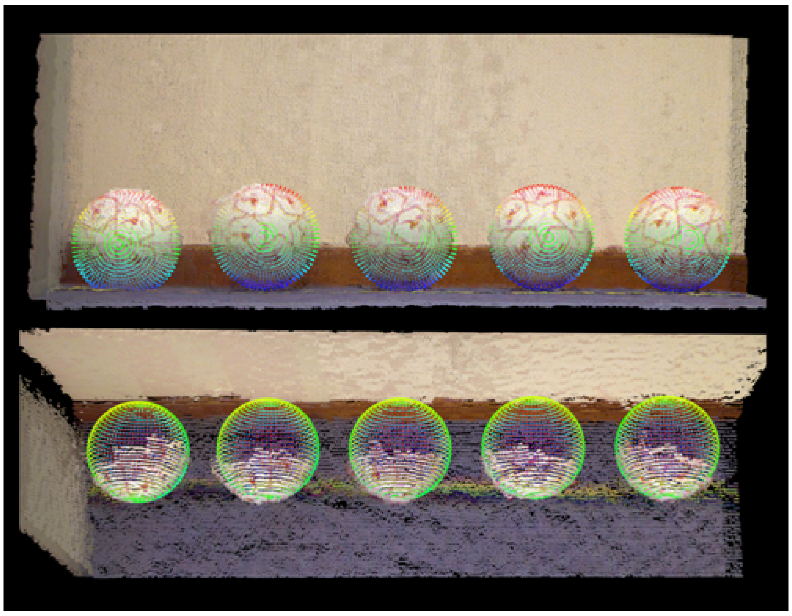


Figure 5.7: Two views of the estimated trajectory by the 3DVHT superimposed on the ground truth trajectory.

Clouds	No. points	Step	Velocities	Acc. size (MB)	No. iterations	Time(s)
6	177k	10	100	12.16	1.14B	50.57

Table 5.4: Real orange sequence.

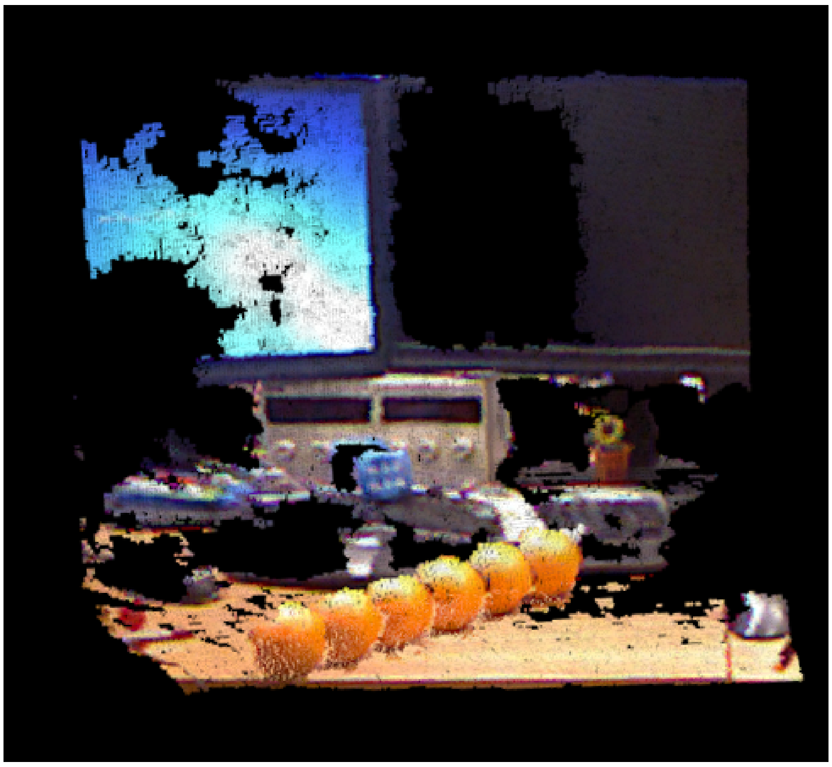


Figure 5.8: Trajectory of orange.

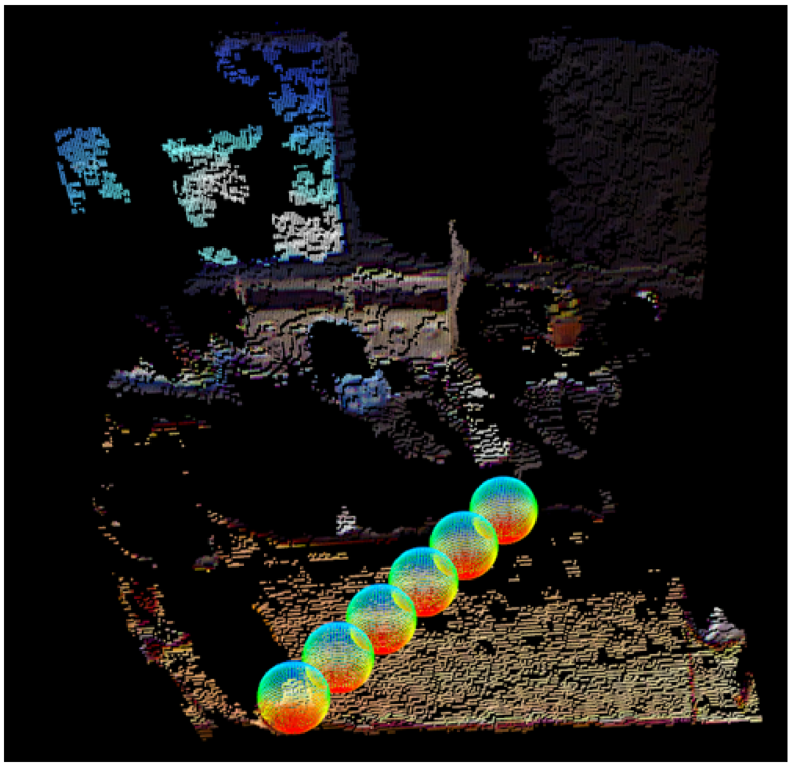


Figure 5.9: Detected trajectory of orange.

5.4.5 Rolling Football Sequence

In sequences 5.4.3 and 5.4.4, the sphere was placed at specified locations in consecutive point clouds to give a constant velocity. In this sequence we deal with a more realistic scenario. This sequence is made of eleven frames taken from a video of point clouds taken at 30 fps of a rolling football moving in x and z velocities, Figures 5.10 and 5.11 . The football in this sequence is moving in non-linear fashion in the x and z direction; i.e. the displacement between each consecutive point clouds is not the same. The football's y velocity is zero.

A range of 49 values (7×7) for x and z velocities, with step size of 1 cm/t, were used to estimate the best "average" velocities which best locate the football at every frame. The 3DVHT estimated an x velocity of 4 cm/t and a z velocity of -5 cm/t. The initial football position was estimated with xyz RMS error of 8.2 cm. Figure 5.12 shows the estimated football trajectory by the 3DVHT superimposed on the true trajectory of the football.

Clouds	No. points	Step	Velocities	Acc. size (MB)	No. iterations	Time(s)
11	1M	10	49	21.3	3.17B	564.5

Table 5.5: Rolling football sequence.

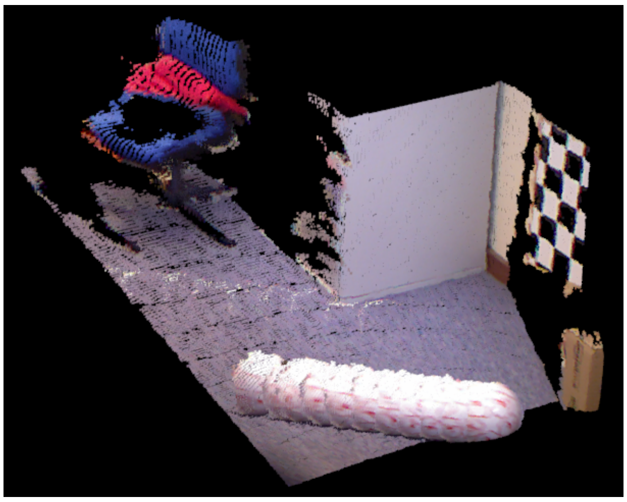


Figure 5.10: Rolling football sequence.

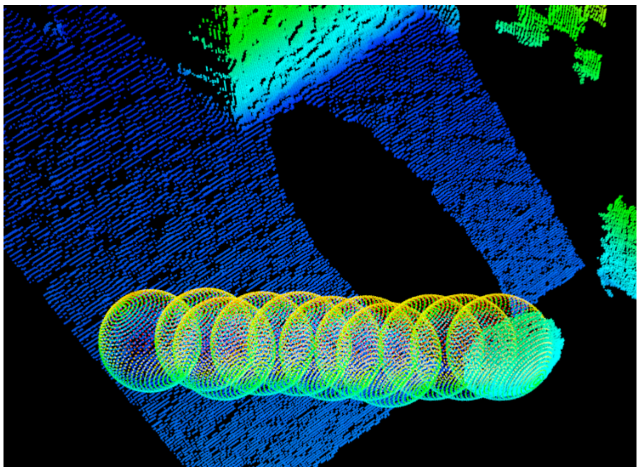


Figure 5.11: True trajectory of football.

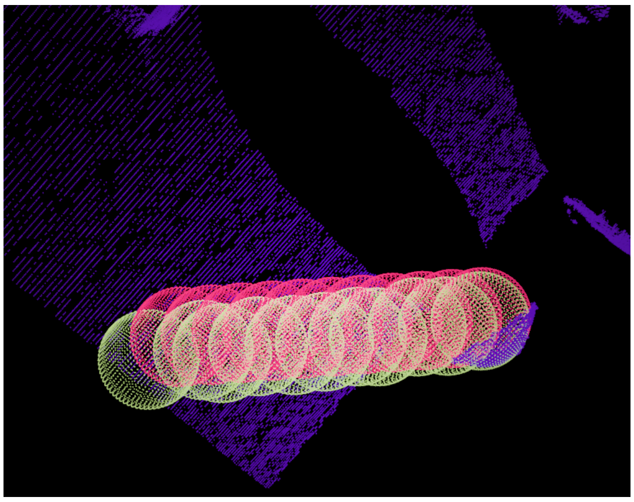


Figure 5.12: Estimated trajectory (pink) superimposed on true trajectory (green).

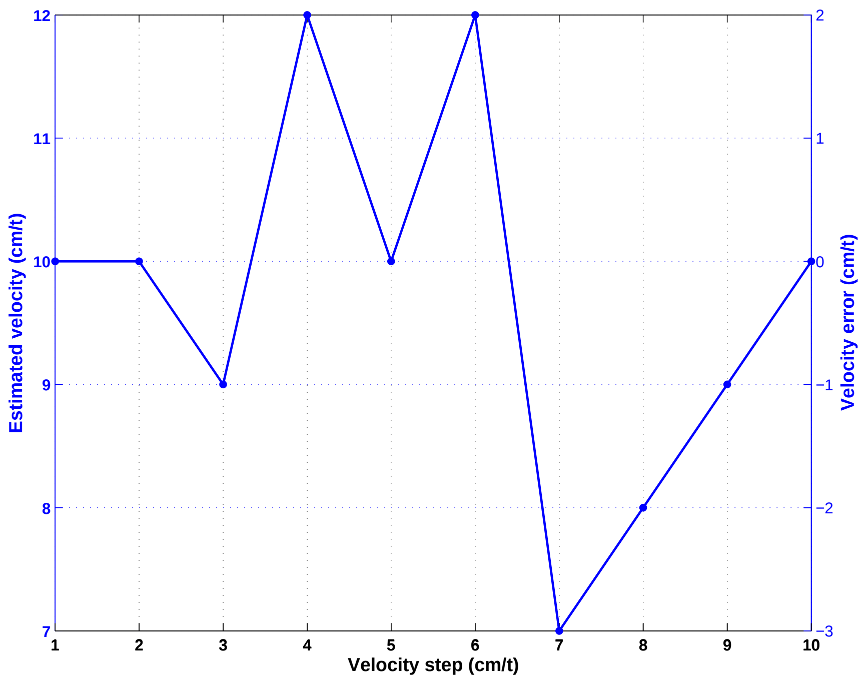


Figure 5.13: Velocity estimation performance at varying velocity step sizes. The velocity error is defined as (estimated velocity − ground truth velocity).

5.5 Analysis

5.5.1 Accumulator Space Resolution (Bin Size)

Similar to the algorithms presented in Chapter 3 and Chapter 4, the performance of the 3DVHT algorithm is affected by the choice of the step size of the parameters. The accuracy of the algorithm depends on the accumulator resolution, as the estimated parameters (the initial 3D centre position, radius and velocity of the sphere) will be rounded to the neatest bin size.

In demonstration experiments 5.4.1 - 5.4.5, we used an accumulator resolution (step size) of 1 cm is used for each of the coordinates of the sphere and for the radius, hence all initial 3D centre positions and radii are estimated to the nearest centimeter. The step size for the three velocity parameters was set to 5 cm/t, hence the estimated velocity will be rounded to the nearest 5 cm/t. In all these sequences, the value of the velocity step was chosen to be a divisor of the correct velocity, so the detected velocity will fall at a multiplication of the bin size.

To show the effect of the resolution of the step size, we vary the value of the steps and the range of the 3D velocity of sequence 5.4.1. Figure 5.13 shows the effect of the bin size on the performance of the algorithm in terms of estimating the moving sphere's

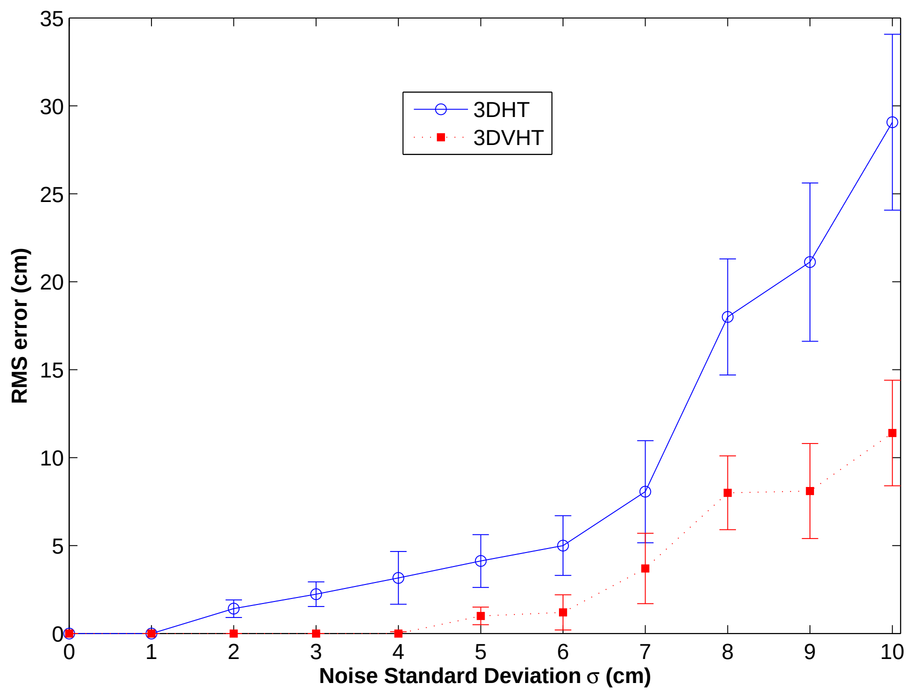


Figure 5.14: Noise performance of the 3D HT and the 3DVHT per frame.

velocity. The plot shown in the figure is the same for all velocities (v_x, v_y, v_z). Note that at step sizes 1, 2, 5 and 10, which are divisors of the correct sphere's velocity, the estimation error is at its minimum. For example if at velocity step 5 cm/t the algorithm will estimate the velocity of the sphere to be 10 cm/t which is the correct velocity and hence zero error. This is similar to the analysis of the angular resolution and rotational velocity in Chapter 3.

Similarly, changing the radius step size will effect the accuracy of the estimated radius, as the estimated value will be to the nearest step size.

5.5.2 Noise

Similar to the previous chapter, the performance against noise is measured by conducting an actual implementation of the 3DVHT algorithm on a synthetic moving sphere. The noise is added by Gaussian distribution model given by Equation 3.15, explained in detail in Subsection 3.6.1.

The noise performance analysis was carried out on a sequence of a synthetic sphere of a of 10 cm, moving in a constant 3D velocity. The sequence has 10 point clouds. A step size 1 cm is used for each of the coordinates of the sphere and for the radius, and a step size of of 1 cm/t for each of the velocities. Hence the accumulator resolution (bin size) of 1 cm[7]. The RMS error is computed 10 times for the same sphere at each standard deviation step.

Figure 5.14 shows a comparison of performance measure against noise between the 3DHT and the 3DVHT per frame; even though both techniques degrade gracefully as the amount of added noise increases. It is clear that the single frame based 3D HT produces inaccurate results for less noisy point clouds than the multi-frame 3DVHT. The 3DVHT is still capable of producing reasonably accurate estimates even when the 3DHT deemed to have failed completely.

The superior performance of the 3DVHT in terms of immunity to noise is due to the fact that the 3DVHT uses the point cloud sequence as a whole. Since the structural information present within each point cloud of the sequence are integrated over the sequence as a whole, even if structural information is corrupted in one frame of the sequence, a complete description of the feature (3D sphere) can be accumulated, as long as the corrupted information is present in another frame of the sequence.

As explained in Subsection 4.6.3, noise causes votes to be distributed in bins around the “correct” bin, which results in a non-salient peak. However, the global nature of the 3DVHT gathers evidence from the all the frames in the sequence and concentrates the votes in the accumulator bin with the correct parameters, and hence an improved performance over the single frame 3D HT.

5.5.3 Occlusion

So far we applied the 3DVHT on a number of synthetic and real sequences to detect the sphere at any point cloud of the given sequence. This could have been done also by applying the 3D HT for sphere detection, or any other sphere detection algorithm, on a frame by frame bases. The significance of the 3DVHT is its ability to predict fully occluded spheres, which is impossible if a frame by frame method is used.

To demonstrate the 3DVHT’s capability of predicting fully occluded spheres, we introduce occlusions to sequences we have previously applied the 3DVHT on. Afterwards, Equation 5.2 is applied to detect the sphere at the point cloud with the occlusion.

Figure 5.15 shows sequence 5.4.3, but with one sphere fully occluded by an object at the second cloud of the sequence. After applying the 3DVHT on the new sequence with occlusion, the 3DVHT estimated the the correct velocity of the sphere ($v_x = 30cm/t, v_y = v_z = 0$) and also estimated the centre coordinates of the first sphere. After that the coordinates of the occluded sphere are calculated, Figure 5.16.

Figure 5.17 shows sequence 5.4.5; an orange moving with x and z velocities, with an addition of occlusion at the third point cloud of the sequence.

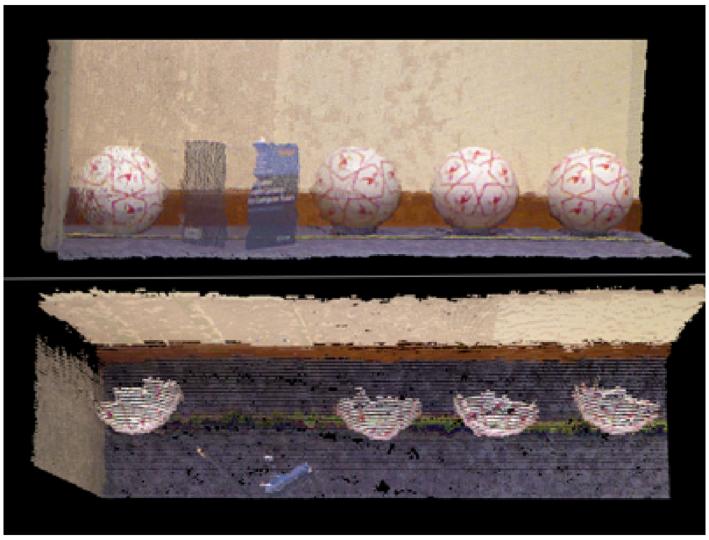


Figure 5.15: Football sequence with occlusion.

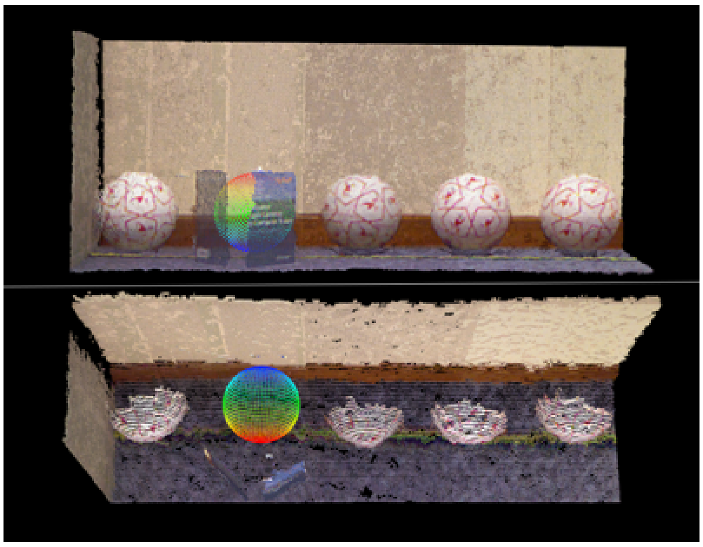


Figure 5.16: Detected occluded sphere.

The 3DVHT is used to analyse a sequence and the extracted sphere parameters, (the initial 3D centre position, radius and velocity of the sphere), are used to estimate the sphere's position at every point cloud of the sequence.

By assuming linear 3D motion, the 3DVHT estimates the trajectory of the moving sphere. This is a useful property of the 3DVHT especially in the presence of occlusion; as it will allow to the sphere to be detected in the case of partial occlusion, and "predicted" in the case of full occlusion. The occlusion experiments indicate the advantage of the 3DVHT which, by considering a whole sequence as a whole and integrating the information present in each individual frame, is able to accumulate a complete description of the sphere, in situations where there is an insufficient evidence in a single frame.

The 3DVHT is able to gather evidence from each frame of the sequence in a single

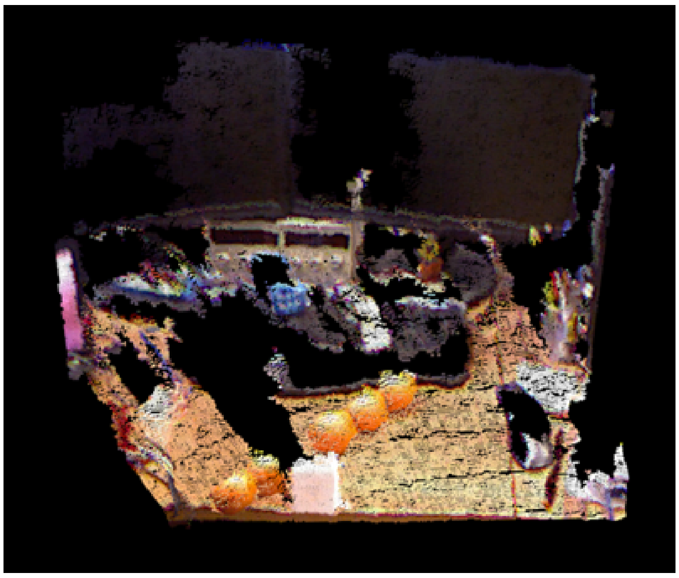


Figure 5.17: Orange sequence with occlusion.

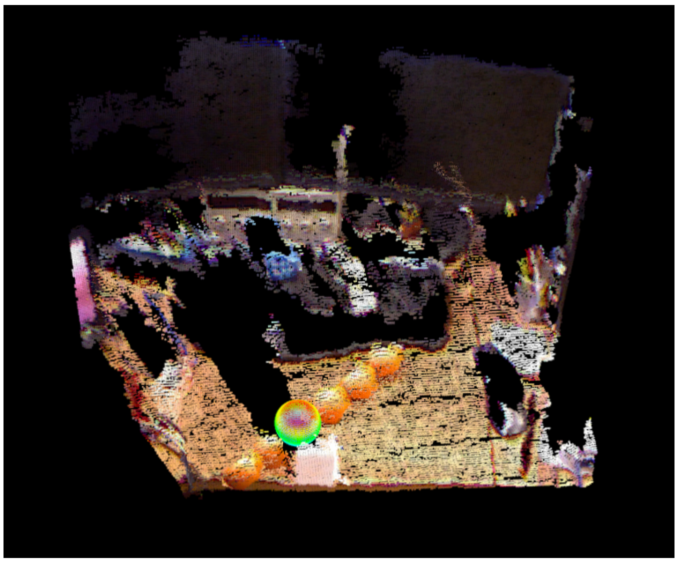


Figure 5.18: Detected occluded sphere.

accumulator space, which enables the concurrent determination of the sphere's structural and motion parameters, also results in the structural information present within each frame of the sequence being integrated over the sequence as a whole. Hence any missing information in one frame due to occlusion, is compensated from other frames. Even though points from the occluded point cloud do not cast votes in the "correct" global accumulator bin, points from other point clouds still cast votes in that bin to give a peak at the correct parameters.

From applying the 3DVHT on these sequences including occlusion, it is clear that the sphere at each sequence was predicted as if there was no occlusion. As explained before, the correct detection of the sphere at the occluded point cloud is due to exploiting the inter-frame correlation within the point clouds of the sequence. The 3DVHT offers

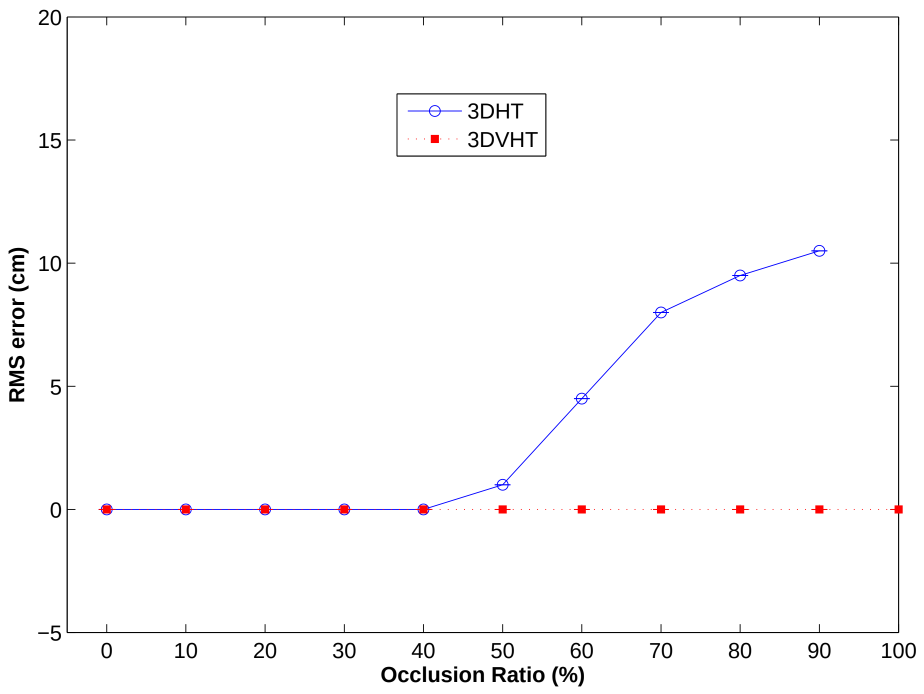


Figure 5.19: 3DVHT (single frame) performance at increasing occlusion ratio.

considerable performance benefits over single frame based methods such as the 3DHT in terms of resilience to occlusion due to the manner in which evidence is gathered from each frame of the sequence.

The performance evaluation in Figure 5.19 is measured if only one point cloud of the sequence includes occlusion. The occlusion ratio is defined in Subsection 4.6.2. It can be seen from the figure that the 3DVHT gives a robust performance even if a single frame is completely occluded. A step size 1 cm is used for each of the coordinates of the sphere and for the radius, and a step size of of 1 cm/t for each of the velocities.

In theory, only two spheres are needed for the 3DVHT to detect centre and velocity parameters; however this is in the case of a perfect scenario or a sphere moving in 3D, as there are no other objects in the point clouds of the sequence which may cast false votes and hence affect the detection process. Applying the 3DVHT on any two frames of sequence 5.4.1 always gives correct detection.

In more realistic scenarios, the performance of the 3DVHT with the presence of occlusion is highly dependent on the nature of the scene. For example, the 3DVHT can correctly detect the centre and velocity values of sequence 5.4.3 up to two full occlusions, if the sphere was occluded at three point clouds, the 3DVHT gives false detection. Figure 5.20 shows the accumulative performance for a real Kinect sequence of five frames (sequence 5.4.3).

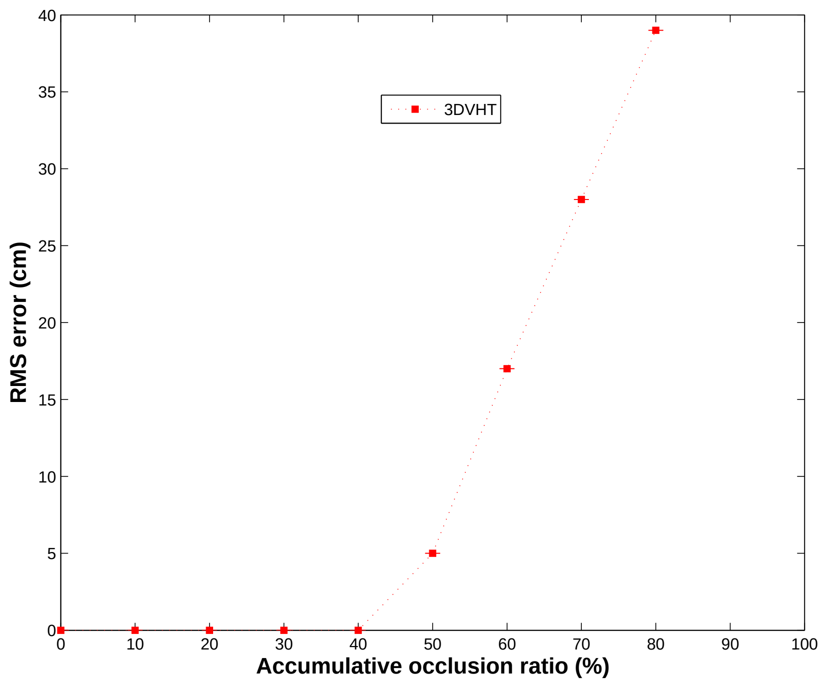


Figure 5.20: Accumulative occlusion ratio of a real sequence.

The accumulative occlusion ratio is defined for the whole sequence, i.e. in Figure 5.15 since the sequence has 5 point clouds and the occlusion is at one fully occluded sphere, the accumulative occlusion ratio is 20%. If one sphere is fully occluded at one frame and at another frame only half the sphere is occluded in another frame, the accumulative occlusion ratio is 30%.

5.6 Conclusion

In this chapter we presented a new global Hough Transform based method for detecting spheres undergoing linear motion in a sequence of point clouds, the three-dimensional Velocity Hough Transform (3DVHT). Analysis has been performed on synthetic and real point cloud sequences. The 3DVHT offers considerable performance benefits over single frame based methods such as the 3DHT in terms of resilience to occlusion and also immunity to noise. We demonstrated the significant advantage of the 3DVHT which is the ability to predict fully occluded spheres at any frame of the sequence. The superior performance is attributed to the concept of incorporating motion in the voting process, accumulating evidence in a single accumulator space for the whole sequence, resulting for the spherical structural information in each cloud to be integrated over the complete sequence. This enables motion and structural parameters to be accurately determined, even in the case of a full occlusion.

Chapter 6

A New Method For Motion Estimation

6.1 Introduction

Accurate motion parameter estimation is based on invariant properties that can be inferred from the motion [7]. In this research we investigate two kinematics theorems, namely the Reuleaux method [111] and Chasles' theorem [112] in the context of computer vision and develop evidence gathering methods employing rigid geometric constraints based on the analysis of corresponding points between 3D point clouds.

More specifically, we are interested in estimating the rigid transformation between 3D datasets (point clouds) without the use of keypoint detection and feature description and matching methods. We present a novel approach that estimates 3D motion parameters directly from the data by exploiting the geometry of rigid transformation using an evidence gathering technique in a Hough-voting-like approach. Our approach provides an alternative to the both local and global feature description and matching pipelines commonly used by numerous 3D object recognition and registration algorithms, discussed in detail in Chapter 2.

The developed algorithms in this chapter are based on the observation that:

There is one and only one invariant axis that synthesises the rigid body transformation in 3D.

When the transformation is rotation only, this invariant axis is the *rotation axis*; when the transformation includes rotation and translation the invariant axis is the *screw axis*.

In this chapter we introduce two evidence gathering algorithms for motion estimation; the first algorithm estimates the motion in the case of rotation around a fixed axis only, we introduce our algorithm that estimates the six degrees of freedom (DoF) parameters of rotation, which are the direction and 3D position of the rotation axis and the rotation angle. The second algorithm extend this algorithm to estimate general motion consisting of rotation as well as translation in 3D space.

The first method tracks the motion between two frames (point clouds) and estimates a number of candidates for the rotation axis using a modified and extended version of the Reuleaux method, and implements a voting algorithm to select the rotation axis with the most likely parameters (3D position and orientation). True correspondences are computed from the estimated axis and finally the rotation angle is estimated by triangulation between the found correspondences and the axis.

Similarly, the second method tracks the motion between two frames and estimates the screw axis using Chasles' theorem, and then computes the rotation angle and linear displacement along the screw axis to estimate the full motion parameters.

According to [113], kinematics theorems such as Chasles' theorem, have been widely used in kinematics and mechanics to analyse the nature and characteristics of rigid body motions. However, the implications of this theory for Computer Vision have not been properly considered.

It appears that this is the first research to use the kinematics theorems in an evidence gathering framework for motion estimation and surface matching without the use of any given correspondences and also without employing feature description and matching.

The capability to estimate the 3D transformation without relying on feature matching and estimating correspondences between 3D datasets is desirable and has many practical applications, especially in scenarios where objects to be matched have smooth surfaces such that keypoints and their associated features are not descriptive, or in the case of having few points on the object's surfaces (sparse point clouds).

We begin this chapter by discussing similar related work, in terms of motion estimation using geometrical analysis, we also discuss methids that use a Hough voting scheme to vote for motion parameters. The next section describes the proposed method for rotation estimation. The third section describes the second proposed method for general motion estimation. After that we give demonstration examples to validate and verify the theory of the two algorithms. Then we provide experimental analysis and evaluation and show the results of implementing the algorithms on real data. Finally, we discuss the properties, contributions and limitations of the proposed algorithms.

6.2 Related Work

6.2.1 Motion Estimation Using Geometrical Analysis

As described in Chapter 2, the vast majority of 3D surface matching methods rely on computing feature descriptors and matching them in order to find correspondences, then use a motion estimation method to compute the transformation parameters from these established correspondences. According to [7], while a large number of these motion methods, such as those based on SVD, orthonormal matrices and quaternions, have been applied with considerable success, a common shortcoming has been lack of efficiency, sensitivity to noise and multiplicity of solutions.

To overcome such shortcomings, a number of researchers have presented geometric constraint frameworks based on invariant properties of 3D rigid transformations to analyse transformation parameters. A key element of the geometric analysis is formulating algebraic constraints between a rotated and translated model and the observed features [114]. The constraints are aimed to define or refine the set of allowable positions for the object, consistent with the observed features.

In the section we list these frameworks in two categories; methods that have focused on estimation of rotation parameters, and methods that estimate the 3D general transformation.

6.2.1.1 Rotation Estimation by the Reuleaux Method

A variety of methods has been developed in different research areas, especially the fields of kinematics and biomechanics, to estimate rotation parameters of moving objects. Many of these methods are based on the method of Reuleaux [111].

The Reuleaux method is a classic 2D graphical method of planar kinematics, which utilizes the perpendicular bisectors of the displacement vectors connecting a set of two corresponding points to find the centre of rotation and calculate the angular change. The method is based on a simple idea; two points on the moving object are tracked and two perpendiculars are constructed on the midpoints. The point of intersection of the perpendiculars represents the location of the centre of rotation at this time, also called the instant centre of rotation, Figure 6.1. The centre of rotation is the point with zero velocity about which a body rotate. It can also be considered the limiting case of the pole of a planar displacement.

In biomechanics, estimating the instant centre of rotation is fundamental for the analysis of clinical movement of body joints and muscle forces, which has been investigated by many researchers using Reuleaux's method, [115, 116, 117, 118, 119, 120, 121].

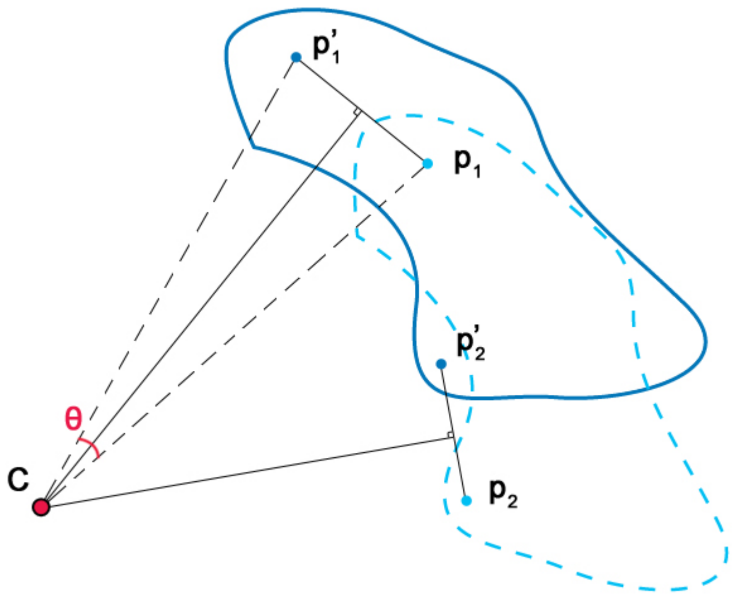


Figure 6.1: 2D rotational motion. The centre of roatation (C) is computed from the intersection of perpendicular bisectors of the displacement vectors. θ is the angle of rotation.

These methods mainly investigate the inaccuracies in finding the centre of rotation and introduce methods based on minimisation of cost functions to estimate the most correct centre of rotation. The limitation of these methods is their requirement of the use of markers and pre-computed point correspondences, which is a significant impracticality in many scenarios.

More recent research efforts have extended the Reuleaux method to three dimensions and estimate the the axis of rotation of 3D rigid objects; [122] presented a computational geometrical method for kinematic registration in 3 dimensions when minimal, over determined, infinitesimal and perturbed sets of corresponding points are given. [9] presented a line geometry based method to estimate screw parameters of helical motion of a rigid body between two positions, given the coordinates of two corresponding lines on the rigid body. Both of these methods consider over-determined systems and provide a linear solution based on least squares minimisation. Similar to the 2D, these methods also require known pre-computed correspondences in order to obtain motion parameters.

Other methods have been developed for estimating the axis of rotation in order to reconstruct surfaces from partial segments; in [123], a multi-step algorithm is employed for the estimation of the axis of rotation of radially symmetric archaeological pottery objects from their fragments. The algorithm uses direct least squares optimization for initial estimation followed by a robust estimation based on M-estimators with iterative refinement. [124] finds the axis of rotation by minimizing the average derivative of curvatures in order to reconstruct rotating surfaces. These methods also have the limitation of being restricted to symmetrical objects, such as pottery; which is also unusual in practical perception scenarios.

A detailed overview, within the context of computer vision, of the Reuleaux method in three dimensions is explained in detail in Section 6.3.

6.2.1.2 General Motion Estimation Using Geometric Properties

In [9, 125], the authors presented a method for kinematic registration that is based on specifications of line features. In this method, kinematic registration involves computation of the screw parameters of a motion from specified positions of geometric features of the moving body. The problem is formulated using a special complex of lines associated with kinematics namely the bisecting linear line complex.

The methods presented in [8, 7, 113] are the first approaches to extend Chasles' screw motion concept to the estimation of motion parameters in computer vision. The authors formalise the relevant geometric properties of corresponding vectors by providing methods to construct the rotation point or axis and estimate a number of solutions for the point or axis. These methods analyse the geometric properties of vectors connecting corresponding feature points and their angular information, synthesised into a single coordinate frame, and define relevant constraints from the point of view of geometric invariants. These methods are superior to methods based on perspective and epipolar geometries because they are based on explicit expressions of distance between feature points and angle measurements rendering them appropriate to calibrate transformation parameters in vision applications.

Several geometric constraints have been identified [114] for estimating surface positions, given information about the shapes of planar surface patches and the relationships between the model and data patches. These constraints include position constraints, defined by algebraic relationships between vectors and data-to-model patch constraints which are based on aligning normals and on proximity and distance between points on the surface.

A method was introduced by [126] for motion estimation based on the analysis of rigid motion equations as expressed in geometric algebra framework. This method finds the correspondences of two 3D points sets by finding a certain 3D plane in a different space that satisfies certain geometric constraints using tensor voting. This method estimates the rigid transformation by producing a set of putative correspondences and uses them to populate joint spaces. If any rigid transformation exists in the set of correspondences, then four planes must appear in these spaces. If four significant planes in these joint spaces are detected, then the points which lie in these planes are related by a rigid transformation. These planes are related by a geometric constraint which limits the search to a single plane which satisfies this constraint.

Note that this method finds the correspondences and estimates the rigid motion; unlike previously discussed methods which estimated the motion from established correspondences, which gives it a significant advantage. However, the tensor voting procedure used in this method is considered as a methodology for feature description, as it consist of two elements: tensor calculus for data representation and tensor voting for data communication. Each input site propagates its information in a neighbourhood and is used to detect lines, curves, points of junction and surfaces. Tensor voting gives this method a computational cost, which is quadratic in the number of points in one scan, and therefore it is not practical. Another impracticality of this method is that one of the sets of points must be rotated in order to densify the plane being looked for in the voting space.

6.2.2 3D Motion Estimation by Hough Voting

There has been a number of 3D motion estimation methods for 3D object recognition and registration that are based on voting schemes. More specifically, these methods employ a voting process to obtain an accurate transformation hypothesis from the matched features. According to [127], these methods estimate the 6 DoF pose between 3D data sets by first generating an empirical distribution of pose through the collation of a set of possible poses, or votes, which are often computed by matching local features, from a test object to those in a library with known pose. And then finding one or more best poses in the distribution which is the maximum (or peak) in a parameter space (also called Hough space).

[128] was one of the earliest approaches to use Hough voting for object recognition; with a Hough voting method in two separate 3D spaces accounting for rotation and translation allowing the detection of objects based on correspondences between vertex points established by matching straight edges.

Another early approach was presented in [129], where a model-based approach for determining the orientation and position of 3D objects in range images. Edges are used as features in the process. Each edge in the model is matched with every edge in the scene to compute the rotation parameters with the help of an arbitrarily preselected reference vector. Translation parameters are determined in a 3D Hough space computed based on every possible rotation obtained from the matchings.

[130] use the 3D generalised Hough transform to detect arbitrary 3D objects, this method quantises the 7D space of 3D translation, rotation and scale. This method stores a 3D model in a 2D R-table and then detects instances of the model in a point cloud by voting. This method is computationally expensive and slow compared to other methods.

[39] Estimates transformation between 3 dimensional surfaces using pairwise geometric histograms and finds correspondences by finding similarities between these histograms.

The global surface correspondence between two 3D surfaces is found by finding the consistent local hypothesis using a probabilistic Hough transform. Correspondences are grouped in pairs and in triplets in order to vote, respectively, into two distinct 3 dimensional Hough spaces, one meant to parametrize rotation and the other to account for translation. This method first estimates the rotation parameters by finding the peaks in the 3D Hough space and then the translation parameters are estimated if a significant peak is found in this space.

[34] introduced 3D SURF features in combination with the probabilistic Hough voting framework for the purpose of 3D shape class recognition. In this method features are detected, described and quantized into the visual vocabulary. Using a previously trained 3D Implicit Shape Model, each visual word then generates a set of votes for the position of the class centre and the relative shape size (scale). Finally, the recognized class is found at the location with maximum density of these votes.

[131] extends the original Hough voting based detection model by introducing a joint Hough space of object location and visibility pattern for object detection and occlusion reasoning. This method use HOG features [132] and incorporates the object 3D position and its visibility pattern and accumulates votes in a 4D space.

[49] deploy Hough voting for hypothesis verification in 3D object recognition. This method selects keypoints, randomly or by means of suitable feature detector such as the SHOT feature descriptor [42], then computes a feature descriptor at each point and matches corresponding features by finding Nearest-Neighbour of each feature by applying an Euclidean distance metric. Corresponding scene feature cast votes in a 3D Hough space to accumulate evidence for possible centroid positions.

[133] introduce extensions to the Hough transform voting process, to reduce memory and computational requirements in applications, and also to improve precision of the Hough transform by utilising an iterative procedure to “demist” the voting space by removing the improbable votes from the voting space, based on the assumption that only one vote is generated by each feature.

To summarise, Hough transform based methods for object recognition largely follow the same pipeline: features are detected and descriptors for the features are computed. The votes are then computed by matching features in the test data with features from training data with ground truth scale and pose. These methods mainly differ in the type of features they employ, and also the metrics used to match the features. For example [39] use Bhattacharyya metric [134] to find the similarity between pairwise geometric histograms, while [49] use the SHOT features with Euclidean Distance metric.

In the next step, pose hypotheses are generated from each matched feature pair, and votes are accumulated in a high dimensional Hough space. The dimensionality of the Hough space depends on the parameters being voted for; for example [34] votes for

Method	Parameters	Hough Space	Feature
Tsui & Chan [128]	R, t	$2 \times$ 3D	Line segments
Hu [129]	t	3D	Edges
Khoshelham [130]	R, t, s	7D	Surface normals
Ashbrook et al.[39]	R, t	$2 \times$ 3D	PGH
Knopp et al.[34]	R, s	4D	3D SURF
Wang et al.[131]	t	4D	HOG
Tombari & Stefano [49]	t	3D	SHOT
R Method	R	1D	-
C Method	R, t	1D	-

Table 6.1: Comparison of Hough-based pose estimation methods. Second column shows pose parameters each method votes for (R: rotation, t: 3D position, s: scale), third column shows the dimensionality of the voting (Hough) space and fourth column shows the feature used for matching. R and C Methods represents the proposed algorithms for rotation estimation and general motion estimation respectively.

the 3D position parameters and scale; hence a 4D Hough space, while [130] votes for position, rotation and scale and hence a 7D Hough space. Finally a search for the highest votes (peaks) is implemented, and the peak parameters are considered to be the transformation hypothesis. Table 6.1 shows details of Hough based methods.

The two developed methods presented in this chapter have the advantage of having a one-dimensional space, which is basically a counter. This significantly reduces the complexity of allocating and searching the high dimensional accumulator spaces that other methods suffer from, Table 6.1. This is achieved by representing the transformation as one axis (rotation or screw), and then voting for similarity between axes in terms of position and direction. More details in Section 6.3.

6.3 Rotation Estimation by Evidence Gathering

As discussed in the previous section, most applications of Reuleaux's method were applied within the fields of kinematics and biomechanics; the method had little deployment in computer vision. In this section we introduce the application of Reuleaux's method to solve the problem of estimating rotation parameters between two 3D datasets without any given point correspondences.

In many applications, there is an interest in estimating 3D rotation parameters; in robotics for example, the rotation of a payload relative to an arm of the robot. When the relative motion of an object and sensor is a rotation around a fixed axis there is potential for more accurate and efficient estimates of motion than from a general 3D surface matching system. For example, for automated high quality scanning of objects it is common to constrain motion of the object or sensor to a fixed axis of rotation.

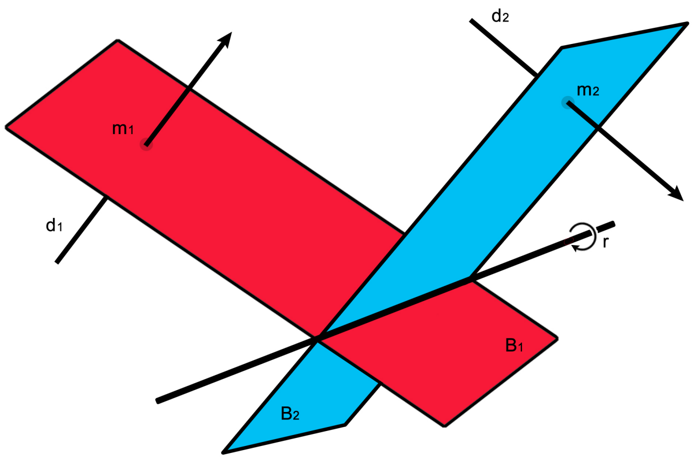


Figure 6.2: Two displacement vectors $\mathbf{d_1}$ and $\mathbf{d_2}$ determine the axis of rotation $\mathbf{r}$. The axis of rotation is the intersection of the two planes ($B1$ and $B2$) normal to each of the displacements and going through the midpoint of each of the displacements ($\mathbf{m}_1$ and $\mathbf{m}_2$).

However, the calibration of relative sensor positions at each frame is likely to shift between recording sessions. As such it would be valuable to automate the alignment of multiple scans with high precision. While generic point cloud alignment techniques exist they can be error prone.

6.3.1 Overview of Reuleaux's Method in 3D

Estimating the motion of a rigid 3D object requires knowledge of the displacements of at least three non-collinear points between the object instances, or two points with their normals, see Section 2.1. However in the case of rotational motion around a fixed unknown axis, only two non-parallel displacements are required. The two displacements $(\mathbf{d_1}, \mathbf{d_2})$ connecting corresponding points between object instances can be used to determine the axis of rotation of the object, which is known as the Reuleaux method [115]. Demonstrated in Figure 6.2, the Reuleaux method states:

The axis of the rotation is the line intersection of the two planes going through and orthogonal to the midpoint of the two displacements.

As explained in Chapter 3, a rotation can be represented by six DoF, five to represent the position and direction of the rotation axis, and one represents the angle. The rotation axis lies on the mid-orthogonal planes of all displacements, and hence any two displacements are enough to find the axis. If more than two displacements are measured, it can be used to verify the estimated axis parameters.

Having two points clouds P and P', where one point cloud is the result of unknown rotation applied to the other; each point cloud represents a set of 3D points distributed in $\mathbb{R}^3$, $P = \{p_0, p_1, \ldots, p_{M-1}\}$ and $P' = \{p'_0, p'_1, \ldots, p'_{M'-1}\}$. Where $p_i = \{x_i, y_i, z_i\}$,

$p'_i = \{x'_i, y'_i, z'_i\}$ are known correspondences, and M, M' are the total number of points in P and P' receptively.

First, the 3D position vectors of the midpoints ($\mathbf{m}_1, \mathbf{m2}$) of the displacement vectors ($\mathbf{d_1}, \mathbf{d_2}$) connecting corresponding points on the two point clouds are computed:

$$\begin{aligned} \mathbf{d_1} &= p_1 - p'_1 = [x_1, y_1, z_1]^\top - [x'_1, y'_1, z'_1]^\top \\ \mathbf{d_2} &= p_2 - p'_2 = [x_2, y_2, z_2]^\top - [x'_2, y'_2, z'_2]^\top \end{aligned} \tag{6.1}$$

$$\begin{aligned} \mathbf{m}_1 &= (p_1 + p'_1)/2 \\ \mathbf{m}_2 &= (p_2 + p'_2)/2 \end{aligned} \tag{6.2}$$

At each midpoint $\mathbf{m}$, a plane B orthogonal to the displacement $\mathbf{d}$, is constructed. The general equation of the plane is of the form:

$$ax + by + cz + e = 0 \tag{6.3}$$

where (a, b, c) represent the nonzero normal vector of the plane, which is the displacement ($\mathbf{d_1}$ or $\mathbf{d_2}$) and e is the dot product of the normal vector and the position vector of the midpoint ($\mathbf{m}_1$ or $\mathbf{m}_2$), hence $e1$ for the first plane (B_1) is given by:

$$e_1 = -(\mathbf{d_1} \cdot \mathbf{m}_1) \tag{6.4}$$

And similarly for the second plane (B_2). For both displacement vectors ($\mathbf{d_1}, \mathbf{d_2}$), two planes ($B_1, B_2$) each represented by four parameters are computed. Substituting in Equation 6.3, the equations of the orthogonal planes $B1$ and $B2$ are:

$$\begin{aligned} B_1 &: \mathbf{d_{1x}}x + \mathbf{d_{1y}}y + \mathbf{d_{1z}}z + e_1 = 0 \\ B_2 &: \mathbf{d_{2x}}x + \mathbf{d_{2y}}y + \mathbf{d_{2z}}z + e_2 = 0 \end{aligned} \tag{6.5}$$

The axis of rotation is the line intersection $\mathbf{r}$ between the nonparallel mid-orthogonal planes. It has a direction $\mathbf{v}$ which is perpendicular to the normals of both planes ($\mathbf{d_1}, \mathbf{d_2}$), and can thus be expressed as:

$$\mathbf{v} = \mathbf{d_1} \times \mathbf{d_2} \tag{6.6}$$

To form a complete description of a line, we also need to provide a point that lies on the line, this can be accomplished by constructing a third plane $B3$ that passes through the origin and has a normal direction $\mathbf{v}$.

$$B_3 : v_x x + v_y y + v_z z = 0 \tag{6.7}$$

We can solve for the unique point q where all three planes intersect, which is guaranteed to exist in this situation, Figure 6.3. Using Equation 6.8, we compute the point q that

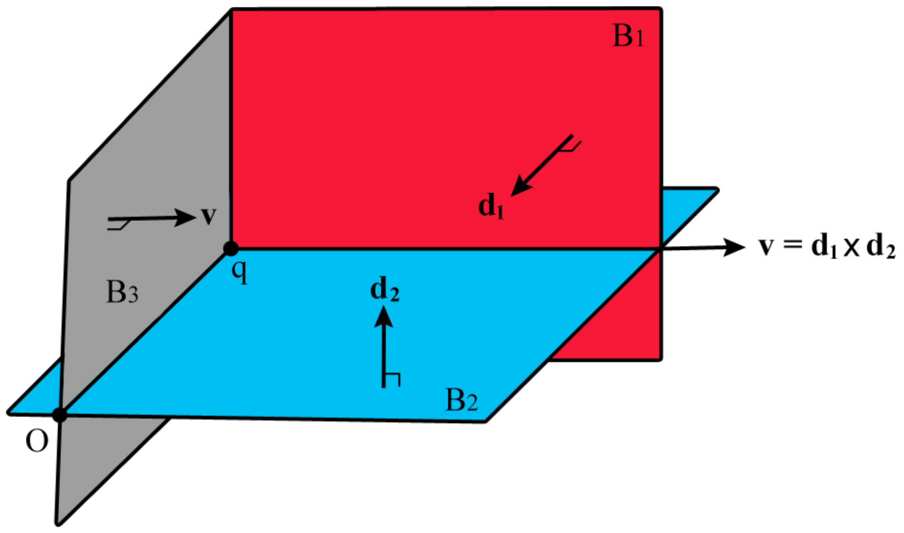


Figure 6.3: Computation of the point q on the axis of rotation. Two planes having normal vectors $\mathbf{d_1}$ and $\mathbf{d_2}$ intersecting at a line running in the direction $\mathbf{v}$. A unique point q on this line can be found by finding the intersection point with a third plane(B_3) passing through the origin and having a normal $\mathbf{v}$.

lies on the line of intersection as follows:

$$q = \begin{bmatrix} d_{1x} & d_{1y} & d_{1z} \\ d_{2x} & d_{2y} & d_{2z} \\ v_x & v_y & v_z \end{bmatrix}^{-1} \begin{bmatrix} -e_1 \\ -e_2 \\ 0 \end{bmatrix} \tag{6.8}$$

The uniqueness of the point q has been confirmed by our experimental exploration. The complete equation of axis of rotation $\mathbf{r}$ is given by:

$$\mathbf{r} = q + u\mathbf{v} \tag{6.9}$$

where u is the free parameter. Note that for rotation around a fixed axis, all point paths are segments of circles, each displacement is normal to the plane containing the axis of rotation and the midpoint of the displacement, these must be non-parallel for the mid-orthogonal planes to intersect, Figure 6.4.

6.3.2 Algorithm for Rotation Estimation

The algorithm is divided into two stages, the evidence gathering algorithm and the voting algorithm. In the evidence gathering algorithm, the Reuleaux method is applied iteratively on the two 3D point clouds, and the estimated axes of rotation parameters are stored along with their two corresponding point pairs. The voting algorithm is applied on the stored axes to find the best estimation of the true axis of rotation. Moreover, true correspondences and the rotation angle are also computed.

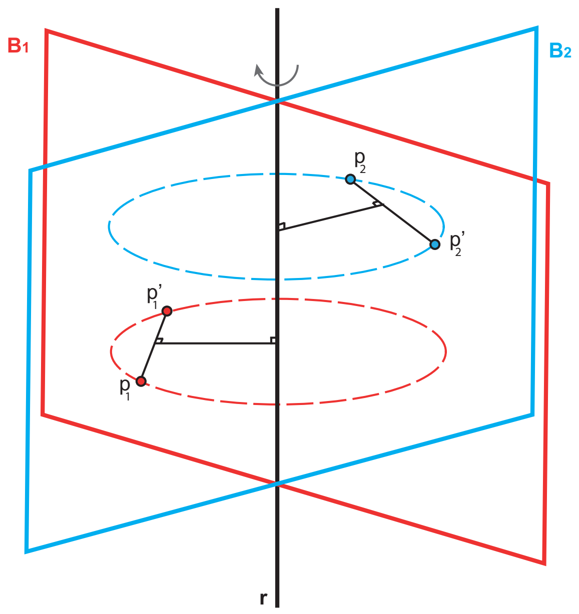


Figure 6.4: 3D rotation invariant geometric properties. All displacements connecting corresponding points are normal to the plane containing the axis of rotation and the midpoint of the displacement.

6.3.2.1 Evidence Gathering Algorithm

The process of computing an axis of rotation from two point pairs, summarised in Equation 6.9 is carried out iteratively for all points on the two input point clouds. At each iteration, the computed axis of rotation $\mathbf{r}$, represented by its position q and direction vector $\mathbf{v}$ is stored in a vector; each axis will have a reference to the four points which it was computed from. These points are used later to find the rotation angle.

There are three scenarios from which the two displacements can be taken; the displacements can be taken between two, three or four frames, as explained in Figure 6.5. In this research we focus on the first case, with only two time instants (frames) of the object, as most recognition and registration problems involve computing correspondences between two frames to estimate the motion.

6.3.2.2 Constraints

Since the algorithm takes 4 points at each iteration in one-to-one mapping approach, a constraint is imposed to avoid repetition; with this constraint the total number of iterations K, which is the number of all possible rotation axes, is given by:

$$K = \frac{(N(N-1))^2}{2} \tag{6.10}$$

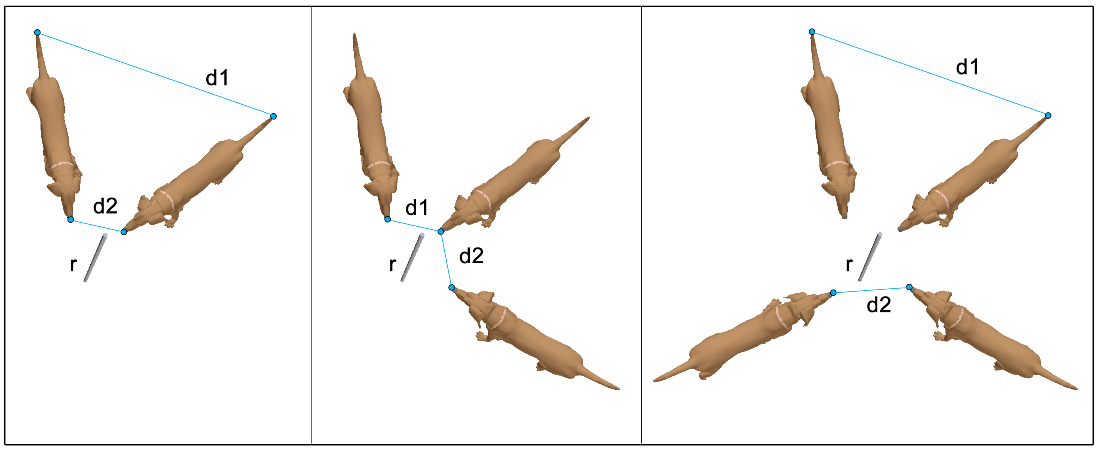


Figure 6.5: Axis of rotation ($\mathbf{r}$) can be estimated from two, three or four frames using the Reuleaux method.

```
for (i = 0; i < M − 1; i++) do
    for (k = i + 1; k < M − 1; k++) do
        for (j = 0; j < M′ − 1; j++) do
            for (l = 0; l < M′ − 1; l++) do
                if ( j ≠ l) then
                    if ( $(\mathbf{d_1} \nparallel \mathbf{d2})$ and $\left( \left| \|p_i - p_k\| - \left\|p'_j - p'_l\right\| \right| < \alpha \right)$ ) then
                        Compute $\mathbf{r}$
                        Save $q, \mathbf{v}$
                        Save $(i, j)$ , $(k, l)$
                    end
                end
            end
        end
    end
end
```

Algorithm 4: Evidence gathering for rotation estimation.

where N is the number of points of the point cloud having the least points. Note that this is the maximum number of the computed axes of rotation, i.e. if the all points are considered in the evidence gathering process.

While considering all points on both clouds can theoretically be implemented, it is not practical (especially for point clouds of large numbers of points such as Kinect clouds). Instead of considering all points, this problem can be solved in a number of approaches:

1. Subsampling: As explained in Subsection 2.5.2, subsampling significantly the number of points, by using the proposed method for subsampling, original surface points are retained while reducing the total number of points in the point cloud.

2. Random points selection: At each iteration, two points on each point cloud are chosen and paired. This process is not guaranteed to compute a true axis as the probability of selecting two correct corresponding pairs is very low. For the general motion estimation algorithm, three point pairs are taken at each iteration, which makes this probability even smaller. However this probability can be increased by having a large number of iterations.

3. Keypoint detection: Our evidence gathering algorithm can be applied on detected keypoints instead on the input clouds; a keypoint detection algorithm will provide a number of keypoints that is significantly less than the number of original point clouds.

These approaches will reduce the number of points N on the point cloud, and subsequently the number of iterations in the evidence gathering process K. However consistency of points between dataset has to be considered; i.e. after reducing the number of points, there must still be corresponding points between the datasets; otherwise the algorithms will fail. In other words the proposed algorithms require “few” but “good” correspondences.

Even with taking any of these approaches, the number of computed axes L can be significantly reduced by imposing geometric constraints on points.

The first constraint guarantees that the two displacements are non parallel, ($\mathbf{d_1} \nparallel \mathbf{d2}$). This constraint is essential to guarantee that that two orthogonal planes intersect and the axis of rotation exists.

The other constraint is similar to the correspondence grouping stage in the local matching pipeline, Chapter 2, where correspondences are discarded by enforcing geometrical consistency between them. Here, since the transformation between the two point clouds is rigid, the Euclidean distance between the two points on the first point cloud must be equal or close to the distance between the corresponding pair on the other cloud. Hence at each iteration, we impose the following relation:

$$|\,\|p_i - p_j\| - \left\|p'_i - p'_j\right\|\,| < \alpha \tag{6.11}$$

where α is a tolerance threshold intuitively representing the consensus set dimension. The value of this threshold depends on the quality and type of 3D sets, it is proportional to the noise of the 3D data; if the point clouds have a small amount of noise then this threshold is tuned to have a small value to discard correspondence subsets supported by a small consensus. Imposing this constraint results in computing the axis or rotation for point pairs that are geometrically consistent. The value of α is determined experimentally; we found that for the best value for it is 1 mm.

At each iteration, the axis of rotation is only computed for the 4 points that satisfy the non-parallelism constraint and the geometric consistency constraint. The number of computed axes L will be significantly less than the total number of iterations, $L \ll K$. This improves the accuracy of the following voting procedure, as the voting will be only on computed axes that are "consistent", and reduces the storage requirements and computational time of the voting algorithm.

6.3.2.3 Voting Algorithm

```
for ($i = 0; i < L; i{+}{+}$) do
    for ($j = i + 1; j < L; j{+}{+}$) do
        if ($|v_i \cdot v_j| > t_{dir}$) then
            if ($\|q_i - q_j\| < t_{pos}$) then
                Accumulator[$i$]++
                Accumulator[$j$]++
            end
        end
    end
end
Search Accumulator for peak.
```

Algorithm 5: Voting for invariant axis.

Due to a potentially large number of invalid correspondences, even with imposing geometrical constraints, it is necessary to use a voting algorithm to obtain accurate estimates for the rotation axis. The algorithm identifies the largest cluster of similar axes. Axes estimated from erroneous correspondences will be randomly distributed in space, whereas correct correspondences will be closely aligned. To calculate the clusters efficiently, a pair of thresholds are used.

The first threshold (t_{dir}), is the minimum value that the dot product between the two axes can take.

$$|\mathbf{v}_i \cdot \mathbf{v}_j| > t_{dir} \tag{6.12}$$

Axes that are in approximately the same direction will have a dot product close to 1. As the axes become more divergent the dot product value will approach -1, this is usually referred to as cosine similarity. The axes might have similar orientations but opposite directions, hence the absolute value is taken, Equation 6.12.

The second threshold (t_{pos}) is the maximum Euclidean distance between the axes.

$$\|q_i - q_j\| < t_{pos} \tag{6.13}$$

The optimal threshold values for the voting algorithm can be identified by calculating the most generous pair of thresholds that correctly identify the true axis in a training

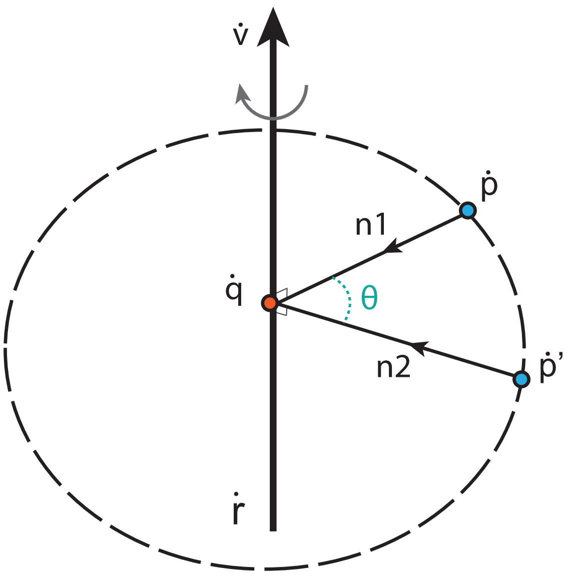


Figure 6.6: Computation of angle of rotation.

data set. The number of iterations U in the voting algorithm is proportional to the number of estimated axis L, and is given by:

$$U = \sum_{i=0}^{L-1} i = \frac{L(L+1)}{2} \tag{6.14}$$

Note that U can be further reduced by voting for a random subset from the computed axes rather than all computed axis.

6.3.2.4 Computation of Rotation Angle

Applying the voting algorithm gives the index of the axis of rotation with most consensus ($\dot{\mathbf{r}}$). As mentioned earlier, at each iteration a reference to two point pairs is saved with each computed axis. The angle of rotation (θ) is computed by using any of the two pairs of corresponding points $((\dot{p_1}, \dot{p_1}'), (\dot{p_2}, \dot{p_2}'))$ referenced to the index of ($\dot{\mathbf{r}}$), Figure 6.6. This is achieved by finding the minimal perpendicular vectors $(\mathbf{n_1}, \mathbf{n_2})$ between the two points and $\dot{\mathbf{r}}$ as follows:

$$\begin{aligned} \mathbf{n_1} &= \dot{p_1} - (\dot{q} + ((\dot{p_1} - q) \cdot \dot{\mathbf{v}}) \times \dot{\mathbf{v}}) \\ \mathbf{n_2} &= \dot{p_1}' - (\dot{q} + ((\dot{p_1}' - q) \cdot \dot{\mathbf{v}}) \times \dot{\mathbf{v}}) \end{aligned} \tag{6.15}$$

where $\dot{q}$ is a point on $\dot{\mathbf{r}}$ and $\dot{\mathbf{v}}$ is the direction vectors of $\dot{\mathbf{r}}$. Then the angle of rotation θ is given by:

$$\theta = \arccos(\frac{\mathbf{n_1} \cdot \mathbf{n_2}}{\|\mathbf{n_1}\| \, \|\mathbf{n_2}\|}) \tag{6.16}$$

6.4 General Motion Estimation by Evidence Gathering

In the previous section we introduced an algorithm to estimate the 3D rotation parameters between using the Reuleaux method. Even though rotation estimation has many practical applications, in most scenarios it is required to estimate general 3D motion consisting of rotation and translation. Unlike rotation around a fixed axis, the instantaneous axis of rotation can not be defined when the motion has a transitional component.

In this section we generalise the proposed algorithm in the previous section to estimate the full motion of rigid objects moving freely in 3D space. We employ Chasles' theorem [112] to estimate the screw axis describing the rigid motion.

Similar to the method described in the previous section, the algorithm is divided into two stages, the evidence gathering algorithm and the voting algorithm. In the evidence gathering algorithm, Chasles's theorem is applied iteratively on two 3D point clouds, taking three correspondences instead of two at each iteration, and the estimated screw parameters are stored along with their three corresponding point pairs. The voting algorithm is applied on the stored axes to find the best estimation of the true screw axis representing the 3D motion.

6.4.1 Chasles' Theorem

Chasles' theorem states:

Any rigid body displacement in 3D can be produced by a rotation around a line followed by a translation along that line.

In other words, any motion in 3D can be represented by a rotation around a single axis and a translation parallel to that axis, this axis is referred to as the *screw axis* and as this motion is reminiscent of the displacement of a screw, it is called a *screw motion.*

To define a screw 3D transformation between two frames, the screw axis $\mathbf{h_c}$, the rotation angle θ and the pitch η must be specified. The pitch is the ratio of the linear displacement to the rotation. For a pure translational motion, the screw axis is not unique; since the rotation is zero, the screw axis will have infinite pitch.

In [7, 8], the authors have investigated Chasles' screw motion analysis in the context of computer vision calibration tasks and developed a method describing rigid geometric constraints based on the analysis of correspondence vectors, see 6.2.1.2. In this research we use their mathematical formalising of relevant geometric properties of correspondence vectors for computing the screw axis in an iterative evidence gathering framework to estimate the 3D motion and compute correspondences.

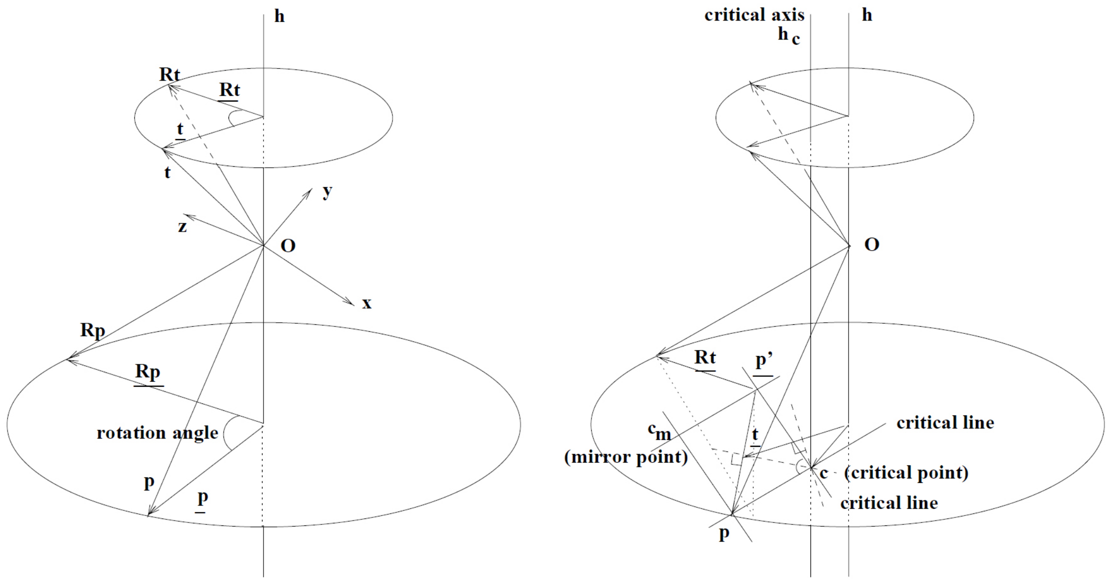


Figure 6.7: The relationships among correspondence (p, p'), critical point $\mathbf{c}$, critical (screw) axis $\mathbf{h_c}$, and translation vector $\mathbf{t}$ in 3D. Left, given a transformation $(R, \mathbf{t})$ the rotation axis (screw axis direction) $\mathbf{h}$ and rotation angle θ are determined. Right, corresponding pairs (p, p') and the translation vector $\mathbf{t}$ are projected on a plane perpendicular to $\mathbf{h}$. The critical point $\mathbf{c}$ is determined at the intersection of the bisector lines $(\underline{p}, \underline{p}')$ and $\mathbf{t}$. The critical point determines the position of the critical axis $\mathbf{h_c}$ where the vectors $\mathbf{c}\underline{p}$ and $\mathbf{c}\underline{p}'$ are of equal length with the including angle equal to the rotation angle θ of the transformation. Source [8].

According to [7], given three 3D correspondences $(p_i, p'_i), \{i = 1, 2, 3\}$, their correspondences vectors $\mathbf{d_i}$ (Equation 6.1) can be evaluated. If the difference of correspondence vectors are not parallel, then the screw axis directional vector $\mathbf{h} = [h_x, h_y, h_z]^\top$ can be uniquely determined as:

$$\mathbf{h} = \frac{(\mathbf{d_2} - \mathbf{d_1}) \times (\mathbf{d_3} - \mathbf{d_1})}{\|(\mathbf{d_2} - \mathbf{d_1}) \times (\mathbf{d_3} - \mathbf{d_1})\|} \tag{6.17}$$

This equation holds for all true correspondences, i.e. all vectors resulting from the cross product of vector difference between all true correspondences coincident at the same axis, which is the screw axis.

Note that Equation 6.17 describes only the direction of the screw axis, and hence a point on the axis is required to complete its definition. Once $\mathbf{h}$ is computed, any correspondences (p_i, p'_i) are projected on a plane perpendicular to $\mathbf{h}$, using the equation:

$$\underline{p} = (I - \mathbf{h}\mathbf{h}^\top)p \ , \ \underline{p}' = (I - \mathbf{h}\mathbf{h}^\top)p' \tag{6.18}$$

where I is the identity matrix. Given two non-parallel projected correspondence vectors $(\underline{\mathbf{d_1}}, \underline{\mathbf{d_2}})$ evaluated from two projected correspondences $(\underline{\mathbf{d_1}} = \underline{p_1} - \underline{p'_1},\ \underline{\mathbf{d_2}} = \underline{p_2} - \underline{p'_2})$, the critical point $\mathbf{c}$ in 3D can be uniquely estimated by:

$$\mathbf{c} = \frac{\underline{p_2}^\top \underline{p_2} - \underline{p'_2}^\top \underline{p'_2}}{2(\underline{\mathbf{d_2}})^\top H(\underline{\mathbf{d_1}})} H(\underline{\mathbf{d_1}}) + \frac{\underline{p_1}^\top \underline{p_1} - \underline{p'_1}^\top \underline{p'_1}}{2(\underline{\mathbf{d_1}})^\top H(\underline{\mathbf{d_2}})} H(\underline{\mathbf{d_2}}) \tag{6.19}$$

Where the matrix H is given by:

$$H = \begin{bmatrix} 0 & -h_z & h_y \\ h_z & 0 & -h_x \\ -h_y & h_x & 0 \end{bmatrix} \tag{6.20}$$

Together the direction of the screw axis $\mathbf{h}$ and the critical point $\mathbf{c}$ define the screw axis $\mathbf{h_c}$ in 3D space, similar to Equation 6.9, the screw axis can be written as:

$$\mathbf{h_c} = \mathbf{c} + u\mathbf{h} \tag{6.21}$$

The screw access $\mathbf{h_c}$ has interesting geometric properties; according to [8], $\mathbf{h_c}$ is equidistant to any 3D correspondences (p_i, p'_i) and the including angle between the lines passing through (p_i, p'_i), perpendicular to and intersecting $\mathbf{h_c}$ is equal to the rotation angle θ of the transformation, Figure 6.7. Moreover, the critical point $\mathbf{c}$ also has interesting geometric properties which are summarised by the following property:

There is one and only one invariant point $\mathbf{c}$ in 3D which is equidistant to the projected correspondences $(\underline{p}, \underline{p}')$ on the plane perpendicular to the screw axis $\mathbf{h} = (h_x, h_y, h_z)^T$ and the including angle between vectors $(\underline{p} - \mathbf{c})$ and $(\underline{p}' - \mathbf{c})$ is equal to the rotation angle θ around the screw axis. Moreover, the vertical bisector of the projected translation vector $\underline{\mathbf{t}}$ on this plane also intercepts at the same point, Figure 6.7. This property can be formalised by the following equations:

$$\frac{\|\underline{p} - \mathbf{c}\|}{\|\underline{p}' - \mathbf{c}\|} = 1 \tag{6.22}$$

$$cos(\theta) = \frac{(\underline{p} - \mathbf{c})^\top (\underline{p}' - \mathbf{c})}{\|\underline{p} - \mathbf{c}\| \, \|\underline{p}' - \mathbf{c}\|} \tag{6.23}$$

Given a homogeneous transformation matrix $(T = [R|\mathbf{t}])$, a conversation to screw axis representation $(\mathbf{h_c} = \mathbf{c} + u\mathbf{h})$ can be achieved by the following equations:

$$\mathbf{h} = \frac{\mathbf{l}}{2\theta \sin\theta} \;,\; \mathbf{c} = \frac{(I - R)\mathbf{t}}{2(1 - \cos\theta)} \tag{6.24}$$

Where I is the identity 3×3 matrix, and $\mathbf{l}$, θ are given by the following equations:

$$\mathbf{l} = \begin{bmatrix} (R_{32} - R_{23}) & (R_{13} - R_{31}) & (R_{21} - R_{12}) \end{bmatrix} \tag{6.25}$$

$$\theta = sign(\mathbf{l} \times \mathbf{t}) \left| cos^{-1}\left(\frac{R_{11} + R_{22} + R_{33} - 1}{2}\right) \right| \tag{6.26}$$

Proofs of the above equations can be found in [135].

6.4.2 Algorithm for Motion Estimation

```
for ($i = 0;\ i < M - 1;\ i{+}{+}$) do
    for ($k = i + 1;\ k < M - 1;\ k{+}{+}$) do
        for ($o = k + 1;\ o < M - 1;\ o{+}{+}$) do
            for ($j = 0;\ j < M' - 1;\ j{+}{+}$) do
                for ($l = 0;\ l < M' - 1;\ l{+}{+}$) do
                    for ($w = 0;\ w < M' - 1;\ w{+}{+}$) do
                        if ( $j \neq l$ and $l \neq w$ and $j \neq w$ ) then
                            if ( $(\mathbf{d_1} \nparallel \mathbf{d_2} \nparallel \mathbf{d3})$ and ( $|d_{min1} - d_{min2}| < \alpha$ ) ) then
                                Compute $\mathbf{h_c}$
                                Save $\mathbf{h}, \mathbf{c}$
                                Save $(i, j)$ , $(k, l)$, $(o, w)$
                            end
                        end
                    end
                end
            end
        end
    end
end
```

Algorithm 6: Evidence gathering for general motion estimation.

Similar to the rotation estimation algorithm, this algorithm is divided into two stages; the evidence gathering algorithm and the voting algorithm. In the evidence gathering algorithm, Chasles' theorem, summarised in Equation 6.21, is applied iteratively on the two 3D point clouds, and the estimated screw parameters are stored along with their three corresponding point pairs, instead of two.

The same constraints apply to this algorithm too; the three displacements must be non-parallel ($\mathbf{d_1} \nparallel \mathbf{d_2} \nparallel \mathbf{d3}$), and the absolute difference of distances between points on the same cloud should be below a certain threshold α. This absolute difference, which is defined by Equation 6.11 for the rotation estimation method, is calculated differently for this method; since three points on each cloud are taken at each iteration, we set this difference to be between the smallest distances between the three pairs on each cloud (d_{min1}, d_{min2}).

Moreover, since three pairs of points are taken at each iteration, the total number of iterations K for the evidence gathering process is given by:

$$K = \frac{(N(N-1)(N-2))^2}{6} \tag{6.27}$$

where N is the number of points of the point cloud having the least points. As explained earlier, this number can be significantly reduced, and the number of computed axes can also be reduced by imposing the non-parallelism as well as the geometric consistency constraints.

Next, the same voting algorithm (Algorithm 5) is applied on the stored axes to find the best estimation of the true screw axis.

6.4.2.1 Computation of Motion Parameters

Once Algorithm 5 is implemented, and the screw axis with the most consensus $\dot{\mathbf{h}}_{\mathbf{c}}$ is determined, the rotation matrix R and the 3D trasnslation vector $\mathbf{t}$ are computed by the following equations:

$$R = I - H\sin\theta + (1-\cos\theta)H^2 \tag{6.28}$$

$$\mathbf{t} = (I - R^\top)\mathbf{c} + \mathbf{h}\mathbf{h}^\top\dot{\mathbf{d}} \tag{6.29}$$

where I is the identity 3×3 matrix, θ is defined in Equation 6.23, and $\dot{\mathbf{d}}$ is the linear displacement $(\dot{p} - \dot{p}')$ computed from any pair of the three pairs of point correspondences indexed with $\dot{\mathbf{h}}_{\mathbf{c}}$. The derivation of equations 6.29 and 6.28 can be found in [8].

6.5 Demonstration Examples

6.5.1 Example 1: Rotation

Figure 6.8 shows a 3D point cloud of an object with dimensions $(112 \times 95 \times 78)$ mm^3 rotated around the ground truth axis at position $(-100, 50, 150)$ mm parallel to the Y axis in 3D space by a rotation angle $\theta = 70°$, note that these ground truth parameters are not used by the algorithm. The transformation matrix (T_g) resulting from this rotation is computed using Equation 3.2 and is given by:

$$T_g = \begin{bmatrix} 0.342 & 0 & 0.940 & -206.75 \\ 0 & 1 & 0 & 0 \\ -0.940 & 0 & 0.342 & 4.727 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6.30}$$

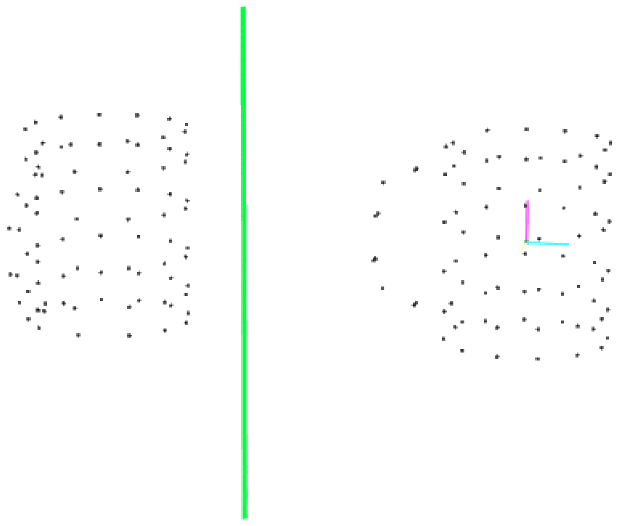


Figure 6.8: Object rotated around unknown axis.

First, we apply Algorithm 4 to gather the evidence for the rotation axis. Figure 6.9 shows three iterations of the algorithm; taking two different pairs and computing the rotation axis at each iteration by intersecting the mid-orthogonal planes of their displacements vectors.

After the computation of all possible rotation axes, Algorithm 5 is implemented to find the axis $\dot{\mathbf{r}} = \dot{q} + u\dot{\mathbf{v}}$ best describing the rotation. Figure 6.10 shows the estimated rotation axis $\dot{\mathbf{r}}$ and the two pairs of estimated corresponding points $((\dot{p}_1, \dot{p}_1'), (\dot{p}_2, \dot{p}_2'))$. It is clear in the figure that the estimated axis $\dot{\mathbf{r}}$ perfectly coincides with the ground truth rotation axis $\mathbf{r_g} = q_g + u\mathbf{v_g}$. The rotation angle θ is computed from the two correspondence pairs stored with $\dot{\mathbf{r}}$ using Equation 6.16.

Table 6.2 summarises the attributes and used algorithm parameters for this example. The positional offset e_{pos} is the Euclidean distance between the estimated axis and the ground truth axis, and directional offset e_{dir} is the direction difference between the axes. Equations 6.31 and 6.32 show how both offsets are computed.

$$e_{pos} = \left| \frac{\mathbf{v_g} \times \dot{\mathbf{v}}}{\|\mathbf{v_g} \times \dot{\mathbf{v}}\|} \cdot (q_g - \dot{q}) \right| \tag{6.31}$$

$$e_{dir} = 1 - (|\dot{\mathbf{r}} \cdot \mathbf{r_g}|) \tag{6.32}$$

Figures 6.11 and 6.12 show the histogram of axes in terms of number of votes. The large number of correctly estimated axes is due to the fact that the point clouds used in this example represent an ideal scenario; no noise or missing points, and each point in one point cloud has a correct correspondence in other point cloud.

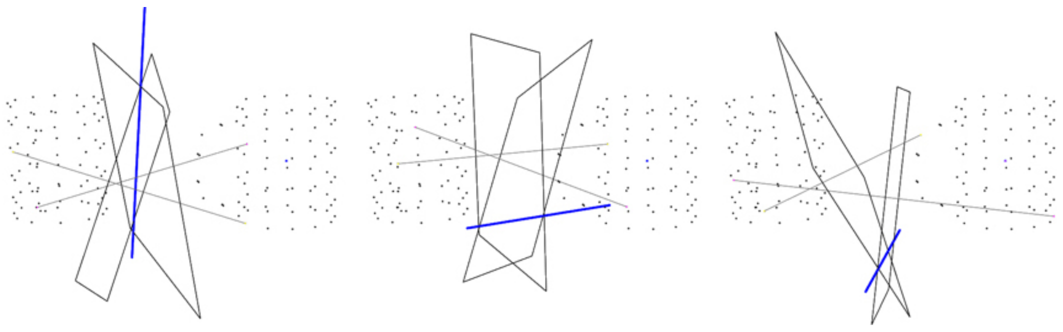


Figure 6.9: Iterations of evidence gathering algorithm. At each iteration a rotation axis is computed by intersecting the mid-orthogonal planes to the two displacements, and stored with it's two corresponding point pairs.

Points	K	L	U	C V I	$t_{pos}(mm)$	t_{dir}
84	2.4×10^7	88296	3.9×10^9	3753993	1	0.99
$\alpha(mm)$	$\dot{\mathbf{r}}$ Votes	$e_{pos}(mm)$	e_{dir}	$e_{\theta}(°)$	EG Time(s)	V Time(s)
1	2350	0	0	0	1.45	87.21

Table 6.2: Demonstration Example 1. Top Columns from left to right: number of points in each point cloud, total number of iterations in the evidence gathering algorithm (Equation 6.10), number of computed axes which satisfy constraints, total number of iterations in the voting algorithm (Equation 6.14), number of counted voting iterations passing thresholds, Euclidean distance threshold, and cosine similarity threshold. Bottom Columns: Correspondence grouping tolerance threshold, max vote count, positional offset (Equation 6.31), directional offset (Equation 6.32), angle offset, evidence gathering time, voting time. Total processing time is 88.66s.

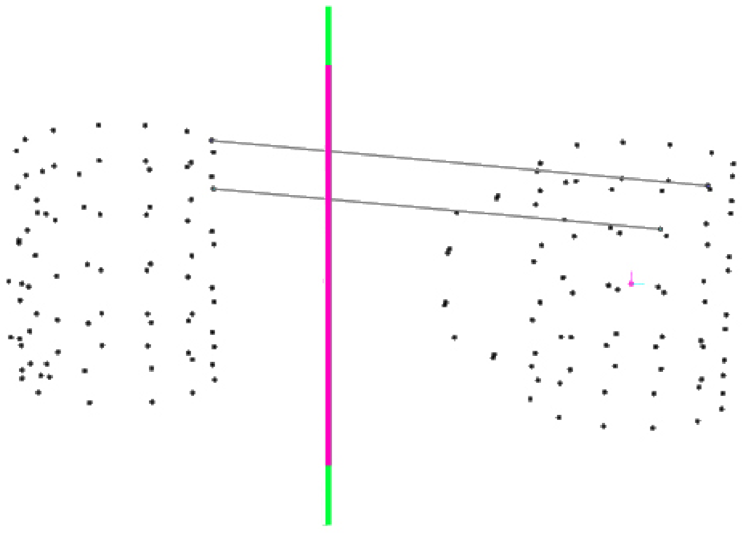


Figure 6.10: Estimated rotation axis with its associated correspondence pairs.

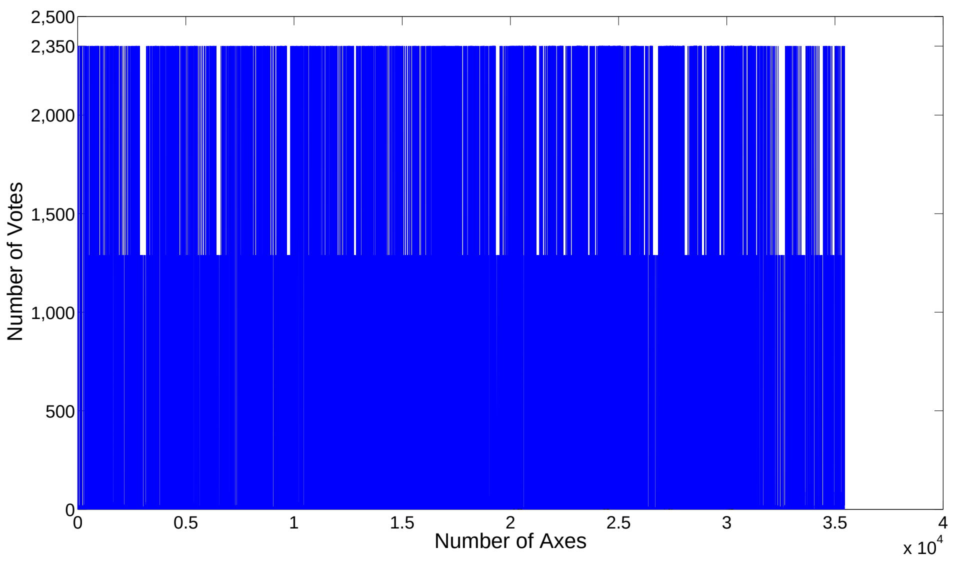


Figure 6.11: Histogram of votes for axes generated by Algorithm 5 for Example 1.

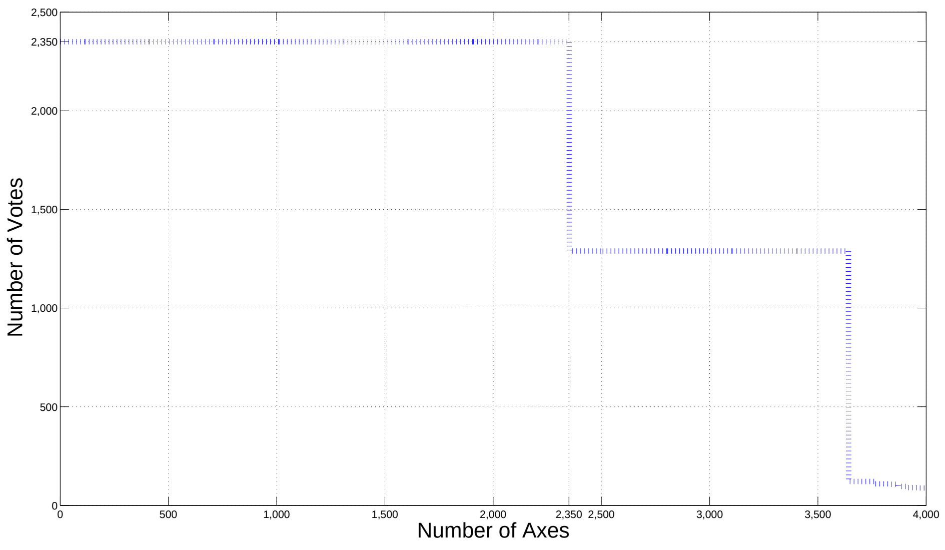


Figure 6.12: Ordered histogram of votes for axes generated by Algorithm 5 for Example 1.

6.5.2 Example 2: General Motion

The previous example demonstrated an ideal scenario with full 3D point clouds and no missing points. In this example we demonstrate a more realistic case; we estimate the general transformation of a 2.5D point cloud with partial overlap, i.e. only a subset of points will have true correspondences, which is a typical scenario for 3D motion estimation algorithms.

Figure 6.13 shows 2.5D point clouds or an object, with dimensions $(112 \times 95 \times 39)\ mm^3$, rotated by an angle $\theta = 90°$ around the Y axis at origin, which is represented by a rotation matrix R_g and translated in by the vector $\mathbf{t_g} = [200, -160, 150]^\top mm$. Together, according to Equation 2.3, the total transformation T_g is given by:

$$T_g = \begin{bmatrix} 0 & 0 & 1 & 200 \\ 0 & 1 & 0 & -160 \\ -1 & 0 & 0 & 150 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6.33}$$

The corresponding ground truth screw axis $\mathbf{h_g}$, shown in Figure 6.13, of this transformation is computed using Equation 6.24 is equal to:

$$\mathbf{h_g} = \begin{bmatrix} 175 \\ 0 \\ -25 \end{bmatrix} + u \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \tag{6.34}$$

The ground truth screw axis has a rotation angle θ of 90° and a pitch $\eta = 3.197$. Similar to Example 1, after the computation of all possible screw axes, Algorithm 5 is implemented to find the screw axis $\dot{\mathbf{h}}_\mathbf{c}$ best describing the 3D transformation. Figure 6.14 shows $\dot{\mathbf{h}}_\mathbf{c}$ and the three pairs of estimated corresponding points $((\dot{p}_1, \dot{p}_1'), (\dot{p}_2, \dot{p}_2'), (\dot{p}_3, \dot{p}_3'))$, it is clear in the figure that the estimated screw axis $\dot{\mathbf{h}}_\mathbf{c}$ perfectly coincides with the ground truth rotation axis. From $\dot{\mathbf{h}}_\mathbf{c}$ and any of the associated three point pairs, the motion parameters can be computed using equations 6.28 and 6.29.

Table 6.3 summarises the attributes and used algorithm parameters for this example and Figures 6.15 and 6.16 show the histogram of axes in terms of number of votes. Note the number of maximum votes is much less than the previous example. This is due to the fact that the point clouds are 2.5D and have partial overlap, and also the number of points taken is less than the previous example.

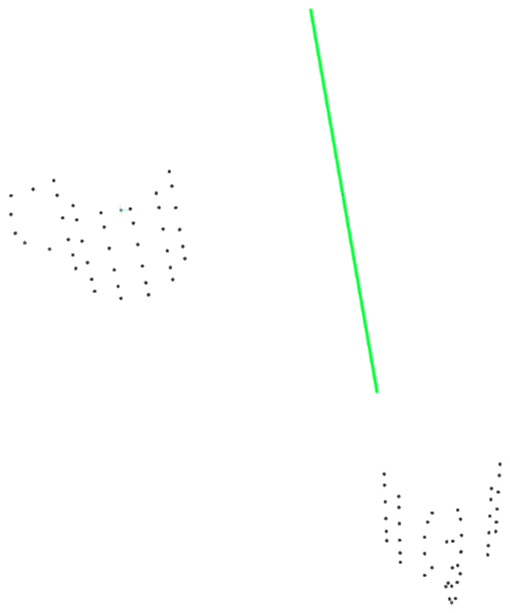


Figure 6.13: 2.5D point cloud rotated and translated in 3D, ground truth screw axis representing motion in green.

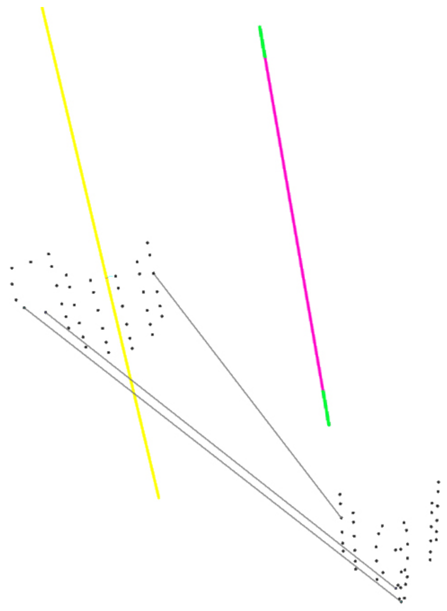


Figure 6.14: Estimated screw axis with its associated three pairs of correspondences. The estimated axis (purple) perfectly coincides with ground truth screw axis (green). Note that the screw axis is parallel to the rotation axis (yellow).

Points	K	L	U	C V I	$t_{pos}(mm)$	t_{dir}
20	7.8×10^7	93048	4.3×10^9	20242	1	0.99
$\alpha(mm)$	$\dot{\mathbf{h}}_\mathbf{c}$ Votes	$e_{pos}(mm)$	e_{dir}	$e_\theta(°)$	EG Time(s)	V Time(s)
1	51	0	0	0	3.35	87.10

Table 6.3: Demonstration Example 2. The total number of iterations (EG Its.) in this example is computed using Equation 6.27. Total processing time is 90.45s.

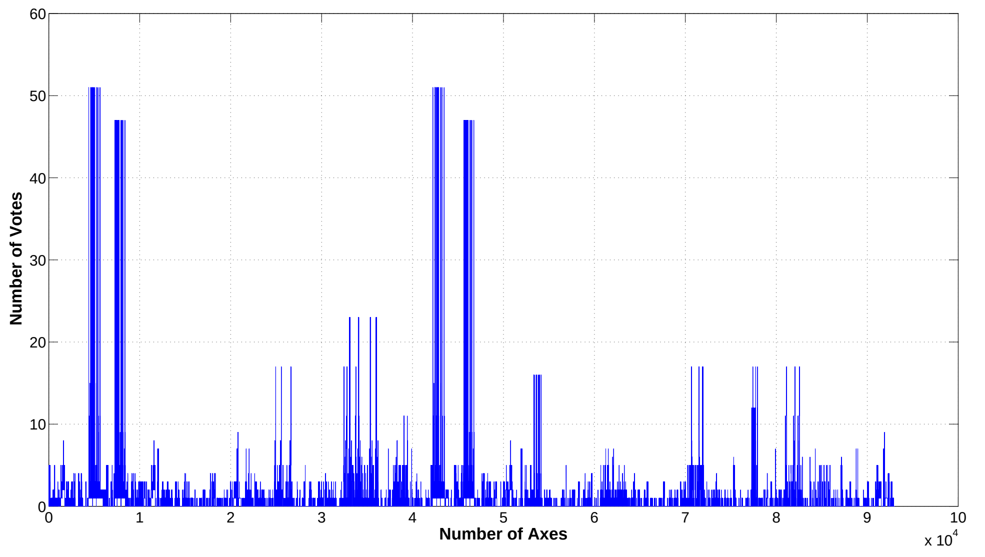


Figure 6.15: Histogram of votes for axes generated by Algorithm 5 for Example 2.

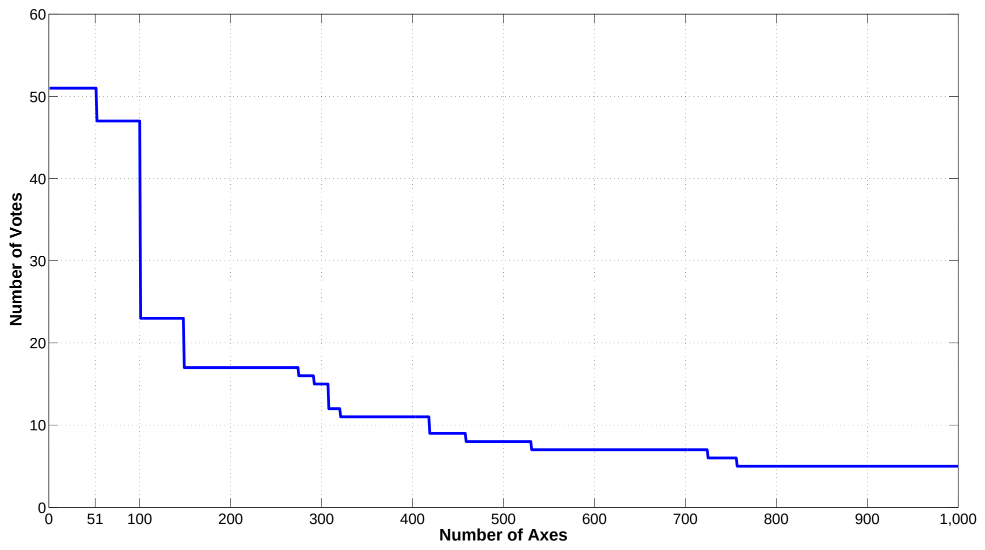
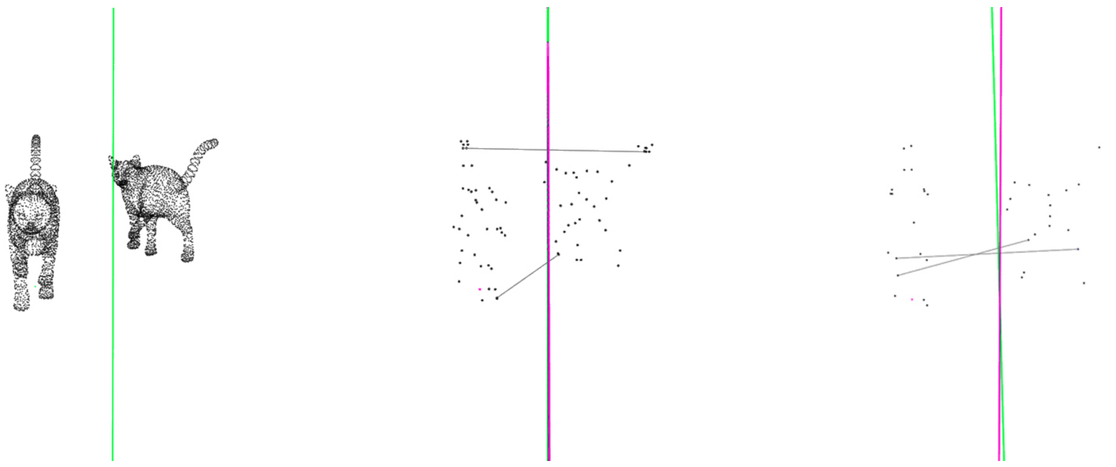


Figure 6.16: Ordered histogram of votes for axes generated by Algorithm 5 for Example 2.



(a) Input point clouds. (b) Estimated $\dot{\mathbf{r}}$, $\beta = 3\ mm^3$. (c) Estimated $\dot{\mathbf{r}}$, $\beta = 5\ mm^3$.

Figure 6.17: Rotation estimation algorithm applied on subsampled point clouds. Ground truth rotation axis $\mathbf{r_g}$ in green, estimated rotation axes $\dot{\mathbf{r}}$ in purple.

6.5.3 Example 3: Rotation - Subsampled

In the two previous examples, the proposed algorithms were iterated over *all* points in the input point clouds. In this example we use point clouds with a relatively large number of points; before we apply the rotation estimation algorithm, we subsample the point cloud using the proposed subsampling method, explained in detail in Subsection 2.5.2.

Figure 6.17(a) shows a 3D point cloud of an object with dimensions $(34 \times 192 \times 96)\ mm^3$ rotated around the ground truth axis at position $(55, 55, 25)\ mm$ parallel to the Z axis in 3D space by a rotation angle $\theta = 165°$.

Both point clouds used in this example have 3400 points before subsampling, after subsampling the number is reduced to 38. The voxel size β used for subsampling is $3\ mm^3$ and the number of octree levels is 5. Table 6.4 summarises the attributes and algorithm parameters used for this example. Note that subsampling introduced errors in the estimated parameters of the rotation axis; 0.57 mm positional offset and an angle offset of 0.68°.

To show the effect of the subsampling voxel size on the performance estimation algorithm, we conduct the same experiment again but with a a voxel size β of 5 mm^3; the resulting subsampled point clouds will have 18 points and the number of octree levels is 4, Table 6.5. Note that the errors in the estimated parameters have increased. Figure 6.17 shows the effect of subsampling and the estimated rotation axes. Detailed analysis of the effect of subsampling on the performance of the rotation estimation algorithm is in Subsection 6.6.2.

Points	K	L	U	C V I	$t_{pos}(mm)$	t_{dir}
38	988418	28756	4.1×10^8	5043	10	0.99
$\alpha(mm)$	$\dot{\mathbf{r}}$ Votes	$e_{pos}(mm)$	e_{dir}	$e_\theta(°)$	EG Time(s)	V Time(s)
2	49	0.57	0	0.68	0.55	28.79

Table 6.4: Demonstration Example 3, subsampled at voxel size 3 mm^3. Total processing time is 29.34s.

Points	K	L	U	C V I	$t_{pos}(mm)$	t_{dir}
18	46818	7582	2.8×10^7	1965	1	0.99
$\alpha(mm)$	$\dot{\mathbf{r}}$ Votes	$e_{pos}(mm)$	e_{dir}	$e_\theta(°)$	EG Time(s)	V Time(s)
10	10	1.90	0.0013	3.73	0.01	0.61

Table 6.5: Demonstration Example 3, subsampled at voxel size 5 mm^3. Total processing time is 0.62s.

6.5.4 Example 4: General Motion - Subsampled

In this example we apply the general motion estimation algorithm on a point cloud translated and rotated in 3D space. Similar to the previous example, we subsample the point clouds before applying the algorithm.

We use the same point cloud in Example 3, Figure 6.17(a). We transform the point cloud with the ground truth screw axis $\mathbf{h_g}$ given by Equation 6.35.

$$\mathbf{h_g} = \begin{bmatrix} -60 \\ 115 \\ 55 \end{bmatrix} + u \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \tag{6.35}$$

The ground truth screw axis has a rotation angle θ of 35° and a pitch $\eta = 2.544$. Tables 6.6 and 6.7 summarise the used parameters of the algorithm when applied on subsampled point cloud at $\beta = 3\ mm^3$ and $\beta = 5\ mm^3$ respectively. Detailed analysis of the effect of subsampling on the performance of the rotation estimation algorithm is in Subsection 6.6.2.

Points	K	L	U	C V I	$t_{pos}(mm)$	t_{dir}
38	4.27×10^8	607224	9.2×10^{10}	326851	1	0.99
$\alpha(mm)$	$\dot{\mathbf{h}}_\mathbf{c}$ Votes	$e_{pos}(mm)$	e_{dir}	$e_\theta(°)$	EG Time(s)	V Time(s)
1	125	4.11	0	6.80	13.14	283.85

Table 6.6: Demonstration Example 4, subsampled at voxel size 3 mm^3. Total processing time is 296.99s.

Points	K	L	U	C V I	$t_{pos}(mm)$	t_{dir}
18	1.2×10^7	80564	1×10^{10}	51562	1	0.99
$\alpha(mm)$	$\dot{\mathbf{h}}_\mathbf{c}$ Votes	$e_{pos}(mm)$	e_{dir}	$e_\theta(°)$	EG Time(s)	V Time(s)
1	73	21.14	0.02	10.1	4.89	106.71

Table 6.7: Demonstration Example 4, subsampled at voxel size 5 mm^3. Total processing time is 111.60s.

6.6 Analysis and Evaluation

It can be seen from Tables 6.2 and 6.3 that approximately 97% of the processing time is consumed in the voting process. This is due to the large number of voting iterations (U). The obvious way to improve the speed of the proposed algorithms is to reduce U, which can be achieved by the following:

- Reducing the number of input points. This can be achieved in a number of approaches as explained in Subsection 6.3.2.2.
- Parameter tuning. The values of parameters such as the geometric consistency constraint (α) can significantly reduce the number of voting iterations.
- Increasing step size in voting algorithm. The “default” implementation of the voting algorithm considers every axis computed by the evidence gathering algorithm (L), as shown in Algorithm 5. Since L is typically large, considering every axis is not necessary; considering every nth axis, or even considering a random fixed number of axes, can significantly reduce U and hence improve processing time.

Figures 6.12 and 6.16 show the ordered histograms of votes for Examples 1 and 2 respectively, note in both figures the number of axes having the maximum number of votes is equal to the maximum number of votes, which confirms the methodology of the voting algorithm.

6.6.1 Time

Note that the proposed algorithm for general motion estimation can be considered as a generalisation of the rotation estimation algorithm, as it can solve the rotation only case; as with no translation the estimated screw axis will be the rotation axis. However the rotation estimation algorithm is much faster as it takes two point pairs at each iteration in the evidence gathering process, and hence the number of evidence gathering iterations is in the order of $O(N^4)$, Equation 6.10. While it is in order of $O(N^6)$ for the general motion estimation algorithm, Equation 6.27. The voting process is the same for both algorithms, and is in the order of $O(L^2)$, where L is the number of estimated axis, Equation 6.14.

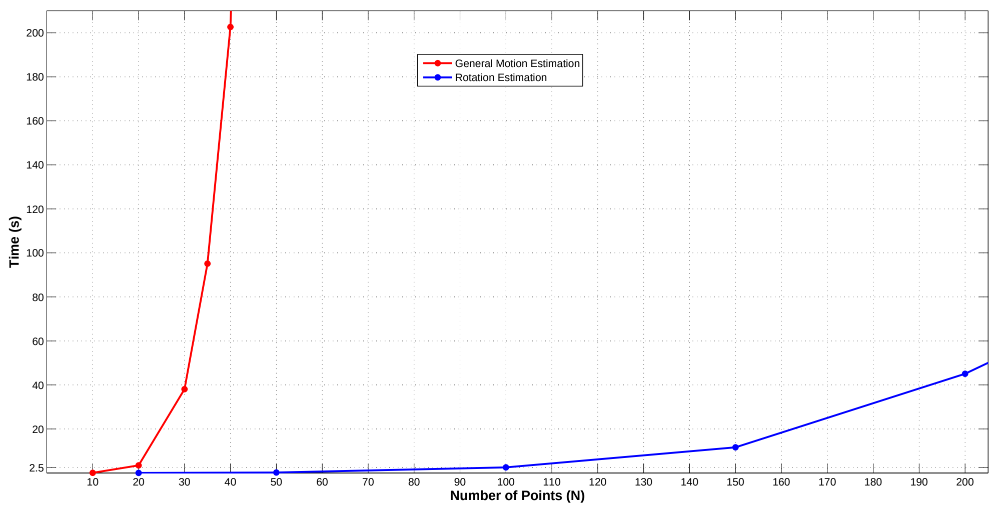


Figure 6.18: Time performance of the evidence gathering process in the rotation estimation algorithm and the general motion estimation algorithm.

For example, taking two point clouds of 20 points each transformed by a rotation, the number of iterations K in evidence gathering step of rotation estimation algorithm will be 7.2×10^4, while it will be $K = 2.3 \times 10^7$ for the general motion estimation algorithm.

To give an example of the processing speed difference, an actual implementation of both algorithms is done on an object of 20 points, rotating about an axis in 3D space. The implementation, done with the same parameter values for both algorithms, showed that the total time of the rotation estimation algorithm is (0.03) seconds, while it is (90.10) seconds for the general motion estimation algorithm, which is 3000 times slower.

Figure 6.18 shows the time performance with increasing number of points for the evidence gathering processes in both the rotation estimation algorithm and the general motion estimation algorithm. The figure is generated by implementing Algorithm 4 and Algorithm 6 on the point cloud pair of Example 1, Figure 6.8. Both algorithms were tuned to iterate only for a certain number of points (N) according to the X axis of the figure.

Figure 6.19 shows the time performance of the voting process, which is the same in both rotation estimation and general motion algorithms. The figure is generated by implementing Algorithm 5 on the axes generated by implementing Algorithm 6 on the point cloud pair of Example 1. The algorithm was tuned to iterate only for a certain number of axes (L) according to the X axis of the figure.

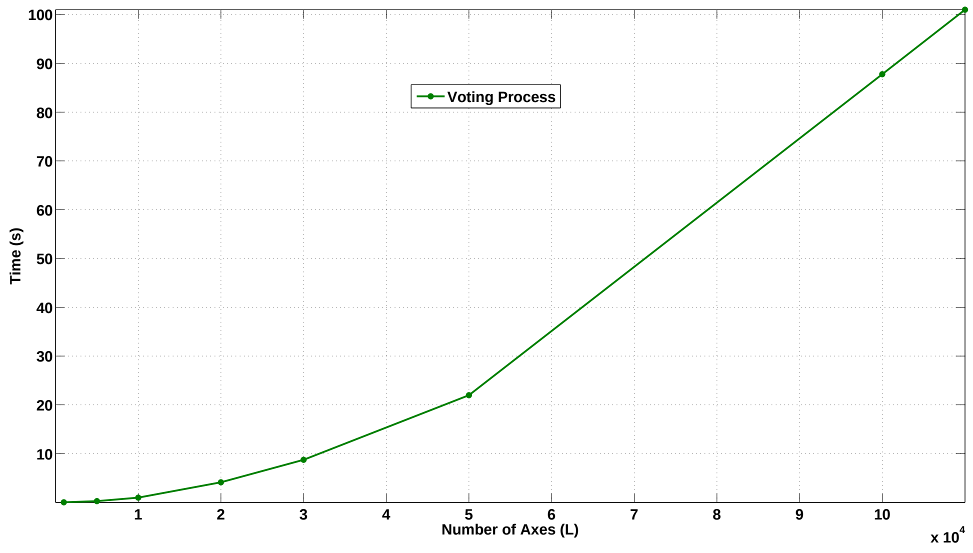


Figure 6.19: Time performance of the voting process in both algorithms.

6.6.2 Subsampling

In Examples 3 and 4 of Section 6.5, we applied the proposed algorithms on point clouds after subsampling. Figure 6.20 shows a real point cloud rotated 45 degrees around an axis in 3D. Note that this is the same point cloud and the rotation is done synthetically. Similar to Examples 3 and 4, the same steps are applied to obtain an estimate for the axis of rotation.

Figure 6.21 shows the estimated rotation axis and the point correspondences. The figure also shows the points of each point cloud after subsampling (in red); the original point cloud has a number of points ($N = 8705$) points, after subsampling this number is reduced to 122 points. The estimated rotation angle θ using Equation 6.16 is 47 degrees. The reason for this slight error in the estimation of rotation angle is due to subsampling.

Subsampling changes the surface of the point cloud; even though our proposed method gives better representation of the surface as it retains original points, the remaining points after subsampling on one point cloud are not guaranteed to perfectly correspond to the subsampled points on the other point cloud. And hence even if the subsampled point pair associated with the correctly estimated rotation axis correspond, it does not imply that the original points (before sampling) are perfect correspondences.

Logically, the finer the subsampling, i.e the higher point cloud resolution, the remaining points will be closer to the original point and the more accurate the estimation is.

To quantify the effect of subsampling on the results of the proposed algorithms, we apply the algorithms on subsampled point clouds at increasing voxel sizes. We use the proposed subsampling method explained in Subsection 2.5.2 applied on the point clouds

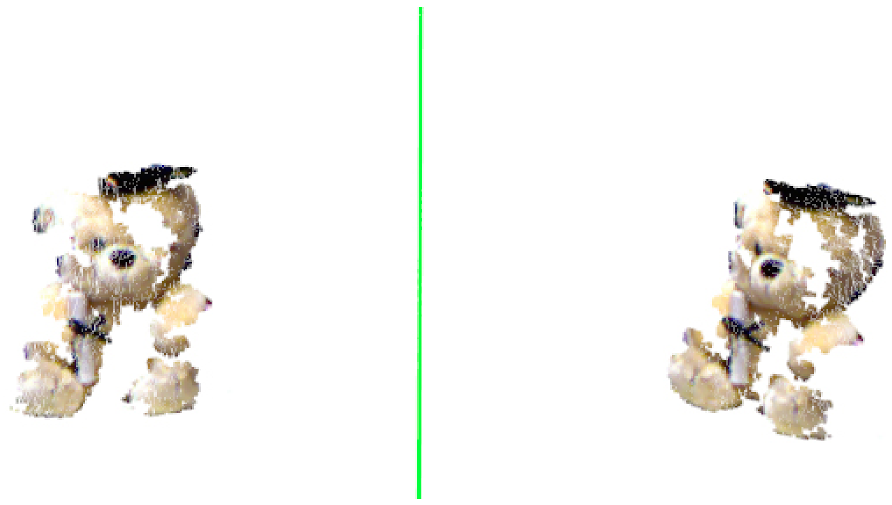


Figure 6.20: Real point cloud rotated around an axis in 3D.

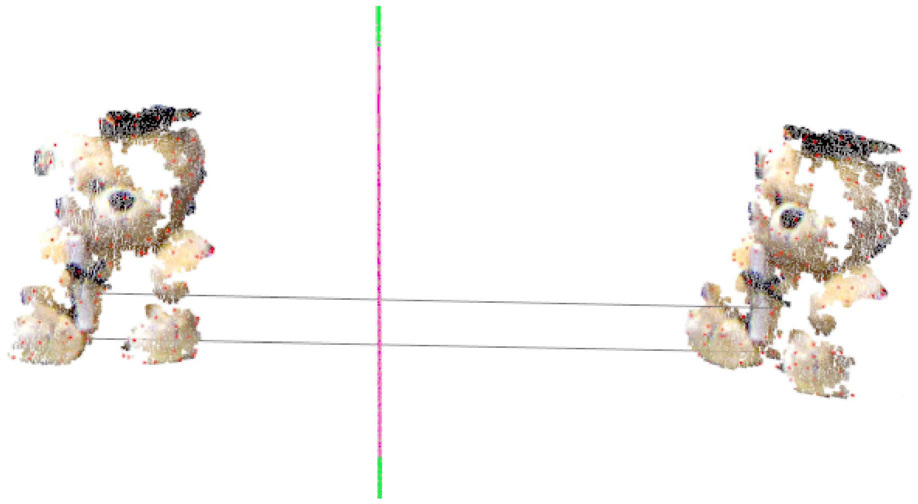


Figure 6.21: Estimated rotation axis after subsampling.

of Examples 3 and 4. Figures 6.22 and 6.25 show the positional offset which is the Euclidean distance between estimated axis and the ground truth axis, Equation 6.31. Figures 6.23 and 6.25 show the directional offset, Equation 6.32. And Figures 6.24 and 6.27 show the angle offset which is the absolute difference between the estimated angle of rotation and the ground truth angle. The three offsets are computed 10 times at each standard deviation step.

Figures 6.22 to 6.27 show how the performance of the algorithms degrade with increasing voxel size. This is because the larger the voxel size used in subsampling, the larger the volume that will contain surface points that will be approximated to one point. As more points are approximated to one point, less points are included in the evidence gathering process. Also, the larger the voxel size is, the larger the Euclidean distance between the subsampled point and the original surface points. Hence the axis estimation becomes less accurate with increasing voxel size.

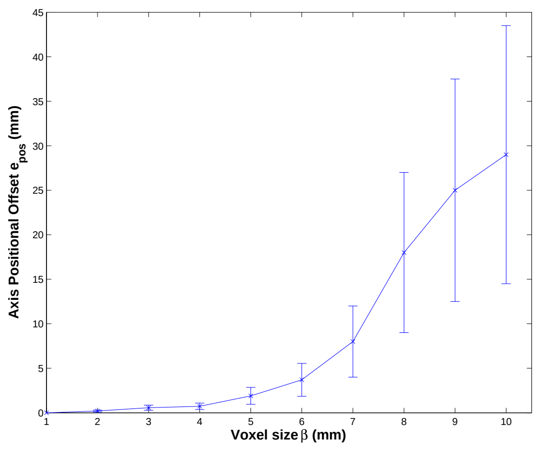


Figure 6.22: Positional offset of the rotation estimation algorithm against increasing voxel size.

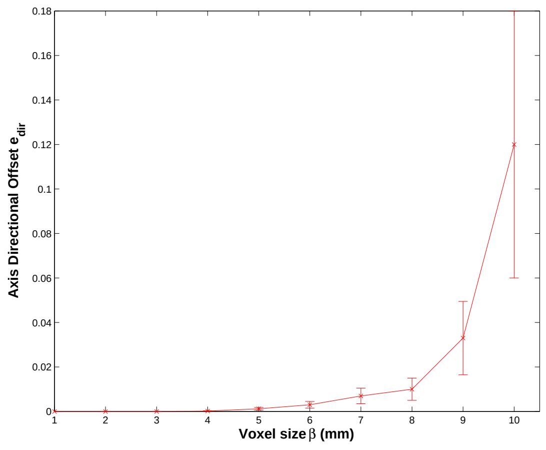


Figure 6.23: Directional offset of the rotation estimation algorithm against increasing voxel size.

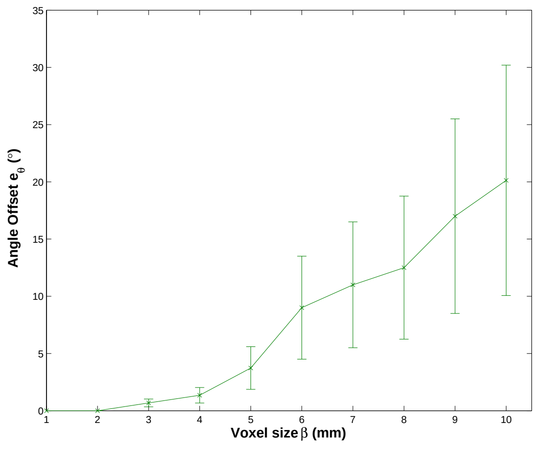


Figure 6.24: Angle offset of the rotation estimation algorithm against increasing voxel size.

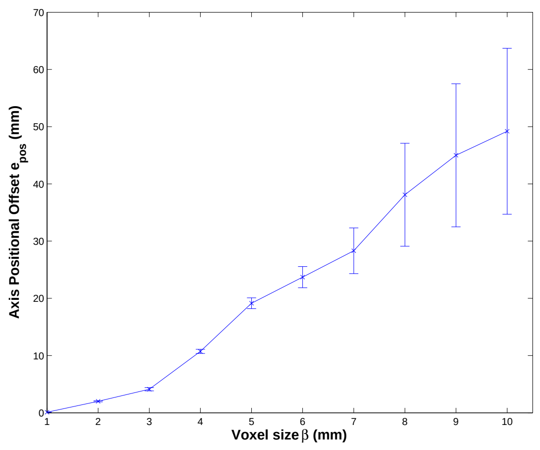


Figure 6.25: Positional offset of the general motion estimation algorithm against increasing voxel size.

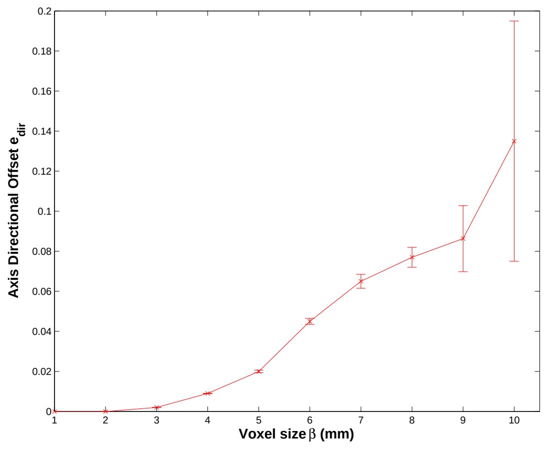


Figure 6.26: Directional offset of the general motion estimation algorithm against increasing voxel size.

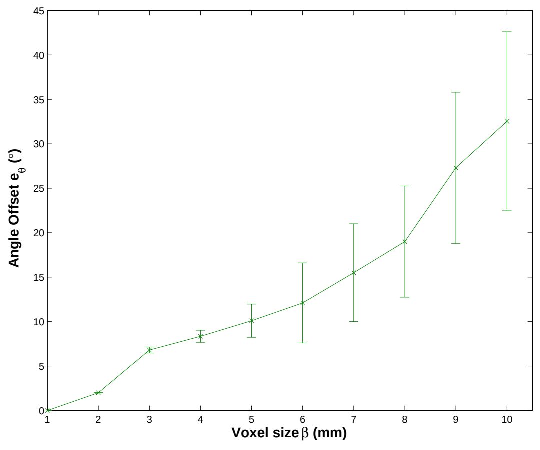


Figure 6.27: Angle offset of the general motion estimation algorithm against increasing voxel size.

Our experimental analysis shows that the estimation of the position and direction of the rotation axis or the screw axis is more robust to changes in the point cloud resolution than the estimation of the rotation angle. The sensitivity of computation of the rotation angle comes from the fact that it is computed from only two points, that are associated with the estimated rotation/screw axis, which might not be exact correspondences due to subsampling. However the estimated axis is more robust to resolution changes as it is computed by the evidence gathering process. One way to obtain better estimates for the angle of rotation, in case of rotation only case, is by using Algorithm 1 explained in Chapter 3.

6.6.3 Noise

Figure 6.28 shows two point clouds of the same scene taken at different times. As explained in Section 2.3, two 3D data sets are unlikely to be acquired such that points exactly correspond; in other words, real point clouds will always have an amount of noise. Figure 6.29 shows the different points on the surfaces of the same point cloud captured at different times.

Moreover, since subsampling changes the surface of the point cloud, it can be considered as a form of noise. To quantify the accuracy of the proposed algorithms, we conduct an implementation on point clouds with increasing amount of noise. The noise is added to the point clouds shown in Figure 6.8 by a Gaussian distribution model given by Equation 3.15, explained in detail in Subsection 3.6.1. The effect of added noise is shown in Appendix A.

Figures 6.30 to 6.35 show that the errors in estimating the rotation and screw axes increase as the noise level increases. Similar to all the algorithms presented in this research, when the level of noise increases, the response of the voting process to noise will also increase until a point is reached where the response due to noise dominates the response due to the presence of a cluster of axes. Once this point is reached, the index of the accumulator peak essentially become random, and hence estimates of the rotation/screw axis parameters cease to be of value.

The added noise will change the coordinates of the points on the point clouds; the larger the level of noise, the further the points will be from their original coordinates. This will have a direct effect on the position and direction of the constructed rotation/screw axis.

With the presence of noise, the constructed axes from true correspondences will not be as densely clustered, and also their directions will vary. In other words the computed axes will not be as similar as axes constructed without noise. Axes that are not similar in terms of position and direction will not cast votes in the same bin, according to the thresholds defined by Equations 6.12 and 6.13. The votes will be casted in bins around

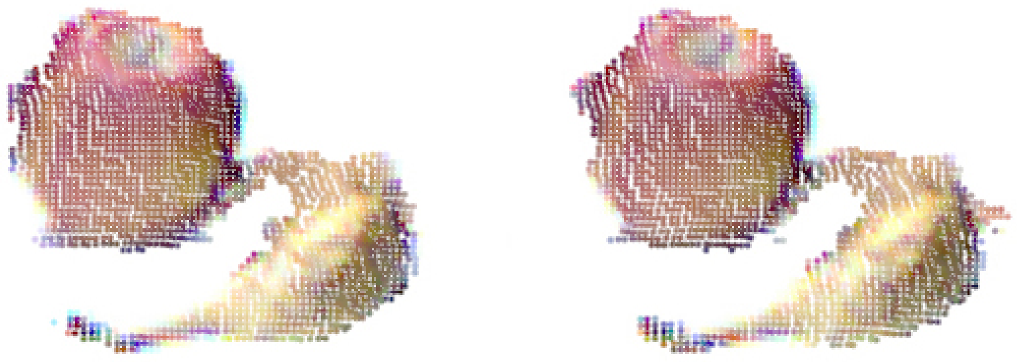


Figure 6.28: Two point clouds of the same scene.

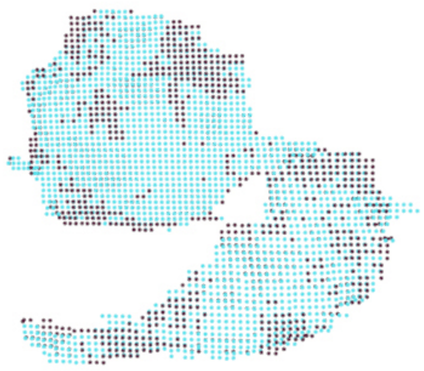


Figure 6.29: Difference of surface points for clouds in Figure 6.28.

the bin corresponding to the true axis, and hence the “correct” bin will not have a peak as high as it would be without noise. The larger the amount of noise, the further the votes will be from the correct bin in the accumulator, and hence the larger the errors in the estimated axis parameters.

Changes in point coordinates due to noise will also cause an error in computing the rotation angle θ by Equations 6.16 and 6.23.

It can be seen from Figures 6.30 to 6.35 that both algorithms degrade gracefully as the amount of added noise increases. However, the general motion estimation algorithm is more sensitive to noise. This is due to the fact that it requires three pairs at each iteration, instead of two as in the rotation estimation algorithm, to compute the axis. The extra pair of points includes more noise in the computation process and hence increases the error in the axis estimation result.

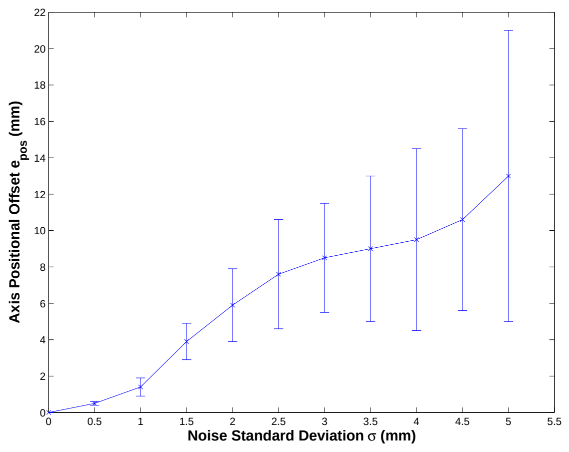


Figure 6.30: Noise performance of rotation estimation algorithm (position offset).

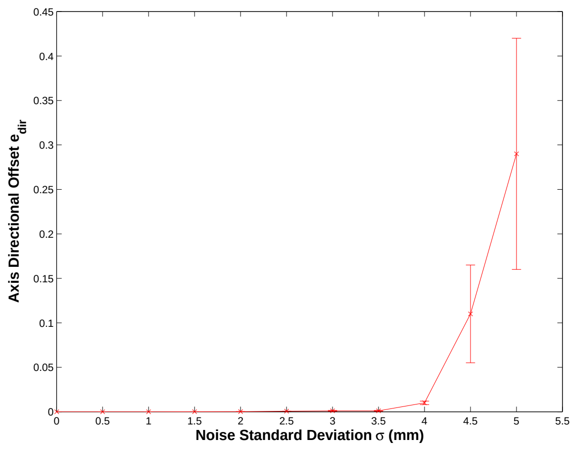


Figure 6.31: Noise performance of rotation estimation algorithm (direction offset).

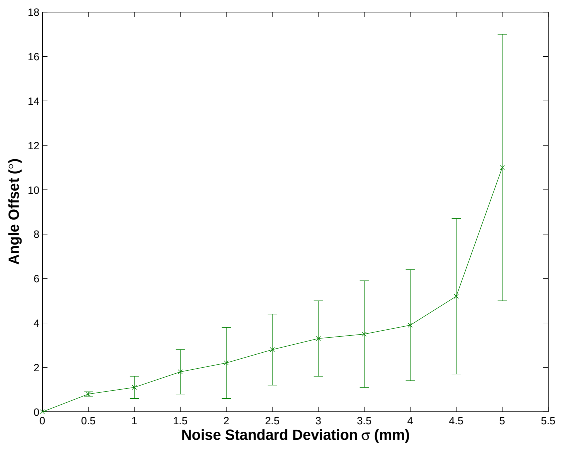


Figure 6.32: Noise performance of rotation estimation algorithm (angle offset).

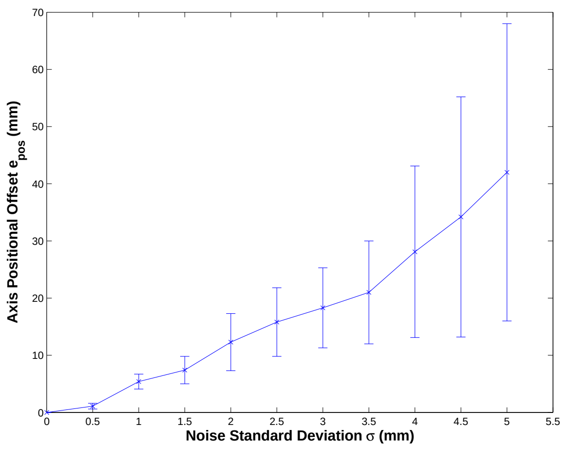


Figure 6.33: Noise performance of general motion estimation algorithm (position offset).

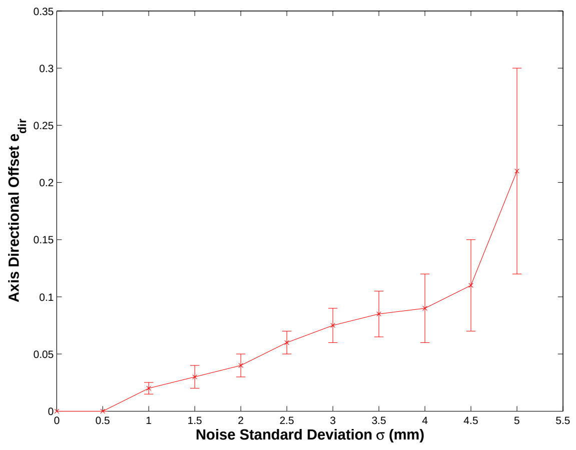


Figure 6.34: Noise performance of general motion estimation algorithm (direction offset).

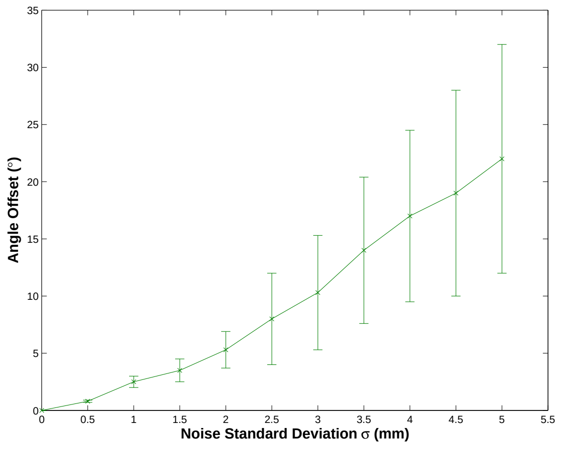


Figure 6.35: Noise performance of general motion estimation algorithm (angle offset).

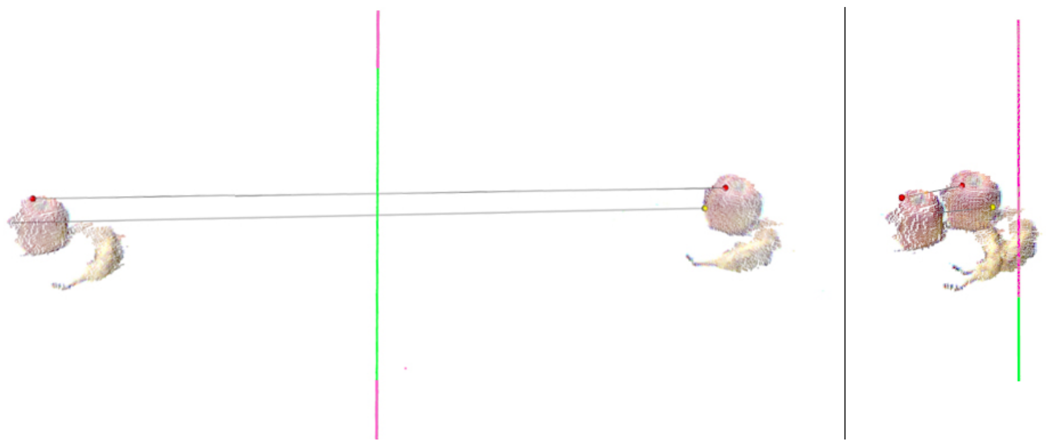


Figure 6.36: Rotation around two different axes. Estimated rotation axis (purple) coincides with ground truth axis (green).

PC	dms (mm^3)	Pts	S.Pts	$e_{pos}(mm)$	e_{dir}	$e_\theta(^\circ)$	Time(s)
Teddy	$172 \times 246 \times 165$	8705	122	0.0	0.04	2.0	102.45
Fruits 1	$175 \times 64 \times 217$	2094	51	0.03	0.0	2.15	71.06
Fruits 2	$175 \times 64 \times 217$	2094	51	0.01	0.01	4.5	72.5
Pitcher	$90 \times 107 \times 114$	5591	74	2.99	0.0	8.4	97.45
Banana	$200 \times 80 \times 60$	5647	60	2.66	0.0	2.0	68.33
Mug	$86 \times 93 \times 98$	5224	60	7.45	0.04	13.2	68.43

Table 6.8: Performance of rotation estimation algorithm implemented on real point clouds. Columns left to right: point cloud name, dimensions of point cloud, number of points, number of points after subsampling, Euclidean distance offset from ground truth rotation axis, directional offset (Equation 6.32), estimated angle error, total processing time.

6.7 Results

6.7.1 Rotation Estimation

Figure 6.36 shows the result of implementing the rotation estimation algorithm when the point cloud is rotated around two different axes. The accurate performance of the algorithm can be be seen from the figures as the rotation axis is estimated with minimal positional error and also a small error in estimating the angle of rotation. Table 6.8 shows the performance of the algorithm implemented on pairs of rotated point clouds.

Figure 6.37 shows point clouds pairs of point clouds rotated on a turntable from the RGBD database of [136], and the result of implementing our algorithm. Note that the ground truth rotation axes for these sets is measured manually, and hence is subject to an error. Nevertheless, the algorithm still provides an accurate performance, Table 6.8. Figure 6.38 shows an example of applying the estimated rotation parameters.

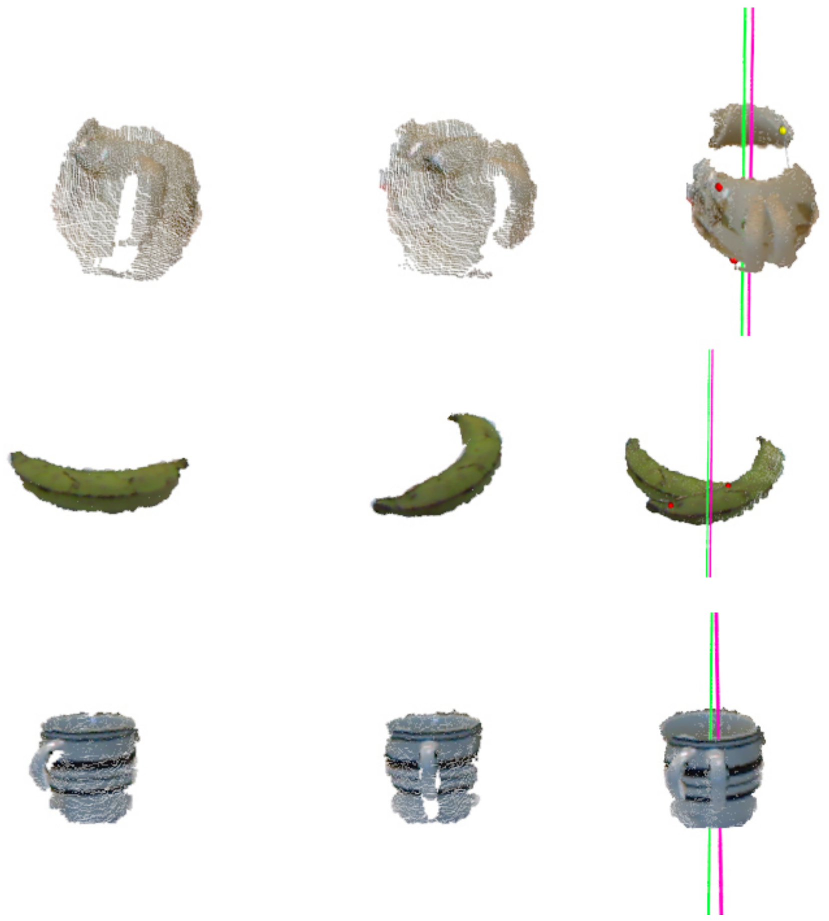


Figure 6.37: Results of implementing rotation estimation algorithm on different point clouds. At each row, the left and middle images show two point clouds taken from the turntable sequence of [136]. Right image shows the estimated axis of rotation (purple) and the ground truth axis (green). Point clouds are from [136].

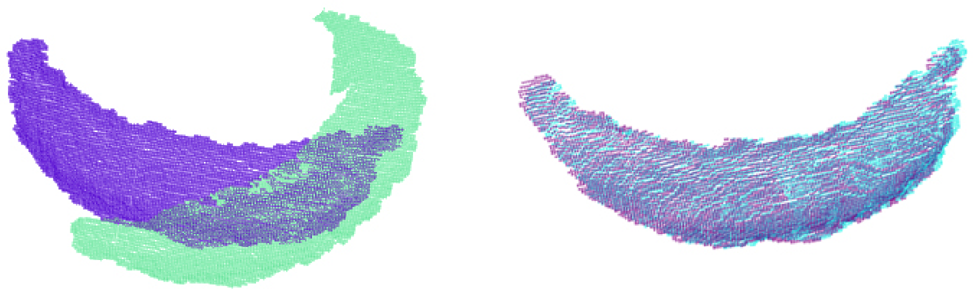


Figure 6.38: Input point clouds before and after applying rotation estimation algorithm.

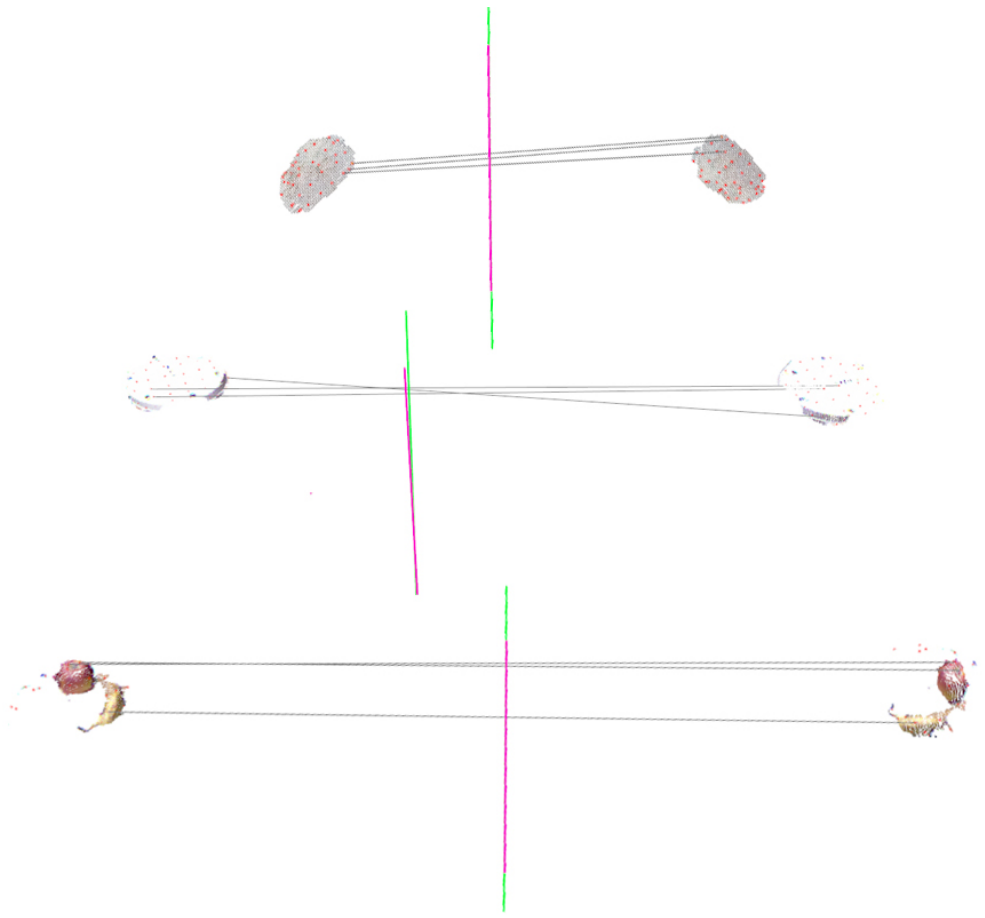


Figure 6.39: Results of implementing general motion estimation algorithm on different point clouds. Garlic point clouds are from [136].

6.7.2 General Motion Estimation

Figure 6.39 shows the results of implementing the general motion estimation algorithm and point clouds rotated and translated in 3D space. Once the screw axis, the rotation angle and the linear displacement are estimated, Equations 6.28 and 6.29 are applied to obtain the homogeneous transformation parameters. Figure 6.40 shows an example of applying the estimated parameters. Details of the implementation are in Table 6.9.

PC	dms (mm^3)	Pts	S.Pts	$e_{pos}(mm)$	e_{dir}	$e_\theta(^\circ)$	$D(mm)$	Time(s)
Garlic	$47 \times 22 \times 28$	880	15	0.04	0.0	0.01	5.3	14.87
Disc	$231 \times 29 \times 197$	4802	35	2.68	0.03	2.53	1.9	57.76
Fruits	$175 \times 64 \times 217$	2094	20	0.01	0.0	0.65	0.1	34.56

Table 6.9: Performance of general motion estimation algorithm implemented on real point clouds. D refers to the difference between the ground truth linear displacement along the screw axes and the measured displacement along estimated axis.

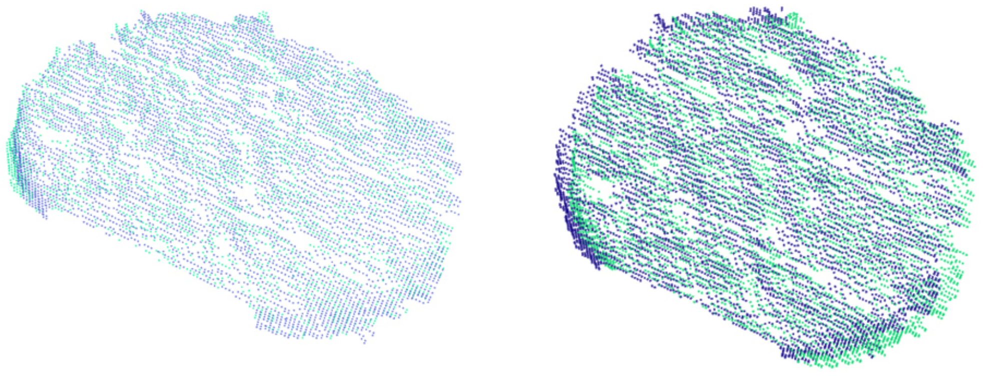


Figure 6.40: Example of general motion parameters estimation. Ground truth transformation (left), and estimated transformation using the general motion estimation algorithm.

6.8 Discussion

The main contribution of the two algorithms presented in this chapter is that they provide the capability of 3D motion estimation, without relying on keypoint detection and feature descriptor computation and matching. According to [53] there are a variety of criteria that one can use for classifying points as being similar or interesting or key points, such as colour, local surface normals, local curvature shape, edgeness, texture, etc. All such features are to varying degrees sensitive to noise and other conditions such as occlusion and thus, the extraction of such features is also a challenging task.

The ability to use data points directly and bypassing feature descriptor computation and matching has practical applications for 3D reconstruction, registration and object recognition when detecting keypoints and feature description and matching is not informative or challenging. Such scenarios are usually present when objects to be matched have smooth surfaces such that keypoints and their associated features are not descriptive, or in the case of having few points on the object's surfaces (sparse point clouds).

The algorithms presented have the advantage of computing the motion between 2.5D frames regardless of the quality of the object's surface. Unlike keypoint-feature based approaches which depend on the local geometry around the detected key point. This local geometry must contain sufficient descriptive information that uniquely characterizes that keypoint. Also, keypoints depend on particular definitions of interestingness, saliency and repeatability. These definitions vary with the type of the keypoint and the feature descriptor used whereas in the new algorithms these definitions are not required.

The representation of motion as a 3D axis gives the presented algorithms the advantage of having a one dimensional accumulator space, which enables voting for motion parameters more efficiently than other methods that use up to 7-dimensional accumulator spaces for voting, Table 6.1. Having a one-dimensional accumulator space significantly reduces the complexity of allocating and searching the high dimensional accumulator spaces.

In the presented algorithms, the voting is not for the actual motion parameters, it is actually for the axis representing the motion. The computational advantage for the presented algorithms comes from the fact that their accumulator space is defined only by the number of points of the object; as opposed to other methods, such as those in Table 6.1, which their accumulator spaces are defined by the resolution (bin size) and ranges of motion and rotation parameters, in addition to the number of points.

For example, the method presented by [39] uses two accumulator spaces, one to vote for the translation parameters and another for the rotation parameters, each of the two spaces is defined by an accumulator resolution, and hence the parameters will be quantised which reduces the accuracy. Also, each of the spaces is defined by a parameter range; if the peak does not exist within that range then the accumulator space is defined incorrectly and must to be redefined.

In contrast, in the presented algorithms the voting space is defined by the number of points and constrained by the axes similarity thresholds (t_{dir} and t_{pos}). By voting for axes, rather than motion parameters, there will be no need to define a range for motion or rotation parameters, as we are looking for the largest cluster of similar axes rather than any parameter values. Additionally, there is no quantisation of any parameters. The axes similarity thresholds (t_{dir} and t_{pos}) constrain the size and the accuracy of the accumulator space, which in other methods are constrained by the resolution and the range of the accumulator.

We showed in previous chapters how the accumulator resolution affects the accuracy of the estimated parameters of the voting process; since the pretested algorithms do not require any quantisation of the parameters, they can provide more accurate results.

Moreover, the application of the axes similarity thresholds (t_{dir}, t_{pos}) discards a large number of axes from the voting process; only axes passing the two thresholds cast votes in the 1D space. However, other methods will vote for motion parameters generated from *all* points (or features) in a high dimensional space. So not only the accumulator space dimensionality of the presented algorithm is smaller, but also the size of the accumulator itself, i.e. number of votes, is smaller. In other words, using axes to vote for motion parameters makes the accumulator array “dynamic” rather than a “large” predefined high-dimensional array. Hence the presented algorithms will have improved memory requirements.

The two algorithms are related as the general motion estimation algorithm can be considered as a generalisation of the rotation estimation algorithm; as with no translation, the screw axis and the rotation axis are identical. However the rotation estimation algorithm is more efficient in the case of rotation only as it performs the evidence gathering process in less time.

As discussed in Section 2.2, the pipeline for feature based methods is to estimate correspondences from matched features and the compute the transformation from these correspondences. Our approach bypasses the feature matching step and estimates the transformation first, and then correspondences can be computed from the estimated transformation.

The two algorithms are presented as a complete "stand alone" solution for 3D motion estimation and surface matching, however they also can be employed within a typical feature-based surface matching pipeline in the correspondence grouping and hypothesis verification steps. Applying our algorithms on estimated correspondences guarantees that they are consistent with estimated transformation.

In Section 2.3 we described the most significant issues driving the different research strands in the area of surface matching and motion estimation. Here we summarise how the developed algorithms handle such issues.

1. Initial registration and range of convergence. Unlike many algorithms, the proposed algorithms do not require any initial relative pose estimate for optimal registration, which is useful in many practical scenarios when no pose priors are available.
2. Inexact correspondences. This issue still exists in the proposed algorithms. As in other methods, the registration errors are a function of inter point-distances and subsequently the resolution of the data. However the proposed algorithms require only few "good" correspondences compared with other algorithms which require a large number of correct correspondences to converge correctly.
3. Outliers/partial overlap. The proposed algorithms identify points which do not have any correspondence between the data sets implicitly as these points will generate axes that are randomly distributed in space and thus ignored by the voting process. Only points with correct correspondences will generate axes that are closely aligned.
4. Pose versus correspondence search. Since the proposed algorithms do not use features, it performs the search for correct alignment in the pose space, similar to other methods as described in Subsection 6.2.2. However our algorithms perform the pose search (voting) in a one-dimensional space rather than multi-dimensional spaces.
5. Registerable feature type. This issue does not exist in the proposed algorithms simply because the algorithms simply use the 3D coordinates of points and do not compute any type of features. This is a significant advantage of the proposed algorithm as the challenging task of computing features and matching them based on different criteria is bypassed.

6. Performance acceleration. Since no features are computed, there will be no feature space to be searched for matching features, the only space search is performed in a one-dimensional accumulator space for the axis representing transformation.

A limitation to the proposed algorithms is that in the case of translation only a transformation can not be estimated. Also, in some cases, the axis with maximum number of votes might not be unique, or the true axis might not have the maximum number of votes. These cases are often related to noise and limited overlap between point clouds. A possible solution to this problem is a more sophisticated voting algorithm, discussed more in our further work, Chapter 7.

6.8.1 Optimisation

The voting algorithm (Algorithm 5), described in Subsection 6.3.2.3, has a time complexity of $O(L^2)$, where L is the number of axis. The algorithm can be further optimised by decomposing it into two stages. The first stage votes for the position of the axes using a recursive divide-and-conquer based algorithm [137], which solves the problem of finding the nearest axes as a closest pair of points problem. The time complexity of of the recursive divide and conquer algorithm is $O(L \log L)$ and a space complexity of $O(L)$. The axes that pass the distance threshold from the first stage go to the second stage, which also uses a modified divide-and-conquer based approach to vote for axes with the same direction. The time complexity will also be of $O(L \log L)$, and a space complexity of $O(L)$.

Having a two-stage voting algorithm, with a time complexity of $O(L \log L)$ and a space complexity of $O(L)$, would give a more optimised performance than the initial $O(L^2)$ voting algorithm. This will be focused on in our future work.

Additionally, the accuracy of the estimated parameters could be refined by introducing a least-squares minimisation process after the voting stage. In both algorithms, the parameters of motion (rotation and general motion) are estimated from the axis with most votes in addition to one pair of associated points as explained in Subsections 6.4.2.1 and 6.3.2.4. A least-squared minimisation method can be applied on the two/three associated point pairs of the rotation/screw axis to increase the certainty of the estimated parameters. Moreover, instead of taking one axis with most votes, a least-squared minimisation method can be applied to N axes with most votes. This will also be focused on our future work.

6.9 Conclusions

In this chapter we introduced two evidence gathering algorithms for motion estimation based on kinematic theorems; the first algorithm estimates the motion in the case of rotation around a fixed axis only. The second algorithm estimates general motion consisting of rotation as well as translation in 3D space. To the best of our knowledge, this is the first research to use the kinematics theorems in an evidence gathering framework for motion estimation and surface matching without the use of any given correspondences and also without employing feature description and matching.

Both algorithms are based on the observation that there is one and only one invariant axis that synthesises the rigid body transformation in 3D, and employ an evidence gathering technique in a Hough-voting-like approach to estimate this axis.

The capability to estimate the 3D transformation without relaying on feature matching and estimating correspondences between 3D datasets is desirable and has many practical applications, especially in scenarios where objects to be matched have smooth surfaces such that keypoints and their associated features are not descriptive, or in the case of having few points on the objects surfaces (sparse point clouds).

We discussed similar related work, in terms of motion estimation using geometrical analysis and Hough voting, and showed that how voting for motion parameters can be achieved in only one-dimensional space instead of a multi-dimensional one, which is more efficient. We described both algorithms in detail and established the mathematical foundations necessary to understand their methodology. We provided experimental analysis and evaluation and showed the results of implementing the algorithms on real point clouds. We discussed the properties and the solutions our algorithms give to common issues driving the different research strands in the area of surface matching and motion estimation. Finally, we discussed how the proposed algorithms can be further optimised for improvement of processing time and accuracy.

Chapter 7

Conclusions and Further Work

7.1 Conclusions

The significant improvement in 3D sensing technologies lead towards the development of numerous algorithms designed specifically to solve 3D perception problems such as recognising objects and registering 3D surfaces.

Unlike methods based on feature computation and matching, methods that are based on evidence gathering have not been popular for solving 3D perception problems.

The dependency on 3D feature description and computation can be a significant limitation to many 3D perception tasks; the fact that there are a variety of criteria used to describe 3D features, such as surface normals and curvature, makes feature-based approaches sensitive to noise and occlusion. In many cases, such as smooth surfaces, computation of feature descriptors can be non informative. Moreover, the process of computing and matching features requires more computational overhead than using points directly.

On the other hand, evidence gathering approaches, which use data directly, have proved to provide robust performance against noise and occlusion. More importantly, evidence gathering approaches do not require initialisation or training, and avoid the need to solve the correspondence problem.

In this thesis we presented evidence gathering algorithms for 3D reconstruction of rotation objects, detection and extraction of static and moving 3D parametric objects and 3D motion parameter estimation.

Chapter 2 of this thesis provided a detailed overview of both local and global 3D motion estimation and surface matching methods; we discussed the procedural pipelines of both categories, explained related concepts, and listed state of the art methods and algorithms and how they are employed to estimate the motion between 3D datasets. We also discussed the main issues driving the different research strands in the area of surface matching and motion estimation. Additionally, we listed the most recent advancements in available 3D sensing technology and described the properties of the output data these sensors provide. We also discussed properties of point clouds and subsampling methods, introducing a variation to current techniques that provides a better representation of the original subsampled point cloud.

In Chapter 3 we presented an algorithm for 3D reconstruction of point clouds that is based on motion-compensated temporal accumulation. This algorithm accumulates surface information over the sequence after applying a series of rigid transformations on the point clouds of the input sequence, and then resamples the accumulated point clouds based on a computed space partitioning and votes for the best reconstruction based on the number of points in each resampled point cloud. We verified the theory of the algorithm, provided experimental analysis, showed experimental results on synthetic and real data which show that successful reconstruction can be achieved.

In Chapter 4 we introduced a fast and robust Hough Transform (HT) based algorithm for detecting spherical structures in 3D point clouds. To our knowledge, our algorithm is the first HT based implementation that detects spherical structures in typical in 3D point clouds generated by consumer depth sensors. Our approach has been designed to be computationally efficient; reducing an established limitation of HT based approaches. We provided experimental analysis of the achieved results, and we showed robust performance for detecting spheres.

In Chapter 5 we presented the three-Dimensional Velocity Hough Transform (3DVHT). We demonstrated the significant advantage of the 3DVHT which is the ability to predict fully occluded spheres at any frame of the sequence. The superior performance is attributed to the concept of incorporating motion in the voting process, accumulating evidence in a single accumulator space for the whole sequence, resulting for the spherical structural information in each cloud to be integrated over the complete sequence. This enables motion and structural parameters to be accurately determined, even in the case of a full occlusion.

In Chapter 6 we introduced two evidence gathering algorithms for motion estimation based on kinematic theorems; the first algorithm employs Reuleaux's method to estimate the motion in the case of rotation around a fixed axis only. The second algorithm employs Chasles' method to estimate the general 3D motion of an object consisting of rotation as well as translation in 3D space. To the best of our knowledge, this is the first research to use the kinematics theorems in an evidence gathering framework for motion estimation and surface matching without the use of any given correspondences and also without employing feature description and matching. We provided experimental analysis and evaluation, showed the results of implementing the algorithms on real point clouds, and discussed the properties and the solutions our algorithms give to common issues driving the different research strands in the area of surface matching and motion estimation.

The capability to estimate the 3D transformation without relying on feature matching and estimating correspondences between 3D datasets is desirable and has many practical applications, especially in scenarios where objects to be matched have smooth surfaces such that keypoints and their associated features are not descriptive, or in the case of having few points on the objects surfaces. We provided alternative means to commonly used feature based approaches by introducing employment of evidence gathering based methods to the processing of 3D range data.

7.2 Further Work

For future work, we will concentrate on the following:

- An advanced method for accumulator space peak detection. The algorithms for 3DHT for sphere detection and 3DVHT determine the correct parameters of the static and moving sphere by a search operation on the 4D and 7D sparse matrix elements, that finds the element with the maximum number of votes. A more sophisticated search operation, that can detect multiple peaks, extends the practicality of the algorithm to detect multiple spheres, and also will give a better understanding of the voting process, which can contribute to improve the performance of the algorithms in terms of computing time and robustness. Moreover, we aim to investigate the application of post-voting accumulator smoothing methods, mentioned in Subsection 4.4.1 to improve the accuracy of peak detection.

- Detection and extraction of other 3D geometric shapes. Similar algorithms can be developed to detect 3D lines, planes, cylinders, cones and other 3D objects that can be parametrised.

- Extraction of 3D parametric shapes moving in arbitrary fashion. So far the 3DVHT assumes the nature of the motion is known, linear for example, which can be easily parametrised and hence included in the evidence gathering process. Any type of motion that can be parametrised, free-fall for example, can be adapted by the 3DVHT to detect moving objects. However if the motion is arbitrary or too complex, i.e. including large number of parameters, it will be not possible to detect the object by the 3DVHT. One solution to this problem is by using motion templates, such as in [72]. Motion templates require a priori knowledge concerning the target object's path before analysis can begin, with these templates it is no longer necessary to accumulate for the parameters describing the motion since they are already known.

- Extending 3DVHT to detected arbitrary 3D objects moving with linear velocity. As previously mentioned there have been also methods which detect arbitrary objects in 3D point clouds, provided a model of the object, such as in [34], [138] and [139]. Extending such methods to include motion parameters will allow the detection of fully occluded objects which these methods cannot achieve.

- A sophisticated voting algorithm for axis estimation. The current voting process in both rotation and general motion estimation algorithms takes the axis with most votes which represents the axis with the most consensus. As explained earlier, in some scenarios the axis with maximum number of votes is not unique or does not refer to the true axis. This can be possible by application of least-squares methods on the estimated axes to improve accuracy and robustness, as explained in Subsection 6.8.1.

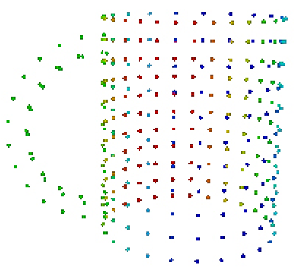
- Optimisation of axes voting algorithm. As explained in Subsection 6.8.1, the current voting algorithm is of time complexity of $O(L^2)$. Dividing the algorithm into two stages of $O(L \log L)$ will improve the computation time.

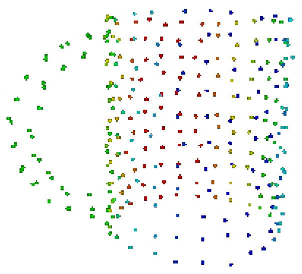
- A voting algorithm for rotation angle estimation. In both algorithms presented in Chapter 6, the angle of rotation is estimated from the points indexed with the axis of the most votes; a more accurate computation of the rotation angle can be achieved by implementing a voting algorithm with a separate accumulator space.

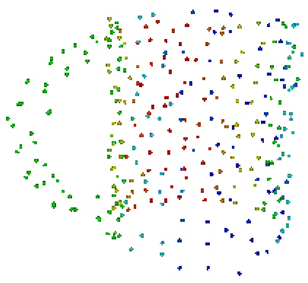
- Parallel implementation of developed algorithms on graphics hardware. The results shown in this thesis were all obtained from implementing the presented algorithms on the CPU. The parallel architecture of graphics hardware can significantly improve the iterative voting algorithms in terms time as well as accuracy performance.

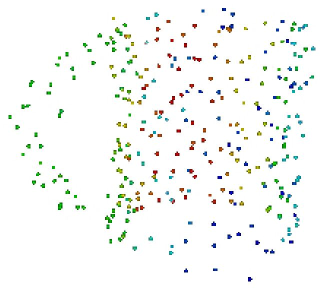
Appendix A

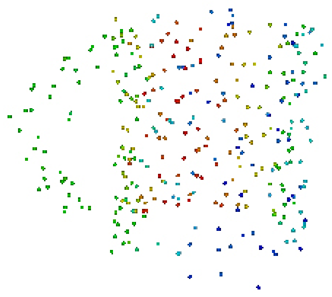
Effect of Added Gaussian Noise

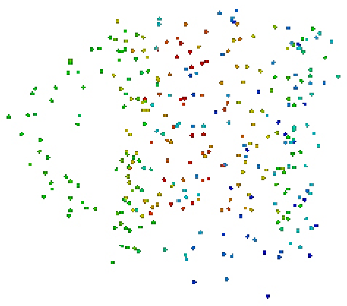

σ = 0mm

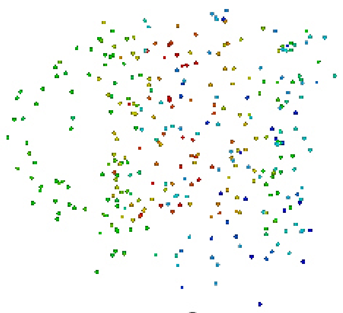

σ = 1mm

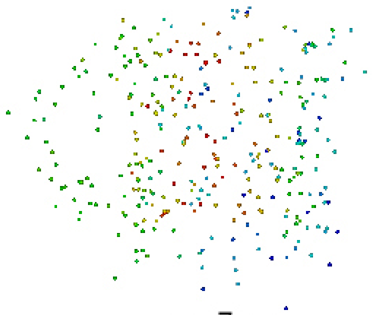

σ = 2mm

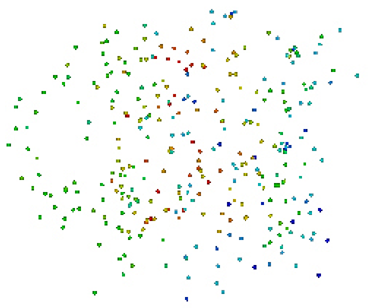

σ = 3mm

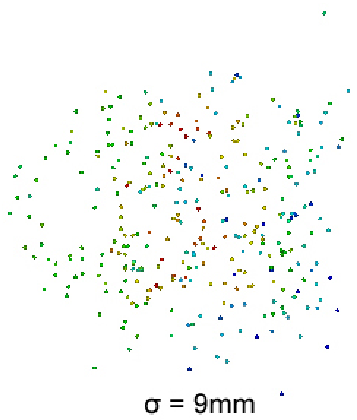

σ = 4mm

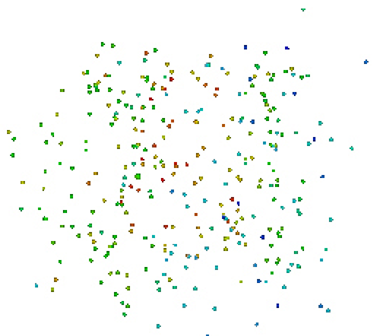

σ = 5mm

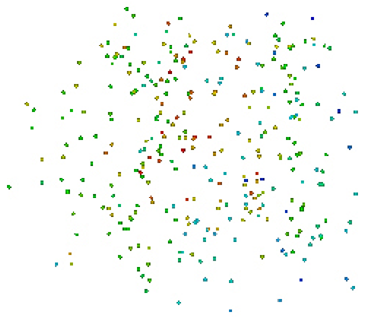

σ = 6mm


σ = 7mm


σ = 8mm

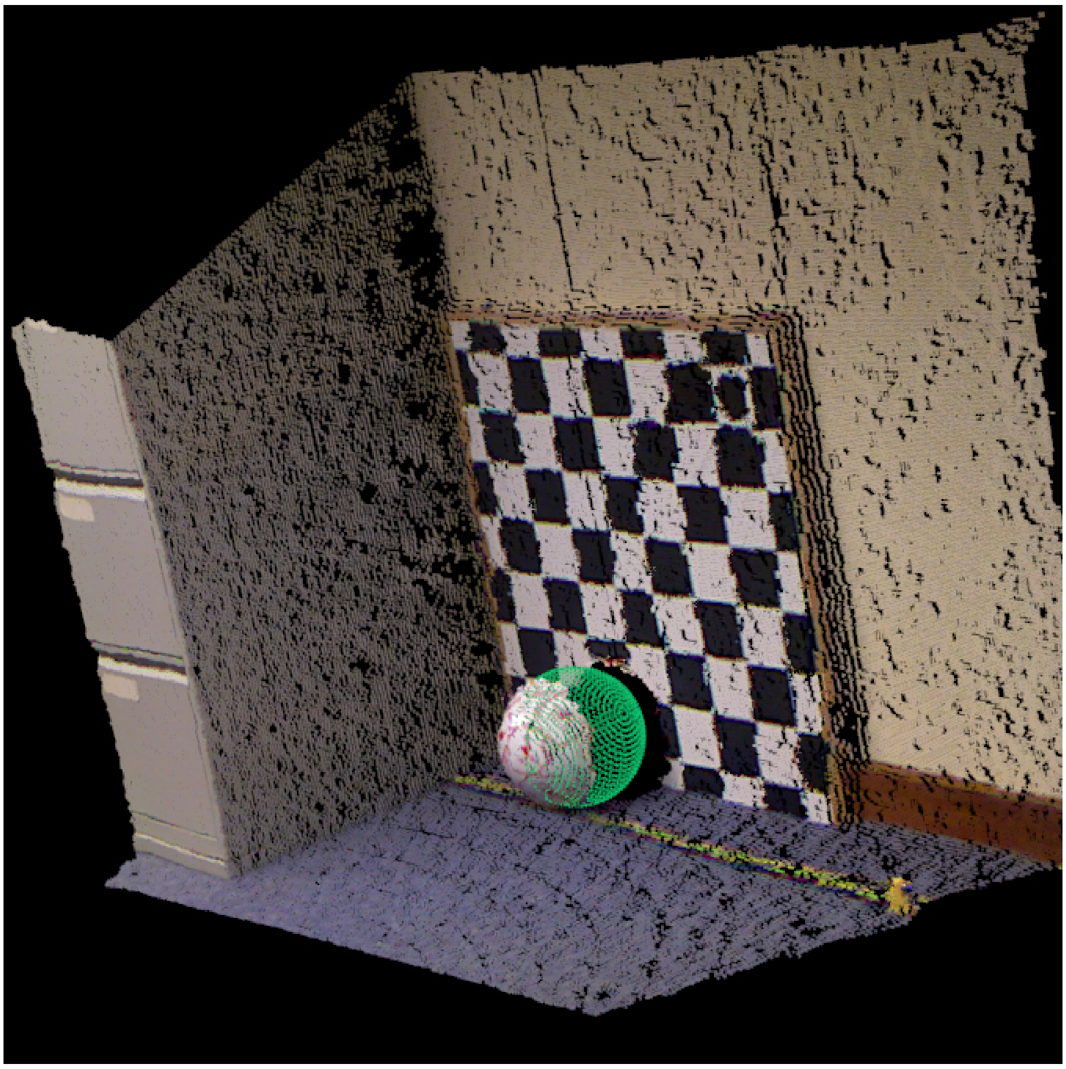

σ = 9mm

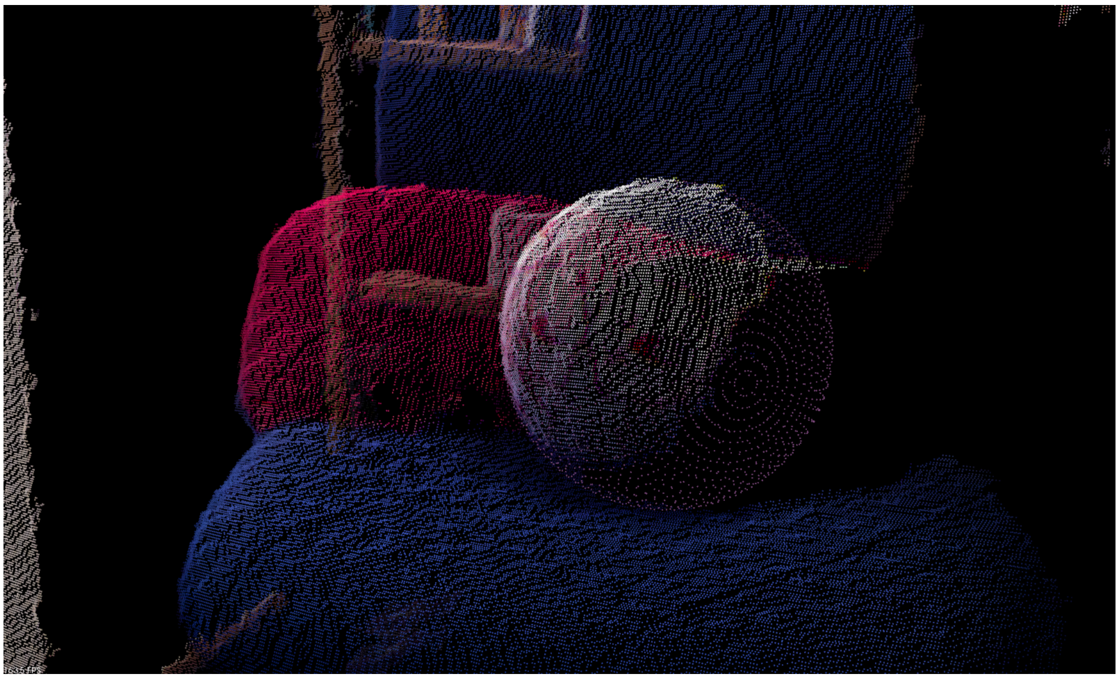

σ = 10mm

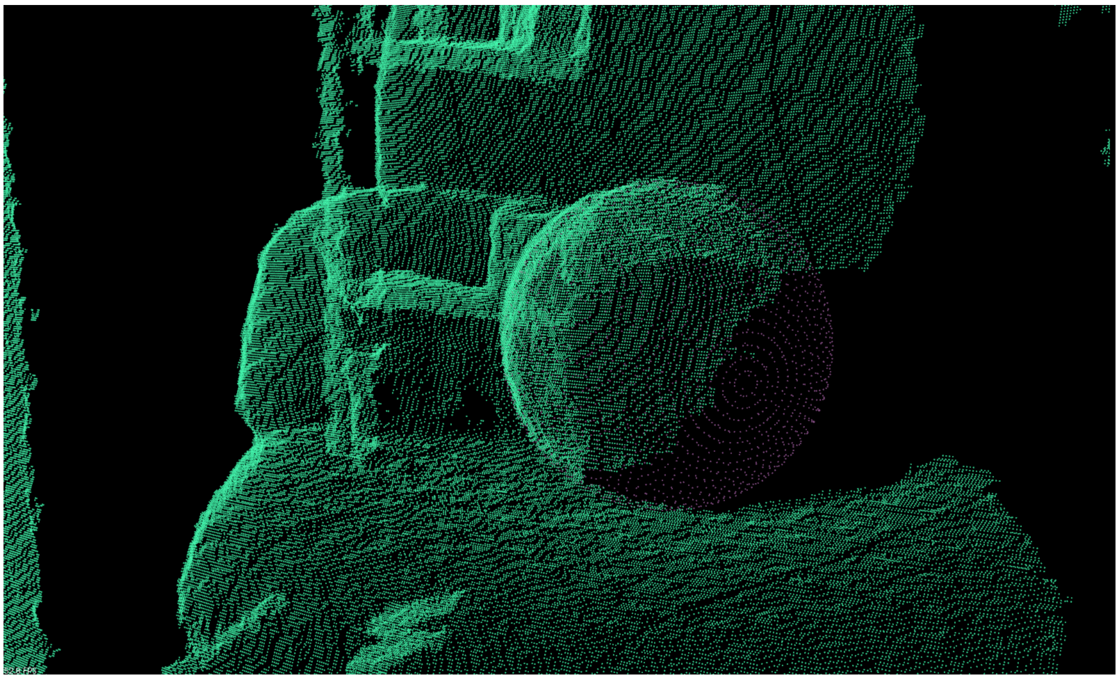

σ = 11mm

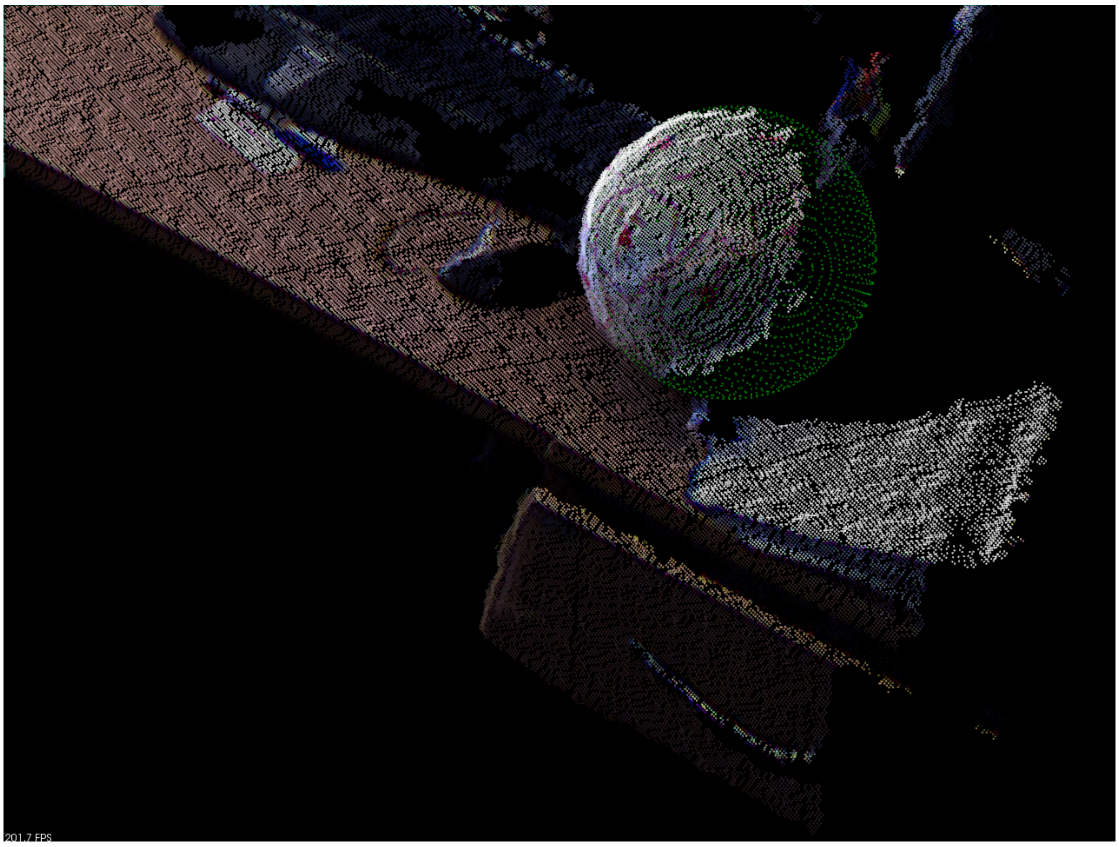
Appendix B

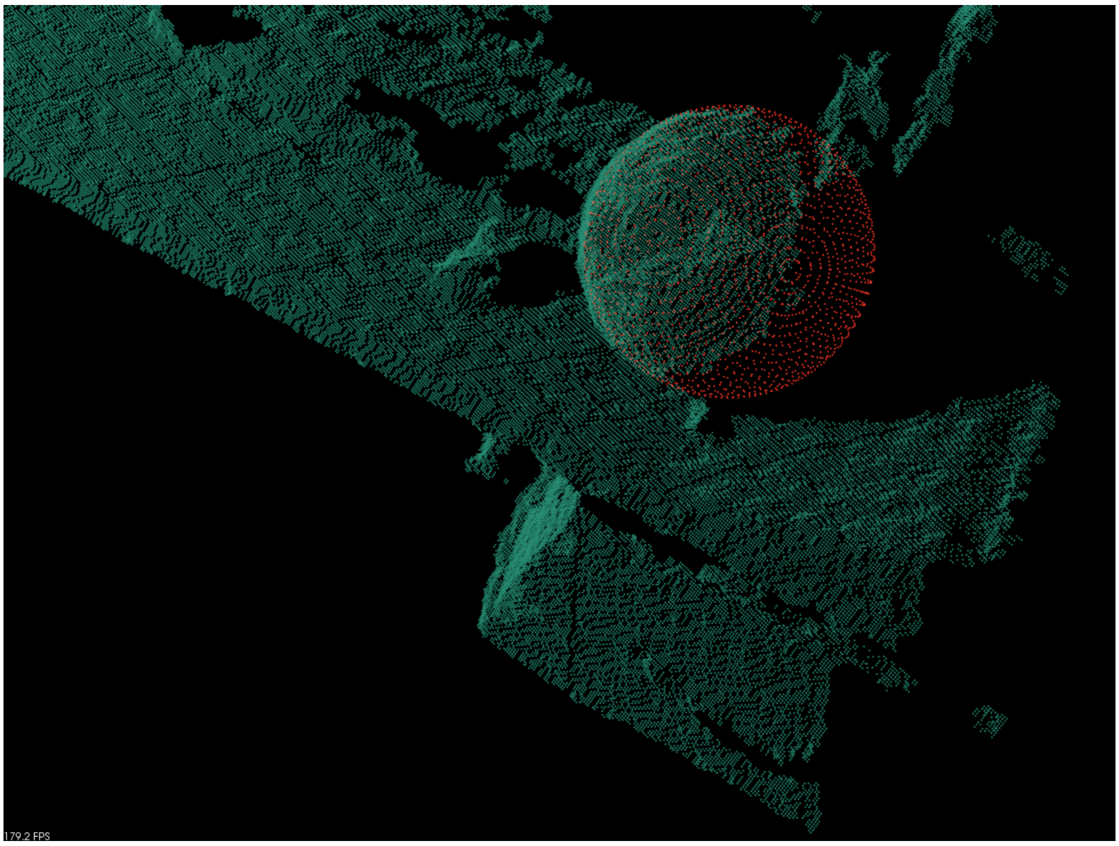
Sphere Detection in Real Point Clouds

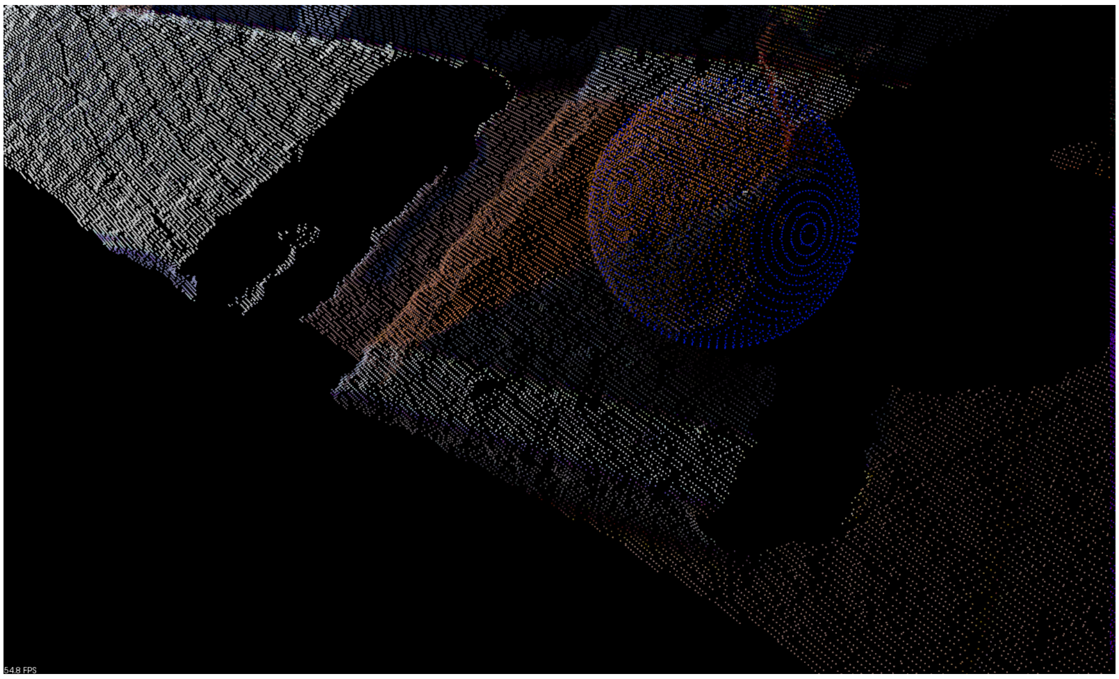


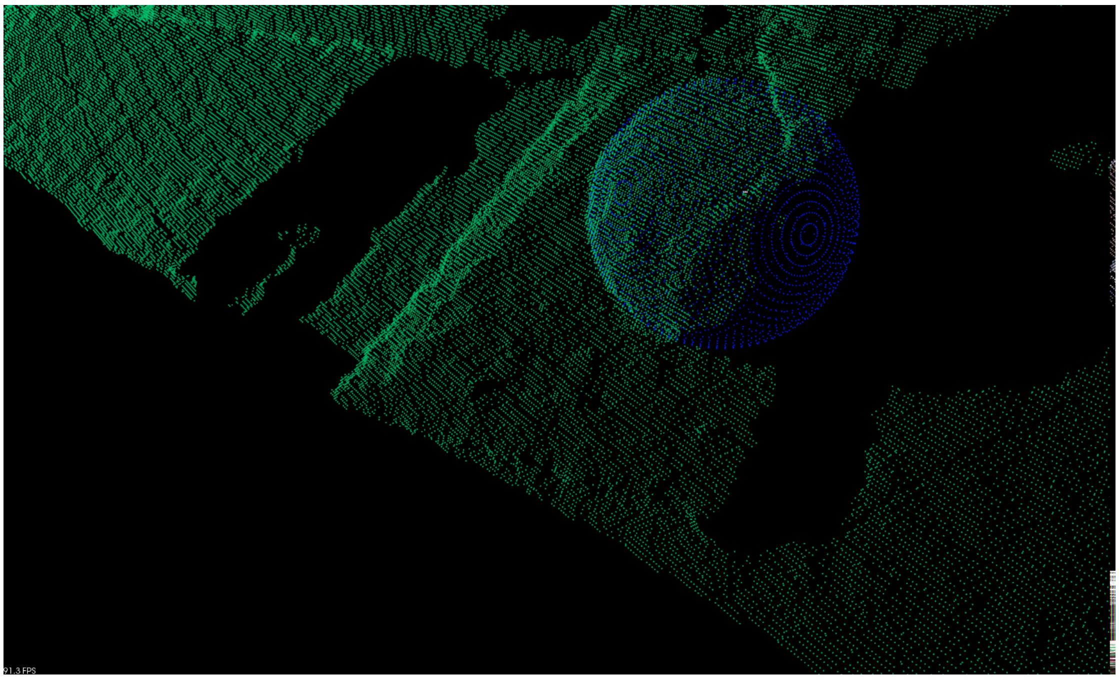


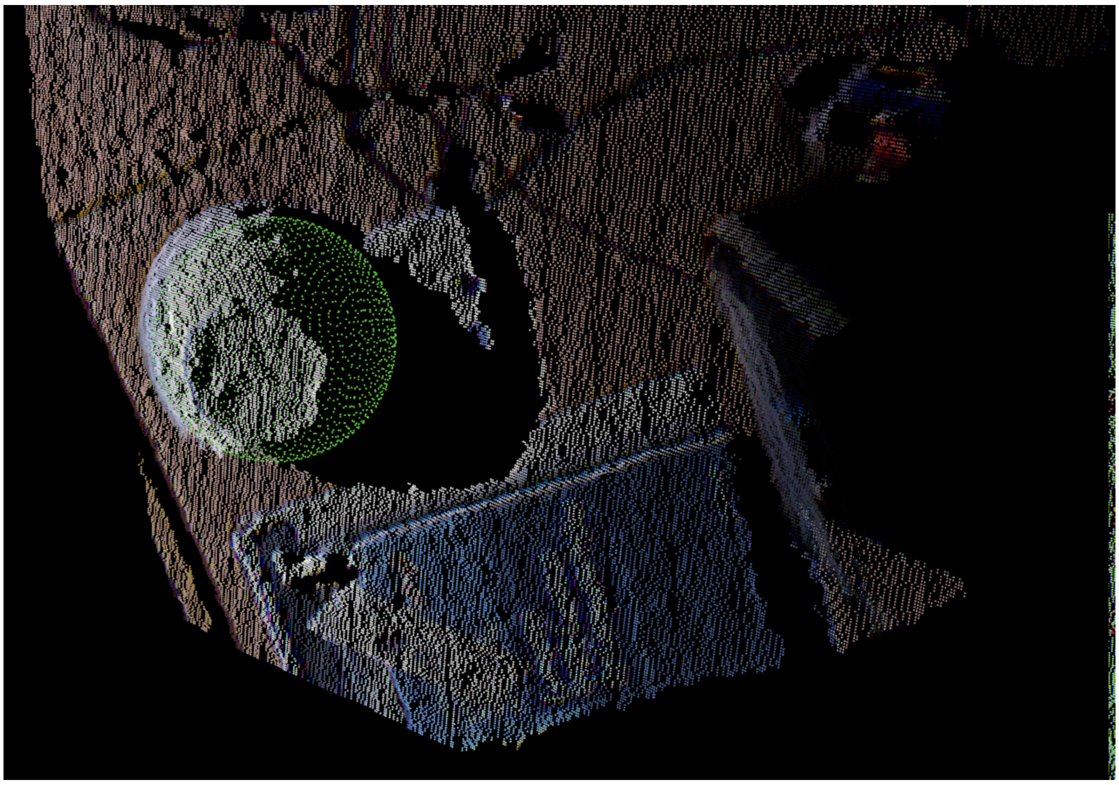


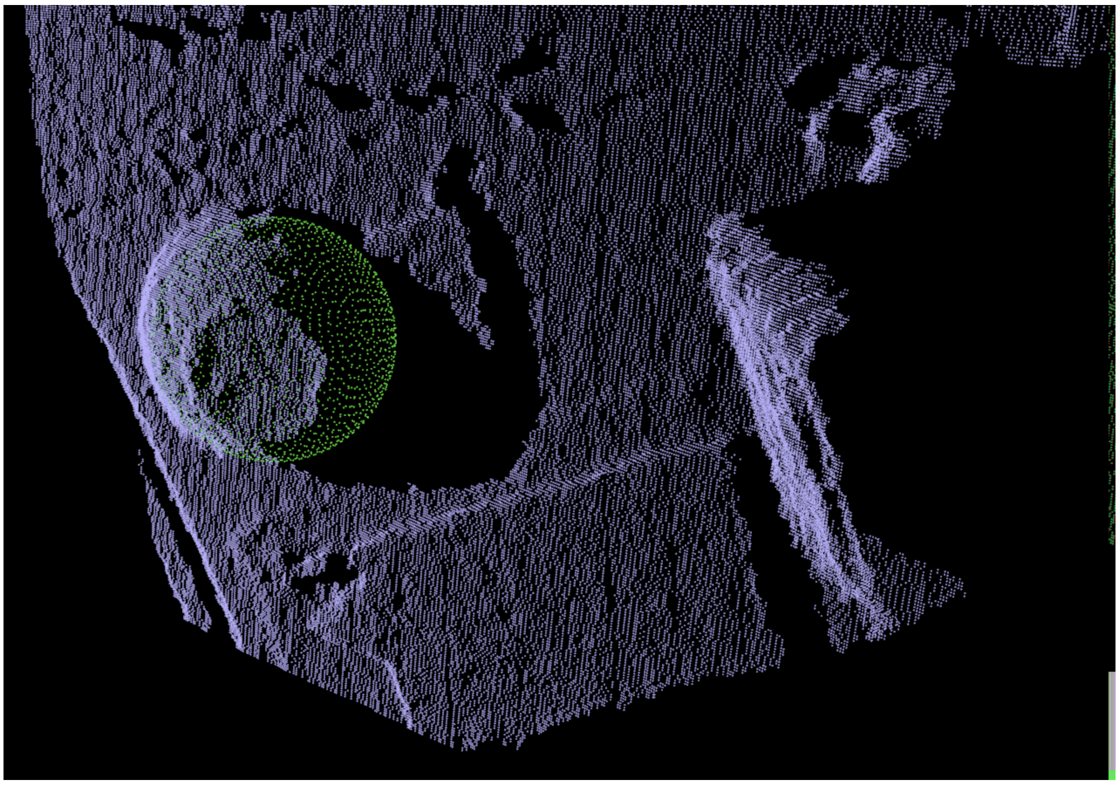

201.7 FPS


179.2 FPS


54.8 FPS


91.3 FPS





References

[1] Anas Abuzaina, Mark S Nixon, and John N Carter. 3D moving object reconstruction by temporal accumulation. In *Pattern Recognition (ICPR), 2014 22st International Conference on*. IEEE, 2014.

[2] Anas Abuzaina, Mark S Nixon, and John N Carter. Sphere detection in Kinect point clouds via the 3D Hough transform. In *Computer Analysis of Images and Patterns*, pages 290–297. Springer, 2013.

[3] Anas Abuzaina, Thamer Alathari, Mark S Nixon, and John N Carter. Detecting moving spheres in 3D point clouds via the 3D velocity Hough transform. In *IVMSP Workshop, 2013 IEEE 11th*, pages 1–4. IEEE, 2013.

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

[5] Bikash Sabata and JK Aggarwal. Estimation of motion from a pair of range images: A review. *CVGIP: Image Understanding*, 54(3):309–324, 1991.

[6] David W Eggert, Adele Lorusso, and Robert B Fisher. Estimating 3-D rigid body transformations: A comparison of four major algorithms. *Machine Vision and Applications*, 9(5-6):272–290, 1997.

[7] Y. Liu and M. A. Rodrigues. Invariant geometric properties of image correspondence vectors as rigid constraints to motion estimation. *International Journal of Pattern Recognition and Artificial Intelligence*, 13(08):1165–1179, 1999.

[8] Yonghuai Liu and Marcos A Rodrigues. Geometrical analysis of two sets of 3D correspondence data patterns for the registration of free-form shapes. *Journal of Intelligent and Robotic Systems*, 33(4):409–436, 2002.

[9] Jasem Baroon and Bahram Ravani. A three-dimensional generalization of reuleauxs method based on line geometry. In *ASME 2006 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, pages 1123–1130. American Society of Mechanical Engineers, 2006.

[10] N Bergstrom, M Bjorkman, and Danica Kragic. Generating object hypotheses in natural scenes through human-robot interaction. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pages 827–833. IEEE, 2011.

[11] Radu Bogdan Rusu, Nico Blodow, Zoltan Csaba Marton, and Michael Beetz. Close-range scene segmentation and reconstruction of 3D point cloud maps for mobile manipulation in domestic environments. In *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pages 1–6. IEEE, 2009.

[12] Radu Bogdan Rusu, Gary Bradski, Romain Thibaux, and John Hsu. Fast 3D recognition and pose using the viewpoint feature histogram. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 2155–2162. IEEE, 2010.

[13] Eric Paquet, Marc Rioux, Anil Murching, Thumpudi Naveen, and Ali Tabatabai. Description of shape information for 2-D and 3-D objects. *Signal Processing: Image Communication*, 16(1):103–122, 2000.

[14] Richard J Campbell and Patrick J Flynn. Eigenshapes for 3D object recognition in range data. In *Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on.*, volume 2. IEEE, 1999.

[15] Robert Osada, Thomas Funkhouser, Bernard Chazelle, and David Dobkin. Shape distributions. *ACM Transactions on Graphics (TOG)*, 21(4):807–832, 2002.

[16] Ameesh Makadia, Alexander Patterson, and Kostas Daniilidis. Fully automatic registration of 3D point clouds. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 1, pages 1297–1304. IEEE, 2006.

[17] A Aldoma, Zoltan-Csaba Marton, F. Tombari, W. Wohlkinger, C. Potthast, B. Zeisl, R.B. Rusu, S. Gedikli, and M. Vincze. Tutorial: Point cloud library: Three-dimensional object recognition and 6 DoF pose estimation. *Robotics Automation Magazine, IEEE*, 19(3):80–91, Sept 2012. ISSN 1070-9932.

[18] Marius Muja and David G Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. In *VISAPP (1)*, pages 331–340, 2009.

[19] Andrea Censi and Stefano Carpin. HSM3D: feature-less global 6DoF scan-matching in the Hough/radon domain. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, pages 3899–3906. IEEE, 2009.

[20] Donghui Wang, Chenyang Cui, and Zheng Wu. Matching 3D models with global geometric feature map. In *Multi-Media Modelling Conference Proceedings, 2006 12th International*, pages 4–pp. IEEE, 2006.

[21] Szymon Rusinkiewicz and Marc Levoy. Efficient variants of the ICP algorithm. In *3-D Digital Imaging and Modeling, 2001. Proceedings. Third International Conference on*, pages 145–152. IEEE, 2001.

[22] Aitor Aldoma, Markus Vincze, Nico Blodow, David Gossow, Suat Gedikli, Radu Bogdan Rusu, and Gary Bradski. Cad-model recognition and 6DoF pose estimation using 3D cues. In *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*, pages 585–592. IEEE, 2011.

[23] Y. Guo, M. Bennamoun, F. Sohel, M. Lu, and J. Wan. 3D object recognition in cluttered scenes with local surface features: A survey, 2014. ISSN 0162-8828.

[24] Federico Tombari, Samuele Salti, and Luigi Di Stefano. Performance evaluation of 3D keypoint detectors. *International Journal of Computer Vision*, 102(1-3): 198–220, 2013.

[25] Ajmal Mian, Mohammed Bennamoun, and R Owens. On the repeatability and quality of keypoints for local feature-based 3D object retrieval from cluttered scenes. *International Journal of Computer Vision*, 89(2-3):348–361, 2010.

[26] Donald JR Meagher. High-speed image generation of complex solid objects using octree encoding, September 15 1987. US Patent 4,694,404.

[27] Sunil Arya, David M Mount, Nathan S Netanyahu, Ruth Silverman, and Angela Y Wu. An optimal algorithm for approximate nearest neighbor searching fixed dimensions. *Journal of the ACM (JACM)*, 45(6):891–923, 1998.

[28] Michael Schwarz and Hans-Peter Seidel. Fast parallel surface and solid voxelization on GPUs. *ACM Transactions on Graphics (TOG)*, 29(6):179, 2010.

[29] Diego Nehab and Philip Shilane. Stratified point sampling of 3D models. In *Proceedings of the First Eurographics conference on Point-Based Graphics*, pages 49–56. Eurographics Association, 2004.

[30] Hui Chen and Bir Bhanu. 3D free-form object recognition in range images using local surface patches. *Pattern Recognition Letters*, 28(10):1252–1262, 2007.

[31] Yulan Guo, Ferdous Sohel, Mohammed Bennamoun, Min Lu, and Jianwei Wan. Rotational projection statistics for 3D local surface description and object recognition. *International journal of computer vision*, 105(1):63–86, 2013.

[32] Yu Zhong. Intrinsic shape signatures: A shape descriptor for 3D object recognition. In *Computer Vision Workshops (ICCV Workshops), 2009 IEEE 12th International Conference on*, pages 689–696. IEEE, 2009.

[33] Ranjith Unnikrishnan and Martial Hebert. Multi-scale interest regions from unorganized point clouds. In *Computer Vision and Pattern Recognition Workshops, 2008. CVPRW'08. IEEE Computer Society Conference on*, pages 1–8. IEEE, 2008.

[34] Jan Knopp, Mukta Prasad, Geert Willems, Radu Timofte, and Luc Van Gool. Hough transform and 3D SURF for robust three dimensional classification. In *Computer Vision–ECCV 2010*, pages 589–602. Springer, 2010.

[35] Yosi Keller and Tal Darom. Scale invariant features for 3D mesh models. *IEEE Transactions on Image Processing*, 21(5), 2012.

[36] Radu Bogdan Rusu. Semantic 3D object maps for everyday manipulation in human living environments. *KI-Künstliche Intelligenz*, 24(4):345–348, 2010.

[37] Bastian Steder, Radu Bogdan Rusu, Kurt Konolige, and Wolfram Burgard. NARF: 3D range image features for object recognition. In *Workshop on Defining and Solving Realistic Perception Problems in Personal Robotics at the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, volume 44, 2010.

[38] Erickson R do Nascimento, Gabriel L Oliveira, Antônio W Vieira, and Mario FM Campos. On the development of a robust, fast and lightweight keypoint descriptor. *Neurocomputing*, 120:141–155, 2013.

[39] A.P. Ashbrook, R.B. Fisher, C. Robertson, and N. Werghi. Finding surface correspondence for object recognition and registration using pairwise geometric histograms. In *Computer Vision ECCV98*, volume 1407 of *Lecture Notes in Computer Science*, pages 674–686. Springer Berlin Heidelberg, 1998.

[40] Shuai Tang, Xiaoyu Wang, Xutao Lv, Tony X Han, James Keller, Zhihai He, Marjorie Skubic, and Shihong Lao. Histogram of oriented normal vectors for object recognition with a depth sensor. In *Computer Vision–ACCV 2012*, pages 525–538. Springer, 2013.

[41] Radu Bogdan Rusu, Nico Blodow, Zoltan Csaba Marton, and Michael Beetz. Aligning point cloud views using persistent feature histograms. In *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, pages 3384–3391. IEEE, 2008.

[42] Federico Tombari, Samuele Salti, and Luigi Di Stefano. Unique signatures of histograms for local surface description. In *Computer Vision–ECCV 2010*, pages 356–369. Springer, 2010.

[43] Andrei Zaharescu, Edmond Boyer, Kiran Varanasi, and Radu Horaud. Surface feature detection and description with applications to mesh matching. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 373–380. IEEE, 2009.

[44] Jian Sun, Maks Ovsjanikov, and Leonidas Guibas. A concise and provably informative multi-scale signature based on heat diffusion. In *Computer Graphics Forum*, volume 28, pages 1383–1392, 2009.

[45] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[46] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. SURF: Speeded up robust features. In *Computer Vision–ECCV 2006*, pages 404–417. Springer, 2006.

[47] Krystian Mikolajczyk and Cordelia Schmid. A performance evaluation of local descriptors. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 27(10):1615–1630, 2005.

[48] Emanuele Rodolà, Andrea Albarelli, Filippo Bergamasco, and Andrea Torsello. A scale independent selection process for 3D object recognition in cluttered scenes. *International journal of computer vision*, 102(1-3):129–145, 2013.

[49] Federico Tombari and Luigi Di Stefano. Hough voting for 3D object recognition under occlusion and clutter. *IPSJ Transactions on Computer Vision and Applications*, 4(0):20–29, 2012.

[50] Aitor Aldoma, Federico Tombari, Luigi Di Stefano, and Markus Vincze. A global hypotheses verification method for 3D object recognition. In *Computer Vision–ECCV 2012*, pages 511–524. Springer, 2012.

[51] Andrew E. Johnson and Martial Hebert. Using spin images for efficient object recognition in cluttered 3D scenes. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 21(5):433–449, 1999.

[52] Paul Scovanner, Saad Ali, and Mubarak Shah. A 3-dimensional sift descriptor and its application to action recognition. In *Proceedings of the 15th international conference on Multimedia*, pages 357–360. ACM, 2007.

[53] Marcos Rodrigues, Robert Fisher, and Yonghuai Liu. Special issue on registration and fusion of range images. *Computer Vision and Image Understanding*, 87(1): 1–7, 2002.

[54] KK Biswas and Saurav Kumar Basu. Gesture recognition using Microsoft Kinect. In *Automation, Robotics and Applications (ICARA), 2011 5th International Conference on*, pages 100–103. IEEE, 2011.

[55] Pushmeet Kohli and Jamie Shotton. Key developments in human pose estimation for Kinect. In *Consumer Depth Cameras for Computer Vision*, pages 63–70. Springer, 2013.

[56] Evan A Suma, Belinda Lange, A Rizzo, DM Krum, and Mark Bolas. FAAST: The flexible action and articulated skeleton toolkit. In *Virtual Reality Conference (VR), 2011 IEEE*, pages 247–248. IEEE, 2011.

[57] Peter Henry, Michael Krainin, Evan Herbst, Xiaofeng Ren, and Dieter Fox. Rgb-d mapping: Using Kinect-style depth cameras for dense 3D modeling of indoor environments. *The International Journal of Robotics Research*, 31(5):647–663, 2012.

[58] Kourosh Khoshelham and Sander Oude Elberink. Accuracy and resolution of Kinect depth data for indoor mapping applications. *Sensors*, 12(2):1437–1454, 2012.

[59] Jan Smisek, Michal Jancosek, and Tomas Pajdla. 3D with Kinect. In *Consumer Depth Cameras for Computer Vision*, pages 3–25. Springer, 2013.

[60] Zoltan Csaba Marton, Radu Bogdan Rusu, and Michael Beetz. On fast surface reconstruction methods for large and noisy point clouds. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, pages 3218–3223. IEEE, 2009.

[61] Gregory P Meyer and Minh N Do. Real-time 3D face modeling with a commodity depth camera. In *Multimedia and Expo Workshops (ICMEW), 2013 IEEE International Conference on*, pages 1–4. IEEE, 2013.

[62] Sidney W Wang and Arie E Kaufman. Volume-sampled 3D modeling. *Computer Graphics and Applications, IEEE*, 14(5):26–32, 1994.

[63] Cyril Crassin and Simon Green. Octree-based sparse voxelization using the GPU hardware rasterizer. *OpenGL Insights*, pages 303–318, 2012.

[64] R.B. Rusu and S. Cousins. 3D is here: Point cloud library (PCL). In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 1–4, May 2011.

[65] Joaquim Salvi, Carles Matabosch, David Fofi, and Josep Forest. A review of recent range image registration methods with accuracy evaluation. *Image and Vision Computing*, 25(5):578–596, 2007.

[66] Paul J Besl and Neil D McKay. Method for registration of 3-D shapes. In *Robotics-DL tentative*, pages 586–606. International Society for Optics and Photonics, 1992.

[67] Thibaut Weise, Thomas Wismer, Bastian Leibe, and Luc Van Gool. In-hand scanning with online loop closure. In *Computer Vision Workshops (ICCV Workshops), 2009 IEEE 12th International Conference on*, pages 1630–1637. IEEE, 2009.

[68] Szymon Rusinkiewicz, Olaf Hall-Holt, and Marc Levoy. Real-time 3D model acquisition. In *ACM Transactions on Graphics (TOG)*, volume 21, pages 438–446. ACM, 2002.

[69] Thibaut Weise, Bastian Leibe, and Luc Van Gool. Accurate and robust registration for in-hand modeling. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008.

[70] Richard A Newcombe, Andrew J Davison, Shahram Izadi, Pushmeet Kohli, Otmar Hilliges, Jamie Shotton, David Molyneaux, Steve Hodges, David Kim, and Andrew Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *Mixed and augmented reality (ISMAR), 2011 10th IEEE international symposium on*, pages 127–136, 2011.

[71] David K Wagg and Mark S Nixon. On automated model-based extraction and analysis of gait. In *Automatic Face and Gesture Recognition, 2004. Proceedings. Sixth IEEE International Conference on*, pages 11–16. IEEE, 2004.

[72] Michael G Grant, Mark S Nixon, and Paul H Lewis. Extracting moving shapes by evidence gathering. *Pattern Recognition*, 35(5):1099–1114, 2002.

[73] José AR Artolazábal and John Illingworth. 3DSVHT: extraction of 3D linear motion via multi-view, temporal evidence accumulation. In *Advanced Concepts for Intelligent Vision Systems*, pages 563–570. Springer, 2005.

[74] D. Held, J. Levinson, and S. Thrun. Precision tracking with sparse 3D and dense color 2D data. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 1138–1145, 2013.

[75] Camillo J Taylor and David J Kriegman. Minimization on the lie group SO(3) and related manifolds. *Yale University*, 1994.

[76] Radu Bogdan Rusu, Nico Blodow, Zoltan Marton, Alina Soos, and Michael Beetz. Towards 3D object maps for autonomous household robots. In *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*, pages 3191–3198. IEEE, 2007.

[77] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. Poisson surface reconstruction. In *Proceedings of the fourth Eurographics symposium on Geometry processing*, 2006.

[78] George Vosselman, Ben GH Gorte, George Sithole, and Tahir Rabbani. Recognising structure in laser scanner point clouds. *International archives of photogrammetry, remote sensing and spatial information sciences*, 46(8):33–38, 2004.

[79] Radu Bogdan Rusu. *Semantic 3D Object Maps for Everyday Manipulation in Human Living Environments.* PhD thesis, Computer Science department, Technische Universitat Munchen, Germany, October 2009.

[80] Martin A. Fischler and Robert C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, June 1981. ISSN 0001-0782.

[81] Paul Hough. Method and Means for Recognizing Complex Patterns. U.S. Patent 3.069.654, December 1962.

[82] Ruwen Schnabel, Roland Wahl, and Reinhard Klein. Efficient RANSAC for point-cloud shape detection. In *Computer Graphics Forum*, volume 26, pages 214–226. Wiley Online Library, 2007.

[83] Dorit Borrmann, Jan Elseberg, Kai Lingemann, and Andreas Nchter. The 3D Hough transform for plane detection in point clouds : A review and a new accumulator design. *3D Research*, 02003(2):32, 2011.

[84] O. O. Ogundana, C. R. Coggrave, R. L. Burguete, and J. M. Huntley. Automated detection of planes in 3-D point clouds using fast Hough transforms. *Optical Engineering*, 50(5):053609 1–11, May 2011.

[85] Olatokunbo O Ogundana, C Russell Coggrave, Richard L Burguete, and Jonathan M Huntley. Fast Hough transform for automated detection of spheres in three-dimensional point clouds. *Optical Engineering*, 46(5):051002 1–11, 2007.

[86] Tahir Rabbani and Frank Van Den Heuvel. Efficient Hough transform for automatic detection of cylinders in point clouds. *ISPRS WG III/3, III/4*, 3:60–65, 2005.

[87] Fayez Tarsha-Kurdi, Tania Landes, Pierre Grussenmeyer, et al. Hough-transform and extended RANSAC algorithms for automatic detection of 3D building roof planes from lidar data. *International Archives of Photogrammetry, Remote Sensing and Spatial Information Systems*, 36:407–412, 2007.

[88] Tobias Kotthäuser and Bärbel Mertsching. Triangulation-based plane extraction for 3D point clouds. In *Proceedings of the 5th international conference on Intelligent Robotics and Applications - Volume Part I*, ICIRA'12, pages 217–228, Berlin, Heidelberg, 2012. Springer-Verlag. ISBN 978-3-642-33508-2.

[89] Russell W Taylor. An efficient implementation of decomposable parameter spaces. In *Pattern Recognition, 1990. Proceedings., 10th International Conference on*, volume 1, pages 613–619. IEEE, 1990.

[90] Chia-Chun Hsu and Jun S. Huang. Partitioned Hough transform for ellipsoid detection. *Pattern Recogn.*, 23(3-4):275–282, March 1990. ISSN 0031-3203.

[91] Marjolein van der Glas, Frans M Vos, Charl P Botha, and Albert M Vossepoel. Determination of position and radius of ball joints. In *Medical Imaging 2002*, pages 1571–1577. International Society for Optics and Photonics, 2002.

[92] MY Cao, CH Ye, O Doessel, and C Liu. Spherical parameter detection based on hierarchical Hough transform. *Pattern recognition letters*, 27(9):980–986, 2006.

[93] M. Kharbat, N. Aouf, A. Tsourdos, and B. White. Sphere detection and tracking for a space capturing operation. In *Advanced Video and Signal Based Surveillance, 2007. AVSS 2007. IEEE Conference on*, pages 182–187, Sept 2007.

[94] Marco Camurri, Roberto Vezzani, and Rita Cucchiara. 3D Hough transform for sphere recognition on point clouds. *Machine Vision and Applications*, 25(7):1877–1891, 2014. ISSN 0932-8092.

[95] Yasutaka Furukawa and Yoshihisa Shinagawa. Accurate and robust line segment extraction by analyzing distribution around peaks in Hough space. *Computer Vision and Image Understanding*, 92(1):1–25, 2003.

[96] Wayne Niblack and Dragutin Petkovic. On improving the accuracy of the Hough transform. *Machine Vision and Applications*, 3(2):87–106, 1990.

[97] Phil L Palmer, Josef Kittler, and Maria Petrou. An optimizing line finder using a Hough transform algorithm. *Computer Vision and Image Understanding*, 67(1): 1–23, 1997.

[98] Jason M Nash, John N Carter, and Mark S Nixon. Dynamic feature extraction via the velocity Hough transform. *Pattern Recognition Letters*, 18(10):1035–1047, 1997.

[99] Douglas J Hunt, Loren W Nolte, and W Howard Ruedger. Performance of the Hough transform and its relationship to statistical signal detection theory. *Computer vision, graphics, and image processing*, 43(2):221–238, 1988.

[100] Mark Nixon and Alberto S Aguado. *Feature extraction & image processing for computer vision.* Academic Press, 2012.

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

[102] A. Azim and O. Aycard. Detection, classification and tracking of moving objects in a 3D environment. In *Intelligent Vehicles Symposium (IV), 2012 IEEE*, pages 802–807, 2012.

[103] Youngmin Park, Vincent Lepetit, and Woontack Woo. Texture-less object tracking with online training using an rgb-d camera. In *Mixed and Augmented Reality (ISMAR), 2011 10th IEEE International Symposium on*, pages 121–126. IEEE, 2011.

[104] Jamie Shotton, Toby Sharp, Alex Kipman, Andrew Fitzgibbon, Mark Finocchio, Andrew Blake, Mat Cook, and Richard Moore. Real-time human pose recognition in parts from single depth images. *Communications of the ACM*, 56(1):116–124, 2013.

[105] Pelopidas Lappas, John N Carter, and Robert I Damper. Object tracking via the dynamic velocity Hough transform. In *Image Processing, 2001. Proceedings. 2001 International Conference on*, volume 2, pages 371–374. IEEE, 2001.

[106] Jason M Nash, John N Carter, and Mark S Nixon. Extraction of moving articulated-objects by evidence gathering. 1998.

[107] Steven Mills, Tony P Pridmore, and Mark Hills. Tracking in a Hough space with the extended kalman filter. In *BMVC*, pages 1–10, 2003.

[108] Koichi Sato and Jake K Aggarwal. Temporal spatio-velocity transform and its application to tracking and interaction. *Computer Vision and Image Understanding*, 96(2):100–128, 2004.

[109] Michael Greenspan, Limin Shang, and Piotr Jasiobedzki. Efficient tracking with the bounded Hough transform. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, volume 1, pages I–520. IEEE, 2004.

[110] Dana H Ballard. Generalizing the Hough transform to detect arbitrary shapes. *Pattern recognition*, 13(2):111–122, 1981.

[111] Franz Reuleaux. *Theoretische kinematik: Grundzuge einer Theorie des Maschinenwesens. English Translation:Kinematics of Machinery. Outlines of a Theory of Machine.* MacMillan, London, 1875.

[112] William B Heard. *Rigid body mechanics: mathematics, physics and applications.* John Wiley & Sons, 2008.

[113] Yonghuai Liu and Marcos A Rodrigues. Geometric understanding of rigid body transformations. In *Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on*, volume 2, pages 1275–1280. IEEE, 1999.

[114] Robert B Fisher. Geometric constraints from planar surface patch matching. *Image and Vision Computing*, 8(2):148–154, 1990.

[115] K Halvorsen, M Lesser, and A Lundberg. A new method for estimating the axis of rotation and the center of rotation. *Journal of biomechanics*, 32(11):1221–1227, 1999.

[116] Lillian Y Chang and Nancy S Pollard. Robust estimation of dominant axis of rotation. *Journal of biomechanics*, 40(12):2707–2715, 2007.

[117] Brendan McCane, J.Haxby Abbott, and Tamara King. On calculating the finite centre of rotation for rigid planar motion. *Medical Engineering and Physics*, 27 (1):75 – 79, 2005. ISSN 1350-4533.

[118] Koon Kiat Teu and Wangdo Kim. Estimation of the axis of a screw motion from noisy data–a new method based on plucker lines. *Journal of biomechanics*, 39(15): 28572862, 2006. ISSN 0021-9290.

[119] J.D. Moorehead, S.C. Montgomery, and D.M. Harvey. Instant center of rotation estimation using the reuleaux technique and a lateral extrapolation technique. *Journal of Biomechanics*, 36(9):1301 – 1307, 2003. ISSN 0021-9290.

[120] Jeffrey J. Spiegelman and Savio L.-Y. Woo. A rigid-body method for finding centers of rotation and angular displacements of planar joint motion. *Journal of Biomechanics*, 20(7):715 – 721, 1987. ISSN 0021-9290.

[121] Manohar M. Panjabi. Centers and angles of rotation of body joints: A study of errors and optimization. *Journal of Biomechanics*, 12(12):911 – 920, 1979. ISSN 0021-9290.

[122] Johannes K Eberharter and Bahram Ravani. Kinematic registration in 3D using the 2D reuleaux method. *Journal of Mechanical Design*, 128(2):349–355, 2006.

[123] Radim Halır. An automatic estimation of the axis of rotation of fragments of archaeological pottery: A multi-step model-based approach. In *Proc. of the 7th International conference in Central Europe on computer graphics, Visualization and Interactive Digital Media (WSCG99)*. Citeseer, 1999.

[124] Xiaoyuan Qian and Xuegang Huang. Reconstruction of surfaces of revolution with partial sampling. *Journal of Computational and Applied Mathematics*, 163(1):211 – 217, 2004. ISSN 0377-0427. Proceedings of the International Symposium on Computational Mathematics and Applications.

[125] Jasem Baroon and Bahram Ravani. A computational geometric solution of the kinematic registration problem using the bisecting linear line complex. *Computer-Aided Design and Applications*, 6(1):1–13, 2009.

[126] Leo Reyes, Gerard Medioni, and Eduardo Bayro. Registration of 3D points using geometric algebra and tensor voting. *International Journal of Computer Vision*, 75(3):351–369, 2007.

[127] Minh-Tri Pham, Oliver J Woodford, Frank Perbet, Atsuto Maki, Björn Stenger, and Roberto Cipolla. A new distance for scale-invariant 3D shape recognition and registration. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 145–152. IEEE, 2011.

[128] H-T Tsui and C-K Chan. Hough technique for 3D object recognition. *Computers and Digital Techniques, IEE Proceedings E*, 136(6):565–568, 1989.

[129] Gongzhu Hu. 3-D object matching in the Hough space. In *Systems, Man and Cybernetics, 1995. Intelligent Systems for the 21st Century., IEEE International Conference on*, volume 3, pages 2718–2723. IEEE, 1995.

[130] Kourosh Khoshelham. Extending generalized Hough transform to detect 3D objects in laser range data. In *ISPRS Workshop on Laser Scanning, Proceedings, LS 2007*, pages 206–210. Citeseer, 2007.

[131] Tao Wang, Xuming He, and Nick Barnes. Learning structured Hough voting for joint object detection and occlusion reasoning. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 1790–1797. IEEE, 2013.

[132] Pedro F Felzenszwalb, Ross B Girshick, David McAllester, and Deva Ramanan. Object detection with discriminatively trained part-based models. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32(9):1627–1645, 2010.

[133] Oliver J Woodford, Minh-Tri Pham, Atsuto Maki, Frank Perbet, and Björn Stenger. Demisting the Hough transform for 3D shape recognition and registration. *International Journal of Computer Vision*, 106(3):332–341, 2014.

[134] Frank J Aherne, Neil A Thacker, and Peter I Rockett. The bhattacharyya metric as an absolute similarity measure for frequency coded data. *Kybernetika*, 34(4): 363–368, 1998.

[135] Bruno Siciliano and Oussama Khatib. *Springer handbook of robotics*. 2008.

[136] Kevin Lai, Liefeng Bo, Xiaofeng Ren, and Dieter Fox. A large-scale hierarchical multi-view rgb-d object dataset. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 1817–1824. IEEE, 2011.

[137] Michael Ian Shamos and Dan Hoey. Closest-point problems. In *Proceedings of the 16th Annual Symposium on Foundations of Computer Science*, SFCS '75, pages 151–162, Washington, DC, USA, 1975. IEEE Computer Society.

[138] Federico Tombari and Luigi Di Stefano. Object recognition in 3D scenes with occlusions and clutter by Hough voting. In *Image and Video Technology (PSIVT), 2010 Fourth Pacific-Rim Symposium on*, pages 349–355. IEEE, 2010.

[139] Min Sun, Gary Bradski, Bing-Xin Xu, and Silvio Savarese. Depth-encoded Hough voting for joint object detection and shape recovery. In *Computer Vision–ECCV 2010*, pages 658–671. Springer, 2010.