

**SIMULTANEOUS REAL-TIME OBJECT RECOGNITION AND POSE ESTIMATION FOR  
ARTIFICIAL SYSTEMS OPERATING IN DYNAMIC ENVIRONMENTS**

by

**Frans-Pieter van Wyk**

Submitted in partial fulfilment of the requirements for the degree

Master of Engineering (Electronic Engineering)

in the

Department of Electrical, Electronic and Computer Engineering  
Faculty of Engineering, Built Environment and Information Technology

UNIVERSITY OF PRETORIA

April 2013

## SUMMARY

---

### **SIMULTANEOUS REAL-TIME OBJECT RECOGNITION AND POSE ESTIMATION FOR ARTIFICIAL SYSTEMS OPERATING IN DYNAMIC ENVIRONMENTS**

by

**Frans-Pieter van Wyk**

Supervisor(s): Mr H. Grobler  
Department: Electrical, Electronic and Computer Engineering  
University: University of Pretoria  
Degree: Master of Engineering (Electronic Engineering)  
Keywords: Object recognition, pose estimation, real-time, partial object matching,  
3D features, free form deformation, data compression, locality sensitive  
hashing, structured light, intelligent systems

Recent advances in technology have increased awareness of the necessity for automated systems in people's everyday lives. Artificial systems are more frequently being introduced into environments previously thought to be too perilous for humans to operate in. Some robots can be used to extract potentially hazardous materials from sites inaccessible to humans, while others are being developed to aid humans with laborious tasks.

A crucial aspect of all artificial systems is the manner in which they interact with their immediate surroundings. Developing such a deceptively simple aspect has proven to be significantly challenging, as it not only entails the methods through which the system perceives its environment, but also its ability to perform critical tasks. These undertakings often involve the coordination of numerous subsystems, each performing its own complex duty. To complicate matters further, it is nowadays becoming increasingly important for these artificial systems to be able to perform their tasks in real-time.

The task of object recognition is typically described as the process of retrieving the object in a database that is most similar to an unknown, or query, object. Pose estimation, on the other hand, involves estimating the position and orientation of an object in three-dimensional space, as seen from an observer's viewpoint. These two tasks are regarded as vital to many computer vision techniques and

regularly serve as input to more complex perception algorithms.

An approach is presented which regards the object recognition and pose estimation procedures as mutually dependent. The core idea is that dissimilar objects might appear similar when observed from certain viewpoints. A feature-based conceptualisation, which makes use of a database, is implemented and used to perform simultaneous object recognition and pose estimation. The design incorporates data compression techniques, originally suggested by the image-processing community, to facilitate fast processing of large databases.

System performance is quantified primarily on object recognition, pose estimation and execution time characteristics. These aspects are investigated under ideal conditions by exploiting three-dimensional models of relevant objects. The performance of the system is also analysed for practical scenarios by acquiring input data from a structured light implementation, which resembles that obtained from many commercial range scanners.

Practical experiments indicate that the system was capable of performing simultaneous object recognition and pose estimation in approximately 230 ms once a novel object has been sensed. An average object recognition accuracy of approximately 73% was achieved. The pose estimation results were reasonable but prompted further research. The results are comparable to what has been achieved using other suggested approaches such as Viewpoint Feature Histograms and Spin Images.

## OPSOMMING

---

### GELYKTYDIGE INTYDSE VOORWERPHERKENNING EN POSE-BERAMING VIR KUNSMATIGE STELSELS WAT IN DINAMIESE OMGEWINGS FUNKSIONEER

deur

**Frans-Pieter van Wyk**

Studieleier(s): Mnr H. Grobler  
Departement: Elektriese, Elektroniese en Rekenaar-Ingenieurswese  
Universiteit: Universiteit van Pretoria  
Graad: Magister in Ingenieurswese (Elektroniese Ingenieurswese)  
Sleutelwoorde: Voorwerpherkenning, pose-beraming, intydse verwerking, gedeeltelike voorwerppassing, 3D-eienskappe, vryevormvervorming, datakompressie, liggings sensitiewe *hashing*, gestruktureerde lig, intelligente stelsels

Die onlangse vooruitgang in tegnologie het navorsers meer bewus gemaak van die noodsaaklikheid vir geoutomatiseerde stelsels in die alledaagse lewe. Kunsmatige stelsels word meer gereeld in omgewings geplaas wat voorheen as te gevaarlik geag is vir mense in te werk. Sekere robotte kan ingespan word om potensieel gevaarlike stowwe te ontgin in areas wat nie bereikbaar is vir mense nie, terwyl ander ontwerp word om mense met moeisame take te help.

'n Deurslaggewende aspek van alle kunsmatige stelsels is die manier waarop hulle met hul nabye omgewing in aksie tree. Die ontwikkeling van so 'n bedrieglik eenvoudig aspek het bewys dat dit besonder uitdagend is, aangesien dit nie net die metodes behels wat deur die stelsel gebruik word om sy omgewing te beskou nie, maar ook sy vermoë om kritiese take te verrig. Hierdie ondernemings behels dikwels die koördinering van verskeie substelsels, wat elk sy eie komplekse plig moet verrig. Om sake verder te kompliseer, is dit deesdae toenemend belangrik vir hierdie kunsmatige stelsels om in staat te wees om hul take intyds te kan uit voer.

Die taak van voorwerpherkenning word dikwels beskryf as 'n proses wat die herwinning van die mees soortgelyke voorwerp in 'n databasis aan 'n onbekende navraagvoorwerp behels. Poseskatting,

aan die ander kant, behels die beraming van die posisie en die oriëntasie van 'n voorwerp in drie-dimensionele ruimte, soos gesien vanaf 'n waarnemer se oogpunt. Hierdie twee take word beskou as noodsaaklik vir baie rekenaarvisietegniese en dien gereeld as inset tot meer komplekse persepsie-algoritmes.

'n Benadering wat voorwerpherkenning en die pose beramingsprosedures as onderling afhanklik beskou, word voorgelê. Die kerngedagte is dat uiteenlopende voorwerpe soortgelyk kan voorkom wanneer hulle waargeneem word uit verskillende oogpunte. 'n Eienskappegebaseerde konseptualisering, wat gebruik maak van 'n databasis, is geïmplementeer en gebruik om gelyktydige voorwerpherkenning en poseskatting uit te voer. Die ontwerp sluit datakompresietegniese in, wat oorspronklik voorgestel is deur die beeldverwerkinggemeenskap, om vinnige verwerking van groot databasisse te fasiliteer.

Stelselprestasie word hoofsaaklik gekwantifiseer deur voorwerpherkenning, poseberaming en uitvoeringstyeienskappe te analiseer. Hierdie aspekte is ondersoek onder ideale omstandighede deur gebruik te maak van drie-dimensionele modelle van relevante voorwerpe. Die werksverrigting van die stelsel is ook ontleed vir praktiese doeleindes deur die verkryging van insetdata, wat ooreenstem met wat verkry kan word deur gebruik te maak van kommersiële afstandskandeerders, van 'n gestruktureerde lig-implementasie.

Praktiese eksperimente dui aan dat die stelsel in staat is om gelyktydige voorwerpherkenning en pose beramingsprosedures in ongeveer 230 ms uit te voer sodra 'n onbekende voorwerp waargeneem is. 'n Gemiddelde voorwerpherkenning akkuraatheid van ongeveer 73% was bewerkstellig. Die pose beraming resultate was redelik, maar het verdere navorsing aangemoedig. Die resultate is vergelykbaar met wat bereik is deur ander voorgestelde benaderings soos Standpunt Kenmerk Histogramme en Spin Images.

## LIST OF ABBREVIATIONS

1D	One-dimensional
2D	Two-dimensional
3D	Three-dimensional
3DMM	3D morphable models
AMM	Active appearance model
ANN	Approximate nearest neighbour
BEM	Boundary element method
BRE	Binary reconstructive embedding
CPU	Central processing unit
DDR SDRAM	Double data rate synchronous dynamic random access memory
DEC	Declination
DSI	Disparity space image
EMD	Earth mover's distance
FEM	Finite element method
FIGTree	Fast improved Gauss transform with tree data structure
FPFH	Fast point feature histogram
FU	Functional unit
GB	Gigabyte
GCM	Gray code methods
HDD	Hard disk drive
ICP	Iterative closest point
KDE	Kernel density estimation
LS	Least-square
LCD	Liquid crystal display
LSH	Locality sensitive hashing
MLH	Minimal loss hashing
OpenCV	Open source computer vision

PCA	Principal component analysis
PDF	Probability distribution function
PFH	Point feature histogram
PSB	Princeton shape benchmark
PSM	Phase shift methods
RA	Right ascension
SFF	Shape from focus
SFM	Structure from motion
SFS	Shape from shading
SH	Spectral hashing
SOPE	Simultaneous object recognition and pose estimation
SSE	Streaming SIMD extensions
SVM	Support vector machine
TD	Temperature distribution
VOXEL	Volumetric picture element
VPFH	Viewpoint feature histogram

## LIST OF FIGURES

2.1	Structured light system setup. . . . .	8
2.2	An overview of structured light codification methods . . . . .	11
2.3	Time-multiplexed binary projection pattern. . . . .	11
2.4	Time-multiplexed Gray code projection pattern. . . . .	11
2.5	Spatial neighbourhood projection pattern based on a De Bruijn sequence. . . . .	13
2.6	Spatial neighbourhood projection pattern using M-array. . . . .	13
2.7	Direct pattern codification using grey values. . . . .	15
2.8	Direct pattern codification using colour values. . . . .	15
2.9	An object with its principal axes derived using the entire object. . . . .	19
2.10	An object with its principal axes derived using only the non-occluded portion of the object. . . . .	19
2.11	Common 3D features and their respective computational complexities. . . . .	21
2.12	Suboptimal sampling scheme . . . . .	28
2.13	Improved sampling scheme . . . . .	28
2.14	Initial Bézier surface patch with control points. . . . .	31
2.15	Deformed Bézier surface patch with altered control points. . . . .	31
3.1	Object recognition and pose estimation system diagram. . . . .	49
3.2	An overview of common scene capture approaches. . . . .	50
4.1	Stereo vision reconstruction concept. . . . .	59
4.2	Typical data pre-processing tasks . . . . .	61
4.3	Diagrammatic breakdown of the feature extraction process. . . . .	69
4.4	Shape descriptor represented as a histogram. . . . .	70
4.5	Shape descriptor processed with the KDE methodology. . . . .	70



4.6	Spherical coordinate system used to position a virtual camera around an object situated at $O$ .	74
4.7	An example with 18 carefully selected viewpoints, represented by the red spheres, around $O$ .	74
5.1	Execution times at which features were extracted.	89
5.2	Execution times at which the extracted features were compressed.	89
5.3	Execution times at which coarse, exhaustive, database queries were performed.	90
5.4	Execution times at which fine, selective, database queries were performed.	90
5.5	The object classification accuracies and percentages of the database flagged for further query.	91
5.6	The average viewpoint estimate error.	91
5.7	The average Hausdorff distances for nearest neighbours.	93
5.8	The average Gromov-Hausdorff distances for nearest neighbours.	93
5.9	Pose estimation results, for the model shown in figure 5.11, using 8 bits for compression and a Hamming distance of 4 bits.	94
5.10	Pose estimation results, for the model shown in figure 5.12, using 32 bits for compression and a Hamming distance of 8 bits.	94
5.11	Model of a toy duck.	94
5.12	Model of a human head.	94
5.13	Experimental structured light setup.	96
5.14	Pattern used for system calibration.	96
5.15	Practical point cloud processing times.	96

# TABLE OF CONTENTS

<b>CHAPTER 1</b>	<b>Introduction</b>	<b>1</b>
1.1	Context of the problem . . . . .	2
1.2	Research gap . . . . .	2
1.3	Research objective and questions . . . . .	3
1.4	Hypothesis and approach . . . . .	4
1.5	Research contribution . . . . .	5
1.6	Structure of this dissertation . . . . .	5
<b>CHAPTER 2</b>	<b>Background</b>	<b>6</b>
2.1	Structured light . . . . .	7
2.1.1	Structured light concept . . . . .	7
2.1.2	Coded structured light techniques . . . . .	10
2.1.3	Indexing uncoded light patterns . . . . .	17
2.1.4	Viewpoint coding . . . . .	17
2.2	Shape representations . . . . .	18
2.2.1	Coordinate system . . . . .	18
2.2.2	Shape context descriptors . . . . .	20
2.2.3	Deformable models . . . . .	29
2.3	Shape and pose information retrieval . . . . .	38
2.3.1	Feature-based methods . . . . .	38
2.3.2	Graph-based methods . . . . .	39
2.3.3	Other methods . . . . .	40
2.4	Data compression techniques . . . . .	40
2.4.1	Orthogonal basis decomposition . . . . .	42
2.4.2	Compact binary representation . . . . .	43
2.5	System prerequisite summary . . . . .	47

<b>CHAPTER 3</b>	<b>System overview</b>	<b>48</b>
3.1	Data capture . . . . .	49
3.2	Data pre-processing . . . . .	51
3.3	Feature extraction . . . . .	52
3.4	Database construction . . . . .	52
3.5	Database pre-processing . . . . .	53
3.6	Searching and matching . . . . .	53
3.7	Data post-processing . . . . .	54
3.8	Object and pose presentation . . . . .	54
<b>CHAPTER 4</b>	<b>System implementation</b>	<b>55</b>
4.1	Data capture . . . . .	55
4.1.1	Stereo vision . . . . .	56
4.1.2	Structured light . . . . .	60
4.2	Data pre-processing . . . . .	61
4.2.1	Outlier removal . . . . .	62
4.2.2	Point cloud thinning . . . . .	64
4.2.3	Point cloud segmentation . . . . .	66
4.2.4	Normalisation . . . . .	67
4.3	Feature extraction . . . . .	68
4.3.1	Coarse point cloud representations . . . . .	68
4.3.2	Deformable point cloud representation . . . . .	70
4.3.3	Fine point cloud representations . . . . .	71
4.4	Database construction . . . . .	71
4.4.1	Point cloud representation . . . . .	72
4.4.2	Viewpoint incorporation . . . . .	73
4.5	Database pre-processing . . . . .	75
4.5.1	Tree structures . . . . .	76
4.5.2	Database compression . . . . .	76
4.6	Searching and matching . . . . .	77
4.6.1	Data compression . . . . .	77
4.6.2	Database query . . . . .	79
4.7	Data post-processing . . . . .	80

<b>CHAPTER 5</b>	<b>Experiments and results</b>	<b>82</b>
5.1	Quantifying system performance . . . . .	83
5.2	Comparing point clouds . . . . .	84
5.2.1	Objects as metric measure spaces . . . . .	85
5.2.2	Mass transportation measures . . . . .	86
5.2.3	Geometric distance measures . . . . .	86
5.3	Simulation experiments . . . . .	87
5.3.1	System execution time analysis . . . . .	88
5.3.2	Joint object recognition and pose estimation analysis . . . . .	91
5.3.3	Pose estimation analysis . . . . .	93
5.4	Practical experiments . . . . .	95
5.4.1	Computational improvements . . . . .	98
<b>CHAPTER 6</b>	<b>Conclusion</b>	<b>99</b>
6.1	Implementation . . . . .	99
6.2	Results . . . . .	100
6.3	Future research . . . . .	102

# CHAPTER 1

## INTRODUCTION

The challenge of simultaneous object recognition and pose estimation (SOPE) entails performing two fundamental tasks frequently addressed in the computer vision community. The first is the identification of an unknown object in a scene, while the second involves acquiring an estimate of the position and orientation of the object in three-dimensional (3D) space.

Object recognition and pose estimation tasks regularly form part of complex perception algorithms often intended for intelligent systems [1]. Any system capable of adequately adapting to, and interacting with its immediate environment can be regarded as an intelligent system. Perception algorithms are of particular interest when implementing intelligent systems related to humanoids, driver assistance modules, mining robotics and combat drones just to name a few.

The object recognition and pose estimation problem can be simplified somewhat, by using a database containing instances of all relevant objects. Such an approach would most likely require a extremely large database, which would severely limit system performance. A more general scenario involves the use of an incomplete database, which implies novel objects will have to be approximated by similar objects in the database [1].

A large body of research regarding object recognition and pose estimation is already available. Many of the approaches however consider the two aspects disjointedly and address them separately [2, 3]. One of the main hypotheses of this research is that object recognition and pose estimation are not mutually exclusive. This is evident when considering that different objects might appear similar when observed from appropriate positions and orientations.

## 1.1 CONTEXT OF THE PROBLEM

Pose estimation has been implemented in a vast number of applications which involve both human and machine. Some human applications include facial [4], head [5], and body pose estimations [6], necessary for creating lifelike computer-generated characters, while most intelligent robotics employ pose estimation [7, 8] to assist with navigation [9, 10, 11], object detection [2, 3] and object recognition [1].

Meaningful data, preferably in 3D, need to be extracted from a scene before pose estimation can be executed. Various methods have been proposed to address the challenge of acquiring high-speed 3D measurements. Most of the proposed methods can be characterised as either active or passive vision techniques [12].

Active vision techniques incorporate methods that introduce additional energy into a scene of interest [13], while passive vision techniques only analyse the available energy in a scene [14, 15]. A significant advantage of active vision methods is the rate at which accurate 3D data can be acquired [16]. This makes active vision techniques attractive for real-time applications.

The fairly recent incorporation of autonomous robots into dynamic environments, such as mining environments and densely populated areas, has increased the necessity for rapid and accurate artificial perception systems [17]. Research related to real-time pose estimation addresses this requirement, as pose information forms a crucial part of perception.

The primary goal of the research is to investigate possible means of designing a system capable of performing SOPE in real-time. Artificial perception is known to be invaluable to autonomous intelligent systems, improving their ability to operate effectively in environments such as common households and natural resource excavation sites. Furthermore, reducing system latencies for real-time operation would be vital for adequate operation in dynamic environments.

## 1.2 RESEARCH GAP

Surface matching facilitates the recognition of free-form objects in a scene by comparing a sensed surface to an object's surface stored in memory. Such an approach to object recognition has an advantage over conventional object recognition methods in the sense that it can also provide object

pose information while performing object recognition. A straightforward attempt at surface matching would entail the alignment of two surfaces in a process also known as registration. Such an approach has proved difficult if no initial information regarding the orientation of the two surfaces is known, which motivates additional research.

Shape representations are often used to collate information stored in dense sensed 3D points, also referred to as point clouds, so that surfaces can be compared efficiently. Many different techniques for representing shape information have already been proposed. Finding an appropriate representation, that guarantees robust and accurate surface matching, is however still being actively researched.

There are a vast number of different shape representations. Some may be classified by the number of parameters used to describe each primitive in the representation, while others can be categorised as either local or global representations. The multitude of proposed shape representations perhaps gives an indication of lack of consensus on the best representation for surface matching [18]. The issue of efficiently storing 3D data, which would facilitate fast object recognition and pose estimation, also needs to be addressed.

Recognition of free-form objects from range data has been, and still remains, a challenging problem. Not only does segmenting arbitrary curved surfaces remain ill-defined, but the computational burden often associated with such challenges seriously hampers real-time implementations. Even if the problem is vastly simplified by assuming that only one object is present in a range scan, most range data will still contain erroneous regions due to object characteristics such as self-occlusions [19].

### 1.3 RESEARCH OBJECTIVE AND QUESTIONS

The reaction time of artificial systems operating in dynamic environments is mostly limited by the latency of their perception systems, which in turn restricts their use in everyday applications [1].

The main objective of the proposed research was to develop an accurate object recognition and pose estimation system capable of operating in real-time. The research addresses possible alterations to existing implementations, which would reduce system latencies.

The following questions needed to be addressed:

- What aspects of current artificial vision and perception systems need to be improved in order to address the aforementioned latency problem?
- What improvements can be incorporated into current state of the art object recognition and pose estimation procedures that will reduce execution time?
- What effects would improvements on execution time have on system accuracy?
- Would an acceptable real-time solution be realisable using existing technology?

A deformable model-based approach to SOPE was investigated as an alternative for improving versatility. The results suggest that developing a system capable of performing SOPE in real-time is possible.

#### **1.4 HYPOTHESIS AND APPROACH**

Various approaches have been proposed which are accepted as viable solutions for the pose estimation problem [20, 21, 22]; however most of these implementations lack the ability to operate in real-time. Real-time performance is generally considered to be application-specific, as the application governs the necessary rate of execution. It is believed that by incorporating an active vision system together with dedicated hardware, the aforementioned inadequacy can be rectified.

A deformable model-based approach to pose estimation was investigated since it would significantly increase the versatility of current fixed model-based implementations in view of the inherent adaptation capabilities of deformable models. The proposed approach also relaxes the precision requirements necessary for 3D measurements in order to maintain accurate pose estimations. Such an approach would therefore be better suited to real-time applications, if a real-time implementation is feasible.

The proposed approach was subdivided into three primary components: 3D model acquisition and 3D data acquisition, as well as object recognition and pose estimation. Accurate object models could be obtained from the world wide web, which simplified the model acquisition procedure somewhat. An active vision system, based on a structured light implementation, was exploited. The system was used to acquire 3D range data similar to what is produced by publicly available 3D range sensors.



The acquired 3D range data served as input to the deformable model-based pose estimation procedure. An analysis of existing implementations suggested that portions of the algorithm would have to be executed on dedicated hardware in order to achieve real-time operation.

## 1.5 RESEARCH CONTRIBUTION

Most research that has been done on pose estimation does not acknowledge the constraints associated with real-time requirements. The chief contribution of the proposed research is to address this shortcoming by developing a system capable of performing SOPE in real-time.

The following key contributions were made:

- A novel approach to simultaneously performing object recognition and pose estimation (SOPE) was presented.
- Areas in current state of the art implementations are identified that needs to be improved in order to reduce system latencies.
- A hierarchical search algorithm was implemented to facilitate fast database search operations.
- Three dimensional features were compressed into bit strings and used as object representations at the coarsest scale.
- A deformable model-based approach was investigated to increase the versatility of the system.
- The implementation provides a test platform for future research.

A full article has been submitted to the Machine Vision and Applications journal.

## 1.6 STRUCTURE OF THIS DISSERTATION

Chapter 2 provides a general overview of structured light techniques, shape representation and retrieval methods, as well as data compression schemes. In chapter 3 the suggested object recognition and pose estimation system layout is presented, while the system implementation is discussed in chapter 4. Chapter 5 provides experimental results and the discussion thereof, while chapter 6 concludes the dissertation and presents ideas for future work.

## CHAPTER 2

### BACKGROUND

Most proposed approaches for pose estimation, primarily for object recognition, can be categorised into three classes: (1) model-based, (2) template-matching and (3) feature-based. Traditional model-based techniques cannot be applied to general objects, as such methods assume that an accurate model of the object is available *a priori*. Template-matching methods, on the other hand, tend to be more applicable to pose recognition rather than pose estimation, as numerous templates need to be recorded in advance. Finally, feature-based approaches rely on distinctive points on objects that can be accurately located and tracked. Feature-based methods tend to be computationally intensive and prone to error if not tailored to a specific application [23].

The quality of artificial perception is largely dependent on the effectiveness of the employed data extraction techniques. Proposed methods for capturing a scene in 3D include structure from motion (SFM) [14], shape from shading (SFS) [24], shape from focus (SFF) [15], stereo vision techniques [25], laser scanners and structured light techniques [12]. SFM, SFS and SFF require multiple observations of a scene, often from different viewpoints, which significantly reduces the rate at which scene data can be acquired. Laser scanners produce exceptionally accurate measurements but require relatively expensive equipment. A structured light approach however, provides an acceptable trade-off between speed, accuracy and system cost.

Numerous alternative proposed methods attempt to register 2D captured images continuously with previously obtained 3D data [26, 27]. Many of these methods incur high expense, as they often necessitate the use of advanced equipment, such as magnetic resonance imaging, laser scanning or x-rays, to acquire the 3D data.

A similar approach is one that introduces fiducial markers into the environment, which can easily

be detected in the captured images and thus serve as points of interest [28]. Motion capture used for special effects in the entertainment industry is a well-known example of this approach. Recent developments however strive not to include any artificial elements into the environment in an attempt to simplify system usage and increase versatility.

The captured data often need to be stored and / or altered to accommodate further processing at a later stage. Storing instances of all known objects in a database would be impractical. A more feasible approach would be to store instances of a limited number of objects and perform a nearest-neighbour search when presented with a novel object. The rigidity of conventional nearest-neighbour search can be relaxed by incorporating deformable models [29]. These models aim at improving traditional computer model-based techniques by permitting computer models to undergo controlled deformations, thereby producing more realistic models.

The concept of deformable models has been successfully implemented in a variety of domains which include, but are not limited to, articulated and non-rigid motion analysis, deformable templates, shape matching [3] and recognition, as well as non-rigid medical image registration [30]. Deformable models are frequently employed to characterise object deformations or spatial mappings and are especially useful when working with non-rigid objects.

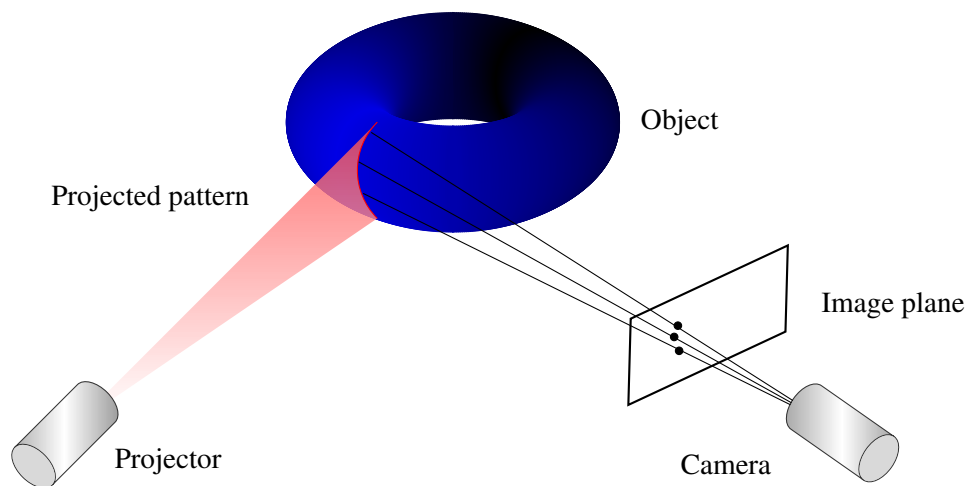
## 2.1 STRUCTURED LIGHT

The structured light method, based on fringe projection, is a well-known 3D shape-measuring method that requires no contact with the object being measured [31]. This process entails the measuring of a 3D object by utilising projected light patterns and a camera system.

A narrow line of illumination can be created by projecting a band of light onto the surface of a 3D object. This line of illumination on the surface of the object appears distorted from perspectives other than the projection point and can be used to reconstruct an accurate estimate of the surface of the target shape [32]. Figure 2.1 illustrates a typical structured light system setup.

### 2.1.1 Structured light concept

A structured light-based system is an active vision system that exploits the triangulation principle. These systems function along the same fundamentals as passive stereo vision systems, with the ex-



**Figure 2.1:** Structured light system setup.

ception that one of the cameras is replaced by a source of controlled illumination, or structured light. The motivation for introducing an illumination source into the design is to simplify the challenge of finding correspondence between stereo images.

A possible implementation is to use a laser and a pair of rotating mirrors to scan a surface sequentially. The position of the spot where the laser strikes the surface of interest can be found as the intersection of the laser beam with the projection ray joining the spot to its image.

A significant difference between passive stereo systems and the aforementioned laser implementation is that the laser spot can typically be identified without any difficulty. This is due to the fact that the laser spot can often be made much brighter than any other scene point. A camera can be fitted with a filter tuned to the rated wavelength of the laser, which in principle should make the camera insensitive to any other energy source. The ability of the system to locate the laser spot in the captured image accurately facilitates the avoidance of any potential correspondence problems that might arise. The Microsoft Kinect sensor embraces this concept with the exception that it employs an infrared projector and camera [13].

An improvement on the laser system discussed above would be to introduce a cylindrical lens into the design, which would transform the laser beam into a plane of light. Not only does this simplify the design of the system, as only one rotating mirror is now required, but it also significantly reduces the time required to scan a scene. The reduction in scan time is due to the ability of the system to acquire a laser stripe, which is equivalent to an entire image column, instead of only a laser dot, the

equivalence of only an image pixel, at each frame. Note that this improved setup does not introduce matching ambiguities, as the laser spot associated with an image pixel can be retrieved as the (unique) intersection of the corresponding projection ray with the plane of light [33, Chp. 21].

Time critical applications might call for additional improvements to decrease data acquisition time further. Projection patterns have been proposed to address this challenge. Projection patterns are carefully designed 2D light patterns, or grids, that are projected by the illumination source and facilitate the identification of multiple image correspondences per frame [12]. An entire scene can thus potentially be reconstructed from a single captured frame, thereby significantly reducing data acquisition time [34]. Complex laser lenses, capable of dispersing the laser beam and creating a grid pattern, have recently become available. Additional improvements can also be incorporated, such as introducing multiple cameras to increase measurement accuracy [35].

The two main drawbacks of active triangulation-based sensors are missing or erroneous data points due to occluded projection points or specularities [33, Chp. 21]. The former cognitive factor also plagues passive stereo systems, as corresponding scene areas are often not visible in all the images captured by the cameras. The latter difficulty tends to be common to most active ranging techniques, as a purely specular surface is characterised by not reflecting light in the direction of the camera unless it happens to lie in the corresponding mirror direction. To complicate matters further, there is a possibility that the reflected beam may induce secondary reflections, which might produce false measurements [33, Chp. 21].

A challenge associated with triangulation-based active range sensors is the problem of keeping the laser strip in focus for the duration of the scan process. The loss of accuracy, inherent to all triangulation techniques as depth increases, also complicates the process [33, Chp. 21]. This is intuitively due to the inverse proportionality between depth and disparity.

Conventional one-shot structured light measurement systems use single black-and-white projection patterns usually encoded according to spatial markings called sub-patterns [12]. The chief advantage of a monochromatic approach is that it produces acceptable results even with strongly coloured scenes. The drawback of the black-and-white approach is however its coarse lateral resolution mainly due to the low transmission capacity of the customary grey encoding schemes implemented. A possible solution to the problem at hand is to incorporate colour-coding schemes which utilise all three primary colour bands. This potentially triples the transmission capacity over monochrome projection

patterns [16].

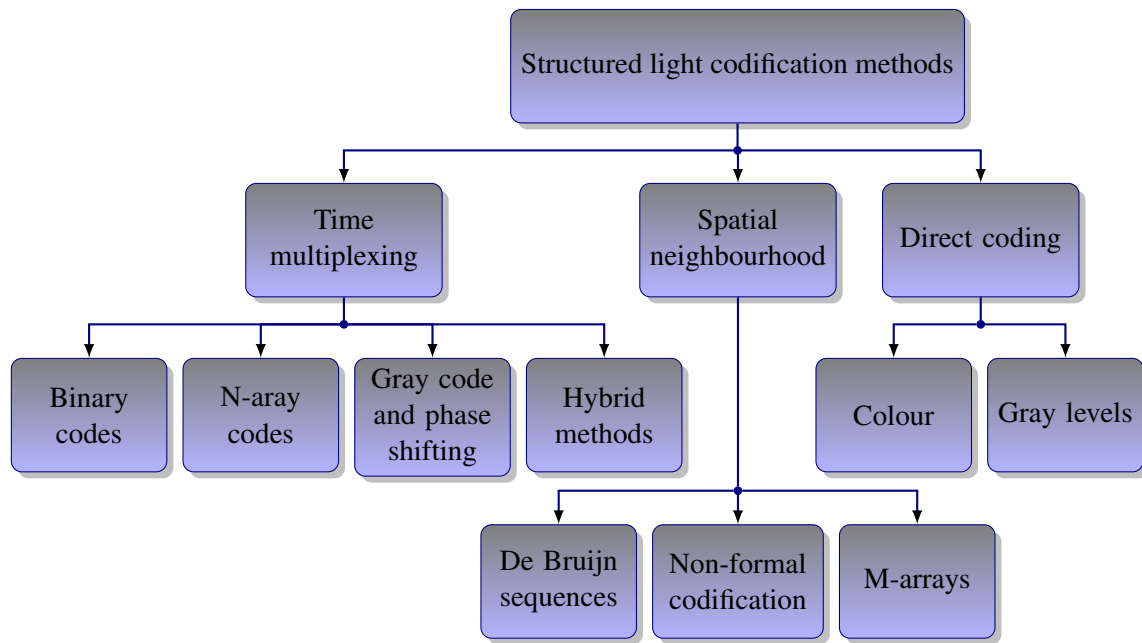
The integration of colour projections into an active imaging system brings about additional complications. Colours reflected from a scene are strongly dependent on the intrinsic reflectivity of the scene. This implies that colour observed by the camera system could differ substantially from the projected colours when implementing a system exploiting the structured light method [16]. A technique known as the rainbow approach was developed in order to compensate for this known difficulty. The rainbow approach employs a projection pattern of monochromatic colours and its encoding using the wavelength [16]. This approach assumes that the reflected light is altered in its intensity but not in its spectral content. Unfortunately mutual reflection and ambient light tend to invalidate this assumption in practice.

### 2.1.2 Coded structured light techniques

A structured light system is based on the notion of projecting a coded pattern, or a set of patterns, onto a target object and analysing pattern deformations [12]. Coded patterns are used to simplify the correspondence problem, which entails identifying correspondences between the projected pattern and the perceived image of a scene or target surface [33, Chp. 21].

The novelty of the patterns is that they are designed to assign codewords to sets of pixels. Every pixel should therefore have its own codeword such that there is direct mapping from the codewords to the corresponding coordinates of the pixels in the pattern [12]. Codewords are simple numbers, which are mapped in the pattern using specific coding techniques such as grey levels, colour or geometrical representations. It is straightforward to realise that the larger the number of points that need to be coded, the larger the codewords would be, which implies that the mapping of such codewords in a pattern is more difficult. The diagram shown in figure 2.2 presents known coding methods used in typical structured light implementations [12].

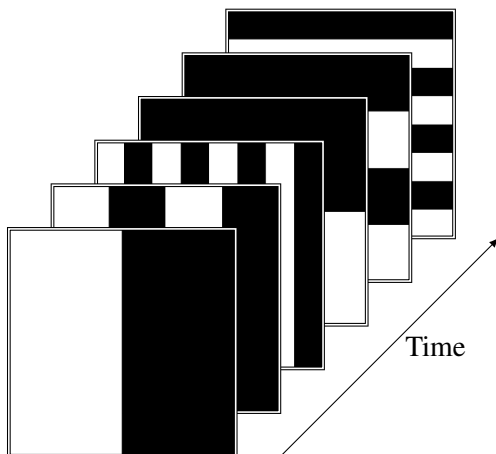
Time-multiplexing coding strategies incorporate different patterns, which are projected onto an object of interest across a specific period of time [12]. The structure of these time-multiplexing patterns could be fairly simple, since multiple patterns contribute to a codeword. Spatial neighbourhood methods aim at encoding different spatial locations of the scene by utilising complex, uniquely structured patterns. Finally, direct codification methods strive to define a codeword for every pixel, which is proportional to its grey level or colour [12].



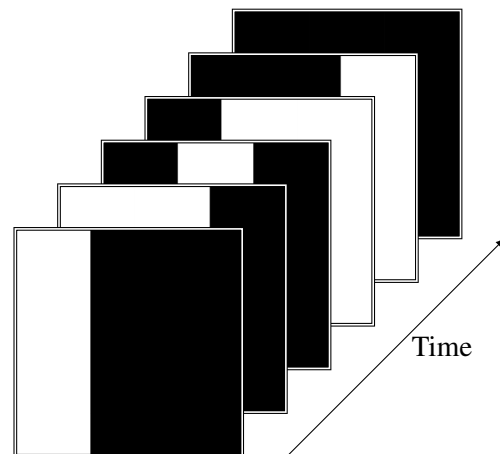
**Figure 2.2:** An overview of structured light codification methods

### 2.1.2.1 Time-multiplexing codification strategies

A common coding strategy is based on temporal coding, which entails successively projecting different patterns on the target surface. The codeword for a specific pixel is formed by the sequence of illumination values projected within a predetermined period of time.



**Figure 2.3:** Time-multiplexed binary projection pattern.



**Figure 2.4:** Time-multiplexed Gray code projection pattern.

In binary coding techniques, as shown in figure 2.3, only two levels of illumination are used, which respectively represent 0 and 1. Each pixel in the pattern has a corresponding codeword formed by the

sequence of 0s and 1s analogous to its value in every projected pattern. A codeword is therefore only obtained once the projection sequence has been completed. It should be noted that this technique only encodes one of the two pattern axes [12].

Plain binary encoded sequence of patterns can be improved slightly by exploiting Gray encoding, as shown in figure 2.4. The advantage of utilising the slightly more complex encoding scheme is that consecutive codewords will have a Hamming distance of one, making the technique more robust to noise [36].

A significant drawback of binary coding schemes is the necessity to project a large number of patterns in order to achieve the desired degree of accuracy, since only two levels of intensity are used in the projections [12]. Although the use of only two intensity levels simplifies image segmentation, it significantly reduces system execution speed. A solution that increases execution speed entails reducing the number of patterns by increasing the number of allowed intensity levels used to encode the stripes [12]. One method is to use an alphabet consisting of multiple grey levels to encode the patterns. An alternative would be to use an extended Gray code, which is based on a special alphabet where each alphabet symbol is represented by a collection of red, green and blue colours [37].

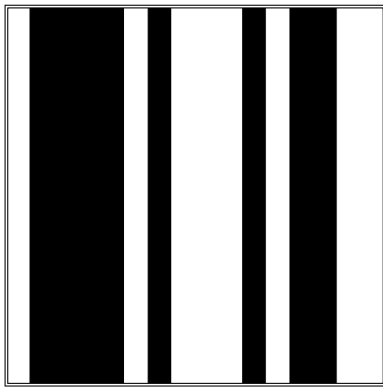
A primary advantage of utilising patterns based on Gray code, binary and n-ary codes, is that the pixel codification is made punctually [12]. This implies that no spatial neighbourhood has to be considered during the codification. The drawback of such methods is that the discrete nature of the patterns limits their range resolution [12]. Phase-shifting methods (PSM) on the other hand, exploit higher spatial resolution as they entail projecting periodic intensity patterns several times by shifting the patterns after every projection [38]. Unfortunately the periodic nature of phase shifting methods introduces ambiguity in the determination of the signal periods in the captured camera images. By integrating Gray code methods (GCM) and PSM into a single strategy, the advantages of these strategies can be combined. The resulting strategy should prove to be unambiguous and robust owing to the use of GCM, but also have high resolution due to PSM. The drawback of exploiting the aforementioned approach is that the number of patterns increases considerably [38].

There are coding methods that exploit multiple patterns and thereby incorporate time-multiplexing techniques, but also factor into account spatial neighbourhood information during the decoding procedure. These techniques are referred to as hybrid methods [12].

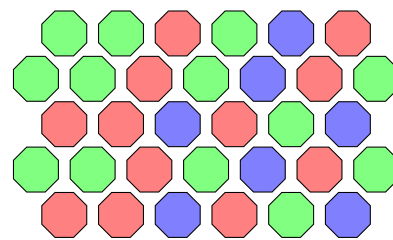


### 2.1.2.2 Spatial neighbourhood codification strategies

The coding techniques in this category tend to concentrate the entire coding scheme in a unique pattern [12]. Each codeword that is associated with a point is determined from the neighbourhood of points around that point. It should be noted that spatial neighbourhoods cannot always be identified and 3D errors might occur, making the decoding stage more difficult. The visual features gathered in a neighbourhood typically include the intensity or colour of the pixels or groups of adjacent pixels in close proximity. Spatial neighbourhood codification strategies have proven to execute significantly faster than time-multiplexing codification strategies, as the number of patterns required is considerably less [12]. Time multiplexing techniques however tend to produce 3D information of higher resolution.



**Figure 2.5:** Spatial neighbourhood projection pattern based on a De Bruijn sequence.



**Figure 2.6:** Spatial neighbourhood projection pattern using M-array.

Non-formal codification strategies entail using neighbourhoods that are generated intuitively. Such techniques propose patterns based on designs that divide the patterns into a number of regions, in which information generates a codeword, without using any mathematical coding theory. Numerous different patterns have been proposed, some of which are discussed in [12].

The non-formal pattern projection techniques were typically generated by following a brute-force approach in order to obtain a pattern with some desirable characteristics. Patterns designed using De Bruijn sequences, as shown in figure 2.5, entail defining neighbourhoods using pseudo-random patterns [12].

A De Bruijn sequence of order  $m$ , consisting of an alphabet of  $n$  symbols, is characterised as a circular string of length  $n^m$  containing each substring of length  $m$  exactly once. Each of these substrings can

be used to uniquely identify an area in a projection pattern. A pseudo-random sequence of length  $n^m - 1$  is defined similarly with the exception that it does not contain the all-zero substring [39]. Various proposed patterns are discussed in [12], most of which incorporate colour slits following De Bruijn sequences in order to encode spatial information.

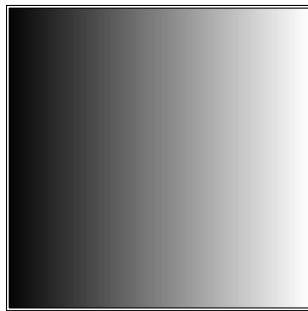
The theory of perfect maps has been embraced by some in order to encode unique patterns by taking advantage of the interesting properties of such matrices. A matrix,  $\mathbf{M}$ , of dimensions  $r \times v$ , where each element is taken from an alphabet of  $k$  symbols, is considered to be a perfect map if  $\mathbf{M}$  satisfies the window property. A matrix satisfies the window property if each different sub-matrix, of predefined dimensions, within that matrix appears exactly once [40]. If a matrix contains all the sub-matrices of predefined dimensions except the all-zero matrix, then that matrix is called an  $M$ -array or pseudo-random array [40]. This special type of array has been widely exploited in pattern codification applications, as shown in figure 2.6, as the window property allows every different sub-matrix to be associated with an absolute position in the array [12].

### 2.1.2.3 Direct codification strategies

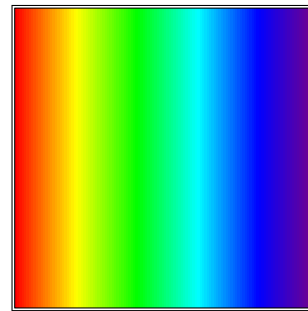
Certain methods have been identified that can be used to design patterns, which facilitates the labelling of every pixel using information presented on it. This implies that an entire codeword for a given point is contained in a unique pixel. Such a method necessitates either the use of a wide range of colour values or the incorporation of periodicity. The drawback of such techniques is that the system is extremely sensitive to noise, as the “distance” between the codewords is nearly zero [12]. The problem of perceived colour is significant and should be taken into consideration. The perceived colour values could differ from the projected colour values because of the intrinsic colour characteristics of the target surface.

Direct codification techniques often require that reference scenes be taken before the pattern projection process commences in order to compensate for any irregularities. These requirements frequently render these techniques inappropriate for dynamic scenes. Furthermore, direct codification strategies are typically constrained to neutral colour or pale objects [12].

Grey level encoding, as shown in figure 2.7, involves exploiting a spectrum of grey levels to encode the points in a pattern [12]. Previous approaches used linear wedges containing a scale of grey levels spread across a dimension of a scene [12]. Pixel locations were typically obtained by calculating the



**Figure 2.7:** Direct pattern codification using grey values.



**Figure 2.8:** Direct pattern codification using colour values.

ratios between the perceived wedge intensities and the intensity under constant illumination at the same pixel. Later, developments introduced the pyramidal intensity-ratio depth sensor or sawtooth sensor. This approach was primarily aimed at reducing the sensitivity to noise of the aforementioned approach.

Colour encoding, as shown in figure 2.8, uses a large spectrum of colour values to encode points in a pattern [12]. These techniques are strikingly similar to those discussed in the previous paragraph, with the exception that colour information is used instead of only grey levels. Some of these methods incorporate time-multiplexing characteristics in order to achieve greater resolution, as was the case with grey level direct codification strategies. Unfortunately such methods are not suitable for dynamic scenes.

#### 2.1.2.4 Codification strategy comparison

It is important to analyse the advantages and disadvantages of the proposed codification strategies. A brief summary, based on data from [12], of the properties of each of the codification strategy categories discussed above is given in table 2.1.

**Table 2.1:** Summary of the properties of the three codification strategies.

Properties of three codification strategies				
	Scene applicability	Object applicability	Equipment requirements	Obtainable resolution
Temporal coding	Static	High	Low	High
Spatial coding	Static/Dynamic	Medium	Medium	Low
Direct coding	Static/Dynamic	Low	High	Medium

Seven different codification strategies were implemented and evaluated in [12]. Three techniques were based on time-multiplexing methods, three techniques were based on spatial neighbourhood coding methods and one technique was based on direct coding methods. These codification techniques were evaluated under identical conditions in order to compare their respective performances.

The time-multiplexing codification strategies yielded the best resolution and accuracy, but also required the most patterns to be projected [12]. The technique that used the most patterns, a set of 14 different patterns, produced the most accurate results, as was expected. These methods are however not suitable for dynamic scenes, but produced the most accurate 3D information of all the techniques evaluated. It should be noted that the time-multiplexing techniques were unaffected by discontinuities in the target surface, unlike the spatial neighbourhood and direct codification strategies.

The De Bruijn sequence-based spatial neighbourhood codification strategy produced the most accurate results, as well as the results containing the highest resolution of the three spatial neighbourhood codification techniques evaluated when the target surface contained no discontinuities. Coding techniques which only encoded one dimension, such as the De Bruijn sequence approach, suffered large amounts of data loss when the target surfaces contained discontinuities. This effect was less severe for techniques encoding two dimensions. All of these codification strategies only made use of one projection pattern, which makes these techniques more appropriate for dynamic scenes and real-time applications. The results obtained using the spatial neighbourhood techniques therefore produced less accurate results than the time-multiplexing techniques but required significantly less execution time.

The implemented direct codification strategy incorporated some time-multiplexing characteristics, as it made use of three projection patterns. The results achieved were slightly more accurate than most of the spatial neighbourhood codification techniques but slightly less accurate than the time-multiplexing codification strategies. It is mentioned that direct codification strategies should be robust against discontinuities in the target surface if no periodicity is included in the patterns. Unfortunately the direct codification technique evaluated in [12] incorporated some periodicity, which caused it to fail when the target surface contained discontinuities. Also, the limited bandwidth of most projectors provoke integration of intensities of adjacent pixels, restricting the resolution of the system. A direct codification strategy therefore tends to be very sensitive to noise.

Techniques capable of locating the pattern stripes with sub-pixel accuracy produced better results than techniques which only located the pattern stripes with pixel accuracy. It is thought that spatial neighbourhood coding strategies failed when the target surface contained discontinuities and the problem could potentially be rectified by incorporating dynamic programming [41]. It was also noted that coding strategies that encoded both axes proved to be more robust against discontinuities in the target surface, as redundancies are incorporated into the pattern.

Finally it was mentioned that the implemented direct codification strategy yielded acceptable results, with the additional advantage of being fairly robust against colourful surfaces [12]. Environment noise should be taken into consideration when developing a structured light system, as the depth per pixel parameter of the system is most likely to decrease as the environmental noise increases. The depth per pixel parameter refers to the number of bits required to accurately indicate the colour, or grey level, of a single pixel.

### 2.1.3 Indexing uncoded light patterns

The approach incorporates an algorithm that indexes uncoded stripe patterns in structured light systems. The proposed method considers all connections and adjacencies among stripe pixels in the form of a weighted directed graph. A maximum spanning tree typically favours clear index transitions, yielding overall optimal indexing. As each stripe in the projected pattern is indexed, its location in the pattern is known, providing sufficient knowledge for 3D information to be acquired [34].

The above-mentioned technique only requires one pattern to be projected, which renders the method suitable for real-time applications. Unfortunately the inherent problem associated with uncoded patterns still exists. The predicament is that discontinuities in the target surface can cause different stripes in the pattern to align from the camera's viewpoint, resulting in erroneous 3D information. It is exactly this drawback codification techniques strive to address.

### 2.1.4 Viewpoint coding

A theoretical framework and practical algorithms for replacing time-coded structured light patterns with viewpoint codes have also been proposed [35]. The approach acknowledges the fact that typical time-multiplexing structured light systems use  $\log(N)$  light patterns, multiplexed in time, to reconstruct  $N$  unique depths unambiguously. An increase in the number of cameras used is proposed. The

notion is that each additional camera introduced into the system may replace one frame in a temporal binary code, which reduces the number of necessary projection patterns.

A theoretical viewpoint coding analysis showed that by exploiting a high frequency stripe pattern and positioning a number of cameras at carefully selected locations, the epipolar projection in each camera can be made to imitate the binary encoding patterns originally projected over time [35]. An attractive feature of such an approach is that it can produce acceptable depth reconstruction without making temporal or spatial continuity assumptions about the target surface being captured [35].

A camera configuration was developed in which possible depths of points in a scene deterministically map to different disparities. It was mentioned that it is possible to distinguish between  $2^k$  depths given that  $k$  different camera positions are available [35]. This implies that  $N$  unique depths can be identified using  $\log(N)$  cameras, which is reminiscent of many structured light methods.

## 2.2 SHAPE REPRESENTATIONS

Many brute force approaches to real-life 3D object representations have already been proposed, including the popular 3D computer models [42]. These digital representations of objects involve storing the location, as numerical values, of thousands of points often accompanied by surface patches, also known as facets. An artificial system would therefore “see” an object as a collection of data points.

Alternative methods for describing objects, or shapes, in digital format have also been suggested, which result in more compact representations. These techniques often incorporate features, or descriptors, which are only relevant to a specific application [43]. A feature-based approach would be more suitable for real-time applications, as processing requirements are significantly reduced from what would be required if a brute force approach is implemented.

### 2.2.1 Coordinate system

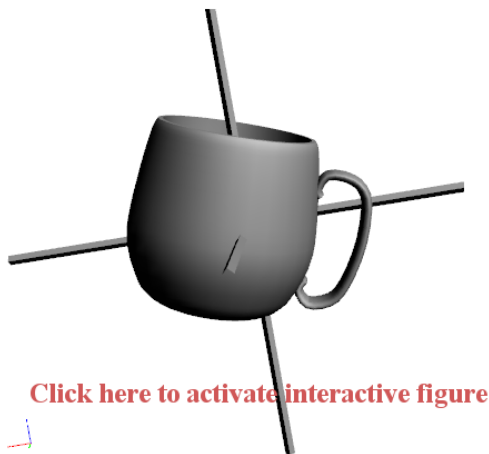
The coordinate system in which surface data are to be described should be considered when deciding which of the various surface representations to incorporate. The two most widely adopted coordinate systems for surface representation are arguably the viewer-centred and object-centred coordinate systems [18].

A viewer-centred coordinate system strives to describe surface data based on a coordinate system dependent on the view of the surface [18]. Although viewer-centred coordinate systems are relatively simple to construct, some cardinal factors should be taken into consideration. Note that the description of a surface will most likely change with a change in viewpoint, and surfaces should be aligned before they can be compared. It is also evident that a separate representation must be available, for example in a database, for each different viewpoint, should a surface need to be represented from multiple viewpoints.

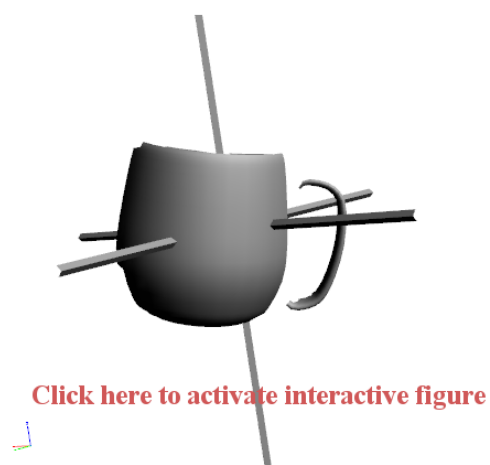
An object-centred coordinate system, on the other hand, attempts to describe the surface of an object in a coordinate system fixed to the object [18]. The description of object surfaces in an object-centred coordinate system is viewpoint-independent, which implies that different surfaces can be compared directly without any alignment prerequisites.

Object-centred coordinate system-based surface representations can be more compact than equivalent viewer-centred coordinate system-based representations, as a single surface representation ideally describes all views of the object [18]. One of the main challenges associated with object-centred coordinate systems is, perhaps ironically, the problem of finding the coordinate system. These systems are generally based on global properties of an object, which might not always be available because of occlusions or noisy data, as illustrated by interactive figures 2.9 and 2.10. The view independence of an object-centred coordinate system arguably prompts its use above a viewer-centred coordinate system given that an object-centred coordinate system can be robustly extracted from the 3D data [18].

Some form of viewpoint or orientation quantisation would be required for storage in either of the aforementioned coordinate systems, whether viewpoint or object-centred. This is to ensure unbiased results when comparing novel instances. A straightforward approach when using a viewer-centred coordinate system would be to translate and rotate a sensed object to predefined coordinates before further processing commences. An object-centred coordinate system, on the other hand, would require all novel objects to be rotated and translated so that their principal axes coincide with a predefined global axes system.



**Figure 2.9:** An object with its principal axes derived using the entire object.



**Figure 2.10:** An object with its principal axes derived using only the non-occluded portion of the object.

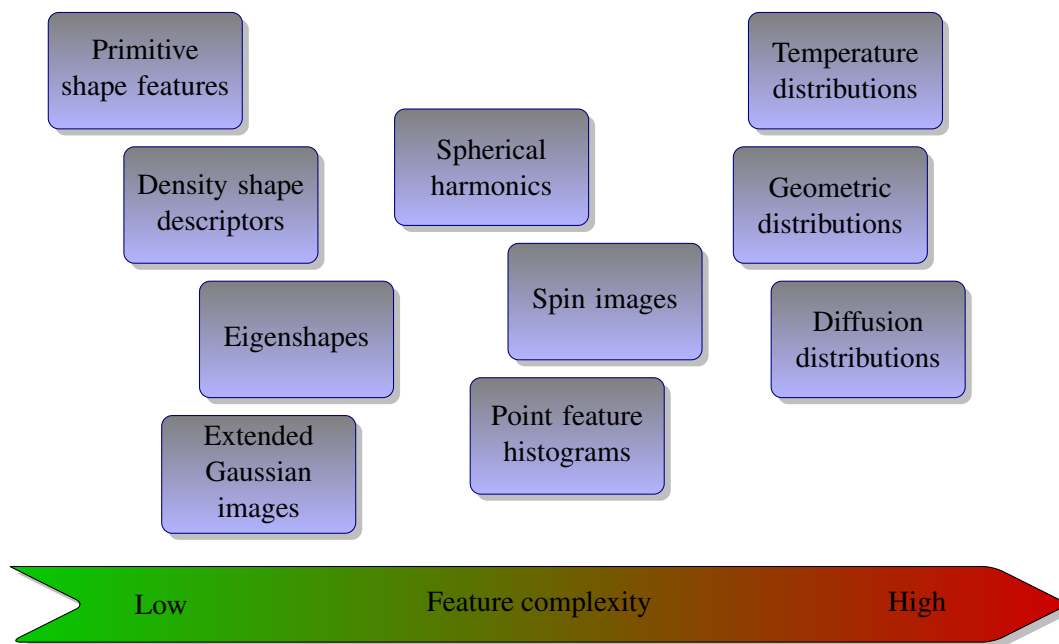
### 2.2.2 Shape context descriptors

A shape descriptor should be able to capture object properties that will ensure acceptable discrimination between objects [44]. The judgement of the similarities between shapes is however somewhat subjective and always depends on user preferences and / or application requirements. For example, it might be sufficient for one application to count only the number of holes seen in an object, while another application might demand additional discriminating information, such as the circumference or outline of a particular shape.

It is inefficient to sequentially match the query object to all the objects in a large database of 3D shapes [45]. Efficient indexing search structures are vital to support effective shape retrieval [46]. It should be kept in mind that the shape descriptor will have to be calculated online, which implies that the computation of the shape descriptor should be fast with respect to the particular implementation. A diagrammatic illustration of popular 3D features, together with their relative computational complexities, is shown in figure 2.11.

Various 3D features can be amalgamated in order to produce descriptors in higher dimensional hyper-spaces. The amalgamation increases the amount of information conveyed by a single descriptor, as additional shape information is encoded in the descriptor. It is however known that at some point the performance of the descriptors will start to decrease as the dimensionality of the descriptor increases. This is a phenomenon better known as the “curse of dimensionality” [47, Chp. 4]. The





**Figure 2.11:** Common 3D features and their respective computational complexities.

following subsections primarily discusses the feature-based approach to surface matching and object recognition.

### 2.2.2.1 Primitive surface features

Primitive features can often be most powerful if used correctly [48]. The computation of such features typically requires low system resources while maintaining rapid throughput. The use of a single primitive feature seldom provides adequate shape information to obtain acceptable discriminative capabilities. It is only when multiple features are combined that the required level of shape discrimination can be achieved. The following primitive local surface features are commonly implemented in feature-based shape-matching techniques [48].

The radial distance,  $R$ , measures the distance of a surface point,  $Q$ , to the origin, or mostly the centre of mass of the shape or object. This is typically not an effective shape feature by itself, but when coupled with other local surface features, it helps to manifest their distribution at varying radii [48].

The radial direction  $\vec{R}$  is a unit vector, collinear with the ray traced from the origin, or centre of mass of the shape or object, to a surface point,  $Q$  [48]. Radial direction is an attractive feature mainly due to it being scale-invariant. The quality of this feature can however be degraded when issues such as occlusions and discontinuities are not taken into consideration.

The normal direction  $\vec{\mathbf{N}}$  is a unit normal vector at a surface point,  $Q$  [48]. This feature, similar to the radial direction, is scale-invariant. Such a vector is typically calculated by fitting a plane to the collection of nearest neighbours to the query point.

The radial-normal alignment,  $A$ , represents the absolute cosine of the angle between the radial direction,  $\vec{\mathbf{R}}$ , and normal direction,  $\vec{\mathbf{N}}$ , and is computed as  $A = \left| \langle \vec{\mathbf{R}}, \vec{\mathbf{N}} \rangle \right|$ . This feature crudely measures how much a query surface deviates locally from sphericity [48].

The tangent-plane distance,  $D$ , resembles the absolute value of the distance between a tangent plane at a surface point and the origin, or centre of mass of the object or shape [48]. This scalar feature is related to the radial distance,  $R$ , by  $D = RA$ .

The shape index,  $SI$ , provides a local characterisation of the shape into primitive forms such as spherical cap and cup, rut, ridge, trough or saddle. A parametrisation often used for shape index is  $SI = \frac{1}{2} - \left( \frac{2}{\pi} \right) \arctan \left( \frac{\kappa_1 + \kappa_2}{\kappa_1 - \kappa_2} \right)$  where  $\kappa_1$  and  $\kappa_2$  are the principal curvatures at the surface point,  $Q$  [48]. Note that  $SI$  is confined within the range  $[0, 1]$  and not defined for a planar patch when  $\kappa_1 = \kappa_2 = 0$ .

Statistical information can also be used to derive basic features. A primitive global feature can be obtained using the statistical moments of the boundary or volume of an object.

### 2.2.2.2 Density-based shape description

A density-based shape descriptor can be obtained by processing primitive feature information with the kernel methodology for density estimation [49]. Using a density-based shape descriptor has many potential advantages. One of these is that noisy data can be accounted for by considering appropriate kernel parameters.

Density-based shape description is a well-known framework used to extract 3D shape descriptors from local surface features characterising an object's geometry. The probability density function (PDF) of a local feature is estimated at specified target points by processing feature information with the methodology for kernel density estimation (KDE) [49]. A shape descriptor vector can then simply be taken to be a discretised version of the resulting PDF. Employing the density-based approach in the aforementioned manner provides a mechanism for converting local shape information, using KDE, into a global shape descriptor [48].

### 2.2.2.3 Eigenshapes

Eigenshape-based approaches to object recognition have demonstrated the ability to recognise large numbers of general objects fairly quickly [50]. These methods aim to encode the variations of the shape and reflectance of an object with respect to its pose and illumination conditions. Recently such an approach has been incorporated into the challenge of 3D object recognition [50].

Let  $\mathbf{Y} = [y_{ij}]$  represent a range image with  $r$  rows and  $c$  columns. It is assumed that the range image,  $\mathbf{Y}$ , contains 2.5D data where each pixel coordinate  $(i, j)$  corresponds to some distance value.  $\mathbf{Y}$  can be converted to an  $n$ -dimensional vector  $\vec{\mathbf{T}} = [y_{1,1}, \dots, y_{1,c}, y_{2,1}, \dots, y_{r,1}, \dots, y_{r,c}]$  by concatenating the rows of  $\mathbf{Y}$ . The resulting vector  $\vec{\mathbf{T}}$  lies in a vector space with dimension  $n$ . The vector space  $I \subset \mathbf{R}^n$  contains all possible range images  $\mathbf{Y}$  [50].

First an eigenspace is constructed from a set of  $m$  training views [50]. A training matrix  $\mathbf{X}$  is obtained by defining each training view  $\vec{\mathbf{T}}_i$  as a column in the training matrix. The goal is then to decompose the  $n \times n$  scatter matrix  $\mathbf{Q} = \mathbf{X}\mathbf{X}^T$  into its corresponding eigenvectors  $\{(\lambda_i, \vec{\mathbf{e}}_i) | 1, \dots, n\}$ . The eigenspace corresponding to  $\mathbf{X}$  is simply the span of  $\vec{\mathbf{e}}_i$  [50].

Once the required number of training samples has been used to obtain sufficient database data, namely the eigenspaces, object recognition can commence. An object is recognised by first projecting the unknown object shape into a query subspace using the set of eigenvectors obtained from the training matrix  $\mathbf{X}$  [50]. This corresponds to mapping the unknown object shape to a point in a multi-dimensional hyper space. The closest point to the unknown object in the hyper space is then considered to be the most similar object in the database [50].

### 2.2.2.4 Extended Gaussian images

A convex polyhedron is fully specified, up to translation, by the area and orientation of its faces [51]. Fortunately area and orientation can be conveniently represented by point masses on a sphere; imagine translating the unit surface normal to each face so that their tails originate at the centre of the sphere [51]. The head of each of the translated surface normals then lies on the surface of the unit sphere, which is also called the Gaussian sphere. Each point on the Gaussian sphere corresponds to a particular surface orientation. The extended Gaussian image of a polyhedron is obtained by placing a mass at each point on the Gaussian sphere equal to the surface area of the corresponding

face [51].

The extended Gaussian image of a shape is not affected by translation of the shape [51]. Rotation of the shape induces an equal rotation of the Gaussian image as the unit surface normals rotate with the shape. Unclosed shapes produce Gaussian images with mass distributions which lie entirely within one hemisphere, that is, the complementary hemisphere is completely empty or sparsely populated. The centre of mass of a Gaussian image needs to be at the origin, which clearly is not possible for unclosed shapes. For more information please refer to [51].

Points on a Gaussian sphere can be associated with points on the surface of a 3D object by finding the point on the Gaussian sphere which has the same surface normal [51]. It is therefore possible to map information associated with points on the object surface onto points on the Gaussian sphere. Such a mapping will be unique for objects with convex surfaces having positive Gaussian curvature everywhere, as no two surface normals will be the same. The mapping from the object surface to the Gaussian surface is in such a case invertible. If however the surface of the object is not convex, then the mapping to the Gaussian sphere will not be unique, as surface information will be lost in the process.

#### 2.2.2.5 Spin images

Spin images are constructed using oriented points, that is, points with associated directions [18]. An oriented point can be defined using the 3D position of the vertex and the surface normal at the vertex. A common technique used to calculate surface normals at specific vertices is to fit a plane to a predefined number of 3D points, which are the nearest neighbours associated with the vertex under investigation.

An oriented point defines a partial, object-centred coordinate system. Two cylindrical coordinates can be defined with respect to an oriented point: The radial coordinate  $\alpha$ , which is defined as the perpendicular distance to the line through the surface normal, and the elevation coordinate  $\beta$ , which is defined as the signed perpendicular distance to the tangent plane defined by the vertex position and surface normal. There is another cylindrical coordinate, the angular coordinate, which can be calculated but is omitted because it cannot be defined robustly and unambiguously on planar surfaces [18].

A spin image is created as follows at each vertex for the oriented point in a collection of 3D points, or point cloud [18]. A 2D accumulator indexed by  $\alpha$  and  $\beta$  is created and all its accumulator cells are reset to zero. The coordinates  $(\alpha, \beta)$  are calculated for a vertex on the surface of an object that is within the support of the spin image. The bin indexed by  $(\alpha, \beta)$  in the accumulator is then incremented, whereafter bilinear interpolation is used to smooth the contribution of the vertex. This procedure is repeated until all the vertices within the support of the spin image have been processed. The resulting accumulator can be thought of as an image, spin image, where dark pixels represent bins that contain many projected points. The position of individual vertices should average out during spin image generation, given that the size of the bins in the accumulator is greater than the median distance between vertices [18].

It is common practice to set the number of rows equal to the number of columns in a spin image to simplify parameter specifications [18]. The result is a square image, which is defined by one parameter. The support distance of a spin image is defined as the image width times the bin size and determines the space swept out by the spin image. The amount of global information contained in a spin image can therefore be controlled by the dimensions, width and height, of the spin image. Decreasing spin image width, or height, while maintaining a constant bin size will decrease the descriptiveness of the spin image, as the amount of global shape included in the spin image will be reduced [18]. On the other hand, decreasing spin image width, or height, will also decrease the amount of clutter that corrupts the spin image.

Spin images can be used for surface matching by constructing a spin image for every vertex in the point cloud. Corresponding surface points are identified by comparing the various spin images, as spin images generated from two different surfaces representing the same object will appear similar because they are based on the shape of the object. The spin images will however not be identical because of the influence of noise.

#### 2.2.2.6 Point feature histograms

Point feature histograms (PFH) encode shape information by analysing pairs of 3D points in a point cloud [52]. Many different relationships between pairs of vertices can be used to construct the PFH. A widely adopted PFH is a histogram that collects the pairwise pan, tilt and yaw angles between every pair of normals on a surface patch. This can be mathematically expressed: for a pair of vertices  $[\mathbf{p}_i, \mathbf{p}_j]$ , and their estimated surface normals  $[\mathbf{n}_i, \mathbf{n}_j]$ , the set of normal angular deviations can be estimated as

follows [43].

$$\alpha = v \cdot \vec{\mathbf{n}}_j \quad (2.1)$$

$$\phi = u \cdot \frac{(\mathbf{p}_j - \mathbf{p}_i)}{d} \quad (2.2)$$

$$\theta = \arctan(w \cdot \vec{\mathbf{n}}_j, u \cdot \vec{\mathbf{n}}_j) \quad (2.3)$$

In the above expressions  $u, v, w$  represent a Darboux frame coordinate system chosen at point  $\mathbf{p}_i$ . The PFH at a patch of points  $\mathbf{P} = \mathbf{p}_i$  with  $i = \{1, 2, \dots, n\}$  captures all the sets of  $[\alpha, \phi, \theta]$  between all pairs of  $\mathbf{p}_i, \mathbf{p}_j$  from  $\mathbf{P}$ , and bin the results in a histogram. Note that such an implementation only relies on surface normal data, which implies that the technique used to estimate these surface normals should be made robust to noise.

The computational complexity of one PFH is  $O(n^2)$  in the number of surface normals, as all possible pairs of points are considered. An alternative implementation, known as the fast point feature histograms (FPFH), was developed to address this problem [52]. The FPFH measures the same angular features as the PFH, but estimates the sets of values only between every point and its  $k$ -nearest neighbours. The results calculated at each vertex are constantly reweighed with the neighbouring histograms, thus reducing the computational complexity to  $O(kn)$ . Note that a point cloud consisting of  $n$  vertices would produce  $n$  PFHs, as a PFH is calculated for each vertex.

It was decided in [43] to incorporate a viewpoint component into the strong recognition capabilities of FPFHs, which produced viewpoint feature histograms (VFH). The viewpoint component is calculated by collecting a histogram of the angles that the viewpoint direction forms with each surface normal. It should be noted that the viewpoint direction is used rather than the viewpoint position itself so as to maintain invariance to scale. A second component measures the relative pan, tilt and yaw as previously discussed but now measured between the viewpoint direction at the central point and each of the surface normals. The computational complexity of VFH is  $O(n)$ , as each point in the point cloud only needs to be visited once. A significant difference between VFHs and FPFHs is that only one histogram is generated per 3D shape when using VFHs as opposed to  $n$  histograms, which would result when employing FPFHs. This implies that a VFH approach to surface matching will require much less computational resources than an equivalent FPFH approach. An FPFH approach, on the other hand, might be more descriptive, as additional local information is retained in the representation.

### 2.2.2.7 Spherical harmonics

Spherical harmonics have been successfully used in literature to describe 3D shapes [53]. Spherical harmonics are the angular portion of a set of solutions to Laplace's equation. Represented in the spherical coordinate system, Laplace's spherical harmonics  $Y_l^m$  are a specific set of spherical harmonics that forms an orthogonal system.

The general solution to Laplace's equation in a ball centred at the origin is a linear combination of the spherical harmonic functions multiplied by an appropriate scale factor,

$$f(r, \theta, \phi) = \sum_{l=0}^{\infty} \sum_{m=-l}^l f_l^m r^l Y_l^m(\theta, \phi), \quad (2.4)$$

where  $f_l^m$  represents the constants and the factors  $r^l$  and  $Y_l^m$  are known as solid harmonics,

$$Y_l^m(\theta, \phi) = N_l^m e^{jm\phi} P_l^m(\cos \theta), \quad (2.5)$$

where  $N_l^m$  represents the normalisation constants and  $P_l^m(\cos \theta)$  are associated Legendre polynomials.

On the unit sphere, any square-integrable function can be expanded as a linear combination of spherical harmonics,

$$f(\theta, \phi) = \sum_{l=0}^{\infty} \sum_{m=-l}^l f_l^m Y_l^m(\theta, \phi). \quad (2.6)$$

In the expression given above  $f_l^m$  represents the spherical harmonic coefficients. It is evident that such an expansion method is closely related to other orthogonal decomposition methods such as, for example, the Fourier transform.

An efficient sampling scheme is vital in order to execute the pixelisation of an object. A sampling scheme based on the astronomers' coordinate system of right ascension (RA) and declination (DEC) is acceptable, as such a sampling scheme could be represented in a compact manner.

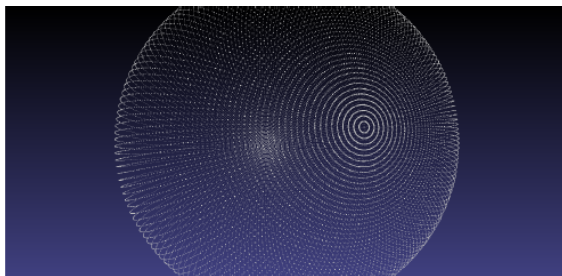
Right ascension is a measure of longitudinal angle in degrees from  $0^\circ$  to  $360^\circ$  increasing counter-clockwise (to the east) around the north pole. Declination is a measure of latitudinal angle in degrees from  $90^\circ$  at the north pole to  $-90^\circ$  at the opposite (south) pole. This is converted into standard

mathematical spherical coordinates  $\theta$  and  $\phi$  as follows,

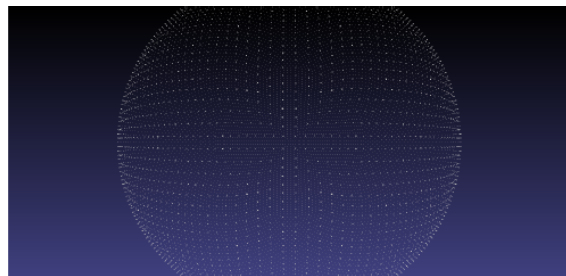
$$\theta = \frac{(90^\circ - DEC)\pi}{180^\circ}, \quad (2.7)$$

$$\phi = \frac{RA\pi}{180^\circ}. \quad (2.8)$$

A straightforward sampling scheme would not suffice and could potentially corrupt any sensed shape data. A simple sampling scheme defining sampling points on the unit sphere according to constant predefined longitudinal,  $0^\circ$  to  $360^\circ$ , and latitudinal,  $90^\circ$  to  $90^\circ$ , increments would be sub-optimal. It should quickly become clear that object areas defined near the polar regions of the sphere are being oversampled while object areas near the equator of the sphere are being under-sampled. A more complex sample scheme was therefore required in order to sample an object at equidistant points on the sphere. A slightly more complex sampling scheme, which resulted in the following points being defined as sampling points on the unit sphere, was used. This sampling scheme is suggested in [54].



**Figure 2.12:** Suboptimal sampling scheme



**Figure 2.13:** Improved sampling scheme

It is evident from the sampling schemes shown in figures 2.12 and 2.13 above, that the improved sampling scheme results in a more evenly sampled object, which should facilitate a more accurate description of the underlying surface.

The process of mapping object surface points to the surface of the unit sphere is yet another concept that needs to be considered. Simply pushing surface points to the surface of the unit sphere might cause significant distortion, especially when dealing with non-starshape-like objects; that is objects that have surface points occluded from their centre of mass by other surface points. Some propositions have been made regarding what the best technique is to accomplish the required surface mapping. Unfortunately all of these propositions are computationally intensive, rendering them impractical for real-time applications.



Once a shape has been mapped to the surface of the unit sphere, its spherical harmonics can be calculated and used to construct a descriptor. These descriptors can then be used to compare the similarities between different 3D shapes. An advantage of using a multi-scale approach, such as the Fourier transform or spherical harmonic transform, is that shapes can be compared at different levels of detail without having to compute additional features. It should however be noted that spherical harmonics tend to be sensitive to irregularities in sample intervals, such as missing data points, and noise.

A natural extension of spherical harmonics is 3D Zernike descriptors which, together with 3D Zernike moments, have been proposed for 3D shape retrieval applications. 3D Zernike moments are obtained by projecting the function defining the shape of an object onto a set of orthonormal functions, 3D Zernike polynomials, within the unit ball [55].

#### 2.2.2.8 Temperature distribution descriptors

Recently, a novel shape descriptor, named the temperature distribution (TD) descriptor, was introduced [56]. This feature is capable of exploring the intrinsic geometric features on a shape. The descriptor is made insensitive to small topological variations and strives to interpret a shape in an isometrically-invariant, shape-aware manner.

The TD descriptor is primarily driven by a heat kernel [56]. It first applies unit heat to each vertex describing the shape, where after it aims to understand the shape by evaluating the surface temperature evolution with time. The extracted information is processed and presented in an one-dimensional (1D) histogram. This histogram can be regarded as a shape signature and can be employed to perform shape matching [56].

#### 2.2.3 Deformable models

The evolution of geometrical models in computer graphics had initially only made provision for the representation of rigid objects. This however changed drastically less than three decades ago. An initial free-form deformation technique was presented, which deformed arbitrary objects by distorting the space in which the objects were contained. Later, the term *deformable models* was introduced when a proposed technique, which incorporated physical properties directly in a graphical object, was presented [57].

Deformable models can be defined in one, two or three dimensions, which respectively entails lines and curves, surfaces and objects. These implementations have been successfully applied in three different areas of research. In object modelling it has been applied to pre-computed animations. In image segmentation it can, for example, be incorporated into a system responsible for automatic 2D interpretation of images obtained from a camera supervising a production line. Lastly, in the field of interactive mechanical simulations, it is employed to emulate the deformable behaviour of non-rigid objects due to external influences [58].

Most deformable models, many of which are intended for surgical simulation applications, can be classified as part of one of three primary categories. The first category consists of ad hoc heuristic approaches, while the second category consists of more technical approaches, which are mainly based on a simplification of the continuum-mechanical model. Techniques that combine the two previously mentioned deformable model methods and are therefore called hybrid methods have also been proposed in [58].

The choice of implementation of deformable models is greatly influenced by the application at hand. Computational-, resource- and execution-rate requirements are just some of the constraints that need to be addressed when attempting a deformable model approach.

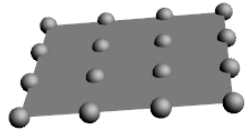
### 2.2.3.1 Heuristic approaches

Heuristic approaches are based on the assumption that classical, relatively exact computation methods, for example finite element models (FEM), are far too complex to yield real-time solutions. This group mainly consists of models that derive the geometry of deformable objects from alternative, rather straightforward modelling schemes that allow for the inclusion of elastic properties [58]. The following heuristic approaches will be discussed in subsequent subsections, Deformable splines, Spring-mass models and Linked volumes.

#### Deformable splines

The general purpose of splines is to obtain smooth and rounded curves, surfaces or volumes, which adjust themselves to a series of control points. A particular curve, surface or volume can be adjusted by altering relevant control points. The Bézier curves and Non-uniform Rational BSplines are some of the more widely known methods among the various existing techniques. Interactive figures 2.14 and 2.15 illustrate how a Bézier surface patch is deformed by manipulating appropriate control

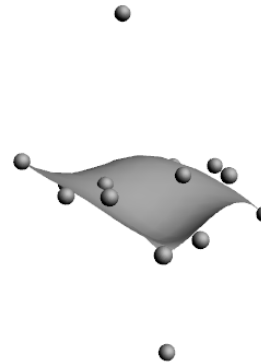
points.



[Click here to activate interactive figure](#)



**Figure 2.14:** Initial Bézier surface patch with control points.



[Click here to activate interactive figure](#)



**Figure 2.15:** Deformed Bézier surface patch with altered control points.

An advantage of deformable splines is the level of control that can be obtained over a shape and the physical properties of an object or mesh. This is primarily due to the elevated number of parameters accompanying such a deformable model. Such an advantage however also brings about some drawbacks. Firstly, the parameters can only be chosen arbitrarily and are often difficult to estimate empirically. A second drawback is the computational cost associated with the model, which can constraint real-time performance. It appears that deformable splines tend to be more complex and computationally intensive than spring-mass models without offering significantly better realism [58].

### Spring-mass models

Spring-mass models are, as indicated by their name, a collection of spring, or spring-damper, elements and discrete mass points. Such a model is conveniently derived from a polygonal mesh representation of an object by simply substituting each vertex with a mass point and each edge, connecting two vertices, with a spring-damper element.

The equilibrium of forces for each node, or vertex,  $i$  in the mesh can be simulated according to the Newtonian law of motion [58],

$$m^i \frac{d^2 \mathbf{r}^i}{dt^2} + \gamma \frac{d \mathbf{r}^i}{dt} \mathbf{f}_{int}^i = \mathbf{f}_{ext}^i. \quad (2.9)$$

In the above expression  $m^i$  represents the mass of node, or vertex,  $i$  while  $\mathbf{r}^i$  represents the current coordinates of node  $i$ . The  $\gamma$  variable is a viscous friction coefficient of the adjacent springs and  $\mathbf{f}_{int}^i$  is the internal force acting on node  $i$ , which tries to maintain the initial position of the latter with respect to the nodes to which it is connected via springs. Finally, an external force, which acts on node  $i$ , is represented by  $\mathbf{f}_{ext}^i$ .

The internal force,  $\mathbf{f}_{int}^i$ , which is opposed to any deformations caused by any external forces, is determined by the degree of deformation of the springs. This can be formally expressed as follows [58]:

$$\mathbf{f}_{int}^i = \sum_{j \in N_i} k_j \frac{|\mathbf{r}^j - \mathbf{r}^i| - |\mathbf{r}^j - \mathbf{r}^i|^0}{|\mathbf{r}^j - \mathbf{r}^i|} (\mathbf{r}^j - \mathbf{r}^i). \quad (2.10)$$

The variable  $N_i$  represents the set of nodes to which node  $i$  is connected via springs in the expression for the internal force  $\mathbf{f}_{int}^i$  given above. Each spring, connecting nodes  $i$  and  $j$ , has rigidity of  $k_j^i$ , while  $|\mathbf{r}^j - \mathbf{r}^i|$  and  $|\mathbf{r}^j - \mathbf{r}^i|^0$  are its current and initial lengths respectively.

A common method for resolving the set of differential equations resulting from the Newtonian law of motion given in equation 2.9, entails the discretisation of the time intervals  $\Delta t$ , which typically employ finite Euler differences. It has been noted that stiffer, more rigid, objects require smaller time intervals to ensure convergence. A more sophisticated time discretisation technique such as the Runge-Kutta method has also been proposed to solve the set of differential equations. This alternative technique is, generally speaking, more stable than the Euler method, which permits the selection of greater time steps  $\Delta t$ , which could result in faster convergence.

An advantage of the spring-mass model is that more sophisticated shape deformations, particularly those of incisions or ruptures, can easily be simulated. Such an occurrence is associated with the removal of appropriate spring-damper elements, or edges, thus eliminating connections that might exist between various mass points, or vertices. It is however advised to also define the interior of a shape, if the possibility of an incision or rupture is sufficiently high.

A significant drawback of the spring-mass model is that its local structure impedes the rapid global propagation of deformations. It is evident that with each iteration step the enforced displacement of one node is only transferred to the next surrounding ring of adjacent nodes. This implies that a significant number of iterations might be required before the effect of a single node displacement is propagated through the entire object. Deformations on one side of an object might therefore only be

noticeable on the alternate side after some, often unrealistic, time delay. It has also proved difficult to maintain an object with constant volume while attempting to perform object deformation [58].

Another disadvantage of the spring-mass model with respect to realism is its tendency to oscillate, as is the case with most iterative structured models. An appropriate choice of time step is vital in this regard. The size of the time step should ensure minimal oscillation while realising rapid convergence.

### Linked volumes

The basic linked volumes model is a straightforward method used to simulate volume objects. This approach is based on the discretisation of the entire volume of a deformable object into evenly spaced, cubic elements. Each cubic element is also assigned a mass value and connected to adjacent cubic elements via spring-damper elements. Such an approach can also be considered as the volumetric extension of the spring-mass model. It should however be noted that the basic linked volumes model would typically be significantly more computationally intensive than that the springmass model. The increase in computational load is primarily inferred from the elevated number of nodes and their accumulated number of connections.

A different approach to volumetric models is the chain-mail model. This model incorporates the same volume discretisation as the basic linked volumes model, but the interconnections between the elements are similar to those found on a chain.

A variation on the volumetric extension of the spring-mass model is based on the discretisation of the entire interior of an object into tetrahedrons. Mass- and spring-damper elements are again respectively placed at the vertices and edges of the mesh. The performance of the resulting model is comparable to that of the basic linked volumes previously discussed. The approach however incorporates continuum mechanics to establish some simplified, linear relationship that best describes the deformational behaviour of the various tetrahedrons. Tensors summarise the stiffness of each of these elements with which the internal forces,  $\mathbf{f}_{int}^i$ , are determined and replace the springs in the spring-mass model, which yields the mass-tensor model. An advantage of the mass-tensor model is that the deformable model is no longer dependent on the topology of the underlying mesh but on the resolution of the mesh.

### 2.2.3.2 Continuum-mechanical approaches

An alternative approach to simulating deformable objects is to base them directly on the laws of continuum mechanics. Continuum mechanics analyse the kinematics and mechanical behaviour of materials regarded as continuous matter and not discrete particles. Such an approach however necessitates that the aforementioned laws be significantly simplified in order to obtain real-time performance.

The simplest relations can be derived when forcing the assumption that a linearly elastic material has small, slow deformations and negligible internal forces such as gravity. If these circumstances can be satisfied, then the resulting differential equation is given by the following second-order Navier equation [58],

$$(\lambda + \mu) \text{grad} (\text{div } \vec{\mathbf{u}}) + \mu \Delta \vec{\mathbf{u}} = 0. \quad (2.11)$$

In the equation above,  $\lambda$  and  $\mu$  are the Lamé constants of the deformable material and  $\vec{\mathbf{u}}$  represents the displacement vector of any point of the object with respect to its initial position. A numerical technique needs to be implemented in order to find the solution of equation 2.11, as no general analytic solution exists. The two most commonly used methods are the FEM and the boundary element method BEM.

#### Finite Element Method

The FEM was originally developed to solve differential equations defined for a certain domain approximately and with some given correspondence boundary condition. First, the entire model is discretised into an acceptable, finite number of elements. Then the magnitude under investigation, in this case displacement, is approximated with polynomial equations over each element and is represented as a function of values at some corresponding control points or nodes. By forcing this quantity to be continuous over element boundaries, an approximate solution can be obtained by minimising the inherent error. The interested reader is referred to [58] for a more in-depth discussion.

A more advanced FEM-based deformable model is the tensor pre-computation model, which is based on the superposition principle. It however adopts the linear elastic hypotheses, which greatly limits its realism for rapid deformations.

### **Boundary Element Method**

Deformable models can also be based on the less well known BEM, which employs so-called fundamental solutions that fulfil the corresponding differential equation within the interior of the respective domain. Boundary value problems are therefore, fittingly, reduced to the boundary of these domains. Such an approach would only necessitate that the boundary, or surface, of an object be discretised, which conveniently coincides with most rendering software. It should however be noted that the mathematics associated with BEM are typically more complex than those of its FEM counterpart.

The BEM unfortunately also adopts the linear elastic hypothesis, which renders it equally limited as the FEM with respect to its realism for rapid deformations. BEM is also equally robust with similar execution time than FEM, but tends to produce more exact results when faced with peaking forces. For additional information please refer to [58]. None of the deformable models proposed thus far, exhibits all of the sought-after characteristics, as all of them suffer from some form of disadvantage or limitation. The interested reader is referred to [58] for a comprehensive comparison between deformable models, which provides a meaningful indication of the differences between the proposed methods.

The comparison mentioned in the previous paragraph suggests that the spring-mass model provides a good trade-off between computation, topology and biomechanical realism and could prove useful in an application where deformable models might be incorporated. It is also noted that the BEM approach shares the best overall score with the spring-mass model approach. It seems that what the BEM approach lacks in biomechanical realism it makes up for in computation and topology. The deciding factor should however be the intended application for which these models are to be used.

#### **2.2.3.3 The application of deformable models to pose and shape estimation**

Humanoids are robots designed to aid humans, giving whatever assistance they might need in any given environment. These mechanical assistants therefore, unlike industrial robots, which are designed to perform activities for somewhat restricted scenarios, not only have to perceive their immediate environment, but also interact in a sensible manner. This task is known to be difficult because of a number of challenges. One of these challenges entails the ever-changing nature of the environment. Deformable models can be a powerful tool to address this difficulty, as changes in objects can be

simulated by, for example, deforming a previously accurate instance of that object.

### Synthetic model generation

One approach to incorporating deformable models into the pose and shape estimation problem is to combine *a priori* 3D shape data in an interactive manner. Assume that 3D models,  $M$ , are given in a boundary representation, that is, by a finite set of triangles. The topology,  $T$ , of the models is defined as the manner in which the graph is described that yields the information on how the vertices are connected to the edges and triangles. The geometric realisation,  $g$ , is defined as the indexed set of 3D vectors corresponding to geometric positions of vertices. A model can be described using its topology and geometric realisation, which can be expressed mathematically as follows [3],

$$M = (T, g). \quad (2.12)$$

If  $M_0, M_1, \dots, M_N$  is a set of  $N + 1$  3D input models with common topology  $\tau$ , then a deformable model can be defined as all the affine combinations of the geometric realisations, which can be stated formally by the following [3],

$$\left( \tau, \sum_{n=0}^N \alpha_n g_n \right) \quad \text{with } \alpha_n \in \mathfrak{R} \text{ and } \sum_{n=0}^N \alpha_n = 1. \quad (2.13)$$

The generic model describes an infinite set of geometric realisations, coded by  $N + 1$  parameters,  $\alpha_0, \alpha_1, \dots, \alpha_N$  where the condition  $\sum_{n=0}^N \alpha_n = 1$  prevents the model from being able to represent the same object in different scales.

### Active Appearance Models

The shape of a 2D active appearance model (AMM) is defined by a 2D triangulated mesh, in particular the vertex locations of the mesh. The shape,  $\mathbf{s}$ , of an AMM is mathematically defined as the 2D coordinates of the  $n$  vertices that make up the mesh. An AMM allows linear shape variation by expressing the shape matrix,  $\mathbf{s}$ , as a combination of a base shape,  $\mathbf{s}_0$ , and  $\mathbf{m}$  shape matrices,  $\mathbf{s}_i$ . The contribution of each shape matrix,  $\mathbf{s}_i$ , is controlled by coefficients,  $p = (p_1, p_2, \dots, p_m)^T$ , known as the shape parameters [59].

$$\mathbf{s} = \mathbf{s}_0 + \sum_{i=1}^m p_i \mathbf{s}_i \quad (2.14)$$



The appearance of the AMM is defined within the base mesh,  $\mathbf{s}_0$ . A convenient notational short-cut is to let  $\mathbf{s}_0$  also denote the set of pixels  $\mathbf{u} = (u, v)^T$  that lie inside the base mesh  $\mathbf{s}_0$ . The appearance of the AMM is then an image  $A(\mathbf{u})$  defined over the pixels  $\mathbf{u} \in \mathbf{s}_0$ . An AMM can then allow appearance variation by expressing the appearance image  $A(\mathbf{u})$  as a combination of a base appearance image  $A_0(\mathbf{u})$  and  $l$  additional appearance images  $A_i(\mathbf{u})$  [59].

The concept of a 2D AMM can be extended to 3D which produces a variation of the 3D morphable model (3DMM). The 3D shape of a deformable model is defined by a 3D triangulated mesh, which is similar to that of a 2D AMM except that an extra dimension has been added.

The appearance of a 3DMM is defined within a 2D texture-map image. In this texture-map, there is a 2D triangulated mesh that has the same topology, vertex connectivity, as the base 3D mesh  $\mathbf{s}_0^{3D}$ . Each 2D mesh point in the texture-map is associated with its corresponding 3D mesh vertex to define a 3D triangulated appearance model.

Three-dimensional models are known to be more compact than their 2D counterparts. Some 2D models may require up to six times as many parameters as a 3D model representing the same shape. It should be noted that 2D models have the capability of representing any geometric phenomenon that 3D models can, with the additional potential of representing unrealisable shapes as well. Such representations are mathematically sound but cannot be acquired physically.

### Comparing deformable models

Algorithms for matching a particular deformable model to a given dataset are typically defined as an energy minimisation problem. A measure of how well the model fits the data therefore needs to be minimised. External energy is usually defined as energy that deforms the model to match the data as well as possible while internal energy, which is mostly known *a priori*, needs to be kept as low as possible. The internal energy simulates the resistance of an object to be pushed by the external forces into directions not coherent with prior knowledge. An optimal solution finds an equilibrium between all applicable forces. Three-dimensional deformable models are also sometimes referred to as 3DMM.

The concept of matching deformable models can be expressed in a more formal framework. A model,  $M$ , needs to be deformed to produce,  $M^*$ , in order to best match a dataset,  $D$ , where the optimally

matched mode,  $M^*$ , minimises the following energy function,

$$E [M] = E_{ext} [M, D] + E_{int} [M]. \quad (2.15)$$

Up to this point the internal energy or prior knowledge of a deformable model has been defined in a very generic sense. Active shape models and 3DMM however incorporate more specific prior knowledge of an object class by learning the typical shapes of various object classes. The principal concept is to assume that all the shapes in an object class are distributed according to some distribution such as the multivariate normal distribution.

### 2.3 SHAPE AND POSE INFORMATION RETRIEVAL

A typical shape retrieval framework consists of an indexed structure, a model database, which is created offline, and a query engine capable of processing queries online. Most 3D models contain too much information to facilitate a straightforward query approach using raw 3D data usually specified in point cloud format. A compact descriptor therefore needs to be obtained for each 3D model in the database, which can be used to identify each 3D model uniquely, in an attempt to reduce query execution time significantly. The aforementioned approach assumes that such compact descriptors are available. An indexing data structure, together with a search algorithm, is required in order to search efficiently through large collections of 3D models online [44].

The online query engine is responsible for generating the query descriptor which best describes an unknown model. The query descriptor is then compared to the descriptors of all the models in the database and only the most similar models in the database are retrieved for further pose investigation or visualisation. The similarity between two descriptors is quantified using a similarity measure.

Most shape-matching methods can be clustered into three broad categories: feature-based methods, graph-based methods and other methods. The interested reader is referred to [44] for a more detailed categorisation and discussion of shape-matching methods.

#### 2.3.1 Feature-based methods

Features denote geometric and topological properties of 3D shapes in the context of 3D shape matching. The intention is to analyse similarities between 3D shapes by comparing various 3D shape

features. Most 3D feature-based shape-matching methods can be divided into four categories, depending on the type of feature used. These are global features, global feature distributions, spatial maps and local features [44].

The first three methods mentioned above rely on the representation of a shape by a single  $d$ -dimensional vector where the dimension,  $d$ , is fixed for all shapes. Each descriptor, representing a shape, can be regarded as a point in a high dimensional space. Two shapes are considered similar if their respective shape descriptors, or points in the high dimensional space, lie within a predefined distance of each other. Retrieving the  $k$  most similar shapes to a query shape can therefore also be considered as solving the  $k$ -nearest neighbour problem in a  $d$ -dimensional hyper space.

The nature of the distance measure and the descriptor space has become non-trivial, especially when working with high dimensional hyper spaces. If a composite descriptor is created by the concatenation of base descriptors, then the resultant composite descriptor may have non-uniform scale and possibly be non-metric. Advanced distance measuring techniques would then have to be employed such as learning distance metrics [60] or scalable clustering [61].

In contrast with the first three methods used in feature-based shape-matching, local feature-based matching methods describe the 3D shape around a number of surface points. A unique descriptor is calculated at each surface point rather than obtaining one descriptor for the entire 3D shape.

The collection of local features do not necessarily provide information of the global shape, unless this is explicitly captured by extracting the geometric arrangement of the features. It is evident that such an approach to shape matching will require significantly higher computational resources than equivalent global descriptor methods, as numerous descriptors have to be compared for each pair of 3D shapes.

Global shape descriptors tend to be less sensitive to finer shape detail than local descriptors, but require significantly fewer computationally intensive algorithms to compare descriptors for different shapes.

### 2.3.2 Graph-based methods

The feature-based shape-matching methods mentioned in the previous section generally only take into consideration the geometry of a shape, which implies, any additional information, such as how distinct

components are linked, might be lost. Graph-based methods attempt to extract geometric meaning from a 3D shape using a graph showing how various shape components are connected. Graph-based methods can be grouped into three categories. These are model graphs, Reeb graphs and skeletons [44].

A significant drawback of graph-based methods is that efficient computation of existing graph metrics for general graphs is not possible. This is due to the fact that computing the edit distance is *NP*-hard and computing the maximal common subgraph is even *NP*-complete [44]. Graph-based methods therefore remain, for the present, too computationally intensive for real-time applications.

### 2.3.3 Other methods

A common approach followed in many view-based shape-matching methods is that two 3D shapes are similar if they look similar from all viewing angles. A natural implementation of such an approach is the incorporation of query interfaces based on defining a query by one or more sketches showing the query from different viewpoints. A descriptor based on this approach is the LightField descriptor discussed in [56].

Deformation-based similarities attempt to identify similar shapes by quantifying the amount of deformation one shape needs to undergo before it completely registers with another shape. Some 2D deformation-based methods rely on natural arc length parametrisation of their contours, which is not straightforwardly generalised to 3D. Based on this, it is known that methods that attempt deformation to shape recovery or shape evolution are very difficult to apply for 3D matching [44].

## 2.4 DATA COMPRESSION TECHNIQUES

The efficient retrieval of information from large databases is considered to be a challenging task common to many data processing problems. Critical aspects that need to be addressed include the processing time required to perform retrieval operations and limited storage space. These aspects are mostly governed by the technology chosen for implementation.

Database retrieval operations often entail performing nearest neighbour searches. A nearest neighbour search is the process of identifying a point in a metric space which is considered to be similar to a query point according to some predefined metric. The problem is regularly formulated as: given a set

of points,  $\mathbf{S}$ , in a metric space,  $M$ , and a query point,  $q \in M$ , find the closest point in  $\mathbf{S}$  to  $q$ .

Nearest neighbour search operations can be computationally expensive and / or resource-intensive when considering moderate to large databases. These requirements can quickly render an implementation impractical because of technology constraints. A common solution to the aforementioned challenge is to relax the similarity constraint and only retrieve approximate nearest neighbours (ANN) [62].

ANN search in large datasets finds its application in a variety of different research fields. In computer vision, for example, it has been used for content-based retrieval [63], object recognition [64] and human pose estimation [65]. An efficient ANN search method can therefore be used to improve database search time significantly. Such an improvement can potentially make a previously impractical implementation feasible.

Sufficient storage space can often be acquired for many basic data storage applications without significant complications. Implementations requiring more complex approaches, such as those intended for mobile platforms, however produce additional challenges. These platforms typically only allow limited storage space to be integrated in the design. Many data compression techniques have been proposed to address the aforementioned challenge.

For this particular research, database query operations are of the utmost importance, as the computational resource requirements for intended implementation should be kept minimal. Methods for constructing compact databases and techniques for efficiently performing search queries can therefore be considered as the main focus.

A system capable of performing object recognition and pose estimation in real-time should consist of at least two major components: i) an effective method for object feature representation and ii) an efficient data structure. It is well known, from results obtained from content-based image retrieval research, that search results greatly rely on the representational power of the chosen features. An efficient data structure is also imperative, as most existing features are high-dimensional. To complicate matters further, it is known that exhaustively comparing a query with entries in a database is extremely slow [46].

Many different data compression techniques have been proposed, which address a variety of storage space limitations. Data compression entails the processing of data such that the result is represented

in a more compact form than the original data. The original data can be obtained from the compressed representation by performing a decompression procedure.

All of the compression methods can be categorised as either lossy- or lossless compression. Lossy compression techniques embrace the approach that some data may be lost during the compression or decompression phase. An exact replica of the original data can therefore not be obtained once the data have been compressed. On the other hand, lossless compression techniques strive to compact the original data in such a manner that no data are lost during compression or decompression.

Most compression techniques were originally developed with the sole purpose of reducing the storage space required. Recently more advanced compression schemes have been proposed, which not only reduce the required storage space, but also facilitate faster processing of the data. Processes such as search queries can now be executed using the compressed representation of the data, which significantly improves system performance.

Only the compression methods outlined in the previous paragraph will be discussed further as these techniques have proven to be more beneficial than straightforward compression schemes. The internal mechanisms of the compression methods are however of less importance to this research, than the usage of the compressed data they produce.

### **2.4.1 Orthogonal basis decomposition**

A common data compression strategy is to map raw data into a different domain using some pre-defined transformation procedure. These procedures typically incorporate orthogonal basis vectors to compute a set of coefficients which represents the transformed data in a compact manner.

After the transformation procedure, only the significant coefficients are stored and used to represent the compressed data. The original data can be obtained by utilising the stored coefficients, together with a different transform which is typically the inverse transform of that used to compress the data.

The degree of loss of data can be manipulated by specifying some threshold value beforehand. This threshold value determines which calculated coefficients can be regarded as insignificant. These insignificant coefficients are then stored as having a numeric value of zero.

Orthogonal basis vector approaches to compression can be tailored to a specific dataset. Numerical methods can be incorporated, which determine the optimal basis vectors that should be used during the transformation procedure. Such a refinement would ensure that the transformation is not only fast and compact, but would also ensure that minimal redundant data are stored.

Database search queries can be executed using only the stored coefficients which represent each database entry. Database entries are considered similar if they have similar coefficient values. In such a case it is possible to perform hierarchical search operations, as some coefficients may be regarded as more important than others. The more essential coefficients can be used to eliminate database entries at an early stage, reducing the number of entries that need to be processed further.

#### 2.4.2 Compact binary representation

Database query operations are significantly influenced by the metric used to define similarity between entries. Similarity measures involving complex mathematical operations would most probably produce more accurate results than less complex similarity measures, but at the expense of a reduced query rate. This should be noted when considering how to store entries in a database.

The Hamming distance,  $\mathbf{d}_{H_m}$ , between two bit strings,  $\mathbf{q}$  and  $\mathbf{q}'$ , is defined as the number of corresponding bits having different bit values. This low-level distance measure can be computed by performing a bit-wise exclusive or operation between the bit strings and counting the number of non-zero bits in the result. Such operations can be executed with astonishing rates on available processing units, making the Hamming distance an attractive metric for database query operations.

Recent research has focussed on obtaining similarity preserving bit strings from extracted features [66]. Similarity preserving implies that the distances between the bit strings obtained for the database entries preserves the distances between the features extracted for the database entries.

Results showed that such an approach would facilitate a fast database search by executing a query operation using the Hamming distance as the metric of choice. Even an exhaustive search would be practical, as the bit count operation can now be carried out in a single instruction using modern central processing unit (CPU) architectures (SSE4.2).

Semantic hashing, introduced by Salakhutdinov and Hinton, is an intelligent method that can be used to store a large database in computer memory efficiently, as well as assist with fast database query

operations [67]. In semantic hashing each database entry is represented by a compact bit string. Each bit string is constructed in a manner that attempts to assign similar bit strings to similar entries. The process of retrieving similar neighbours in the database is reduced to finding all bit strings within a small Hamming distance of the query bit string and returning the corresponding database entries [68].

There are three main challenges associated with semantic hashing. A compact binary code is required that [68],

1. is easily computed for novel inputs,
2. requires a small number of bits to encode the entire database and
3. maps similar database entries to similar codes.

The first challenge is often addressed by deriving a feed-forward network, which can be used to map novel inputs to their corresponding binary code words. The second and third challenges however typically produce more complex solutions.

#### 2.4.2.1 Locality sensitive hashing

The semantic hashing problem can be simplified by assuming that the requirement for compact binary code words can be relaxed. This implies that code words do not necessarily have to be as compact as possible.

Another plausible assumption would be that the database items have already been embedded in a Euclidean space. Such an embedding would imply that the Euclidean distance correlates to the desired similarities. The problem of finding such an embedding has been addressed in a number of machine learning algorithms [68].

The two assumptions mentioned above form the basis for the well known locality sensitive hashing (LSH) method [69]. It is shown in [69] that the Hamming distance between code words would asymptotically approach the Euclidean distance between items, if every bit in the code word is calculated using random projections followed by random thresholds.

Although LSH is a popular semantic hashing method it is most likely to produce inefficient code



words. This is due to the fact that the feed-forward network, used to map the inputs to their binary code words, does not make any prior assumption about the data distribution. It therefore seems logical that any alternative method that exploits such information should produce more optimal code words. Several authors have pursued machine learning approaches to acquire methods other than simply using random projections. Three of such methods are discussed in the following subsections.

### 2.4.2.2 Spectral hashing

An efficient binary code is defined as a code consisting of bits having exactly a 50% chance of being either one or zero. It is also required that the bits be independent of one another in order to eliminate redundancies. Only the codes having a minimal average Hamming distance between similar points and having all the above-mentioned properties should be identified as possible solutions [68].

To obtain a viable solution let  $\{y_i\}_{i=1}^n$  be the codewords, that is, the binary strings of length  $k$ , for  $n$  data points, and  $\mathbf{W}_{n \times n}$  be the affinity matrix. The affinity matrix can be constructed using,

$$\mathbf{W}(i, j) = e^{-\frac{\|x_i - x_j\|^2}{\epsilon^2}}. \quad (2.16)$$

The  $\epsilon$  parameter defines the distance in the Euclidean space which corresponds to similar items. The following problem is obtained by relaxing the independence requirement and demanding that the bits be uncorrelated [68],

$$\begin{aligned} \text{minimise: } & \text{trace}(\mathbf{Y}^T (\mathbf{D} - \mathbf{W}) \mathbf{Y}), \\ \text{subject to: } & \mathbf{Y}(i, j) \in \{-1, 1\}, \\ & \mathbf{Y}^T \mathbf{1} = 0, \\ & \mathbf{Y}^T \mathbf{Y} = \mathbf{I}. \end{aligned} \quad (2.17)$$

The constraint  $\mathbf{Y}^T \mathbf{1} = 0$  obliges each bit to fire 50% of the time while  $\mathbf{Y}^T \mathbf{Y} = \mathbf{I}$  ensures that the bits are uncorrelated.

It is noted in [68] that for a single bit, solving the problem stated above is equivalent to balanced graph partitioning, which is known to be *NP* hard. The problem can be simplified by removing the  $\mathbf{Y}(i, j) \in \{-1, 1\}$  constraint. Now the solutions are simply the  $k$  eigenvectors of  $\mathbf{D} - \mathbf{W}$  with minimal eigenvalue. Note that the trivial eigenvector  $\mathbf{1}$ , which has eigenvalue 0, should be excluded from the solution.

### 2.4.2.3 Minimal loss hashing

The chief task associated with minimal loss hashing (MLH) is to find a hash function that maps high-dimensional inputs embedded in a metric space,  $x \in R^p$ , onto binary codes,  $\mathbf{h} \in H \equiv \{0, 1\}^q$ , which preserves some notion of similarity.

The canonical approach assumes centred (mean-subtracted) inputs, linear projection and binary quantisation. These hash functions, parametrised by the matrix  $\mathbf{W} \in R^{q \times p}$ , are given by,

$$b(x; w) = \text{thr}(\mathbf{W}x), \quad (2.18)$$

where  $\vec{w} \equiv \text{vec}(\mathbf{W})$  and the  $i^{\text{th}}$  bit of the vector represented by  $\text{thr}(\mathbf{W}x)$  is 1 if, and only if, the  $i^{\text{th}}$  element of  $(\mathbf{W}x)$  is positive [66].

The matrix  $\mathbf{W}$  can also be seen as a set of hyperplanes which partition the input space. The  $i^{\text{th}}$  row of  $\mathbf{W}$  determines the  $i^{\text{th}}$  bit of the hash function in the input space; 0 is assigned to points on one side of the hyperplane, and 1 to points on the other side.

The goal of MLH can be explained more formally as follows. Let  $\mathbf{D}$  consist of  $N$  centred,  $p$ -dimensional training points  $\{x_i\}_{i=1}^N$  and let  $\mathbf{S}$  represent the set of pairs for which similarity labels exist. Straightforward binary similarity labels are specified by  $\{s_{ij}\}_{(i,j) \in \mathbf{S}}$  where two exemplars,  $x_i$  and  $x_j$ , are similar if  $s_{ij} = 1$  and dissimilar if  $s_{ij} = 0$ .

Mapping is performed using equation 2.18. The quality of a specific mapping is defined by a loss function  $L: H \times H \times \{0, 1\} \rightarrow R$  that assigns a cost to a pair of binary codes and their corresponding similarity label. The purpose of the loss function,  $L(h, g, s)$ , is to measure how compatible binary codes  $h, g \in H$  are with label  $s \in \{0, 1\}$ . The loss function therefore assigns a small cost if  $h$  and  $g$  are nearby codes and  $s = 1$  and a large cost otherwise. Ultimately, to learn the hash functions,  $w$ , the empirical loss over the training pairs needs to be minimised,

$$L(w) = \sum_{(i,j) \in \mathbf{S}} L(b(x_i; w), b(x_j; w), s_{ij}). \quad (2.19)$$

The loss function used in [66] is specifically intended for learning binary hash functions, and bears resemblance to the hinge loss often used in SVMs. A hyper-parameter  $\rho$  is included, which serves as

a threshold in the Hamming space, distinguishing neighbours from non-neighbours. Such a parameter is important when learning hash keys, since the goal is to map similar training points to binary codes that differ by no more than  $\rho$  bits.

The hinge-like loss function,  $l_p$ , depends on the Hamming distance,  $\|h - g\|_H$ , between codes and not on the individual codes,

$$l_p(m, s) = \begin{cases} \max(m - p + 1, 0) & \text{for } s = 1 \\ \lambda \max(p - m + 1, 0) & \text{for } s = 0, \end{cases} \quad (2.20)$$

where  $m \equiv \|h - g\|_H$ , and  $\lambda$  represents a loss hyper-parameter which governs the ratio of the slopes of the penalties incurred for similar, or dissimilar, points when they are too far apart, or too close. It should also be noted that when similar points are sufficiently close, or dissimilar points are distant, the loss does not impose any penalty [66].

#### 2.4.2.4 Binary reconstructive embedding

Binary reconstructive embedding (BRE) incorporates a loss function that penalises the difference between Euclidean distance in the input space and Hamming distance between binary codes. This loss function can formally be expressed as follows [66],

$$l_{BRE}(m_{ij}, d_{ij}) = \left( \frac{1}{q} m_{ij} - \frac{1}{2} d_{ij} \right)^2. \quad (2.21)$$

In the expression presented above,  $d_{ij}$  represents the Euclidean distance between two input vectors of unit length, while  $m_{ij}$  represents the Hamming distance between two  $q$ -bit binary codes. Hash functions are derived by minimising the empirical loss over training pairs. The BRE approach however, incurs high storage requirements during training which could potentially render large datasets impractical.

## 2.5 SYSTEM PREREQUISITE SUMMARY

A system capable of producing acceptable pose estimations in real-time will have to elegantly integrate all of the ideas mentioned in this chapter. This entails the effective collection of scene data, efficiently processing and storing the data and rapidly servicing novel queries.

## CHAPTER 3

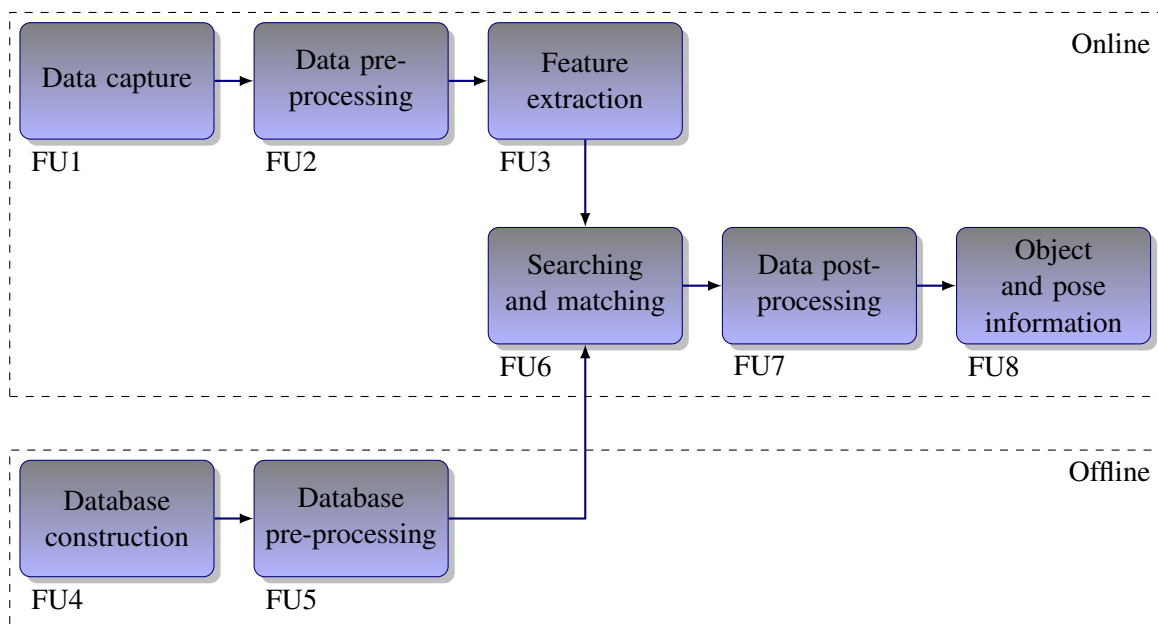
### SYSTEM OVERVIEW

A potential object recognition and pose estimation system would arguably consist of eight fundamental sub-components. Following the divide-and-conquer approach, each of these sub-components is responsible for performing a subset of processes which, when integrated, should solve the pose estimation and object recognition problem.

Computational requirements are of significant importance for all processes that need to be performed online, that is, while the system is expected to produce an output. This is due to the real-time performance requirements of the system. It should be noted that real-time performance is considered to be application-specific and system performance should be analysed accordingly. The system diagram, shown in figure 3.1, presents the eight fundamental sub-components, as functional units (FU), of a potential pose-estimation and object recognition system.

The operation of the system can be explained briefly as follows. A scene is sensed by FU1 and the captured data is passed to FU2. FU2 is responsible for performing pre-processing tasks on the data to ensure the stability and robustness of all subsequent feature extraction algorithms employed in FU3. The extracted features are compared to data in a database in FU6 and the foremost choices are sent to FU7. Data post-processing is performed by FU7, which entails refining the results suggested by FU6 to increase system performance. Finally the results are presented in FU8.

The database used in FU6 to compare the sensed data to, needs be constructed off-line. This is to ensure increased system throughput and comply with real-time requirements. A database is constructed in FU4 by pooling appropriate samples. These samples might either be 2D or 3D samples, depending on the implementation. The database suggested by FU4 is passed to FU5, where it undergoes pre-processing. This pre-processing involves deriving an indexing scheme that would facilitate fast



**Figure 3.1:** Object recognition and pose estimation system diagram.

access to all the elements in the database. The database suggested by FU4 can also serve as a training set and be used to tailor a particular implementation. The resulting indexing scheme is forwarded to FU6 during system operation and is used to perform the required matching operation.

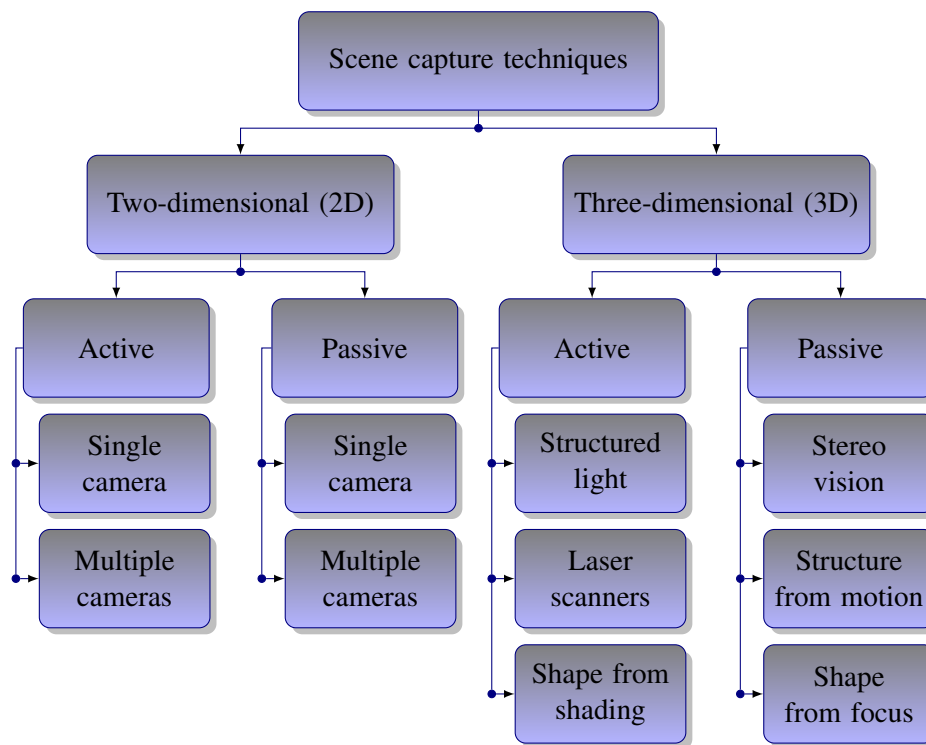
It is worth noting that FU1 and FU8 jointly form the interface of the system. FU1 can be regarded as the system's senses, which should react to stimuli; it is responsible for providing the system with adequate input information. FU8 can be regarded as a user-machine interface, as it should present the results to either a user or a master system. Each of the individual subcomponents is briefly discussed in the subsequent subsections, whereafter they are discussed in more detail.

### 3.1 DATA CAPTURE

The study of computer vision involves methods that can be used to model and understand the way machines perceive their surroundings. Like humans, machines can also extract information regarding their immediate surroundings through visual stimuli. Such stimuli are commonly associated with various radiation sources concentrated in a particular frequency band of the electromagnetic spectrum. A very important band is that of the visual spectrum, which contains the frequencies of electromagnetic radiation humans perceive as visible light.

FU1 is the data capturing subcomponent and responsible for extracting data from the system's im-

mediate environment. The extracted data should be converted into some form that will facilitate rapid processing of the data by the rest of the system so as to comply with real-time requirements. Numerous methods are used to extract data from a scene, ranging from 2D to 3D techniques. The illustration shown in figure 3.2, presents the categories with which many of the most popular methods can be associated.



**Figure 3.2:** An overview of common scene capture approaches.

Two-dimensional methods primarily make use of only one sensor with which data are acquired. The most common sensors used are those sensitive to a specific band of frequencies in the electromagnetic spectrum, such as cameras. The captured data are processed and presented in a 2D fashion. The most common method of representing these 2D data is as an image.

Active techniques introduce additional energy into a scene in an attempt to acquire high-quality data. Such techniques entail, for example, using an additional light source to illuminate the scene if the scene does not have sufficient ambient light. Passive techniques, on the other hand, only use what energy is available when the data are required.

Three-dimensional data capture methods aim at reconstructing the pre-image of a scene. The goal of these methods is to acquire 3D data, which can be regarded as the addition of depth information to

data obtained by a 2D technique. Three-dimensional techniques provide sufficient data such that 3D problems, such as pose estimation, can be addressed directly.

Similar to the 2D case, active 3D methods introduce additional energy into a scene. Structured light techniques project light patterns into a scene and extract 3D information by analysing the deformation of the projected patterns by the scene. Time-of-flight cameras, on the other hand, transmit energy signals and measure the time lapsed before the signal returns to the sensor. Depth information can then be obtained by assuming the recorded time period is proportional to distance.

Most passive 3D data acquisition systems are based on the triangulation principle. Multiple images of the same scene are taken from different viewpoints and the results are compared. Correspondences between the images are then identified and used to extract the required depth information.

### 3.2 DATA PRE-PROCESSING

Most data acquisition techniques have inherent flaws, which can be accounted for by processing the captured data before piping it to more sensitive algorithms. These flaws sometimes result in measurement errors and are due to environmental effects, sensor limitations and scene characteristics [33, Chp. 21]. Even though environmental effects, such as illumination and temperature conditions, can be predicted with high levels of accuracy, it is often still necessary to process the captured data to account for unexpected occurrences.

The pre-processing of 2D scene data often entails correcting effects such as irregular light distribution, shadowing and image distortion. Pre-processing might call for image enhancement techniques such as enhancing important colours or emphasising object boundaries by means of image sharpening. It is also fairly common to align images before further processing commences, in which case some form of image registration would be required.

The pre-processing of 3D data might include the aforementioned 2D pre-processing as well as additional procedures. It is typical for depth sensors to produce data which are corrupted by measurement errors due to scene ambiguities and ill-posed object surfaces. Such data need to be processed in order to correct, or at least significantly reduce, the number of outliers present in the dataset. This data-cleaning procedure is known to be challenging, as uncorrupted data can just as easily be, unintentionally, trimmed from the dataset. The 3D data might need to be segmented into clusters according

to their position in space. Another process which is often included in the pre-processing phase is the normalisation of the captured data.

### 3.3 FEATURE EXTRACTION

A feature-based approach to object recognition and pose estimation was adopted. Not only can features be extracted at rates adequate for real-time processing, features can also be extracted at multiple scales, or resolutions, producing a hierarchical description structure.

Another attractive characteristic of a feature-based implementation is that features can be made less sensitive to practical imperfections, such as measurement noise and object deformations. It would however have to be assumed that some information regarding system imperfections is available *a priori*. This attribute allows features to be tailored to a specific application, which could significantly enhance their contribution and improve system performance.

The idea behind a hierarchical structure is to identify a small percentage of the database using features that are computationally inexpensive to extract. The identified database entries can then be analysed at finer resolutions using more expensive features, in order to refine the results.

Hierarchical descriptions can further increase system throughput, as insignificant objects can potentially be identified without having to perform complex algorithms. For example, consider the scenario where a robot is instructed to collect a mug from an arbitrary environment cluttered with irrelevant objects. A brute force approach would be to identify all the objects in the environment in order to locate the mug. An improved approach would be to assign priority to the objects which most resemble a mug, without having to identify all the objects. Only these objects are analysed further, which reduces the number of objects that need to be identified in the environment.

### 3.4 DATABASE CONSTRUCTION

A database is necessary to compare unknown instances to, in an attempt to identify the unknown instance. Such a database can be constructed by pooling appropriate samples. Multiple 2D and 3D samples of relevant objects need to be collected and pooled in such a manner as to enforce uniformity.

Uniformity, in this sense, simply implies that all the objects in the database should be represented in a



similar fashion. Two-dimensional samples should therefore be used to construct 3D shapes or models if a 3D representation is favoured. Alternatively the 2D projections of the 3D shapes or models should be obtained if a 2D representation is preferred. Uniformity is important, as this would imply that all samples in the database can be processed similarly without having to convert between the different representations.

### 3.5 DATABASE PRE-PROCESSING

The construction of the database is interlinked with the methods that will be used to perform the searching and matching operations. This is evident, as the database should preferably contain all the necessary information required during the matching process, otherwise sensible matching would not be practical. The method by which the database is constructed is therefore dependent on the techniques that will be used to represent each sample. For example, if a feature-based matching approach is adopted, then the corresponding features should be extracted from each sample and stored in the database.

The pre-processing phase should also incorporate some form of compression scheme. The role of the compression scheme is twofold. Firstly, compression schemes allow databases to be stored efficiently, as current databases tend to be very large, often consisting of millions of entries. Secondly, recently suggested compression techniques not only reduce the size of databases, but also facilitate fast processing thereof. Such rapid processing would be vital for real-time performance.

### 3.6 SEARCHING AND MATCHING

Once the captured data have been processed and a database indexing scheme has been made available, a matching procedure should be performed which identifies the most similar element to the sensed data in the database. Searching entails foraging through the database in the most efficient manner possible, while matching involves finding the most similar element in the database to the unknown instance. The concept of similarity is somewhat subjective, as it is always application-specific.

An effective searching scheme is required and should be tailored to a particular database. It is clear that the complexity of the searching scheme will increase as the size of the database increases. This is evident, as an increase in the number of database elements will naturally increase the time required to process all the elements in the database. A complex searching scheme could however obviate the

requirement to process each element in the database, decreasing the search time.

A method commonly adopted for quantising similarity between two sets of data is the use of similarity measures. Similarity measures provide a mathematical framework for comparing datasets which could be implemented on current processing units. The multitude of proposed similarity measures, as well as dissimilarity measures, suggests lack of consensus regarding the optimal similarity measure. A combination of measures might therefore be optimal.

### **3.7 DATA POST-PROCESSING**

The results suggested by the searching and matching subcomponent might not be accurate enough for any given application. Note that it might be feasible to implement a searching and matching subcomponent that is capable of only providing a crude estimate of object class and pose in order to improve system throughput. The data post-processing subcomponent can then decide whether the suggested results are adequate for the application at hand.

Additional refinement is required, should the subcomponent decide that the suggested results are not satisfactory. Such refinement can include post-processing, which entails more advanced algorithms, which would have been too computationally intensive to perform during the initial searching and matching procedure. It is common to perform refinement iteratively in a closed loop configuration until a predefined error specification has been met.

### **3.8 OBJECT AND POSE PRESENTATION**

The final functional unit of the system is responsible for presenting the results suggested by the system. If this is a machine-user interface, the information can be presented on a display device such as a monitor, otherwise the information can be passed to subsequent subcomponents.

The two most important pieces of information that need to be provided is the object type or class and the estimated object pose. The object class or type can be represented by assigning integer values or names to each object class. The pose information can be presented by providing the parameter values of the estimated yaw, pitch, roll and translation of the object according to some reference model.

## CHAPTER 4

# SYSTEM IMPLEMENTATION

An object recognition and pose estimation system was implemented according to the system overview presented in the previous chapter. Practical data were obtained using a structured light implementation employing a binary, time-multiplexed projection pattern.

The captured 3D data, represented as point clouds, were pre-processed to account for measurement errors and enforce uniformity by normalising the data, before extracting features. The extracted features were then compressed and used to query a database which had been constructed offline. The most similar database entries, flagged by the query process, were taken to be estimates of the sensed object, together with its pose. The system was implemented on a Linux operating system and coded in C++ primarily making use of the following third party libraries,

- the Point Cloud Library [70],
- the OpenCV library [71],
- the Eigen library [72],
- the ccSHT library [54],
- and the FIGTree library [73].

### 4.1 DATA CAPTURE

Methods for registering visual stimuli are of particular interest in computer vision, as this is known to be a primary cue used by humans in order to perform endless tasks such as object identification,

navigation and interaction just to name a few. Humans are not only able to acquire vast amounts of visual information, such as scene colours, in a relatively short time span, but are also able to infer spatial information, such as range and object shape, from those data. It therefore seems logical to assume that visual cues would be appropriate sources of input data to a system which needs to interact with an ever changing environment.

A structured light implementation, based on the discussion and source code presented in [74], was used as the primary sensor for acquiring practical scene data during experiments. The implementation consisted of a light source, a projector and a light sensor, a Logitech C100 webcam. The camera, as device, has been studied extensively since its invention in the 16th century. The interested reader is referred to [33, Chp. 1-3] for an in-depth discussion on camera models. Stereo vision is briefly discussed in the following subsections, as the fundamentals of structured light implementations are based on the stereo vision concept.

#### 4.1.1 Stereo vision

Humans are able to gain a strong sense of depth by fusing the images recorded by their eyes and exploiting the difference, or disparity, between them. This task is known as stereopsis and many algorithms that mimic the human ability to perform this task have been proposed [33, Chp. 11]. Computer programs that are able to perform stereoscopic perception reliably are invaluable in visual robot navigation, cartography and aerial reconnaissance, as well as close-range photogrammetry.

Stereo vision entails two principal processes: The fusion of features observed by two, or possibly more, vision sensors, cameras, and the reconstruction of their 3D pre-image. The reconstruction process is considered to be relatively straightforward. The pre-image of corresponding points can, in theory, be found by obtaining the intersection of the rays that pass through these points and the associated camera pinhole. Stereo vision therefore seems relatively simple when a single image feature is observed at any given time instance.

The complexity presents itself when the fact that each picture consists of a vast number of pixels, with potentially tens of thousands of features, such as edge elements and texture, is considered. This complicates matters, as a method capable of acquiring correct correspondences needs to be devised, otherwise erroneous depth measurements will result. Constraints, such as those inferred by epipolar geometry, play a fundamental part in the process of finding correspondences, as they restrict the

search for image correspondences to matching epipolar lines [33, Chp. 10].

Note that stereo vision implementations require that the sensors, or cameras, be calibrated and that their respective intrinsic and extrinsic parameters are precisely known relative to some fixed world coordinate system.

Most computer vision algorithms have to make certain assumptions regarding the physical world and the image formation process in order to produce practical implementations. A common assumption is that all surfaces are Lambertian surfaces, which implies that the surface appearance does not vary with respect to the viewpoint. More complex algorithm models include specific variations of noise and the possible differences in camera gain or bias [25]. The camera gain and bias could also be adjusted by performing simpler image processing procedures, such as histogram normalisation.

One of the desired outputs of a stereo vision algorithm is a univalued disparity map,  $d(u, v)$ , which is a function of the pixel coordinates,  $(u, v)$ , as it is directly related to the differences in the two images [33, Chp. 11]. A concept which is central to such methods is the concept of disparity space, which is a space defined by  $(u, v, d)$ . A disparity space image (DSI) can be formed from a disparity space and is defined as any image or function over a continuous or discrete version of the disparity space. The DSI typically represents the confidence or cost of a particular match implied by the disparity map,  $d(u, v)$ .

#### 4.1.1.1 Key point recognition

Key point recognition entails recognising points of interest across one or more images that may have been acquired from very different viewpoints [33, Chp. 11]. Such tasks are required to address many fundamental computer vision problems, which range from image registration to object recognition.

It is customary to approach the key point recognition problem by incorporating multiple descriptors of local image regions, which are preferably affine-invariant to account for possible image deformations. The extracted descriptors are then compared across images in an attempt to identify possible correspondences. Such an implementation is known to be computationally intensive, since it typically involves fine scale selection, intensity normalisation and rotation correction in order to produce acceptable results [75].

A different approach to the key point recognition problem has recently been proposed, which regards the matching problem as a generic classification problem. Such an approach performs an offline training procedure during which multiple different views of the key points to be matched are used to train randomised trees or classifiers. Each set of possible appearances of an image patch surrounding a key point is therefore treated as a class. The features used as inputs to the classifiers are obtained by executing simple intensity comparison operations. These techniques have proven to be effective for object detection, yielding robustness to viewpoint and lighting changes [75].

#### 4.1.1.2 Binocular fusion

Binocular fusion entails finding correspondences in images of the same scene captured by more than one camera which have been positioned at different orientations with respect to some global coordinate system.

Stereo algorithms tend to perform subsets of the following four steps [25]:

1. Matching cost computation.
2. Cost, or support, aggregation.
3. Disparity computation and optimisation.
4. Disparity refinement.

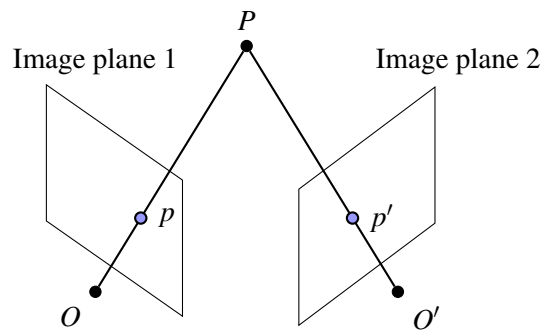
Most of the existing stereo vision algorithms can be clustered into three main categories. The first category involves local, or window-based, algorithms where the disparity at a given point solely depends on the values of neighbouring pixels within a finite window. Such implementations make implicit smoothness assumptions by aggregating support.

The second stereo vision algorithm category entails global algorithms, which make explicit smoothness assumptions, whereafter the problem is solved and optimised. These algorithms typically do not perform aggregation but rather attempt to obtain a disparity assignment that minimises a predetermined global cost function, which combines data and a smoothness term. Global algorithms have only recently become practical owing to their high computational demand, whereas local algorithms tend to trade accuracy for speed.

The other major stereo vision algorithm category consists of cooperative algorithms which are primarily inspired by computational models of human stereo vision [25]. These algorithms strive to perform local computations iteratively and incorporate non-linear operations. The behaviour of such algorithms is similar to global optimisation algorithms.

#### 4.1.1.3 Reconstruction

The process of reconstruction involves obtaining the 3D pre-image of a scene which has been captured by two or more cameras. This can be illustrated as shown in figure 4.1.



**Figure 4.1:** Stereo vision reconstruction concept.

It is, in principle, straightforward to reconstruct the corresponding scene point  $P$  from images captured by two or more cameras, given that the stereo rig is calibrated, its intrinsic and extrinsic parameters are known, and the two matching image points  $p$  and  $p'$  are given. In theory this is performed by finding the intersection of the two rays defined by  $R = Op$  and  $R' = O'p'$ . The optical centres of the cameras are respectively represented by  $O$  and  $O'$ . Practical interferences however complicate the matter somewhat. The rays  $R$  and  $R'$  will most probably never intersect in practice because of practical inaccuracies such as calibration and feature localisation errors.

Various methods have been proposed that address the aforementioned complication. One approach is to construct a line segment which intersects and is perpendicular to both rays  $R$  and  $R'$ . The mid-point of this line, taken at the closest point between the two rays, can be considered as the pre-image of the 3D point  $P$ .

An alternative approach to reconstructing a scene point entails a pure algebraic formulation. If it can be assumed that the projection matrices  $\mathbf{M}$  and  $\mathbf{M}'$  of the individual cameras, together with the matching points  $p$  and  $p'$ , are known, then the constraints  $z\mathbf{p} = \mathbf{M}\mathbf{P}$  and  $z'\mathbf{p}' = \mathbf{M}'\mathbf{P}$  can be rewritten

as follows [33, Chp. 11],

$$\begin{aligned} \mathbf{p} \times \mathbf{M}\mathbf{P} = 0 \\ \mathbf{p}' \times \mathbf{M}'\mathbf{P} = 0 \end{aligned} \Leftrightarrow \begin{pmatrix} [\mathbf{p}_\times] \mathbf{M} \\ [\mathbf{p}'_\times] \mathbf{M}' \end{pmatrix} \mathbf{P}. \quad (4.1)$$

The system shown in equation 4.1 is an over-constrained system of four independent linear equations in the coordinates of  $\mathbf{P}$  which can be solved with relative ease using linear least-square (LS) techniques. Unfortunately this approach to reconstruction does not have an obvious geometric interpretation. The advantage of using this implementation is that it generalises readily to the case where more than two cameras are used. Each new camera image simply introduces two additional constraints.

A third method used to perform reconstruction involves reconstructing a scene point  $P$  with associated image points  $p$  and  $p'$  as the point  $Q$ , which has associated image points  $q$  and  $q'$ , by minimising the following error,

$$E = d^2(p, q) + d^2(p', q'). \quad (4.2)$$

This approach is different from the previously mentioned reconstruction methods in the sense that it does not allow the closed-form computation of the reconstructed point. This implies that the scene point needs to be estimated using some form of non-linear LS technique. The approach can be combined with any of the previously discussed methods by using the reconstruction obtained by either of the other two methods as a reasonable guess to initialise the optimisation process. It is worth noting that this non-linear approach also readily generalises to the case of multiple images [33, Chp. 11].

#### 4.1.2 Structured light

The structured light approach to scene reconstruction is similar to the stereo vision approach discussed in the previous subsection. The exception is that one of the cameras is replaced by a controlled light source. The scene reconstruction process is identical to what is explained in section 4.1.1.3, except that one of the incoming rays is replaced by the projected ray originating from the light source.

A structured light approach was adopted specifically to simplify the key point recognition problem. Areas in a scene were encoded with binary codewords using binary, time-multiplexed projection



patterns. These binary codewords were identified in a set of images captured by the camera and facilitated the process of finding point correspondences.

The implementation is based on the assumption that the projected light patterns are much brighter than the illumination sources in the scene. The assumption ensures that the projection patterns, and their deformations due to depth variations in the scene, can easily be identified in a captured image. The structured light approach also renders the implementation significantly less sensitive to colour variations in the scene. Finding correspondences between the projected patterns and captured images was therefore reduced to simply extracting and reading the binary codewords in the images.

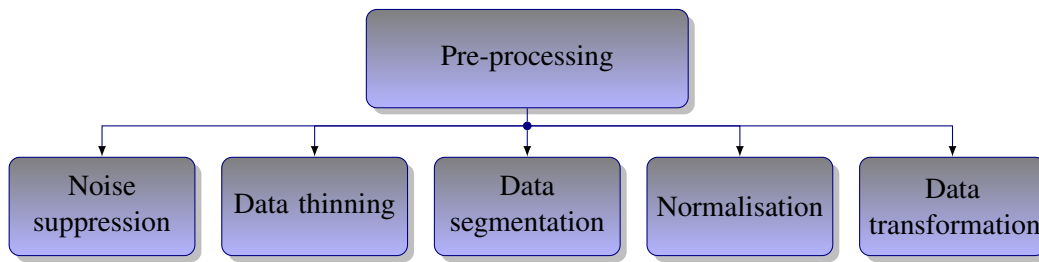
Structured light implementations, like stereo vision implementations, need to be calibrated in order to obtain system parameters. The camera calibration process described in [76] was used to estimate the required camera and projector parameters. The pre-processing of captured images necessary to negate camera imperfections, such as the various distortions and irregular image light distributions, was also performed to increase system accuracy.

Note that although the abovementioned structured light implementation does not adhere to real-time requirements, its data closely resemble those produced by commercial range scanners capable of operating in real-time. This particular structured light implementation was only used to acquire practical data which would serve as input to the object recognition and pose estimation subsystem.

## 4.2 DATA PRE-PROCESSING

The process of data pre-processing is common to most, if not all, signal processing approaches. This procedure involves a number of different tasks that need to be performed on the input data before more complex, often more sensitive, processing can commence. These tasks might include operations from identifying irregularities in sensed data to projecting the incoming data to different dimensions, as illustrated in figure 4.2.

It is often observed that measurement errors introduce sparse outliers into sensed data. These artefacts can be caused by non-ideal object surface characteristics, ambiguities present in the measured data, unforeseen reflections due to environmental setups, etc. Outliers such as these complicate the estimation of local point cloud characteristics and need to be addressed before additional processing on the sensed point cloud commences.



**Figure 4.2:** Typical data pre-processing tasks

To complicate matters further, typical range sensors generally produce point clouds of varying point densities. Factors such as the type of sensors used, alternating sensing distances, object surface characteristics and occlusions contribute to varying point densities. Such variation inherent to the data capturing process can cause algorithms to function less robustly and even become unstable.

Appropriate features were extracted once the sensed data had been processed to account for measurement errors and normalised. After the feature extraction procedure, the pre-processing process was concluded by compressing the extracted features in a manner that would facilitate fast database query operations.

#### 4.2.1 Outlier removal

Measurement errors corrupt the sensed data and complicate the estimation of local point cloud characteristics such as surface normals and curvature changes. Any sensed data should therefore be processed with the intention of removing outliers before passing the data to more complex algorithms.

The removal of outliers from a point cloud is known to be challenging as object points, which are considered to be non-corrupted data points, can easily be removed in the process. This is evident when considering phenomena such as occlusions. It might happen that an area of an object appears to be dislocated from the object itself owing to self-occlusions. Some outlier removal algorithms might flag such points as outliers and remove valuable information from the data.

##### 4.2.1.1 Statistical outlier removal

Outliers, or measurement errors, can be rectified by incorporating statistical analyses on individual points in a point cloud [47, Chp. 10]. Statistics are used to analyse the characteristics of a neighbour-

hood of points around a point of interest. Any points that do not satisfy statistical criteria are flagged as outliers and removed from the point cloud.

Statistical analyses are especially effective at removing isolated points in a point cloud. Such points can constantly be classified as outliers when working under the assumption that relevant object surfaces cannot be represented by only one point.

A statistical-based outlier removal approach was adopted, which utilises relative point distances. A distribution of point-to-neighbour distances is computed. This involved estimating the mean and standard deviation of the distances of a point from all its neighbours. The resulting, estimated mean and standard deviation values can then be taken as the actual mean and standard deviation parameters of a Gaussian distribution. Any points with mean distances outside a predefined interval, which can be considered as global mean and standard deviation parameters, were considered as outliers and trimmed from the point cloud [70]. The statistical outlier filter provided in the PCL [70] was used in the implementation.

#### 4.2.1.2 Conditional outlier removal

A conditional outlier removal filter identifies a point as an outlier and discards the point from a point cloud if it violates any condition specified in a set of constraints [70]. Such a filter is similar to a point pass filter, discussed in the next section, with the exception that additional constraints can be included to facilitate advanced control over how points are rejected from a point cloud. It is a simple method for eliminating potentially corrupted data points.

For example, a set of constraints might define an outlier as a point having a  $y$ -axis coordinate greater than 2 while at the same time having an  $x$ -axis coordinate smaller than 1. If any of the points present in a point cloud satisfies the abovementioned requirements, then the point is discarded from the point cloud.

It is straightforward to deduce that a conditional outlier removal filter can be employed to isolate a specific scene area, which can then be processed separately from the rest of the data in the dataset at a later stage. Note that the constraints characterising a particular conditional outlier removal filter do not necessarily have to be limited to the dimensions of the data. Additional constraints, such as those related to neighbouring points, can also be included to increase the efficiency of the filter.

The conditional outlier removal filter provided with the structure light implementation [74] used to acquire scene information was used in the pre-processing module. This filter rejected any 3D points that did not fall within a predefined distance from the sensor.

#### 4.2.1.3 Radius outlier removal

Outliers can also be removed based on the number of neighbouring points located within a predefined radius from a point of interest. This is accomplished by counting the number of points located within a globally specified radius at each point in the point cloud. A point is then removed from the point cloud if it does not have at least this number of neighbours within the globally specified radius [70].

It is straightforward to deduce that a radius outlier removal filter can be used to trim isolated points from a point cloud. This is also the intention of a statistical outlier removal filter. Radius outlier removal filters can process data at higher rates than statistical-based filters, as fewer calculations are required. Inspection showed that statistical-based analyses produced more accurate results as they incorporate underlying point cloud characteristics. The statistical outlier removal filter was therefore incorporated and not the radius outlier removal filter.

#### 4.2.2 Point cloud thinning

Time-critical applications should only accept the most informative data as inputs in order to be able to ensure that vital tasks are performed as rapidly as possible. Any inputs that provide redundant information or that could result in unnecessary processing should therefore be identified as quickly as possible and discarded before more intensive operations commence.

Current range sensors strive to produce dense 3D data of an observed scene. It could be possible for the sensor to produce a point cloud that is too densely populated for the purpose of a particular application. An approximation of the surface of an object could, for example, be sufficient to perform crude navigation operations. It is therefore appropriate to investigate proposed techniques that could be used to discard redundant 3D vertices present in a point cloud. Note that the choice of a data removal implementation, if any is considered, should be application-specific. This implies that sufficient knowledge regarding the characteristics of the sensor that will be used in the implementation should be available before any data removal is considered.

#### 4.2.2.1 Point pass filter

The point pass filter is a straightforward filter that can be applied to reduce the number of data points present in a point cloud. The filter removes 3D vertices from a point cloud based on predefined prerequisites along a specific dimension. Vertices in a point cloud can therefore be removed from a given point cloud if they are positioned either inside or outside a specified range [70].

One application of the point pass filter would be to cluster a sensed point cloud into sub-point clouds according to a collection of predefined range intervals. Such an implementation is especially useful in robot navigation, as objects closer to the robot can be isolated and processed differently from objects located further from the robot. A point pass filter was implemented to isolate objects in the operational region of the 3D range scanner.

A point pass filter can be employed efficiently if there is prior knowledge of the environment. For example, if for some reason it can be assumed that the majority of objects present in a scene will be clustered at specific range intervals, then a point pass filter can be used to extract the objects at the various distances. Alternatively, objects in close proximity to the sensor can be processed with higher priority than objects at a distance, and so forth.

A conditional outlier removal filter is similar to a point pass filter since both filters discard points from the point cloud when they do not satisfy predefined conditions. There was therefore no need to implement a point pass filter for this design since the conditional outlier removal filter, discussed previously, already performed a similar task.

#### 4.2.2.2 Voxel grid filter

A volumetric pixel, or more correctly a volumetric picture element or voxel, is defined as a volumetric element which represents a value on a regular grid in 3D space [70]. A single voxel can therefore be regarded as a tiny 3D box positioned somewhere in 3D space.

A voxel grid filter reduces the number of points in a point cloud by means of resampling. The initial step in the resampling process involves creating a voxelised grid over the original point cloud. The size of the grid elements, the voxels, is important, as this parameter controls the resolution at which resampling is performed. Once a grid has been constructed, resampling is performed by substituting

for all the points located inside a voxel, a single point which is positioned at the centroid of these points [70].

The centroid of a collection of vertices is defined as the centre of gravity of the points, which can be calculated as follows,

$$\bar{\mathbf{x}} = \frac{1}{N} \sum_{i=1}^N \mathbf{x}_i. \quad (4.3)$$

A more crude approximation can be obtained by simply approximating all the points in a voxel by the centre of the voxel. Although this method is less computationally intensive than approximating the collection of points by their centroid, it tends to produce a more distorted approximation of the point cloud, which might discard vital underlying surface information. The surface of an object can therefore be represented more accurately using the resampling method which substitutes for a collection of vertices their centroid instead of their corresponding voxel centre.

The voxel grid filter provided in the PCL [70] was incorporated in the system to perform the necessary task of point cloud thinning. A resampling method that substituted for a collection of vertices their centroid was used due to the advantages discussed in the previous paragraph.

### 4.2.3 Point cloud segmentation

Most object recognition algorithms tend to perform optimally when presented with a point cloud consisting of vertices belonging to only one object. Such an assumption would certainly be violated in most practical applications, as real world scenes often contain more than one object that introduces clutter into the environment. Computer vision algorithms therefore need to formulate some form of point cloud segmentation strategy in order to suppress clutter in a sensed scene. Many of the proposed strategies involve clustering algorithms, which aim at collecting vertices and forming groups based on their similarities.

Range segmentation is a straightforward technique that can be used to cluster a sensed scene into various areas of interest. The segmentation is primarily based on the proximity of objects in the scene to the sensor. An example of range-based segmentation would be the implementation of a conditional outlier removal filter or point pass filter, as discussed previously. Such an approach was adopted for this implementation. Objects were isolated by assuming that they could be segregated from a scene based on depth information.

More advanced segmentation techniques incorporate clustering methods such as agglomerative, divisive and  $K$ -means clustering, to name just a few. The interested reader is referred to [33, Chp. 14] for an additional discussion. It should be noted that fairly complex segmentation techniques, such as those exploiting clustering approaches, often incur significant resource requirements and tend to be computationally intensive. Because of the aforementioned possibilities, such segmentation techniques were noted, but not integrated in the development of the system.

#### 4.2.4 Normalisation

It can be assumed that the sensed 3D data, representing an object in a scene, will differ considerably from the ideal data stored in a database. This is due to environmental effects such as occlusions, sensor imperfections and orientation, etc. System performance can be improved significantly by including normalisation algorithms in the pre-processing phase. These algorithms strive to realise an environment where data can be compared in a sensible manner. For example, the size of an object should not hamper system performance if the goal is to assign unknown objects to crude classes.

##### 4.2.4.1 Coordinate normalisation

The two most widely adopted coordinate systems for surface representation are arguably the viewer-centred and object-centred coordinate systems [18]. Many shape-matching algorithms embrace a viewer-centred coordinate system, which requires that objects be normalised with respect to a predefined coordinate system. This typically entails rotating and translating the object to some reference point from which similarities can be established. Any additional processing, such as feature extraction and matching, is then performed using the normalised data.

Object-centred coordinate system-based surface representations can be more compact than equivalent viewer-centred ones. A common approach is to incorporate a method such as principal component analysis (PCA) to identify the principal axes of an object. The object is then normalised by rotating the object such that its principal axes coincides with the three principal axes of some predefined coordinate system such as the Cartesian space. These systems are generally based on global properties of an object, which might not always be available because of occlusions or noisy data.

A viewer-centred coordinate system was adopted in this case because of the real possibility of not being able to extract coordinate systems robustly from the sensed objects. This is evident when

acknowledging the fact that sections of the objects will be occluded from view and would greatly influence the coordinate system that could be extracted.

#### 4.2.4.2 Object size normalisation

Objects were normalised with respect to size by scaling the object so that it fit completely inside the unit ball. Such an approach to object size normalisation entails identifying the most distant vertex from the point of origin and then scaling the coordinates of the remaining vertices so that the most distance point lies on the edge of the unit ball.

Note that the aforementioned approach is largely susceptible to noise, as the performance of the normalisation process is entirely dependent on the accuracy with which the most distant point is identified. An outlier can easily be mistaken for the most distant point, which would result in erroneous normalisation. Pre-processing techniques, specifically a statistically-based outlier removal filter, was used to improve the robustness of the normalisation process significantly.

An alternative to the aforementioned normalisation process would be to employ a median-based analysis in order to identify the most distant point in the point cloud. Such an approach would involve ranking all points beyond a predefined threshold, based on their distance from the origin. The point located at the median of this ranking can then be used during scaling.

### 4.3 FEATURE EXTRACTION

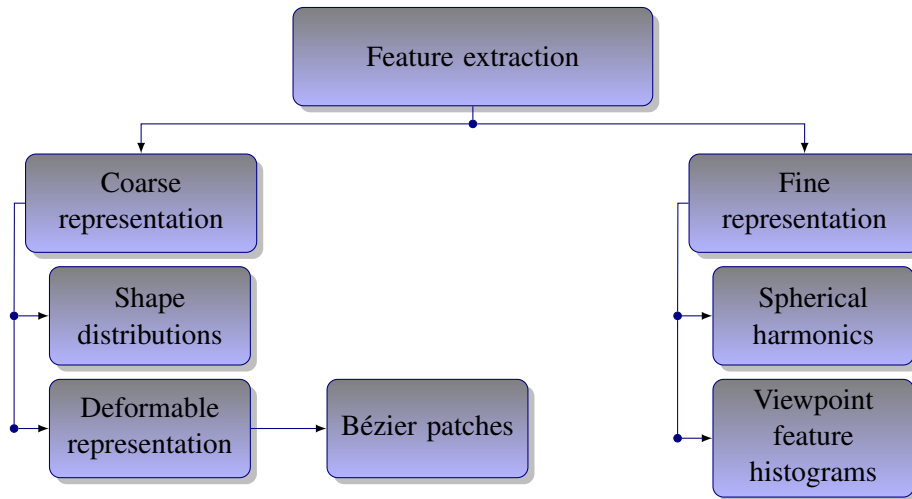
A hierarchical approach to feature extraction was adopted. Coarse features, represented as shape signatures, deformable surfaces, in the form of Bézier patches, and finer features, in particular VFHs, were exploited. A diagrammatic breakdown of the feature extraction process is shown in figure 4.3.

Each of these features represents the 3D point cloud data at different levels of detail.

#### 4.3.1 Coarse point cloud representations

Signatures, in the form of shape distributions, were used as inexpensive features primarily due to their relatively low resource requirements. These features were regarded as coarse representations of the global characteristics of a sensed point cloud. A concatenation of the following primitive surface





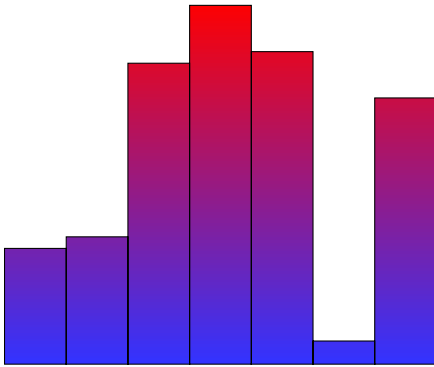
**Figure 4.3:** Diagrammatic breakdown of the feature extraction process.

features were employed: radial direction and surface normal alignments, radial direction vectors, radial distances and surface normals.

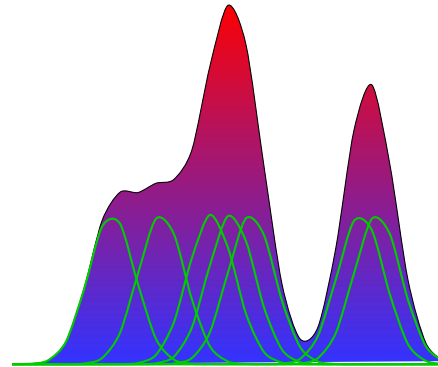
The features were considered based on the motivation in section 2.2.2.1 and experimental results provided in [48]. The results in [48] indicate that a combination of these primitive surface features yields improved descriptive qualities. Deformable model parameters were also included as coarse point cloud representations and are discussed in the subsequent subsection.

Shape distributions can be created by binning primitive shape descriptors and thereby constructing histograms. An alternative method would be to process primitive shape descriptors with the Gaussian kernel using the KDE methodology. These two approaches are respectively illustrated in figures 4.4 and 4.5.

The KDE approach to acquiring shape distributions was adopted using the Gaussian kernel with its bandwidth fixed at  $\frac{1}{2\sqrt{2}}$ . The bandwidth was chosen through inspection. The KDE approach was employed as more accurate distributions could be obtained with the KDE approach compared to a histogram-based approach. The cost associated with this improved accuracy was an increase in computational demand. The FIGTree library was used to address this drawback, as it was designed for the rapid execution of the Gauss transform [73].



**Figure 4.4:** Shape descriptor represented as a histogram.



**Figure 4.5:** Shape descriptor processed with the KDE methodology.

### 4.3.2 Deformable point cloud representation

A deformable model-based approach to object recognition and pose estimation was incorporated by fitting a single deformable patch, described by a Bézier surface, to the sensed point cloud. Interactive figures 2.14 and 2.15 illustrate the Bézier surface concept. Informally the approach can be regarded as folding a material, such as a table cloth, over a specific object and thereby creating an imprint of the object in the material. Different objects can then be compared based on the captured imprints.

A Bézier surface of order  $(m, n)$  is defined by a set of  $(n + 1)(m + 1)$  control points  $k_{ij}$  [77]. This can be expressed formally as,

$$p(u, v) = \sum_{i=0}^n \sum_{j=0}^m B_i^n(u) B_j^m(v) k_{ij} \text{ with } u, v \in [0, 1], \quad (4.4)$$

and

$$B_i^n(u) = \binom{n}{i} u^i (1 - u)^{n-i}. \quad (4.5)$$

The Bézier surface was fitted to the point cloud by performing the following:

1. Choose equidistant control points  $k_{ij}$  on a plane which is spanned by the two eigenvectors with the largest eigenvalues of the covariance matrix of the point cloud.
2. Find the closest point on the parametric surface to each point in the point cloud, i.e. find the  $u_k$  and  $v_k$  values that define the  $k$  foot points.

3. Minimise the distance between the parametric plane and the foot points,  $\|p(u_k, v_k) - p_k\|^2$ , by adjusting the control points.
4. Repeat steps 2 and 3 until convergence is reached.

The control points were adjusted using least square methods in order to minimise the distance between the parametric plane and the foot points. The vectors spanning the initial parametric surface, as well as the resulting control points, were used as global descriptors together with the signatures generated from primitive shape descriptors.

The chief contribution of the deformable patches is their ability to adapt when presented with a novel surface. These patches increase the versatility of the system by accounting for limited deformations that an object might undergo. Including deformable surface information therefore addresses the adaptation problem associated with fixed model-based approaches.

### 4.3.3 Fine point cloud representations

More descriptive features were also computed. An attempt was made to quantify the frequency content of a sensed point cloud by computing the spherical harmonic coefficients of the data. The ccSHT library was used, which ensured the rapid execution of the spherical harmonic transform (SHT) [54]. Preliminary experiments however suggested that spherical harmonics are extremely sensitive to noise and occlusions. It was also noted that a spherical harmonics implementation does not cope well when subjected to planar objects such as walls and tables. Spherical harmonics were not included as features, based on the aforementioned complications.

VFHs were used instead, for comparing point clouds at higher levels of detail. These histograms could be acquired using the Point Cloud library [70]. VFHs were of particular interest because of their ability to incorporate object pose information. These histograms also contain PFH information which presents descriptive information useful when performing object recognition [43].

## 4.4 DATABASE CONSTRUCTION

A multitude of 3D models has recently been introduced into the world wide web in reaction to the ever increasing popularity of 3D effects in the various fields of multimedia. Publicly available benchmark databases have been made available primarily with the intention of providing researchers with

some means of comparing different shape-matching techniques. These benchmark databases typically comprise models classified into different categories containing similar shapes. Examples of shape benchmark databases can be found on web pages from Princeton University [42], the University of Konstanz and Utrecht University.

Benchmark databases can also be used as a readily available source of 3D models. These models can be incorporated into training algorithms to bypass the tedious procedure of generating 3D data by hand. The drawback of this approach to obtaining training data is that the data might be erroneous or unrealistic, as many of the models are mere approximations of real-life objects.

A database was constructed by processing a subset of models in the Princeton Shape Benchmark (PSB). The subset was chosen such that it would contain objects that are most likely to be found in typical households. Household setups are constantly changing, setting a dynamic environment in which potential artificial systems should be able to function.

#### **4.4.1 Point cloud representation**

An optimal 3D model is one which has fine resolution, or many vertices, at areas of fine detail and coarse resolution, or few vertices at areas of lesser detail. This method of representing a model would ensure the optimal use of storage space, as no redundant information would be stored. Such a model representation is however far from what would actually be captured by most range sensors. A typical range sensor captures a scene as a dense collection of vertices, which is also referred to as a point cloud. The 3D shape models presented in the publicly available databases had to be slightly altered before they could be related to data captured by a range sensor.

All the 3D models in the PSB database are represented as a collection of faces. These faces are planes defined by three or four vertices positioned on the face. It is straightforward to deduce that additional data points should be populated into each model to obtain an equivalent point cloud representation of the model. Other standard 3D model file formats could also have been used. The use of other formats however imply resampling would be more complex as faces could be represented as Non-uniform Rational B-spline patches [78].

The simplest approach to converting a mesh representation of a shape into an equivalent point cloud representation of the same shape is to generate additional vertices on each face of the shape. The

three, or more, vertices that define a face can be used to derive an equation for the plane that is represented by the face. This technique was employed to populated additional vertices onto the plane at predefined intervals.

Two expressions are commonly used to represent a plane, the implicit and parametric plane representations. The implicit representation of a plane is described by one point  $q$  and a normal vector  $\vec{n}$ . A point,  $p$ , belongs to the plane if the vector spanned by the two points  $p$  and  $q$  is orthogonal to the normal vector  $\vec{n}$ .

A more convenient method for representing the plane spanned by the three vertices  $P_1$ ,  $P_2$  and  $P_3$  would be the parametric representation of a plane. The parametric representation of a plane is described by specifying one point,  $q$ , and two linearly independent direction vectors,  $\vec{v}_1$  and  $\vec{v}_2$ . Any other point,  $p$ , on the plane is a result of adding multiples of the direction vectors  $\vec{v}_1$  and  $\vec{v}_2$  to  $q$ . This can formally be expressed as follows,

$$P = \{p = q + \lambda_1 \vec{v}_1 + \lambda_2 \vec{v}_2 : \lambda_1, \lambda_2 \in \mathfrak{R}\}. \quad (4.6)$$

The parametric plane representation appears to be more convenient, as the values of parameters  $\lambda_1$  and  $\lambda_2$  can be varied to produce a point  $p$  at any position on the plane. Note that the choice of values for the parameters  $\lambda_1$  and  $\lambda_2$  is of utmost importance if current range sensor data have to be simulated accurately. This is because the direction vectors  $\vec{v}_1$  and  $\vec{v}_2$  need not be normalised, which might lead to oversampled or undersampled 3D models.

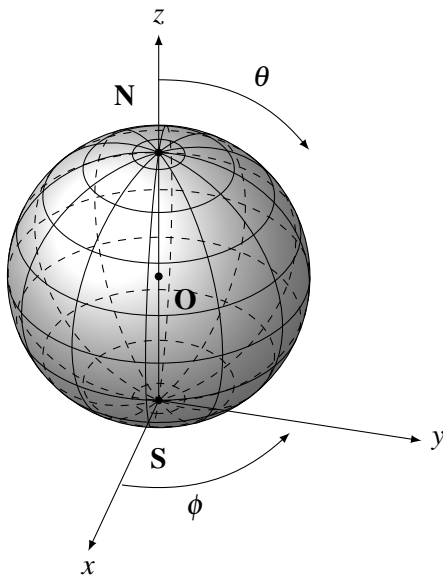
#### 4.4.2 Viewpoint incorporation

The effect of viewpoint change can be incorporated once a model with sufficient 3D data points has been generated. When an object is viewed from different perspectives some points, or areas, on the object might become occluded owing to object characteristics and shape.

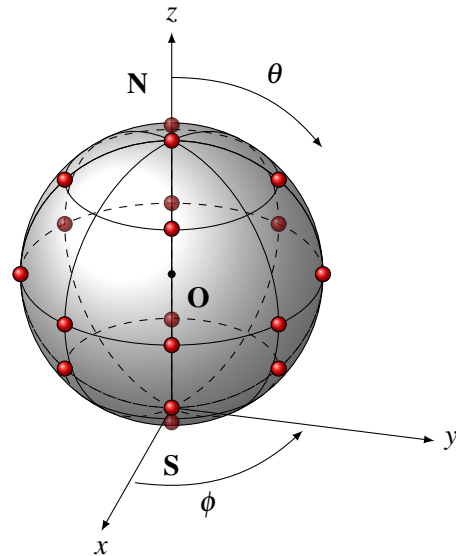
A viewpoint change was simulated by positioning a virtual camera at various locations on a sphere having the object under investigation at its origin,  $O$ . The camera position can be conveniently specified using the spherical coordinate system,  $(\theta, \phi, r)$ . Figure 4.6 illustrates the general concept, while figure 4.7 presents the viewpoints that were used in the implementation.

A trade-off between execution time and pose estimation accuracy needed to be acknowledged. Observing objects from a large number of viewpoints would improve system accuracy but it would also significantly decrease system throughput, making the system less suited for real-time applications.

The 18 viewpoints presented in figure 4.7 were considered sufficient to capture the majority of object pose variations, resulting in an acceptable trade-off between database size and object information. These viewpoints were selected based on the discussion on sampling schemes in section 2.2.2.7. Future research will focus on means to improve pose estimation accuracy by integrating information obtained from multiple viewpoints and investigating viewpoints other than the 18 suggested above.



**Figure 4.6:** Spherical coordinate system used to position a virtual camera around an object situated at  $O$ .



**Figure 4.7:** An example with 18 carefully selected viewpoints, represented by the red spheres, around  $O$ .

The variable  $\theta$  is sometimes referred to as the declination angle, which indicates latitudinal movement from the north pole of a sphere to its south pole and lies within the range  $\theta \in [0, 180^\circ]$ . The variable  $\phi$  is also known as the right ascension, which indicates longitudinal movement and is confined within the range  $\phi \in [0, 360^\circ]$ . The  $r$  variable indicates the radius of the sphere or the distance of the viewpoint from the centre of the sphere.

The occlusion process can be simulated by tracing a ray from each vertex on the surface of the object to the viewpoint. If a ray traced from a vertex to the viewpoint intersects a face along its path, it can

be assumed that that particular vertex is occluded from that particular viewpoint.

A point is occluded from view only if the ray intersection lies on the plane, and more importantly in the triangle, spanned by the three, or more, vertices that define a face. There are numerous methods that can be used to determine whether a point lies within a triangle. The Barycentric technique [79] was considered, as it was designed with the goal of optimising execution speed.

Using the parametric plane equation given in equation 4.6, all three of the following conditions must hold for a point  $p$  to lie within the triangle spanned by the two vectors  $\vec{v}_1$  and  $\vec{v}_2$ ,

$$\lambda_1 \geq 0, \quad \lambda_2 \geq 0, \quad \lambda_1 + \lambda_2 \leq 1. \quad (4.7)$$

The two parameters  $\lambda_1$  and  $\lambda_2$  can be solved from the equation if a point  $p$  is given,

$$p - q = \lambda_1 (P_2 - P_1) + \lambda_2 (P_3 - P_1) : q = P_1. \quad (4.8)$$

Points  $P_1$ ,  $P_2$  and  $P_3$  in the above expressions represent the vertices spanning the plane. Simplifying the equations yields the following expressions for the parameters  $\lambda_1$  and  $\lambda_2$ ,

$$\lambda_1 = \frac{(\vec{v}_1 \cdot \vec{v}_1)(\vec{v}_2 \cdot \vec{v}_0) - (\vec{v}_1 \cdot \vec{v}_0)(\vec{v}_2 \cdot \vec{v}_1)}{(\vec{v}_0 \cdot \vec{v}_0)(\vec{v}_1 \cdot \vec{v}_1) - (\vec{v}_0 \cdot \vec{v}_1)(\vec{v}_1 \cdot \vec{v}_0)} \quad (4.9)$$

$$\lambda_2 = \frac{(\vec{v}_0 \cdot \vec{v}_0)(\vec{v}_2 \cdot \vec{v}_1) - (\vec{v}_0 \cdot \vec{v}_1)(\vec{v}_2 \cdot \vec{v}_0)}{(\vec{v}_0 \cdot \vec{v}_0)(\vec{v}_1 \cdot \vec{v}_1) - (\vec{v}_0 \cdot \vec{v}_1)(\vec{v}_1 \cdot \vec{v}_0)}. \quad (4.10)$$

All the points that were considered occluded, according to the discussion presented above, were removed from the point cloud representing the object.

#### 4.5 DATABASE PRE-PROCESSING

Pre-processing databases with the intent of improving the performance of subsequent database operations is by no means uncommon. Hierarchical data structures are often attractive, as they facilitate rapid query operations.

### 4.5.1 Tree structures

Large databases are frequently rearranged to resemble tree structures, which significantly reduces query latencies. A-trees [80], SR-trees [81], TV-trees [82] and X-trees [81] are just a few of the vast number of suggested tree structures.

Most tree structure implementations are outperformed by brute force approaches, such as exhaustive searches, when dealing with high-dimensional data. Some tree structures, such as the TV-tree [82] and iDistance [83], have been designed to address this limitation of tree structures.

The complexity of most existing tree algorithms, such as k-d trees and R-trees, however increases exponentially with dimension. The aforementioned complication renders these search structures impractical for dimensionality above 15 [84]. An alternative database pre-processing scheme, which addresses the aforementioned complication, involves data compression techniques.

### 4.5.2 Database compression

Recently proposed data compression techniques have not only attempted to reduce storage space requirements, but also to make rapid querying of the database possible. Binary hashing, as discussed in section 2.4.2, is an example of such a compression scheme.

A binary hashing approach was adopted for this system based on three primary observations. First, a binary database representation would be more compact than an equivalent tree data structure. A binary representation of the database can be exhaustively searched at extremely rapid rates with dedicated hardware. Finally, more importantly, the binary hashing approach is less sensitive to the dimensionality of the problem.

The database pre-processing phase entailed performing two critical tasks. Firstly, adequate features were extracted from each of the database entries, as discussed in the previous section. These features would be used in subsequent operations as accurate representations of the respective entries. Secondly, the extracted features were compressed into bit strings, as discussed in the next section, which are then used to perform database query operations.



## 4.6 SEARCHING AND MATCHING

The searching and matching subcomponent accepts features describing a sensed point cloud, and compressed representations of point cloud instances from the data capturing device and database respectively. Its primary objective is to compare the sensed point cloud, represented as a collection of features, with entries stored in the database and, produce the database entries that are most similar to the query point cloud.

The aforementioned requirement is addressed by first compressing the received features. The compressed data are then used to perform the first task in the hierarchical search procedure, a coarse search of the database, whereafter a more refined search improves preliminary suggestions.

### 4.6.1 Data compression

Binary hashing techniques have proven to be advantageous when attempting to perform content-based queries on large databases [45]. This inspired the incorporation of recently suggested binary hashing methods into the searching and matching subcomponent in order to facilitate fast database query operations. Existing implementations of four variations of binary hashing were exploited, LSH, BRE, MLH [66] and SH [68].

For LSH, BRE and MLH, the canonical approach was adopted and centred (mean-subtracted) inputs, linear projection, and binary quantisation were assumed. The hash functions for BRE and MLH were obtained using the training implementation discussed in [66], while for LSH the functions were generated randomly. The binary quantisation was performed according to equation 2.18, which is repeated here for convenience.

$$b(x; w) = \text{thr}(\mathbf{W}x) \quad (4.11)$$

The SH implementation called for additional data processing to ensure robustness. With the solution to the relaxed problem stated in subsection 2.4.2.2 known, a suggested method to obtain the binary code words would be simply to threshold the  $k$  eigenvectors of  $\mathbf{D} - \mathbf{W}$  with minimal eigenvalue. Such an approach however would only be applicable to computing the code representation of items in the training set. This is the well-known problem of out-of-sample extension associated with spectral methods.

### Processing novel inputs

Several learning algorithms, particularly those based on eigendecomposition, provide embedding only for training points with no direct extension for novel, or out-of-sample, instances. A possible solution would be to recompute the eigenvectors every time a new sample is taken [85]. Such an approach is straightforward but inefficient and impractical for large sample sets.

The Nyström method is an alternative often used to address the out-of-sample extension challenge [86]. Unfortunately the cost of calculating the Nystrom extension of a new data point is linear to the size of the dataset. This implies that calculating the Nystrom extension for large databases would most likely be as expensive as performing an exhaustive nearest-neighbour search [68].

A more efficient method for enabling out-of-sample extension is to assume that the data points,  $x_i$ , are samples from a probability distribution  $p(x)$ . The equations in the problem defined in 2.17 are now seen to be sample averages, which can be replaced by their corresponding expected values. A weighted Laplacian,  $L_p$ , can be defined as an operator that maps a function  $f$  to  $g = L_p f$ . The solution to the relaxed problem is functions that satisfy  $L_p f = \lambda f$  with minimal eigenvalue, once again ignoring the trivial solution  $f(x) = 1$ , which has eigenvalue 0. For a more detailed discussion please refer to [68].

The resulting eigenfunctions,  $\Phi_k(x)$ , and their eigenvalues,  $\lambda_k$ , of the 1D Laplacian,  $L_p$ , specifically for the case of a uniform distribution on  $[a, b]$  are given as,

$$\Phi_k(x) = \sin\left(\frac{\pi}{2} + \frac{k\pi}{b-a}x\right), \quad (4.12)$$

$$\lambda_k = 1 - e^{-\frac{\epsilon^2}{2} \left| \frac{k\pi}{b-a} \right|^2}. \quad (4.13)$$

A similar equation is also available for the 1D Gaussian distributions and should be used if it is known that the probability distribution is not uniform. A binary code can be built by thresholding the  $k$  eigenfunctions of  $L_p$  with minimal eigenvalue  $y(x) = \text{sign}(\Phi_k(x))$ . Note that outer-product eigenfunctions should be avoided when building a hashing code in order to reduce redundancies [68].

## Spectral hashing implementation

Given a training set of points  $\{x_i\}$  and a desired number of bits  $k$  the spectral hashing (SH) algorithm is employed as follows to obtain the required binary code words:

1. Find the principal components of the data using PCA.
2. Calculate the  $k$  smallest one-dimension analytical eigenfunctions of the Laplacian  $L_p$  using a rectangular approximation along every PCA direction. These functions are identified by evaluating the  $k$  smallest eigenvalues for each direction using equation 4.12. A list of  $dk$  eigenvalues is created, and then sorted to find the  $k$  smallest eigenvalues, with  $d$  being the dimensionality of the problem.
3. Threshold the analytical eigenfunctions at zero, to obtain the binary codes.

This simple algorithm has two obvious limitations. Firstly it assumes that the data were generated from a multidimensional uniform distribution. Secondly, high-order dependencies between the bits may exist even though the algorithm avoids the trivial three-way dependencies that arise from outer-product eigenfunctions. The rate at which training sets can be processed with the SH training algorithm however outperforms that of BRE and MLH significantly.

### 4.6.2 Database query

The primary goal of a database query operation is to suggest database entries that are similar to a query description. A database search operation is largely governed by the metric used to define similarity between entries. Using a metric that does not involve complex mathematical operations could significantly reduce query time, as less computational resources would be needed. The chosen metric should however remain powerful enough to ensure acceptable retrieval results.

A hierarchical approach to database query operations was adopted. A coarse database search is first performed using the bit strings resulting from processing the extracted features, which were acquired online, by one of the data compression schemes discussed in the previous subsection. The coarse search only flags potential matches in the database and passes the flagged database entries to a refinement process. The refinement process then analyses features, which are more descriptive than those used in the coarse search, to produce more accurate database suggestions.

Bit strings were produced by compressing primitive and deformable 3D features. These features were distributions of radial direction and surface normal alignments, radial direction vectors, radial distances and surface normals, as well as the positions of parametric surface control points of Bézier patches. Only the database entries which fall within a specified Hamming distance of the query bit string were considered further, using their VFH and radial direction values.

The aforementioned primitive 3D features were chosen based on their simplicity, ensuring fast feature extraction. Radial direction and surface normal alignments crudely measure how much a query surface deviates locally from sphericity. This information is valuable when segregating block-like objects from round objects. Distributions of radial direction vectors and distances coarsely encode pose information, as they are sensitive to viewpoint changes. Surface normals were included to incorporate surface curvature information.

It was sufficient to record only the positions of the control points of the deformable patches, as they completely describe the patch. Deformable information could therefore be stored in a very compact manner. The Bézier patches were added, as they account for limited variations that different objects might undergo. This information addresses the versatility challenge associated with most fixed model-based approaches.

The Hamming distance,  $d_{H_m}$ , between two bit strings,  $q$  and  $q'$ , is defined as the number of corresponding bits having different bit values. This can formally be expressed as [87],

$$d_{H_m}(q, q') = \frac{m}{2} - \frac{1}{2} \sum_{i=1}^m \text{sign}(q_i, q'_i). \quad (4.14)$$

Current CPU architectures allow Hamming distances to be calculated at significant rates owing to dedicated hardware instructions. This uncomplicated similarity measure is therefore the measure of choice for coarse search operations. The subset of possible matches flagged by the coarse search operation is then compared, based on their respective VFHs, and the most similar entries suggested as estimates of the class and pose of the query instance.

#### 4.7 DATA POST-PROCESSING

No post-processing was performed on the object and pose suggestions provided by the implementation discussed in this section thus far, as this module will be focused on in the future. The post-

processing subsystem can however provide a more accurate object and pose estimate than what is currently suggested.

A straightforward approach to refining the pose estimate produced by the system would be to use the iterative closest point (ICP) algorithm to align the suggested and observed object surfaces. A known drawback of the ICP algorithm is its sensitivity to initial conditions, that is, the original alignment of the two surfaces. This challenge should however be significantly simplified, since the pose estimate produced by the system should be accurate enough to ensure convergence to a global optimum.

An alternative approach to refining system suggestions would be to incorporate previous object recognitions and pose estimations. The idea would be to combine observations of the same object as seen from multiple viewpoints. This can be realised by acquiring additional information from different perspectives in a scene. The system can perform this task by either moving around in a scene or by being fitted with multiple 3D range sensors.

## CHAPTER 5

# EXPERIMENTS AND RESULTS

The process of system validation is unique to each system and should be defined accordingly. An appropriate framework can be developed by considering what is expected from the system. The most important information an object recognition and pose estimation system provides, is the object and pose information. A system validation procedure should therefore primarily focus on these aspects.

The main objective of the proposed research was to develop an accurate object recognition and pose estimation system capable of operating in real-time. The following questions address key features that needed to be investigated:

- What aspects of current artificial vision and perception systems need to be improved in order to address current latency problems?
- What improvements can be incorporated into current state of the art object recognition and pose estimation procedures that will reduce execution time?
- What effects would improvements on execution time have on system accuracy?
- Would an acceptable real-time solution be realisable using existing technology?

In this chapter, experiments are defined and results presented that quantify system performance and address the questions stated above. Specific attention was paid to execution times, object recognition accuracies and pose estimation accuracies, as these characteristics would define the object recognition and pose estimation system. The system was analysed using both simulated and practical data.

## 5.1 QUANTIFYING SYSTEM PERFORMANCE

An approach often adopted in the machine learning community to quantify recognition performance entails partitioning a database into two subsets. These subsets are frequently referred to as the training and testing subsets and are employed during the validation process [19, 50, 43, 88, 89, 90, 91]. Such an approach will henceforth be referred to as a database approach to system validation.

The database approach to system validation incurs two requirements. It not only assumes that an adequate database is available, but also requires the training and test subsets to be well-defined. That is, each instance in the dataset needs to be labelled to make supervised learning possible. Technological advances have made it possible to acquire vast amounts of 3D data over the Internet. This agrees with the first assumption, making it feasible to construct large databases. The second requirement however is known to be extremely tedious, was frequently carried out by the researcher and often proved impractical when large databases were being considered.

An alternative to the database approach for system validation involves the use of mathematical measures or metrics. These measures, or metrics, are capable of independently providing accurate information related to the aspect of a system which needs to be validated. For example, similarity measures are often exploited when evaluating the accuracy of a recognition-based system. Such measures facilitate the convenient representation of significant amounts of information in a compact fashion when combined with confusion matrices. This approach to system validation will be referred to as a distance measure-based approach.

Both the aforementioned approaches to quantifying system performance have limitations. A database approach would probably be impractical when large databases need to be analysed, while most measure, or metric, based approaches are known to be computationally intensive and slow. The process of identifying appropriate measures, or metrics, which would guarantee robust performance, is also known to be exceedingly challenging.

A combination of the two aforementioned methods was used to quantify system performance. Such an approach was adopted in order to find an acceptable trade-off between data preparation time and the computational complexity of the experiments. The approach acknowledged that objects from different classes might appear similar from appropriate viewpoints. A straightforward database approach to system analysis will not accommodate the aforementioned possibility, producing inaccurate system

performance indicators. The inclusion of a database approach made it possible to compare the results to published results [43].

## 5.2 COMPARING POINT CLOUDS

All shapes, or objects, in their basic form, were stored and represented as point clouds during experimentation. This form of representation was adopted as commercial 3D range scanners provide shape information in point cloud format.

Point clouds are one of the most primitive and fundamental methods for representing surfaces. The recent increase in popularity and broad use of this source of data have motivated research into developing methods of working directly with point cloud data. Previous techniques would typically manipulate the data into a more manageable format, such as fitting surfaces to the data points, before processing commenced. The importance of this type of shape representation has led to a significant increase in the fundamental study of point clouds [92].

A well-known challenge related to the analysis of non-rigid shapes is the problem of shape similarity, that is, how the degree of similarity or dissimilarity of two given objects can be quantified. A serious difficulty in such a comparison stems from the vast number of degrees of freedom present in the problem [88]. Means of robustly quantifying shape similarity had to be identified in order to generate some form of ground truth data to which the object recognition performance of the system could be compared.

Various distance metrics have been proposed that attempt to address the problem. Some of the metrics are still only theoretical, while the practical implementations of others enforce the acknowledgement of numerous conditions, which cannot always be satisfied. A metric that can robustly calculate the distance between various point clouds could be used to generate ground truth data.

The following subsections discuss three different measures that could be used to generate ground truth data, as well as their respective implementations used during the experiments. The mass transportation and geometric distance measures were implemented, but not exploited during system validation. This was due to the complex nature of the proposed methods, which significantly hampered the rate at which these measures could be calculated.



### 5.2.1 Objects as metric measure spaces

A common approach to defining distance measures is to describe objects, represented by point clouds, as metric measure spaces. The Hausdorff distance between two point clouds  $X$  and  $Y$ , each representing an object, is defined as [93],

$$d_H^Z(X, Y) = \max \left( \sup_{x \in X} \inf_{y \in Y} d_Z(x, y), \sup_{y \in Y} \inf_{x \in X} d_Z(x, y) \right). \quad (5.1)$$

In the equation presented above the distance metric used between any two data points is represented as  $d_Z(x, y)$ , where  $x$  and  $y$  represent the respective data points. The Hausdorff distance is often considered inappropriate when comparing complex data structures, such as objects represented by point clouds. This is because the Hausdorff metric does not account for deformation within the data structures.

An improvement over the Hausdorff distance is the Gromov-Hausdorff distance, which could be made blind to specific isometric transformations, such as bends or rigid transformations, and is defined as,

$$d_{GH}(X, Y) = \inf_{Z, f, g} d_H^Z(f(X), g(Y)), \quad (5.2)$$

where  $f: X \rightarrow Z$  and  $g: Y \rightarrow Z$  are isometric embeddings (distance preserving) into the metric space  $Z$  [93].

The Gromov-Hausdorff distance however poses a number of limitations. Firstly, the computation of the metric is an *NP*-hard problem which suggests its computational intensiveness [88]. Secondly, the isometric embeddings,  $f: X \rightarrow Z$  and  $g: Y \rightarrow Z$ , must be application-specific, as the chosen family of functions will govern the invariance of the metric to specific transformations. Thirdly, most of the work done using the Gromov-Hausdorff distance was done under the assumption of smooth surfaces.

A crude implementation of the Gromov-Hausdorff distance would be to use the ICP algorithm as the isometric transformation. Point cloud similarity is then defined as the Hausdorff distance of the aligned point clouds. A significant drawback of this approach is the sensitivity of the ICP algorithm to initial conditions.

The Gromov-Hausdorff distance can be modified with the goal of modelling and addressing the practical problems of object matching and comparison. The result is a Gromov-Wasserstein type of distance, which is defined by regarding the objects as metric measure spaces and incorporating ideas of mass transportation [89].

### 5.2.2 Mass transportation measures

Transportation theory involves the study of optimal transportation and allocation of resources [94]. A suggested distance measure which is based on a solution to the transportation problem from linear optimisation is the earth mover's distance (EMD). The EMD is a measure of distance between two distributions. This distance measure analyses the minimal cost that is associated with the transformation of one distribution into another [94]. It is also referred to as the Wasserstein metric.

The EMD can informally be explained as follows. The two distributions are interpreted as piles of dirt over a specified region. The calculated distance is then the minimum cost associated with turning one pile into the other. This cost is defined as the product of the amount of dirt moved and the distance by which it has been moved.

Note that the EMD has been improved to be able to process multidimensional data [95]. The implementation can therefore be used to compare point clouds as well.

### 5.2.3 Geometric distance measures

The geodesic and diffusion distances are two intrinsic (geometric) distances, which can informally be described as measuring constrained paths by only travelling on the surface of point clouds or shapes [90].

The geodesic distance, between two points, is defined as the length of the shortest surface-path between the two points [90]. A shape signature can be constructed by pooling numerous geodesic distances, or the probability thereof, between points in a point cloud. Note that the geodesic distance is susceptible to noise, as it only considers a single path between two points.

The diffusion distance is more robust against noise than the geodesic distance, as it is related to the probability of travelling from one point to another within a specific number of random steps. A shape signature can also be constructed by pooling numerous diffusion distances between points in a point

cloud. Complex 3D data are often corrupted by noise, especially if obtained from range scanners, which indicates that a diffusion-based distance measure would be more suitable than a geodesic-based equivalent.

The similarity between two point clouds was quantised by comparing distributions using the well-known  $\chi^2$  test. The distributions were constructed from diffusion distances extracted from the point clouds. The implementation is based on the discussion in [90], to which the interested reader is referred. The probability of a random walker on a point cloud to step from point  $x$  to point  $y$  in the point cloud is defined as

$$p(x,y) = \frac{k(x,y)}{v(x)}, \quad (5.3)$$

where  $v(x) := \sum_y k(x,y)$  is the sum of the elements in each row of matrix  $P$  composed of elements  $p(x,y)$  and  $k(x,y)$  is an affinity function defined over all pairs of points in the point cloud. In this case the affinity function was taken as the Gaussian kernel, in accordance with [90], with variance equal to the average Euclidean distance between all pairs of points in the point cloud.

A diffusion map is obtained after the geometry of the point cloud has been separated from its density. The diffusion distance between points  $x$  and  $y$  in the point cloud can then easily be obtained as the Euclidean distance between the mapped points of  $x$  and  $y$ . The interested reader is referred to [90] for a more in-depth discussion.

### 5.3 SIMULATION EXPERIMENTS

A database, consisting of 206 objects, was constructed by selecting relevant objects from the PSB. An object was regarded as relevant if it could be classified as an object that would be encountered in everyday life. Objects such as tables, chairs, mugs, vases, cabinets, etc. were therefore considered and manually labelled accordingly.

Viewpoint information was incorporated by viewing each object in the database from 18 different viewpoints, as discussed in subsection 4.4.2 and illustrated in figure 4.7. This process enlarged the database such that it contained a total of 3708 entries. Finally the enlarged database was partitioned into two subsets labelled the training and test subsets, which were respectively used for training the bit compression schemes and evaluating system performance.

System performance was evaluated by presenting 252 test instances, 14 objects viewed from 18 dif-

ferent viewpoints, to the system. The test instances were presented to the system as point clouds which consisted on average of 9838.03 points, with a standard deviation of 9255.40 points. The large standard deviation was used to simulate objects at various distances from the range scanner. An inverse relationship exists between point cloud size and the distance from the range scanner to the object.

The system retrieved four objects from the database and produced an estimation of their poses for each of the test instances. The four retrieved database entries were considered as the best matches for the query objects and were taken as estimates for the query objects together with their poses. The number of objects that were retrieved from the database were chosen based on inspection to account for objects that subjectively appear similar when observed from different viewpoints. Table 5.1 summarises the specifications of the system on which the experiments were conducted.

**Table 5.1:** Summary of the system specifications on which experiments were conducted.

System specifications	
Feature	Description
<b>Hardware specifications</b>	
Operating system	Linux 3.2.0-3-amd64, Debian, 64 bit
Processor/Speed	Intel Xeon 5355 (8MB L2 cache, 2.66 GHz, 1333 MHz FSB)
System memory	8 GB Dual-channel DDR3 SDRAM
<b>Software specifications</b>	
Programming language	C++
Compiler	GCC, GNU Compiler collection (g++ 4.7.2)

All the experimental results presented next were obtained using a single core of the Intel Xeon 5355, 2.66 GHz quad core processor running Linux. The compressed information of all the database entries, that is, the bit strings which represent each database entry at coarse scale, was loaded into memory before time-critical operations were performed. The more descriptive information, which represented each database entry at a higher level of detail, was stored on the hard drive and only retrieved when required.

### 5.3.1 System execution time analysis

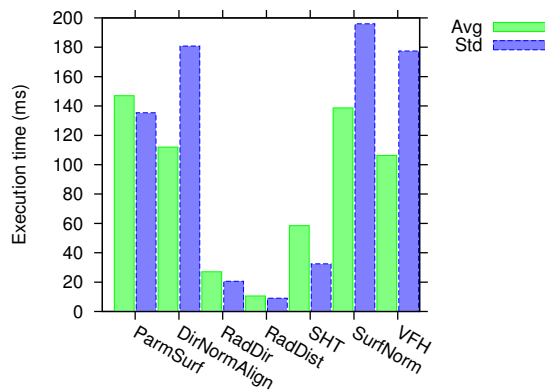
The suggested implementation incorporates bit compression techniques to realise a hierarchical search structure and address the latency problem associated with current object recognition and pose estimation methods. Various system execution times were analysed in order to investigate the effects

that these alterations brought about.

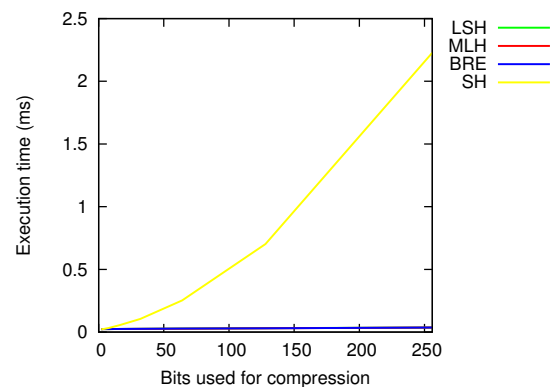
The average size of the point clouds, used during simulation circumstances, was somewhat larger than what is typically obtained with practical range scanners. This is due to the high resolution of the 3D computer models used. It was therefore expected that the feature extraction process would take much longer under simulation circumstances than during practical experiments.

The figures shown next present feature extraction and compression times noted during simulations. Feature extraction times are shown in figure 5.1, where average values are presented in green and standard deviation values are shown in blue. These values were recorded during the processing of the 252 test instances. No additional processes were initiated on the active processor core, once a simulation was started. This was necessary to limit external interferences which could corrupt time-critical measurements.

The features used in the implementation, as displayed from left to right in figure 5.1, are parametric surface control points, radial direction and surface normal alignments, radial direction vectors, radial distances, surface normals and VFHs. Note that even though the execution times for the SHT are shown in figure 5.1, it was not used in the implementation and is only included for interest's sake.



**Figure 5.1:** Execution times at which features were extracted.



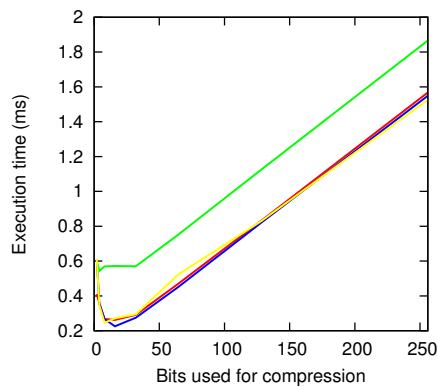
**Figure 5.2:** Execution times at which the extracted features were compressed.

The increase in the time it takes to compress extracted features into bit strings, when using SH, associated with an increase in the number of bits used, was expected. This is because the number of Laplace functions that need to be evaluated is linear to the number of bits used during compression. Other bit compression techniques, such as MLH, BRE and LSH, employ a simple feed-forward network, which

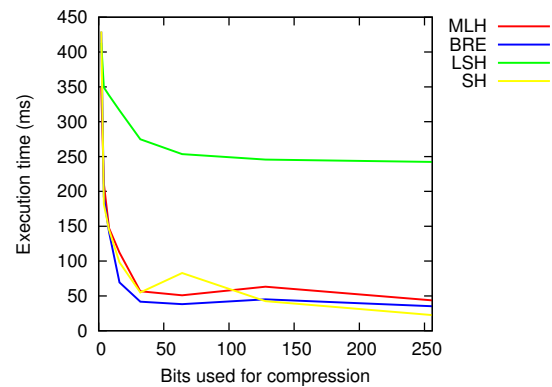
is less sensitive to the number of bits used. Implementations for simple feed-forward networks, such as those incorporated by MLH, BRE and LSH, can be optimised, which further improves execution rates.

The execution times associated with database query operations were also investigated. Two search operations were performed. Firstly, a coarse search was performed, which entailed exhaustively comparing the bit string representations of the database entries to those of the query instance. Only the database entries which fall within a specified Hamming distance of the query bit string are considered further, using their VFH and radial direction values. These Hamming distances were defined as percentages of the respective bit string lengths.

The following results were obtained by setting the query Hamming distance to 25% of the bit string length for the SH implementation, 50% for the LSH implementation and 15% for the MLH and BRE implementations. The larger Hamming distance assigned to the LSH implementation was necessary to force the system to suggest objects from the database. Any smaller Hamming distance resulted in the LSH implementation not making any suggestions from the database, which defied the purpose of the system.



**Figure 5.3:** Execution times at which coarse, exhaustive, database queries were performed.



**Figure 5.4:** Execution times at which fine, selective, database queries were performed.

The time it takes to search through a database selectively can be decreased by increasing the bit string lengths used for compression, as was explained in the previous subsection. It is interesting to note the rate at which a large database can be searched exhaustively when using bit strings to represent the database entries.

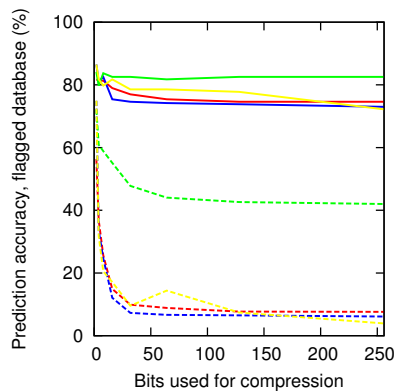
It is evident that simply increasing the number of bits used for compression might not be optimal.

The results indicate that the method used for compression is highly important and largely governs the performance of the system as is illustrated in figure 5.4.

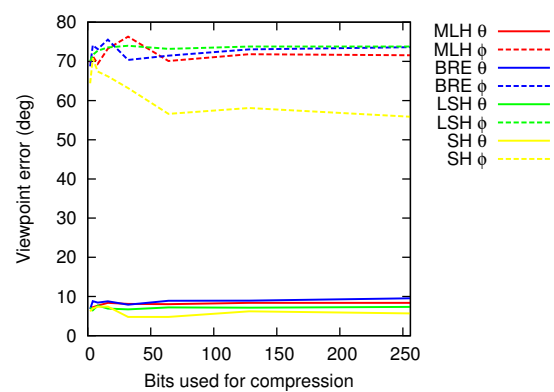
### 5.3.2 Joint object recognition and pose estimation analysis

Object and pose information is of primary concern when evaluating an object recognition and pose estimation system. The effects of improved execution time were analysed based on these aspects.

Each test instance was manually labelled to indicate the class, or classes, to which it belonged, before it was presented to the system. A test instance was considered correctly classified if the system suggested one of the classes assigned to that test instance. The following object classification accuracies and viewpoint estimation errors were noted during simulations, as discussed in the previous subsection.



**Figure 5.5:** The object classification accuracies and percentages of the database flagged for further query.



**Figure 5.6:** The average viewpoint estimate error.

The results indicate an average object classification accuracy of approximately 73% for the BRE, MLH and SH implementations with the LSH implementation producing a slightly higher average. The LSH implementation however, requires a much larger portion of the database to be searched at fine resolution. Results shown in figure 5.5 illustrate this, as a minimum of 42% of the database still needed to be processed after a coarse search, when using LSH. The lack of database rejection shown by the LSH implementation is largely due to implementation not factoring into account shape similarities during the compression phase.

The results are comparable to other suggested implementations, such as VFHs [43] and spin images [18]. Experimental results given in [43] indicate that VFHs have the potential of realising object

recognition and pose prediction accuracies of 98.52%. Spin images, on the other hand, produced object recognition accuracies of approximately 75.3%. These results were obtained using a database containing 60 different kitchenware objects.

It is worth noting that the object classification accuracy, as presented in figure 5.5 for the MLH, BRE and SH implementations, appears not to be heavily dependent on the number of bits used for compression. This is due to the hierarchical search structure of the implementation, as discussed in section 4.6. Only the four most appropriate objects in the database, according to their respective VFHs and radial direction values, are suggested by the system. The coarse search operation however, which is dependent on the number of bits used for compression, only reduces the number of potential candidates in the database that need to be compared based on their VFH and radial direction values.

Note the rapid decrease in the percentage of the database that is flagged for further processing, represented by the dashed lines in figure 5.5, as the length of the bit strings, used to compress the extracted features, increases. This was expected, as longer bit strings should provide more descriptive information, which can be used to increase the number of database entries rejected by a coarse search.

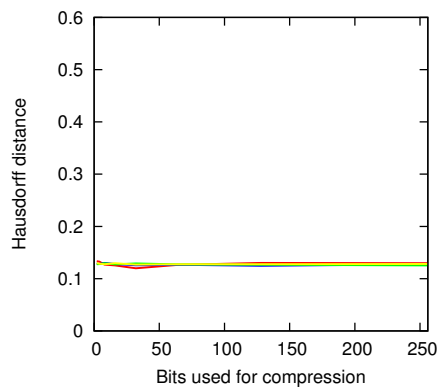
The LSH implementation did not improve as much as the other three implementations did with an increase in bit string length. This is primarily due to the randomness of the LSH implementation. Based on the results obtained, it appears that little improvement is achieved when using bit strings longer than 128 bits.

The orientation of an object is directly related to the viewpoint from where it is being observed. This implies that orientation parameters, such as pitch, roll, tilt and yaw, can easily be deduced from an object in a database, given that the point of observation is known. The viewpoint estimation errors presented in figure 5.6 therefore correlate directly with the object orientation estimation error of the system. The  $\theta$  and  $\phi$  parameters respectively represent the declination angle and right ascension, as illustrated in figure 4.6 in section 4.4.2. A possible explanation for the significant right ascension error is the effect of notably symmetrical objects.

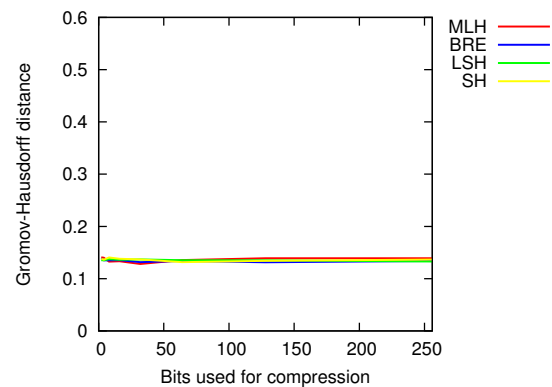
Distance measures were calculated to account for objects that might appear similar when observed from different viewpoints. The results are shown in figures 5.7 and 5.8.

The average Hausdorff and Gromov-Hausdorff distances presented in the previous figures were calculated for query objects and the objects considered to be most similar to each of the query objects.





**Figure 5.7:** The average Hausdorff distances for nearest neighbours.



**Figure 5.8:** The average Gromov-Hausdorff distances for nearest neighbours.

Only the best candidates were therefore considered.

During these experiments it was noted that point clouds representing dissimilar objects produced Hausdorff distances of approximately 0.6, while significantly similar point clouds resulted in Hausdorff distances around 0.06. A Hausdorff distance of 0.0 was only possible when comparing a point cloud with itself.

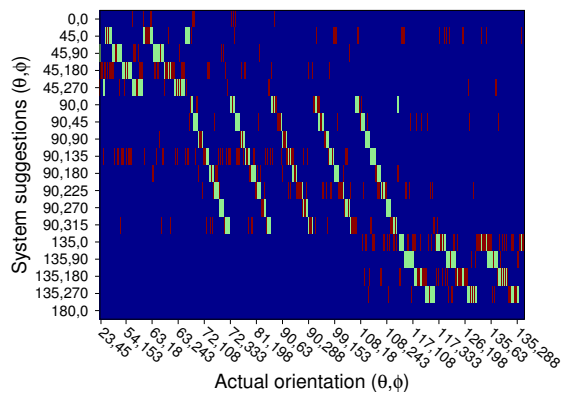
Note the small, almost negligible, difference between the Hausdorff and Gromov-Hausdorff distances presented in figures 5.7 and 5.8. Recall that the Gromov-Hausdorff distance entailed aligning the point clouds before calculating the distance measures, as was discussed previously. The results therefore propose that the majority of the suggestions produced by the system seem to be closely related to the query object.

It is evident from the results shown in figures 5.5 and 5.6 that complex compression schemes, such as MLH, BRE and SH, significantly outperform straightforward implementations such as LSH. This was expected.

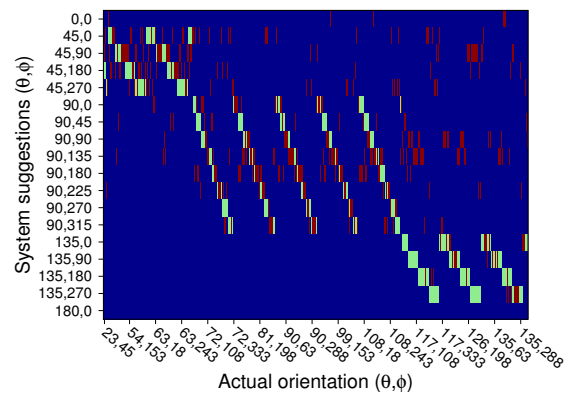
### 5.3.3 Pose estimation analysis

The pose estimation aspect of the system was also analysed independently from its object recognition capabilities. Additional test instances were generated by viewing 3D objects from viewpoints other than those used to train the system. That is, viewpoints different from those suggested in figure 4.7. These novel test instances were then presented to the system which estimated their poses.

The pose estimation analysis experiments were different from the experiments discussed in the previous subsection in the sense that the class of the objects was made known to the system *a priori*. The system therefore knew to which class the test instance belonged and only needed to estimate its pose. The results obtained for the models shown in figures 5.11 and 5.12 are shown in figures 5.9 and 5.10.



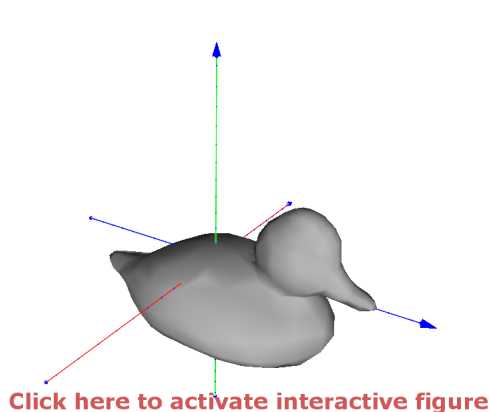
**Figure 5.9:** Pose estimation results, for the model shown in figure 5.11, using 8 bits for compression and a Hamming distance of 4 bits.



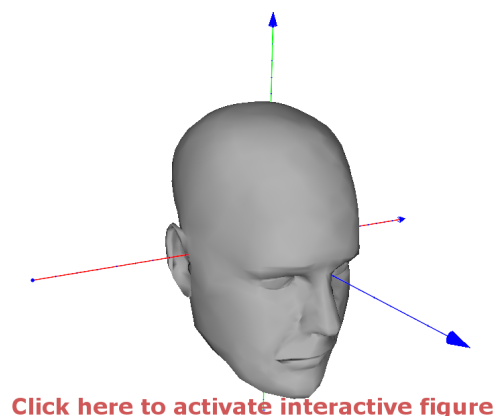
**Figure 5.10:** Pose estimation results, for the model shown in figure 5.12, using 32 bits for compression and a Hamming distance of 8 bits.

The benchmark, or ideal pose suggestions, are indicated in green in figures 5.9 and 5.10, while the poses suggested by the system are presented in red. The system could only suggest poses available on the vertical axis as these were used to train the system.

The vertical axes in figures 5.9 and 5.10 indicate that five discrete declination angles  $\theta$ , were specified to transition from the north pole to the south pole as shown in figure 4.7. They also indicate the discrete increases in right ascension  $\phi$ , which corresponds to a displacement in the longitudinal.



**Figure 5.11:** Model of a toy duck.



**Figure 5.12:** Model of a human head.

The results shown in figure 5.9 indicate that most of the poses suggested by the system are clustered around the ideal object poses. This advocates that the system is functioning as expected and encourages further refinement.

Results shown in figure 5.10 motivate further improvement as it is evident that the system struggled to estimate adequate poses for a human head when observed from below. These results however support the idea that objects may appear similar when viewed from different viewpoints, as is pointed out throughout this dissertation. The human head appears like a mere convex shape when viewed from above or below when considering that the finer detail, such as the eyes, nose, mouth and ears, might be too subtle for the extracted features to quantify. This is a strong possibility when using a hierarchical search structure in which coarse features are used during the initial search operation.

It is postulated that the difference in pose estimation accuracies shown in figures 5.9 and 5.10 is the result, or the lack, of prominent 3D features characterising the models. For example, the position of the head of the toy duck, shown in figure 5.11, relative to its body, is a feature quantifiable by coarse features. The human head, shown in figure 5.12, has no such prominent features and is therefore not easily quantified by the coarse features used for this experiment. This lack of descriptive information caused the system to repeatedly mistake the bottom of the head for the top of the head.

Further insight into the behaviour of the system can be obtained by presenting additional viewpoints to the system during training. The new information should primarily focus on the areas attributing to the pose error.

#### **5.4 PRACTICAL EXPERIMENTS**

The accuracy with which simulations can be performed is generally governed by the amount of data that is available. Accurate simulations however typically incur significant computational demands due to the vast number of variables that need to be accounted for. Such system analyses do not necessarily take into consideration all practical limitations the system might have. Many systems are susceptible to real-life complications, such as measurement noise and lighting conditions, which supports the necessity for practical system analyses.

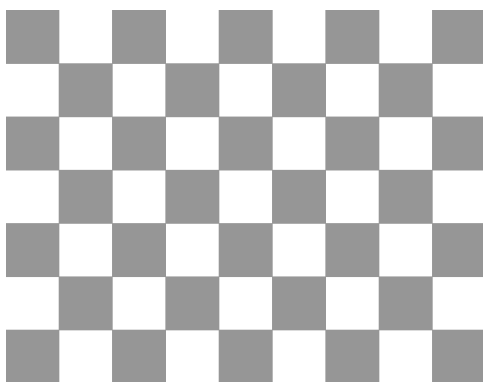
Practical data were obtained through sensing objects using the structured light implementation discussed in section 4.1. A Panasonic LCD projector was used with its output resolution set to 1024 x

768 pixels. A Logitech C100 webcam was exploited, with its capturing resolution set to 640 x 480 pixels. The respective output and input resolutions were set according to hardware limitations. The experimental setup is shown in figure 5.13.

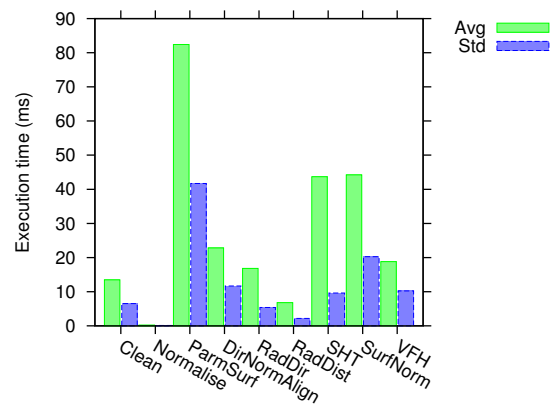


**Figure 5.13:** Experimental structured light setup.

The pattern shown in figure 5.14 was used for camera and projector calibration purposes. This procedure entailed estimating not only the internal and external projection matrices of both the camera and projector, but also the camera-projector alignment, as discussed in [74, 76].



**Figure 5.14:** Pattern used for system calibration.








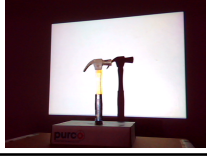






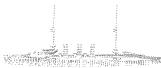












**Figure 5.15:** Practical point cloud processing times.

The data, captured by the structured light implementation, were presented to the system, which suggested appropriate objects, with pose information, from the database. All primitive features were compressed to bit strings of 64 bits and used for coarse database queries.

Execution times noted for practical feature extraction, as shown in the simulation system analysis subsection, together with the time taken to clean and normalise a point cloud, are presented in figure 5.15. Average execution times are shown in green, with associated standard deviation values in blue.

Practical object recognition and pose estimation results are shown in table 5.2. The sensed query object is shown in the first column, followed by the best database suggestion provided by the various implementations.

**Table 5.2:** Practical object recognition and pose estimation results.

Object recognition and pose estimation results				
Query object	Best suggested object and pose			
	MLH	BRE	LSH	SH
				
				
				
				
				

Most of the suggested database objects closely resemble the query object except for the cases where

a shoe and an upside-down hammer were presented to the system. The inaccuracies associated with the hammer can be related to the incomplete database used. No upside-down hammers were present in the database, which forced the system to suggest an alternative object, in this case a shovel.

The MLH and BRE implementations produced the most accurate results and regularly suggested similar objects from the database. This is consistent with the results obtained from the simulations discussed in the previous subsection. The LSH implementation, on the other hand, made the least accurate suggestions, which is also consistent with the simulations.

The execution times, presented in figures 5.2, 5.3, 5.4 and 5.15, indicate that the system should be able to produce a suggestion within approximately 230 ms after the scene was captured, using a single core of the processor. Such processing rates can be considered adequate for real-time applications when coupled with commercially available high-speed 3D range sensors.

#### 5.4.1 Computational improvements

The results presented in figure 5.15 only give an indication of the relative overhead of the individual extractions, but they do not clearly state what times are possible. Methods for optimising these implementations still need to be investigated of which some postulations are provided next.

The current implementation only makes use of a single core to extract and process all the information. This forced the system to extract features in a serial fashion. A multicore implementation would allow some, even all, of the features to be executed in parallel, which would increase system throughput considerably. The large number of matrix operations associated with extracting parameteric surface information could even be implemented on a graphical processing unit.

It is believed that not all system operations should be implemented as multicore operations. The coarse database search operation was optimised using specific SSE instructions for calculating the Hamming distance between bit strings. An attempt to implement the search technique as a multicore operation might incur significant overhead that could potentially degrade system throughput.

## CHAPTER 6

### CONCLUSION

The challenge of object recognition and pose estimation is frequently encountered across various fields of research involving artificial systems. Many proposed solutions adopt an approach which considers object recognition and pose estimation as disjointed, and deal with these issues separately. The research presented in this dissertation however, is based on an approach which regards object recognition and pose estimation as intertwined and attempts to solve these challenges simultaneously.

The task of object recognition and pose estimation entails simultaneously identifying an unknown object together with its position and orientation in 3D space. Such a task is vital to many artificial systems, as the information is often used in more complex perception algorithms. Autonomous systems would arguably benefit most from an accurate object recognition and pose estimation system. The versatility of such systems would be significantly improved in dynamic environments in response to enhanced perception capabilities.

The solution proposed in this dissertation integrates structured light, feature extraction and data compression techniques to address the challenge of SOPE. The recent introduction of affordable 3D range scanners into the commercial market has motivated the exploitation of these devices in autonomous robotic applications. The challenge of object recognition and pose estimation can now be directly addressed in all three dimensions. The implementation of the system is discussed in the subsequent section.

#### 6.1 IMPLEMENTATION

A database consisting of objects commonly found in typical households was constructed offline, using a subset of 3D models from the PSB. Additional vertices were populated into the 3D models in order

to simulate practical data and represent the objects as dense point clouds. Viewpoint information was included by positioning an artificial camera around the objects and trimming occluded vertices from the point cloud. The database did not incorporate object data obtained from the structure light implementation, as the process of acquiring such data was considered impractical using the current 3D sensor.

Primitive shape features and free-form deformation information, in the form of Bézier patches, were extracted from all the point clouds in the database. These primitive features and Bézier patches were then compressed into bit strings using data compression schemes originally suggested by the image processing community. More descriptive features, such as VFHs, were also extracted but not compressed into bit strings.

Novel objects were sensed in a scene and presented to the system as point clouds, using a structured light-based implementation. The implementation employed a binary, time-multiplexed projection pattern and produced range measurements typical to many commercially available 3D range scanners. More accurate 3D range measurements can be acquired, at increased rates, using application-specific computer vision equipment.

All novel inputs were processed similarly to the database entries in terms of feature extraction and compression. The novel inputs were however pre-processed to account for measurement errors and size variations before features were extracted. Novel inputs were also processed online.

The bit strings, resulting from compressing extracted features, were used to realise a hierarchical search structure. The search procedure produced objects from the database similar to the novel object. A coarse database search, using bit strings, flagged possible candidates which were analysed further, using more descriptive features, to produce the database objects that would be suggested by the system as estimates of the query objects.

## 6.2 RESULTS

The reaction time of artificial systems operating in dynamic environments is mostly limited by the latency of their perception systems, which in turn restricts their use in everyday applications [1]. The main objective of the research was to investigate means of developing a rapid and accurate object recognition and pose estimation system. The four important research questions listed in section 1.3



are addressed next.

- What aspects of current artificial vision and perception systems need to be improved in order to address the aforementioned latency problem?

Many artificial perception systems still rely on passive sensors to acquire scene information. These techniques incur significant latencies as oppose to active vision techniques. Structured light methods provide means for acquiring accurate 3D data at rates suitable for real-time applications. Active vision techniques are more robust against variations in illumination, as the projection patterns can be made much brighter than any other illumination source. This characteristic makes structured light implementations less prone to ambiguities and attractive for real-time applications, as accurate data can be acquired at rapid rates.

- What improvements can be incorporated into current state of the art object recognition and pose estimation procedures that will reduce execution time?

Data obtained using structured light implementations are often represented as dense point clouds and incur significant computational resources when processed. An alternative representation entails extracting relevant features from each point cloud. Similar point clouds should produce similar features, which reduces the task of comparing point clouds to comparing their respective features.

Extracting adequate features from point cloud data does not necessarily allow for fast data processing, specifically the challenge of comparing features. Recently proposed data compression schemes not only reduce storage requirements but also facilitate fast data processing. Bit compression techniques, originally suggested by the image processing community, are specifically developed to accelerate search queries when working with large databases.

It was noted that current system latencies can be reduced by incorporating a hierarchical database search structure. Bit strings should be used to represent database entries at the coarsest scale. This hierarchical approach to database querying significantly increased system throughput, as bit string queries can be performed at astonishing rates using available hardware. The idea of integrating a deformable model-based approach to SOPE was also investigated.

Experimental results indicate that the system was be able to produce an object and pose suggestion within approximately 230 ms after the scene was captured, using a single core of the processor. An

average object classification accuracy of approximately 73% was achieved, which is comparable to other suggested implementations such as VFHs [43] and spin images [18].

Database retrieval results presented in [46] suggest that the 3D features used for this implementation are far from optimal. The results indicate more accurate database retrievals when only using bit strings. The concept of content-based image retrieval is however more mature than content-based shape retrieval and motivates further research into more descriptive 3D features [18].

- What effects would improvements on execution time have on system accuracy?

Results indicate that an increase in system throughput can be achieved by compressing features into longer bit strings. Longer bit strings provide additional descriptive information, which is vital when performing a coarse database query operation. The drawback associated with the longer bit strings is an increase in storage requirements. A trade-off between performance, computational overhead and storage resources therefore needs to be considered.

The scalability of longer bits strings to significantly larger databases still needs to be investigated. The ultimate idea is to have large databases of objects to allow the system to operate accurately in most real-world environments. It would however be straightforward to expand the current database to accommodate such an experiment. This would simply entail introducing additional 3D models which can be obtained from publicly available 3D model databases.

- Would an acceptable real-time solution be realisable using existing technology?

The current implementation is capable of producing object and pose estimations in near real-time rather than in real-time. It should be noted that the implementation only makes use of a single core of the processing unit. Improvements allowing for the exploitation of multiple cores would reduce system latencies further. Also, replacing the current structured light implementation with dedicated hardware will reduce data capture latencies considerably. An acceptable real-time solution would be realisable given that the two aforementioned inadequacies are addressed.

### 6.3 FUTURE RESEARCH

Three subcomponents of the system were identified as primary candidates for future research and improvement. The first is the data capture component, of which the data acquisition procedure needs to

be accelerated. Secondly, and perhaps most importantly, are the features used to describe 3D objects. Thirdly is the post-processing component, which was neglected in this implementation.

The method used to acquire 3D scene data was sufficient for research purposes but can be improved significantly. A recently proposed technique, coined compressed sensing, suggests that sufficient data sampling can be achieved at rates lower than the Nyquist rate [96]. This implies that the 3D reconstruction process, discussed in section 4.1.1.3, can be accelerated, as fewer points correspondences need to be identified. Preliminary research suggests that a trade-off between computational complexity and accuracy will have to be considered.

Many proposed solutions that address the object recognition and object pose estimation challenges adopt a feature-based approach. It is however only recently that features have been proposed which attempt to address the object recognition and pose estimation issues simultaneously. This fairly recent introduction of these methods suggests that the development of appropriate 3D features, specifically intended for real-time applications, has yet to reach maturity [18].

Data compression techniques, originally suggested by the image processing community, were employed to compress extracted features. These compression schemes were considered specifically because they also facilitate fast query operations when working with large databases. Most of these techniques however were developed for processing features extracted from images, which might indicate that a more optimal compression scheme can be developed specifically for features extracted from point clouds.

Finally, it was pointed out in the section discussing the system layout that no post-processing of the suggested results was performed. This is because there are numerous methods that could be used to refine system outputs. A straightforward approach would be to employ the ICP algorithm as suggested previously. An alternative, and perhaps more fitting, approach would be to amalgamate multiple views of the same scene. Such an approach would provide the system with multiple observations of the same object, as seen from various known viewpoints.

The aforementioned improvement would however present some challenges. Firstly, an appropriate method of registering numerous viewpoints within a global coordinate system needs to be defined. Another issue would involve the process of specifying the optimal positions of alternative viewpoints, once an unknown object has been located in a scene.

## REFERENCES

- [1] M. Martinez, A. Collet, and S. Srinivasa, “MOPED: A Scalable and Low Latency Object Recognition and Pose Estimation System,” in *Proc. IEEE Int. Conf. on Robotics and Automation*, May 2010, pp. 2043–2049.
- [2] K. Al, F. Fleuret, D. Hasler, and P. Fua, “Joint Pose Estimator and Feature Learning for Object Detection,” in *Proc. 12th IEEE Int. Conf. on Computer Vision*, 2009, pp. 1373–1380.
- [3] A. Laubenheimer, S. Richter, and K. Kroschel, “3D Pose and Shape Estimation with Deformable Models in Lifelike Scenes,” in *Proc. 7th IEEE-RAS Int. Conf. on Humanoid Robots*, Nov.-Dec. 2008, pp. 159–166.
- [4] P. Jiménez, J. Nuevo, and L. Bergasa, “Face Pose Estimation and Tracking Using Automatic 3D Model Construction,” in *Proc. IEEE Computer Society Conf. on Computer Vision and Pattern Recognition Workshops*, Jun. 2008, pp. 1–7.
- [5] W. Ma, G. Hamarneh, G. Mori, K. Dinelle, and V. Sossi, “Motion Estimation for Functional Medical Imaging Studies Using a Stereo Video Head Pose Tracking System,” in *IEEE Nuclear Science Symp. Conf. Record*, Oct. 2008, pp. 4086–4090.
- [6] Q. Mühlbauer, K. Kühnlenz, and M. Buss, “A Model-based Algorithm to Estimate Body Poses using Stereo Vision,” in *Proc. 17th IEEE Int. Symp. on Robot and Human Interactive Communication*, Aug. 2008, pp. 285–290.
- [7] M. Yang, Q. Yu, H. Wang, and B. Zhang, “Vision based Real-Time Pose Estimation for Intelligent Vehicles,” in *Proc. IEEE Symp. on Intelligent Vehicles*, Jun. 2004, pp. 262–267.
- [8] A. Agrawal, Y. Sun, J. Barnwell, and R. Raskar, “Vision-guided Robot System for Picking

## References

---

- Objects by Casting Shadows,” *International Journal of Robotics Research*, vol. 29, no. 2-3, pp. 155–173, 2010.
- [9] E. Montijano and C. Sagues, “Fast Pose Estimation For Visual Navigation Using Homographies,” in *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, Oct. 2009, pp. 2704–2709.
- [10] D. Hähnel, W. Burgard, and S. Thrun, “Learning compact 3D models of indoor and outdoor environments with a mobile robot,” *Robotics and Autonomous Systems*, vol. 44, no. 1, pp. 15–27, Jul. 2003.
- [11] H. de Ruiter and B. Benhabib, “Visual-model-based, real-time 3D pose tracking for autonomous navigation: methodology and experiments,” *Autonomous Robots*, vol. 25, no. 3, pp. 267–286, Oct. 2008.
- [12] J. Salvi, J. Pages, and J. Batlle, “Pattern codification strategies in structured light systems,” *Pattern Recognition*, vol. 37, no. 4, pp. 827–849, Apr. 2004.
- [13] R. El-laithy, J. Huang, and M. Yeh, “Study on the Use of Microsoft Kinect for Robotics Applications,” in *Position Location and Navigation Symposium (PLANS)*, Apr. 2012, pp. 1280–1288.
- [14] F. Dellaert, S. Seitz, C. Thorpe, and S. Thrun, “Structure from Motion without Correspondence,” in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, vol. 2, Jun. 2000, pp. 557–564.
- [15] S. Nayar and Y. Nakagawa, “Shape from Focus,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 16, no. 8, pp. 824–831, Aug 1994.
- [16] F. Tsalakanidou, F. Forster, S. Malassiotis, and M. Strintzis, “Real-time acquisition of depth and color images using structured light and its application to 3D face recognition,” *Real-Time Imaging*, vol. 11, pp. 358–369, Aug. 2005.
- [17] F. Sandakly and G. Giraudon, “3D scene interpretation for a mobile robot,” *Robotics and Autonomous Systems*, vol. 21, no. 4, pp. 399–414, Oct. 1997.
- [18] A. Johnson and M. Hebert, “Using Spin Images for Efficient Object Recognition in Cluttered 3D Scenes,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 21, no. 5, pp. 433–449, 1999.

## References

---

- [19] G. Hetzel, B. Leibe, P. Levi, and B. Schiele, “3D Object Recognition from Range Images using Local Feature Histograms,” in *Proc. IEEE Computer Society Conf. on Computer Vision and Pattern Recognition*, vol. 2, Dec. 2001, pp. II394–II399.
- [20] S. McKenna and S. Gong, “Real-Time Face Pose Estimation,” *Real-Time Imaging*, vol. 4, no. 5, pp. 333–347, 1998.
- [21] J. Romero, H. Kjellstrom, and D. Kragic, “Monocular Real-Time 3D Articulated Hand Pose Estimation,” in *Proc. 9th IEEE-RAS Int. Conf. on Humanoid Robots*, Dec. 2009, pp. 87–92.
- [22] A. Erol, G. Bebis, M. Nicolescu, R. Boyle, and X. Twombly, “Vision-based hand pose estimation: A review,” *Computer Vision and Image Understanding*, vol. 108, no. 1-2, pp. 52–73, Oct. 2007.
- [23] A. Avanaki, B. Hamidzadeh, and F. Kossentini, “Object reconstruction and pose indexing by volume feedback,” in *Proc. Int. Conf. on Image Processing*, vol. 2, Sept. 2003, pp. 13–16.
- [24] M. Breuss, O. Vogel, and A. Tankus, “Modern Shape from Shading and Beyond,” in *Proc. 18th IEEE Int. Conf. on Image Processing*, Sep. 2011, pp. 1–4.
- [25] D. Scharstein and R. Szeliski, “A taxonomy and evaluation of dense two-frame stereo correspondence algorithms,” *International Journal of Computer Vision*, vol. 47, no. 1-3, pp. 7–42, Apr. 2002.
- [26] P. Gamage, S. Xie, P. Delmas, and P. Xu, “Pose Estimation of Femur Fracture Segments for Image Guided Orthopedic Surgery,” in *Proc. 24th Int. Conf. on Image and Vision Computing New Zealand*, Nov. 2009, pp. 288–292.
- [27] L. Rai, S. Merritt, and W. Higgins, “Real-time Image-Based Guidance Method for Lung-Cancer Assessment,” in *Proc. IEEE Computer Society Conf. on Computer Vision and Pattern Recognition*, vol. 2, Jun. 2006, pp. 2437–2444.
- [28] M. Ayad, J. Lee, A. Deguet, E. Burdette, and J. Prince, “C-arm Pose Estimation using a Set of Coplanar Ellipses in Correspondence,” in *Proc. IEEE Int. Symp. on Biomedical Imaging: From Nano to Macro*, april 2010, pp. 1401–1404.

## References

---

- [29] D. Terzopoulos and K. Fleischer, “Deformable models,” *The Visual Computer*, vol. 4, pp. 306–331, Nov. 1988.
- [30] A. Rangarajan, H. Chui, and E. Mjolsness, “A Relationship between Spline-based Deformable Models and Weighted Graphs in Non-rigid Matching,” in *Proc. IEEE Computer Society Conf. on Computer Vision and Pattern Recognition*, vol. 1, Dec. 2001, pp. I897–I904.
- [31] W. Ryu, Y. Kang, S. Baik, and S. Kang, “A study on the 3-D measurement by using digital projection moiré method,” *Optik*, vol. 119, no. 10, pp. 453–458, Aug. 2008.
- [32] K. Creath and J. C. Wyant, *Moiré and Fringe Projection Techniques*, 2nd ed. John Wiley & Sons, inc., 1992, ch. 16, pp. 653–685.
- [33] D. Forsyth and J. Ponce, *Computer Vision. A modern approach*. Prentice Hall, 2003.
- [34] W. Brink, “Real-time surface tracking with uncoded structured light,” in *Proc. Annu. 19th Symp. of the Pattern Recognition Association of South Africa*, 2008, pp. 91–95.
- [35] M. Young, E. Beeson, J. Davis, S. Rusinkiewicz, and R. Ramamoorthi, “Viewpoint-Coded Structured Light,” in *Proc. IEEE Computer Society Conf. on Computer Vision and Pattern Recognition*, Jun. 2007.
- [36] S. Inokuchi, K. Sato, and F. Matsuda, “Range-Imaging System for 3-D Object Recognition.” in *Proc. - International Conference on Pattern Recognition*, vol. 2, 1984, pp. 806–808.
- [37] D. Caspi, N. Kiryati, and J. Shamir, “Range Imaging With Adaptive Color Structured Light,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 20, no. 5, pp. 470–480, 1998.
- [38] G. Wiora, “High Resolution Measurement of Phase-Shift Amplitude and Numeric Object Phase Calculation,” in *Proc. of SPIE - The International Society for Optical Engineering*, vol. 4117, 2000, pp. 289–299.
- [39] F. MacWilliams and N. Sloane, “Pseudo-Random Sequences and Arrays,” *Proc. of the IEEE*, vol. 64, no. 12, pp. 1715–1729, Dec. 1976.
- [40] T. Etzion, “Constructions for Perfect Maps and Pseudorandom Arrays,” *IEEE Trans. on Inform. Theory*, vol. 34, no. 5, pp. 1308–1316, Sep. 1988.

## References

---

- [41] L. Zhang, B. Curless, and S. Seitz, "Rapid Shape Acquisition Using Color Structured Light and Multi-pass Dynamic Programming," in *The 1st IEEE Int. Symp. on 3D Data Processing, Visualization, and Transmission*, June 2002, pp. 24–36.
- [42] P. Shilane, P. Min, M. Kazhdan, and T. Funkhouser, "The Princeton Shape Benchmark," in *Proc. Shape Modeling International*, Jun. 2004, pp. 167–178.
- [43] R. Rusu, G. Bradski, R. Thibaux, and J. Hsu, "Fast 3D Recognition and Pose using the View-point Feature Histogram," in *Proc. IEEE Int. Conf. on Intelligent Robots and Systems*, Oct. 2010, pp. 2155–2162.
- [44] J. Tangelder and R. Veltkamp, "A survey of content based 3D shape retrieval methods," *Multimedia Tools and Applications*, vol. 39, no. 3, pp. 441–471, Sep. 2008.
- [45] A. Torralba, R. Fergus, and Y. Weiss, "Small Codes and Large Image Databases for Recognition," in *Proc. 26th IEEE Conf. on Computer Vision and Pattern Recognition*, Jun. 2008.
- [46] Y. Jiang, J. Wang, and S. Chang, "Lost in Binarization: Query-Adaptive Ranking for Similar Image Search with Compact Codes," in *Proc. 1st ACM Int. Conf. on Multimedia Retrieval*, Apr. 2011.
- [47] R. Duda, P. Hart, and D. Stork, *Pattern Classification*. John Wiley & Sons, inc., 2001.
- [48] C. Akgül, B. Sankur, F. Schmitt, and Y. Yemez, "Multivariate Density-Based 3D Shape Descriptors," in *Proc. IEEE Int. Conf. on Shape Modeling and Applications*, Jun. 2007, pp. 3–12.
- [49] S. Sheather, "Density Estimation," *Statistical Science*, vol. 19, no. 4, pp. 588–597, 2004.
- [50] R. Campbell and P. Flynn, "Eigenshapes for 3D Object Recognition in Range Data," in *Proc. IEEE Computer Society Conf. on Computer Vision and Pattern Recognition*, vol. 2, Jun. 1999, pp. 505–510.
- [51] B. K. P. Horn, "Extended Gaussian Images," *Proc. IEEE*, vol. 72, no. 12, pp. 1671–1686, Dec. 1984.
- [52] R. Rusu, N. Blodow, and M. Beetz, "Fast Point Feature Histograms (FPFH) for 3D registration,"



## References

---

- in *Proc. IEEE Int. Conf. on Robotics and Automation*, May 2009, pp. 3212–3217.
- [53] M. Mousa, R. Chaine, and S. Akkouche, “Frequency-based Representation of 3D Point-based Surfaces using Spherical Harmonics,” *Machine Graphics and Vision*, vol. 15, no. 3-4, pp. 537–545, 2006.
- [54] C. Cantalupo. (2004, Nov.) ccSHT: A fast parallel spherical harmonic transform. [Online]. Available: <http://crd-legacy.lbl.gov/~cmc/ccSHTlib/doc/>
- [55] M. Novotni and R. Klein, “Shape retrieval using 3D Zernike descriptors,” *CAD Computer Aided Design*, vol. 36, no. 11, pp. 1047–1062, Sep. 2004.
- [56] Y. Fang, M. Sun, and K. Ramani, “Temperature Distribution Descriptor for Robust 3D Shape Retrieval,” in *Proc. IEEE Computer Society Conf. on Computer Vision and Pattern Recognition Workshops*, Jun. 2011.
- [57] D. Terzopoulou, J. Platt, A. Barr, and K. Fleischert, “Elastically Deformable Models,” *Computer Graphics*, vol. 21, pp. 205–214, Aug. 1987.
- [58] U. Meier, O. López, C. Monserrat, M. C. Juan, and M. Alcañiz, “Real-time deformable models for surgery simulation: a survey,” *Computer methods and programs in biomedicine*, vol. 77, no. 3, pp. 183–197, Mar. 2005.
- [59] I. Matthews, J. Xiao, and S. Baker, “2D vs. 3D Deformable Face Models: Representational Power, Construction, and Real-Time Fitting,” *International Journal of Computer Vision*, vol. 75, no. 1, pp. 93–113, Oct. 2007.
- [60] M. Schultz and T. Joachims, “Learning a Distance Metric from Relative Comparisons,” in *Advances in Neural Information Processing Systems 16*. MIT Press, 2004.
- [61] Y. Lai, R. Orlandic, W. Yee, and S. Kulkarni, “Scalable Clustering for Large High-Dimensional Data Based on Data Summarization,” in *Proc. IEEE Symp. on Computational Intelligence and Data Mining*, Apr. 2007, pp. 456–461.
- [62] K.-I. Lin and C. Yang, “The ANN-tree: An index for efficient approximate nearest neighbor search,” in *Proc. of 7th Int. Conf. on Database Systems for Advanced Applications*, Apr. 2001,

## References

---

- pp. 174–181.
- [63] H. Jégou, M. Douze, and C. Schmid, “Hamming embedding and weak geometric consistency for large scale image search,” in *Proc. 10th European Conf. on Computer Vision: Part I*, Oct. 2008, pp. 304–317.
- [64] D. Lowe, “Distinctive Image Features from Scale-Invariant Keypoints,” *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, Nov. 2004.
- [65] G. Shakhnarovich, P. Viola, and T. Darrell, “Fast Pose Estimation with Parameter-Sensitive Hashing,” in *Proc. IEEE Int. Conf. on Computer Vision*, vol. 2, Oct. 2003, pp. 750–757.
- [66] M. Norouzi and D. Fleet, “Minimal Loss Hashing for Compact Binary Codes,” in *Proc. 28th Int. Conf. on Machine Learning*, Jun.-Jul. 2011, pp. 353–360.
- [67] R. Salakhutdinov and G. Hinton, “Semantic Hashing,” *International Journal of Approximate Reasoning*, vol. 50, no. 7, pp. 969–978, Jul. 2009.
- [68] Y. Weiss, A. Torralba, and R. Fergus, “Spectral Hashing,” in *Proc. 22nd Annu. Conf. on Advances in Neural Information Processing Systems*, Dec. 2009, pp. 1753–1760.
- [69] A. Andoni and P. Indyk, “Near-Optimal Hashing Algorithms for Approximate Nearest Neighbor in High Dimensions,” in *Proc. 47th Annu. IEEE Symp. on Foundations of Computer Science*, Oct. 2006, pp. 459–468.
- [70] R. Rusu and S. Cousins, “3D is here: Point Cloud Library (PCL),” in *Proc. IEEE Int. Conf. on Robotics and Automation*, May 2011.
- [71] G. Bradski, “The OpenCV Library,” *Dr. Dobb’s Journal of Software Tools*, 2000.
- [72] G. Guennebaud, B. Jacob, and et al. (2010, Aug.) Eigen v3. [Online]. Available: <http://eigen.tuxfamily.org>
- [73] V. Morariu, B. Srinivasan, V. Raykar, R. Duraiswami, and L. Davis, “Automatic online tuning for fast Gaussian summation,” in *Proc. 22nd Annu. Conf. on Advances in Neural Information Processing Systems*, Dec. 2009, pp. 1114–1120.

## References

---

- [74] D. Lanman and G. Taubin, “Build Your Own 3D Scanner: 3D Photography for Beginners,” in *ACM SIGGRAPH 2009 Courses*, Aug. 2009.
- [75] M. Özuysal, P. Fua, and V. Lepetit, “Fast Keypoint Recognition in Ten Lines of Code,” in *Proc. IEEE Computer Society Conf. on Computer Vision and Pattern Recognition*, Jun. 2007.
- [76] Z. Zhang, “A Flexible New Technique For Camera Calibration,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 11, pp. 1330–1334, Nov. 2000.
- [77] J. Lang and O. R"oschel, “Developable (1, n) - Bézier surfaces,” *Computer Aided Geometric Design*, vol. 9, no. 4, pp. 291–298, Sep. 1992.
- [78] W. Shengli and Z. Chongming, “NURBS Surface Generation by Control Points,” in *IEEE 3rd Int. Conf. on Communication Software and Networks*, May 2011, pp. 544–547.
- [79] E. Weisstein. (2013, Jan.) Barycentric Coordinates. [Online]. Available: <http://mathworld.wolfram.com/BarycentricCoordinates.html>
- [80] Y. Sakurai, M. Yoshikawa, S. Uemura, and H. Kojima, “The A-tree: An Index Structure for High-Dimensional Spaces using Relative Approximation,” in *Proc. 26th Int. Conf. on Very Large Data Bases*, Sep. 2000, pp. 516–526.
- [81] N. Katayama and S. Satoh, “The SR-tree: An Index Structure for High-Dimensional Nearest Neighbor Queries,” *SIGMOD Record (ACM Special Interest Group on Management of Data)*, vol. 26, no. 2, pp. 369–380, Jun 1997.
- [82] K.-I. Lin, H. V. Jagadish, and C. Faloutsos, “The TV-tree: An index structure for high-dimensional data,” *The VLDB Journal*, vol. 3, no. 4, pp. 517–542, Oct. 1994.
- [83] H. Jagadish, B. Ooi, K. Tan, C. Yu, and R. Zhang, “IDistance: An adaptive B+-tree Based Indexing Method for Nearest Neighbor Search,” *ACM Transactions on Database Systems*, vol. 30, no. 2, pp. 364–397, Jun. 2005.
- [84] S. Nene and S. Nayar, “Closest Point Search in High Dimensions,” in *Proc. IEEE Computer Society Conf. on Computer Vision and Pattern Recognition*, Jun 1996, pp. 859–865.
- [85] Y. Bengio, J. Paiement, P. Vincent, O. Delalleau, N. Le Roux, and M. Ouimet, “Out-of-Sample

## References

---

- Extensions for LLE, Isomap, MDS, Eigenmaps, and Spectral Clustering,” in *Proc. Conf. on Advances in Neural Information Processing Systems*. MIT Press, 2004, pp. 177–184.
- [86] C. Fowlkes, S. Belongie, F. Chung, and J. Malik, “Spectral Grouping Using the Nyström Method,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, no. 2, pp. 214–225, 2004.
- [87] C. Strecha, A. Bronstein, M. Bronstein, and P. Fua, “LDAHash: Improved matching with smaller descriptors,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 1, pp. 66–78, 2012.
- [88] A. Bronstein, M. Bronstein, R. Kimmel, M. Mahmoudi, and G. Sapiro, “A Gromov-Hausdorff Framework with Diffusion Geometry for Topologically-Robust Non-Rigid Shape Matching,” *International Journal of Computer Vision*, vol. 89, no. 2-3, pp. 266–286, Sep. 2010.
- [89] F. Mémoli, “Gromov-Wasserstein Distances and the Metric Approach to Object Matching,” *Foundations of Computational Mathematics*, vol. 11, no. 4, pp. 417–487, Aug. 2011.
- [90] M. Mahmoudi and G. Sapiro, “Three-Dimensional Point Cloud Recognition via Distributions of Geometric Distances,” *Graphical Models*, vol. 71, no. 1, pp. 22–31, Jan. 2009.
- [91] R. Osada, T. Funkhouser, B. Chazelle, and D. Dobkin, “Shape Distributions,” *ACM Transactions on Graphics*, vol. 21, no. 4, pp. 807–832, Oct. 2002.
- [92] F. Mémoli and G. Sapiro, “Comparing Point Clouds,” in *ACM Int. Conf. Proc. Series*, vol. 71, Jul. 2004, pp. 32–40.
- [93] F. Mémoli, “Gromov-Hausdorff distances in Euclidean spaces,” in *Proc. IEEE Computer Society Conf. on Computer Vision and Pattern Recognition Workshops*, Jun. 2008.
- [94] Y. Rubner, C. Tomasi, and L. Guibas, “Earth Mover’s Distance as a Metric for Image Retrieval,” *International Journal of Computer Vision*, vol. 40, no. 2, pp. 99–121, Nov. 2000.
- [95] A. Andoni, P. Indyk, and R. Krauthgamer, “Earth Mover Distance over High-Dimensional Spaces,” in *Proc. Annu. ACM-SIAM Symp. on Discrete Algorithms*, Jan. 2008, pp. 343–352.
- [96] L. Gan, “Block compressed sensing of natural images,” in *Proc. 15th Int. Conf. on Digital Signal Processing*, Jul. 2007, pp. 403–406.