UNIVERSITY OF SOUTHAMPTON

3-D Content-Based Retrieval and Classification with Applications to Museum Data

by

Simon Goodall

A thesis submitted in partial fulfillment for the degree of Doctor of Philosophy

in the
Faculty of Engineering, Science and Mathematics
School of Electronics and Computer Science

March 2007

UNIVERSITY OF SOUTHAMPTON

ABSTRACT

FACULTY OF ENGINEERING, SCIENCE AND MATHEMATICS
SCHOOL OF ELECTRONICS AND COMPUTER SCIENCE

Doctor of Philosophy

by Simon Goodall

There is an increasing number of multimedia collections arising in areas once only the domain of text and 2-D images. Richer types of multimedia such as audio, video and 3-D objects are becoming more and more common place. However, current retrieval techniques in these areas are not as sophisticated as textual and 2-D image techniques and in many cases rely upon textual searching through associated keywords. This thesis is concerned with the retrieval of 3-D objects and with the application of these techniques to the problem of 3-D object annotation. The majority of the work in this thesis has been driven by the European project, SCULPTEUR.

This thesis provides an in-depth analysis of a range of 3-D shape descriptors for their suitability for general purpose and specific retrieval tasks using a publicly available data set, the Princeton Shape Benchmark, and using real world museum objects evaluated using a variety of performance metrics. This thesis also investigates the use of 3-D shape descriptors as inputs to popular classification algorithms and a novel classifier agent for use with the SCULPTEUR system is designed and developed and its performance analysed. Several techniques are investigated to improve individual classifier performance. One set of techniques combines several classifiers whereas the other set of techniques aim to find the optimal training parameters for a classifier. The final chapter of this thesis explores a possible application of these techniques to the problem of 3-D object annotation.

Contents

List of Figures

List of Tables

Acknowledgements

Thanks to my supervisors, Paul H. Lewis and Kirk Martinez for all their support. Thanks to Hewlett-Packard for donations of equipment and to the SCULPTEUR project (IST-2001-35372) for funding, access to 3-D models of museum objects, and users for testing of the content-based retrieval system and classifier agent versions. Tony Tung and Francis Schmitt from GET-ENST for initial implementation of 3-D shape representation algorithms (Area Volume Ratio, Cord Histograms, Shape Distributions, Extended Gaussian Image, 3-D Hough Transform and the augmented Multi-resolution Reeb Graph. Thanks to the Princeton Shape Retrieval and Analysis Group for providing an extensive collection of 3-D objects and tools to evaluate retrieval performance. Finally I would like to thank my wife, Heather, and my parents for their continued support and encouragement throughout.

Chapter 1

Introduction

The growing number of large multimedia collections has led to an increased interest in content-based retrieval research. Content-based retrieval is concerned with retrieval based upon the data contained within a multimedia object (image, 3-D model, video etc), such as it's shape or colour rather than associated keywords which may or may not reflect the full semantics of the object. Applications of content-based techniques to image retrieval is an active research area but much less work has been reported on content-based retrieval of 3-D objects in a multimedia database context. Such objects are increasingly being captured and added to multimedia collections and the European project, SCULPTEUR, developed a museum information system which includes the introduction of facilities for content-based retrieval of the 3-D representations. The project was also concerned with another rapidly developing area: the semantic web, which will aid the use of semantically described data both for machine processing and enhanced human interaction.

Content-Based Retrieval has been an active research area for the last few decades with many advances in the area of image-based retrieval. This research has naturally progressed to other forms of multimedia; audio, video and now 3-D objects. 3-D objects offer many interesting advantages over traditional 2-D image retrieval. Such representations are increasingly becoming attractive to museums wishing to digitise sculptures and other objects as the cost of 3-D acquisition becomes cheaper. A 3-D object is an explicit representation of the surface of an object and in some cases may contain the internal structure as well. To gain the same information with a 2-D representation requires many images and special techniques to estimate the 3-D structure. In some cases such techniques can be used to create an actual 3-D representation of the object. Of course each different type of multimedia brings its own challenges and 3-D is no exception. A 3-D object can be represented in any orientation, position or scale. Additionally how the 3-D object is represented can differ vastly while still being visually similar. Two 3-D representations of the same object may differ considerably when looking closely at its representation. For example one version the surface can be composed of triangles

whereas in the other version it could be composed of quadrilaterals. Being able to cope with such differences is a key feature in any 3-D retrieval algorithm.

Closely linked to object retrieval is object recognition. Once it is possible to retrieve similar objects to a query object, it should then be possible to use the same retrieval techniques to begin to classify objects into different groupings. Of course visual similarity does not necessarily equate to similar classes of object and not all techniques can be expected to yield good classification results.

Another area of research gaining attention is that of annotation. Annotation assigns textual keywords to an arbitrary item of data to help describe that item and to enable retrieval through textual search engines. It is also possible to annotate items of data with concepts in an ontology and this provides a much richer description as any relations associated with that concept can be associated with the data item as well. In one sense, annotation can be thought if as an extension of classification. Once some data has been classified, that classification can be used as an annotation for that data.

A large portion of this work was performed as part of the SCULPTEUR project and as such there was some collaboration in the software produced. The main software component produced was FVS, the underlying component that facilitated 2-D and 3-D content-based retrieval. The 2-D algorithms were ported from the FVG tool in the ARTISTE project and most of the 3-D algorithms were implemented by GET-ENST. My contribution to FVS was a re-write of the internal architecture of the FVG tool to more easily allow new algorithms to be integrated, added support for 3-D objects and to allow easy addition of further types of media, fixed existing bugs and improved performance and memory utilisation. Additionally a MySQL UDF interface was written to allow fast retrieval when integrated with the rest of the SCULPTEUR system. A Java Native Interface was also written for experimentation within other systems. An ASCII based feature vector file format was implemented to allow experimentation with descriptor data in MATLAB. Multiple distance metrics have been implemented and integrated into some of the 3-D algorithms to investigate which distance metrics worked best with which descriptor. A novel component, the 3-D thumbnail generator, was also implemented using the FVS data structures for representing a 3-D object and has been integrated with Nautilus to allow previews of VRML objects on the file system. Nautilus is the file manager application that is part of the Gnome Desktop Environment. FVS has recently been released under the LGPL open source license. As part of the FVS development, numerous bug fixes and enhancements have been passed back to the supporting libraries, most notably VIPS and Cyber X3D. Support was added in Cyber X3D for compressed VRML objects and the VRML parser was fixed to allow 3-D objects with an arbitrary number of faces rather than only 3-D objects with a small number of faces.

As part of the classification work, several versions of the classifier agent were produced and additional code was produced for further experimentation. The first classifier agent

wrapped a JSP web front-end around a MATLAB back end. This used a Support Vector Machine implementation and some Java to MATLAB code that is freely available on the web. This agent was a proof of concept system to investigate whether 3-D shape descriptors were usable as inputs to classifiers. The second classifier agent wrapped a PHP front-end around a C++ back-end. This used a custom implementation the k-Nearest Neighbour classifier and k-Means clustering algorithms. A Particle Swarm Optimisation and Genetic Algorithm implementation was also written to investigate automatic classifier optimisation. A small application was written to communicate with a SCULPTEUR system to create training data sets (using the SRW client application to encode a query and retrieve the response). This version of the agent provided a test bed for user trials on the interface and potential uses of the system. Final experimentation was performed in MATLAB to make use of existing classifier implementations in the Netlab toolbox. Genetic Algorithms, Dynamic Classifier Selection and Classifier Ensembles were implemented in order to investigate how well these techniques can improve base classifier performance. Much of this software is described in Chapter 4.

Two main conference papers have been published from this work as well as a number of contributions to more general papers on the SCULPTEUR project (Addis et al., 2003b; Sinclair et al., 2005b; Kim et al., 2004; Addis et al., 2005a; Goodall et al., 2004a; Addis et al., 2003a). The first paper, Goodall et al. 2005a, is a comparison of a number of 3-D shape descriptors using both the Princeton Shape Benchmark and a data set composed of museum objects. The second paper, Goodall et al. 2005b, presents our initial experimentation with the Particle Swarm Optimisation algorithm that was part of the final version of the Classifier Agent.

The research objectives for this thesis are to design and develop a content-based retrieval system for 3-D objects using collections of multimedia objects from museums. This will include research into 3-D content-based retrieval algorithms, and classification techniques to speed up retrieval and provide recognition capabilities. The SCULPTEUR system has been tested with substantial museum collections and real users.

There are several main objectives for the work.

1. Evaluate suitability of various 3-D Content-Based Retrieval Algorithms for general purpose and specific retrieval operations.
2. Design and develop classifiers using 3-D Content-Based Retrieval Algorithms as inputs to different classification techniques.
3. Explore and evaluate the use of 3-D classification for annotating 3-D objects.

General purpose retrieval is concerned with finding similar objects to a query object without any prior knowledge about the query object. Specific retrieval is concerned

with finding similar objects to the query using knowledge about the query to refine the retrieval process. For example if the query object is a vase, a technique that works well on vases can be selected.

This thesis begins with some background material relating to 3-D content-based retrieval in Chapter 2 and classification techniques in Chapter 3. Chapter 4 introduces the SCULPTEUR project in more detail and describes the areas with which this thesis is concerned. This chapter describes the context in which the thesis has been written. This chapter also describes FVS, the content-base retrieval software developed during the course of this thesis and is the fundamental building block for content-based retrieval in SCULPTEUR and in the experimental work undertaken.

Chapter 5 gives an in depth analysis of various 3-D content-based retrieval algorithms against a publicly available data set, the Princeton Shape Benchmark and against a data set composed of 3-D objects provided by the museum partners. The 3-D descriptors are analysed for both their overall performance and on their performance for specific classes of object. Additionally, a number of different distance metrics are evaluated for each descriptor in order to determine what distance metrics improve the retrieval performance of a descriptor. A large range of different performance metrics are used to perform the evaluation.

Chapter 6 uses classification techniques to classify 3-D objects using as input the 3-D content-based retrieval techniques described in Chapter 5. Three popular classification techniques are used in the main body of work providing a large comparison for suitability as base classifiers and for a combined approach.

Chapter 7 describes how the content-based retrieval and classification techniques can be integrated with semantic web technologies to facilitate 3-D object annotation.

The thesis then finishes with some conclusions and future work in Chapter 8.

Chapter 2

Content-Based Retrieval Background

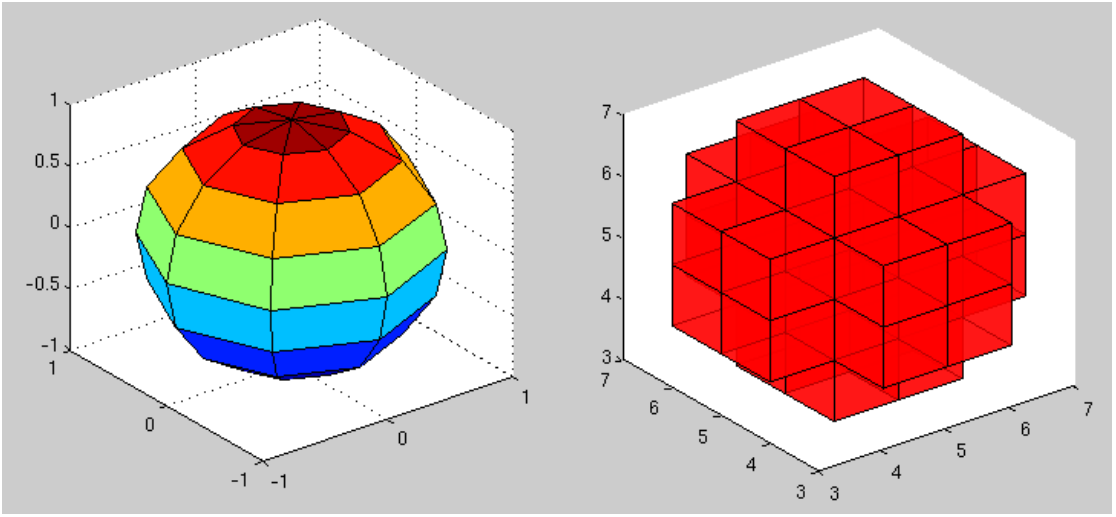
2.1 Introduction

In this chapter we review content-based retrieval techniques, in particular 3-D object retrieval. The chapter begins by discussing some of the issues involved with the retrieval of 3-D objects before describing a range of current 3-D descriptors and distance metrics. This chapter is also concerned with evaluating the retrieval performance of 3-D descriptors across different data sets, both for general purpose retrieval and for specific retrieval tasks.

2.2 3-D Object Representation

There are two main methods for representing arbitrary 3-D objects. One such method of representing a 3-D object is the mesh format. This is a collection of connected polygons forming either part of or the whole surface of an object. Many 3-D techniques assume that a mesh is composed of triangles rather than arbitrary sized polygons as this greatly simplifies calculations. A 3-D object can be composed of one or more meshes.

The other main method of representing a 3-D object is by using voxels. A voxel is a volume pixel, the 3-D equivalent of a pixel in a 2-D image. Unlike the mesh representation which models the surface of the object, a voxel models the whole volume of the object. As with 2-D images, increasing the scale of the model can result in blocky edges (pixelation). Often a model will be represented as a mesh and converted to voxels as needed. See Figure 2.1 for an example of a mesh and voxel representation of a sphere. The sphere has been shown deliberately low resolution to emphasise its construction.



(a) Mesh Representation (b) Voxel Representation

FIGURE 2.1: Example Mesh and Voxel representation of a Sphere

Another less commonly used method is to represent a 3-D object by a set of parameters. For example a sphere can be defined by a position and radius size. This method is only suitable for representing primitive objects, or those that can be represented easily by a pre-defined function.

2.3 3-D Content-Based Retrieval

3-D object matching is a growing research area and a wide range of differing techniques have been developed. 3-D content-based retrieval typically consists of four stages. The CBR descriptor generates a *feature vector* which contains the data representing a 3-D object according to the algorithm. In many cases this is a histogram.

- The first stage is to convert the object into a suitable format that is understandable by the rest of the process. This process may also involve re-sampling the object to provide a more even spread of vertices on the mesh. The initial sampling of the object (the creation of the mesh approximation at the time of acquisition or creation) may result in areas of the mesh being more densely populated than other areas. Typically flatter areas can be represented in a few large faces and very curved areas require many small faces. This process may also try to correct problems in the object, such as holes in the mesh or triangle orientation inconsistencies. This can be done once and the result saved for future use as this process is independent of the CBR algorithm.

- The next stage is to normalise the object into a canonical co-ordinate frame; that is to transform each object into a common co-ordinate system. However, some

algorithms are invariant to some aspects of possible transformations, e.g. rotation, scale and translation. The exact requirements depend on the properties of the algorithm.

- Stage three is to generate the feature vector for the descriptor from the object mesh.
- Stage four is to compare the feature vector with other feature vectors of the same type using an appropriate distance metric.

2.4 3-D Storage Formats

There are a wide range of storage formats for 3-D objects. Most formats represent an object as a collection of polygons (typically triangles) that form a mesh. Additional information such as surface normals, texture co-ordinates (and texture maps) and colour information are also commonly stored. Some formats (e.g. VRML (Web 3D Consortium, 1997) and X3D (Web 3D Consortium, 2004)) also allow 3-D objects to be represented by parameters (e.g. radius and position for a sphere). Many formats have been developed in association with a 3-D modelling packages (such as 3-D Studio (Autodesk, Inc, No Year) and Blender (Blender Foundation, No Year)) and may represent an entire 3-D scene containing camera information, animations etc. Other formats are designed to be quick to load for use in high performance games (e.g. MD3/MD4 file format in Quake (ID Software, 1999)).

VRML is the Virtual Reality Modelling Language and is a widely used format for distributing 3-D objects across the World Wide Web. It is a highly flexible ASCII based format and there are numerous viewer applications and plug-ins for this format and many 3-D packages list VRML as a supported file type. However the format allows for too much variability in describing an object which can cause problems when processing an object. An often referred to problem is called *polygon soup,* typical to VRML*,* meaning that an object can be represented by any number of unconnected or unstructured polygons which may visually look fine as a whole, but are horrendous for processing. VRML has been superseded by X3D (Web 3D Consortium, 2004), an XML version of VRML. However, it does not yet have such widespread usage.

The OFF file format has been used by the Princeton Shape Benchmark data set (Shilane et al., 2004) to represent all the 3-D objects. It is a simple text based format storing only the vertex information for each object.

The TRI file format is used to represent 3-D objects created by GET-ENST's 3-D object acquisition process (format not published). This is a binary representation storing information about a single mesh in the form of vertices and faces.

2.5 3-D Object Pre-Processing

The huge range of 3-D file formats, methods of object creation and user ability mean that the same object could be represented in many different ways but still be visually similar. Often some pre-processing of an object is required to bring it "in line" with other objects. Typically some processing would be required to transform all polygons into triangles, make sure the mesh is closed, make sure triangles connect to other triangles, fix normals, redistribute vertices to give a uniform sampling. Typically, it would be expected that objects obtained from the World Wide Web (WWW) will require considerably more pre-processing than objects created from a 3-D acquisition system.

In order to be able to perform a good comparison between objects, they should be geometrically similar, in their scale, orientation and position. This is important for some descriptor schemes as almost identical objects with even a slight rotation between the two can have a large difference in the resulting feature space. There are two main methods for achieving this. One is to build invariance into the descriptor itself (e.g. Saupe and Vranić, 2001), the other is to pre-process the model to transform it into a common reference frame (Vranić et al., 2001; Paquet et al., 2000).

Typically, most descriptors require transforming an object into a canonical co-ordinate frame, i.e. to normalise the object. This is to ensure that a given object of an arbitrary scale, orientation and position will always produce an identical feature vector for an identical model with a different geometric transformation. Often a descriptor will not be invariant to all geometric transforms, only some of them.

2.5.1 Translation Invariance

Typically, an object is translated so it's centre of mass is at the origin of the co-ordinate system. Care needs to be taken for meshes with an uneven distribution of vertices as this can cause a bias in the centre point.

2.5.2 Rotation Invariance

Principal Components Analysis (PCA), also known as the Karhunen-Loeve or Hotelling transform, is a commonly used method to provide rotation invariance to an arbitrary 3-D object. As part of this process, translation invariance is usually applied to the object. PCA is more commonly used to reduce the dimensionality of feature vectors.

PCA is more commonly used to reduce the dimensionality of feature vectors. A 3x3 matrix, M, is calculated as $M = X \cdot X^T$ where X is the set of all vertices in the mesh translated such that the centre of mass of the mesh lies at the origin. The eigenvalues of the matrix are used to sort the eigenvectors of the matrix to produce a rotation

matrix. This transforms the vertices in the mesh such that the greatest variation in vertex positions is along the x-axis. The y-axis points in the direction of greatest variation in vertex position in the yz plane.

The problem with applying PCA to mesh data is that typically vertices are not uniformly distributed across the mesh. This can cause problems during the rotation stage as areas of higher vertex density will have a greater effect than areas of lower vertex density. Several researchers have tackled this problem and have come up with several different methods to provide a solution. Vranić et al. (2001) weights each vertex against the surface area it represents, whilst Paquet and Rioux (1999b) use the centre of mass of the triangle as the input (instead of vertex position) weighted against the mass of the triangle. Ohbuchi et al. (2003a) use the point selected algorithm they modified from Osada et al. (2001) to provide a uniform distribution of points. An alternative method is to resample the object mesh to provide an even distribution of the vertices.

Sometimes PCA can align the object, but an axis can become flipped when the variance is equal in both directions along that axis. Körtgen et al. (2003) flips the object such that the "heavier" side of the object points along the positive direction on the axis. The heavier side is the side with the most triangles, or most "mass".

2.5.3 Scaling Invariance

Typically, scale invariance is achieved by scaling the object so that the maximum extent of the object along one (isotropic) or all (anisotropic) axes is of unit length, or fits within a unit cube bounding box. The actual size does not matter particularly as long as it is consistent across all objects put through this stage.

2.5.4 Mesh Invariance

While not necessarily a pre-processing step, we mention mesh invariance here for completeness. Depending on how a 3-D model was created for a given object, it could differ greatly in how the mesh is composed. Different models of the same object could vary in the number and types of polygons composing the mesh, the size of the polygons and to the degree of which the mesh approximates the object surface. Ideally, a 3-D descriptor will be able to overcome these differences, but some techniques still require a helping hand.

In the ideal case, it would be composed of many equally sized triangles. The larger the number of triangles, the better the approximation of the object surface can be, although this will increase the size of the model and computation time for processing. A typical pre-processing technique will re-sample the object to make the mesh consist of equally sized triangles or at least uniformly spaced vertices.

The different shape descriptors described in the following section use a range of techniques to overcome this limitation. Paquet and Rioux (1999b) use the surface area of the triangle under consideration as a weighting factor. Vranić et al. (2001) use a similar approach, however they weight the individual vertices rather than the triangle surface area. Osada et al. (2001) have developed a technique to pick random points on the surface of the object rather than a particular vertex or triangle.

With all of these techniques, there is still an element of variation as the mesh only approximates the surface of an object. Shape descriptors therefore need to be tolerant to variations in the surface between model representations.

2.6 3-D Algorithms

The majority of the work on 3-D model matching is based on finding similar shaped objects (Shilane et al., 2004; Tangelder and Veltkamp, 2004; Iyer et al., 2005). There have been some attempts at finding similarly coloured objects (Paquet and Rioux, 1999b) however this is not an area that has received much attention so far. The following provides an overview of a range of 3-D content-based descriptors.

The Area Volume Ratio descriptor, Cord Histograms, Shape Distributions, Modified Shape Distributions, Extended Gaussian Image, 3-D Hough Transform and the Augmented Multi-Resolution Reeb graph are used in the work presented in this thesis. These are the algorithms implemented within the SCULPTEUR project (see Chapter 4). The descriptions of the other algorithms are included for completeness.

2.6.1 Area Volume Ratio Descriptor

A simple geometric descriptor described by Tung and Schmitt (2004) is the ratio between surface area and volume of an object. Equation 2.1 shows the ratio in a dimensionless form. This is a single valued descriptor capturing only basic geometric properties, however it is invariant to rotation, scale and translation transforms and relatively quick to compute.

$$ratio = \frac{Area^3}{Volume^2} \tag{2.1}$$

$$Area = \frac{1}{2}\sum_{i}^{N} |(\mathbf{V}_{i,1} - \mathbf{V}_{i,0}) \times (\mathbf{V}_{i,2} - \mathbf{V}_{i,0})| \tag{2.2}$$

where $\mathbf{V}_{i,j}$ is the j^{th} vertex, $\mathbf{V}$ (a vector of x, y and z components), from triangle i.

$$Volume = \frac{1}{6}\sum_{i}^{N}(-V_{i,2}^{x}V_{i,1}^{y}V_{i,0}^{z} + V_{i,1}^{x}V_{i,2}^{y}V_{i,0}^{z} + V_{i,2}^{x}V_{i,0}^{y}V_{i,1}^{z} - V_{i,0}^{x}V_{i,2}^{y}V_{i,1}^{z} - V_{i,1}^{x}V_{i,0}^{y}V_{i,2}^{z} + V_{i,0}^{x}V_{i,1}^{y}V_{i,2}^{z}) \quad (2.3)$$

where $V_{i,j}^{a}$ is the x, y or z component (a) from the j^{th} vertex (0, 1 or 2), $\mathbf{V}$, of triangle i (Zhang and Chen, 2001).

2.6.2 Cord Histograms

The Cord Histograms by Paquet and Rioux (1999b) define an object in terms of *cords*. A cord is defined as the vector between the centre of mass of an object and a point on its surface. Three versions of the Cord Histogram are defined. The first is a histogram of cord lengths. The second type is a feature vector containing two histograms; a histogram of angles between a cord and the first principal axis, and a histogram of angles between a cord and the second principal axis. The third type is a bi-dimensional histogram indexed by angles between a cord and the first principal axis along one dimension and angles between a cord and the second principal axis along the other dimension. The cord histograms are defined using the centre point of each face, weighted according to the relative surface area of the face. The histograms are rotation and translation independent. Normalisation for scale is required for the first histogram type, however the other histograms are inherently invariant to scale. The Cord Histograms capture basic geometric information and so while relatively quick to compute, they will not be very discriminating.

2.6.3 Colour Descriptor

In addition to the Cord Histograms, Paquet and Rioux (1999b) also describe one of the few colour based descriptors for a 3-D object. This technique uses a voxel representation of the objects and each voxel has a colour value associated with it. A colour histogram is then used to describe the colour distribution. Colour is defined as a combination of the texture map, material properties and vertex colour information stored within the object representation. This method is dependent on not only the size of the bins used in the histogram (quantisation of the colour space), but also the resolution of the voxel representation which will typically be generated from a mesh based representation.

2.6.4 Shape Distributions

The Shape Distributions by Osada et al. (2001) are a collection of descriptors that capture distributions of various features of the shape of an object. The study performed by Osada et al. (2001) determined that the D2 variant resulting in a probability density function performed best overall. The D2 variant captures the distribution of the distances between pairs of random points on the surface of a 3-D object. This descriptor is invariant to translation and rotation transforms. It is also robust against changes in mesh resolution for a given object. It is however sensitive to changes in object scale and so requires normalisation for scale.

The different variations are:

A3 Measures the angle between three random points (A, B, C)on the surface of a 3-D model. This is the angle between vectors $\overrightarrow{BA}$ and $\overrightarrow{BC}$.

D1 Measures the distance between a fixed point (e.g. centroid) and a random point on the surface of the model. This is similar to Cord length in the first Cord Histogram of Paquet and Rioux (1999b).

D2 Measures the distance between two random points on the surface.

D3 Measures the square root of the area of the triangle between three random points on the surface.

D4 Measures the cube root of the volume of the tetrahedron between four random points on the the surface.

The point selection algorithm used is important as it treats an object as a surface instead of individual triangles. This provides invariance to mesh resolution in a way that can be applied to many different techniques. To select a point on the surface, a table of the cumulative triangle surface area is generated. A random number generator is used to obtain a cumulative area value which corresponds to a triangle in the table. Two random numbers, r_1 and r_2, are generated in the range [0.0, 1.0]. Equation 2.4 generates a point, P, on the triangle $(\mathbf{A}, \mathbf{B}, \mathbf{C})$ surface given r_1 and r_2. As a global descriptor, the shape distributions may not be able to capture the finer details of more complicated objects. Figure 2.2 helps illustrate this equation.

$$P = (1 - \sqrt{r_1})\mathbf{A} + \sqrt{r_1}(1 - r_2)\mathbf{B} + \sqrt{r_1}r_2\mathbf{C} \tag{2.4}$$

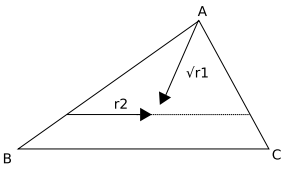


FIGURE 2.2: Selecting a point on the surface of a triangle

2.6.5 Modified Shape Distribution

Based on the work on Shape Distributions by Osada et al. (2001), Ohbuchi et al. (2003a) proposes several descriptors based upon the Shape D2 descriptor. These are the modified Shape D2 (mD2), the Angle-Distance (AD) histogram and the Absolute-Angle Distance (AAD) histogram descriptors, the latter two additionally take into account surface orientation. These versions of the Shape D2 descriptor both calculate the distribution from *all possible parings* of points selected, using a quasi-random number sequence (QRNS) to select the inputs to the point selection algorithm, (r_1 and r_2), on the surface of the triangle. This differs from a pseudo-random number sequence (PRNS) in that the QRNS produces more consistent feature vectors as the same points will always be selected for a given model.

A pseudo random number generator is the more common type. It calculates a number in the range [0.0, 1.0] given an initial seed which is updated each time a number is requested. It is not truly random as given the same seed, the same sequence of random numbers can be produced. With *enough* random numbers sampled, the distribution will be uniform. This, however, could be a large number of samples. The quasi random number generator again is not really a random generator as a predictable sequence of numbers is generated. The advantage is that the numbers generated will provide uniform sampling (again, for proper uniform sampling a suitable number of samples is required, however, this is more easily calculated). An easy way to think about the QRNS is to sub-divide a line (e.g. from 0.0 to 1.0). The division point can be thought of as the first random number. Each segment can then be subdivided again and the next set of random numbers returned.

The AD and AAD descriptors measure the mutual orientation of the surfaces on which the pair of points are located. The mutual orientation is the angle calculated as the inner product of the two surface normals. The additional information is stored in a 2-D histogram (indexed by distance and angle) as opposed to a 1-D histogram (indexed by distance). The difference between the AD histogram and the AAD histogram is that the AD histogram respects the sign of the angle (and so requires consistently orientated surface normals). The choice of descriptor (AD or AAD) depends on whether the surface normals of the models are properly and consistently orientated. If they are the AD descriptor is used and AAD if they are not. The histograms are normalised to improve comparison results. Out of the four normalisation methods proposed, normalisation by average produced the best results. The other normalisation methods are called maximum,

median and mode. The maximum method splits the values between the maximum and the minimum distances into equally spaced intervals. The average method is similar, except the intervals above the average values can be of different spacing to those below the average. The median and mode methods are similar, except using the median and mode instead of the average respectively. While the mD2 is still very similar to the Shape D2 descriptor, the AD and AAD give better retrieval performance results.

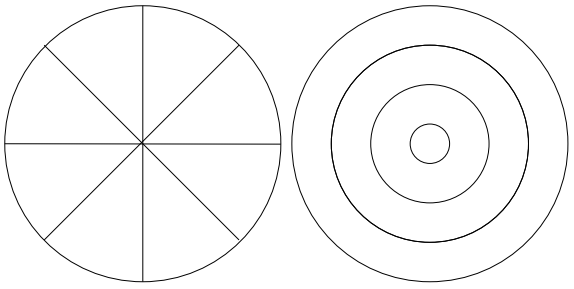
2.6.6 Parameter Methods

Ohbuchi et al. (2002) developed a descriptor based on a parametrised approach. They reason that a collection of descriptors will perform better than any single descriptor. The descriptor is composed of three statistics applied to the three principal axes. These are moments of inertia about the axis, the average distance to surface points from the axis and the variance of the distance to the surface points from the axis. The Euclidean and elastic distance metrics are used for matching on the moments of inertia and the other statistics are used as a weighting factor if required. While the authors do not compare this descriptor against any others, they are keen to point out that this is a more general descriptor framework dependant upon the choice of statistics used.

The parametrisation splits an object into slices along each of the principal axis in turn. A sliding window is applied to all consecutive pairs of slices to allow for mis-alignment of the object during pose normalisation. The statistics are calculated on each window position. This results in nine vectors, one per statistic per axis, which are concatenated together into a single vector.

2.6.7 Multiple Orientation Depth Fourier Descriptor

The Multiple Orientation Depth Fourier Descriptor (MODFD) by Ohbuchi et al. (2003b) uses the generic Fourier descriptor by Zhang and Lu (2002c) applied to 42 different 2-D views generated from a 3-D object. It requires normalisation for scale and translation invariance. However, the method by Zhang and Lu (2002c) provides rotation invariance by representing the object in terms of polar co-ordinates instead of Cartesian. The mapping from Cartesian to polar co-ordinates changes the rotation into a translation and takes advantage of the translation invariance provided by a Fourier transform. Similarity is calculated by the sum of the minimum distances between views on one object and all other views on the other. This descriptor performs slightly better than the AAD descriptor in the comparison by Ohbuchi et al. (2003a).



(a) The Sector Model (b) The Shell Model

FIGURE 2.3: The Sector and Shell Models for the Shape Histograms

2.6.8 3-D Shape Histograms

The Shape Histograms by Ankerst et al. (1999) partition the space inside the bounding sphere of an object either with shells (concentric spheres), sectors (planar slices) or a combination of the two. The shell based approach records the distance between the centre of mass to points on the surface (see Figure 2.3 (b)). This is similar to the first kind of cord histogram by Paquet and Rioux (1999b) and the Shape D1 descriptor by Osada et al. (2001). The sector based approach records the area of the model contained within each sector (see Figure 2.3 (a)). The shell model is rotation and translation invariant but requires normalisation for scale. The sector model is scale and translation invariant, however normalisation for rotation in Ankerst et al. (1999) use models represented by uniformly distributed points, such that the shells and sectors bins can be calculated by the number of points within the partitioned space. However the volume of the object contained within the space could also be used. These descriptors were evaluated by Shilane et al. (2004) and showed that the sector-shell model performed quite well, whilst the shell model performed worst out of the descriptors evaluated. The sector model gave better performance than the Shape D2, but still significantly worse than the sector-shell model.

2.6.9 3-D Shape Contexts

Körtgen et al. (2003) combines the work on Shape Histograms (Ankerst et al., 1999) with Shape Contexts (Belongie et al., 2002) and provides a set of descriptors called 3-D Shape Contexts. The shape contexts take a number of sampled points, and for each point generates the histogram of relative positions of all the other sampled points. In 3-D, the histogram is one of Ankerst's shape histograms centred upon the point. An additional change from Ankerst's shell model is to use a logarithmic scaling to determine boundary positions to give a more even distribution of the volume (as outer shells bound a larger volume than inner shells). The shape histograms are orientated about a point such that the first axis points towards the centre of mass. The remaining axes are determined by

projecting the principal axis onto the plane defined by the selected point and the first axis (as if it was the normal to the plane).

Similarity calculation compares the features points on one object to the points on another object. Three features are described; The Shape Term compares the shape histograms of the two points. The Appearance Term measures the distance between the orientation of the two histograms. The Position Term measures the distance between the two points, using a function similar to the squared Euclidean distance. Finally, the three terms are combined using a a set of user or automatically defined weights. The results showed good retrieval results, however they were not compared against other descriptors.

2.6.10 Extent Descriptor

Heczko et al. (2001) introduced a simple shape descriptor that measures the extent of an object along a fixed set of uniformly distributed vectors radiating from the centre of mass. It is referred to as the Extent Descriptor here, but it is sometimes referred to as the Radial Descriptor in the literature. In the case of multiple surface intersections along a vector, the furthest extent is used. Matching can then be performed by comparing the distance between corresponding vectors on different objects. This descriptor requires normalisation for scale and rotation invariance. The descriptor is sensitive to noise.

2.6.11 Spherical Harmonics

Further to the original extent descriptor, Vranić et al. (2001) makes use of spherical harmonics to improve the robustness while representing the object with a few coefficients in the spectral domain. Spherical harmonics allow any spherical function $f(\theta, \phi)$ to be decomposed into the sum of its harmonics;

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

where $Y_l^m(\theta,\phi)$ is the spherical harmonic function and $\hat{r}(l,m)$ is a weighting for the spherical harmonic. Different weightings can be used to characterise different functions.

The extent vector is converted into a simple spherical function. Given a spherical co-ordinate, the extend of the vector from the centre of the object to the furthest surface point of the object in the direction defined by the spherical co-ordinate is returned. Combining the Fourier transform applied to a sphere with this spherical function forms the basis of the descriptor. Vranić and Saupe (2002) further improve the descriptor by also taking into account the orientation of the surface along the extent vector.

2.6.12 Discrete Fourier Transform

In alternative work by Vranić and Saupe (2001a), a voxel representation of an object is first normalised for orientation and a discrete Fourier transform is applied to it. This represents the object in the frequency domain. It requires a suitably high degree of resolution for the voxelisation process in order to capture the finer details of the object. The Fourier transform provides a small number of co-efficients which compose the feature vector. For a octree of N^3 cells, each coefficient g_{uvw} can be calculated by Equation 2.5 where q_{ikl} is a cell in the octree. Vranić and Saupe (2001a) suggest a value of 128 for N with values of u, v and w in the range of [-3:3].

$$g_{uvw} = \frac{1}{\sqrt{N^3}} \sum_{i=-\frac{N}{2}}^{\frac{N}{2}-1} \sum_{k=-\frac{N}{2}}^{\frac{N}{2}-1} \sum_{l=-\frac{N}{2}}^{\frac{N}{2}-1} q_{ikl} \exp\left(-j\frac{2\pi}{N}(iu + kv + lw)\right) \tag{2.5}$$

2.6.13 3-D Moments

3-D Moments are a popular type of descriptor that has received the attention of several researchers. Paquet and Rioux (1999a) calculate moments in terms of the centre of mass for all triangles with respect to the mass of the triangle (See Equation 2.6). Saupe and Vranić (2001) calculate moments in terms of the extent of an object in a given direction with respect to the surface area of the object. In this case m_i is the surface area of triangle i multiplied by the distance from the centre of mass of the object. In both cases, a normalisation step is required for rotation invariance. The 3-D moments usually have low retrieval performance, although this strongly depends upon the order used.

$$M_{qrs} = \sum_{i=1}^{N} m_i (x_i - x_{cm})^q (y_i - y_{cm})^r (z_i - z_{cm})^s \tag{2.6}$$

where q, r, s are the moments order, (x_{cm}, y_{cm}, z_{cm}) is the centre of mass of the object, m_i is the mass of the triangle with a centre of mass at x_i, y_i and z_i. N is the number of triangles.

Saupe and Vranić (2001) use $1 \leq q + r + s \leq m$ for m values ranging from 2 to 6.

2.6.14 Extended Gaussian Image

The Extended Gaussian Image (EGI), originally developed by Horn (1984), is a spherical function resulting in a histogram of the distribution of the surface normals of an object. While translation and scale independent, it still requires normalisation for rotation. Typically the orientation histogram, a discrete approximation of the EGI, will be used for

mesh based objects. For each triangle, the surface area is added to the histogram bin representing the direction of the surface normal. The Complex EGI (Kang and Ikeuchi, 1993) stores a complex number where the real component represents surface area and the phase component represents the distance of the surface from the origin.

2.6.15 3-D Hough Transform

Zaharia and Preteux have developed several successive versions the Hough Transform for use in 3-D. The original development (Zaharia and Prêteux, 2001b) produced the Optimised 3-D Hough Transform Descriptor (O3DHTD) then in later work (Zaharia and Prêteux, 2002) the Canonical 3D Hough Transform Descriptor (C3DHTD). The Hough Transform transforms an object into Hough Space; an accumulator which gathers evidence of how similar the query is to the reference. For each object, a look up table is generated to perform this mapping.

The 3-D Hough Transform requires calculating a Hough Transform (HT) from all possible orientations of the x, y and z axes from views down each axis, however this number can be reduced by taking into account the fact that some pairs of orientation are equivalent, and that other views can be generated through a simple geometric transform. This culminated in the O3DHTD based on three views. The C3DHTD reduced this to a single HT by defining the object in such a way that all views become equivalent. The largest disadvantage of using a HT is that it requires a large amount of processing to provide a comparison as the computationally expensive part (populating the accumulator) cannot be pre-computed. It also requires normalisation for rotation, scale and translation.

Similarity matching is performed by comparing the tables treated as histograms. The true Hough Transform method creates an accumulator that maps one object into another one and sees how well it matches. However, this is quite slow compared to matching just the histograms.

2.6.16 The 3-D Shape Spectrum Descriptor

The 3-D Shape Spectrum Descriptor (3DSSD) by Zaharia and Prêteux (2001a) is defined as the distribution of the shape index over the entire mesh. The shape index is the function of the two principal curvatures. It is a local geometrical attribute of a 3-D surface. It is expressed as the angular co-ordinate of a polar representation of the principal curvature vector. It provides a scale for representing salient elementary shapes such as convex, concave, rut, ridge and saddle. It is invariant to rotation and translation transforms. The 3DSSD is a continuous function and for use with polygonal models, the descriptor is estimated. The 3DSSD is sensitive to topological changes meaning that objects differing only in some pose (e.g. a human with arms out and arms by the

sides) will be treated differently. The 3DSSD is the descriptor used by MPEG-7 (see Section 2.7).

2.6.17 Light Field Descriptors

The Light Field Descriptors by Chen et al. (2003) convert the 3-D shape matching problem into a 2-D shape matching problem by generating 2-D silhouettes of the 3-D object at various camera positions and orientations. These 2-D silhouettes are compared by using a combination of 2-D shape matching techniques to determine the similarity. These are Zernike moments (Zhang and Lu, 2002b) and Fourier descriptors (Zhang and Lu, 2002a). A combined feature vector based on the moments and Fourier coefficients is the result.

Similarity is performed by calculating the sum similarity of each image match. Images are matched by finding the orientation of images that gives the maximum similarity. This provides a degree rotation invariance.

Ten silhouettes were determined to be sufficient to represent the whole 3-D object. The use of silhouettes exploits the fact that they are mirrored when the object is rotated by 180 degrees as twenty views would otherwise be required.

2.6.18 Reflective Symmetry Descriptor

The Reflective Symmetry descriptor by Kazhdan et al. (2002) is a descriptor that measures the amount of symmetry (or not) in an object. In the 2-D case, it works by averaging an image against itself reflected along a line of symmetry. The descriptor is defined for all planes that go through the centre of mass. To do this efficiently, the fast Fourier Transform is used to calculate the symmetry. Extended to the 3-D case, "slices" or projections of a sphere are used to make into multiple 2-D problems. Visually, this is represented by deforming a unit sphere. Areas of higher symmetry cause the sphere to extend outwards, whereas areas of lower symmetry will not. Comparisons with the Shape Distributions of Osada et al. (2001), moments and random retrievals show that the reflective symmetry descriptor performs significantly better.

The 3-D object is converted to a voxel representation and decomposed into a series of concentric spheres. A Fourier Transform is then applied. The use of a FT defined on a sphere allows for rotation invariance.

2.6.19 Sphere Projection

The Sphere Projection descriptor by Leifman et al. (2003) computes the amount of "energy" required to deform an object into a predefined 3-D shape such as an enclosing

sphere. Energy is proportional to the average distance between the pairs of points on the surface of the object and the corresponding points on the sphere that lie in the same direction with respect to centre of mass. The feature vector is composed of two parts. The first part represents the minimal distances from the sphere to the object's surface. The second part represents the object's surface in terms of spherical coordinates.

2.6.20 Octree

An octree is the 3-D equivalent of a quad-tree (Ayala et al., 1985). It recursively decomposes a bounded 3-D space into eight equally sized partitions. Typically this method is used to efficiently store a voxel representation of a 3-D object. Each cell of an octree will either contain no voxels, be completely full of voxels, or it will be further partitioned into another eight cells. Leifman et al. (2003) use the octree as a descriptor by comparing the difference in volume between corresponding tree nodes. The octree is translation and scale independent, however it requires normalisation for rotation.

2.6.21 Reeb Graphs

The Reeb graph represents the skeletal and topological structure of an object. This is represented in a graph of interconnected nodes based upon a suitable function. The most common function is the height function on a 2-D manifold. Hilaga et al. (2001) proposes a multi-resolution version that construct a Reeb graph at various resolutions by re-partitioning at each node. Tung and Schmitt (2004) takes this approach further and store geometrical attributes at each node on the graph. These features are the Cord histograms and colour statistics of Paquet and Rioux (1999b), local curvature as used in the 3-D Shape Spectrum Descriptor by Zaharia and Prêteux (2001a) and volume associated with the node. It is invariant to rotation and translation transforms.

2.6.22 Descriptor Summary

Table 2.1 gives a summary of the different descriptors. The * denotes tested experimentally in Chapter 5. It classifies the descriptors into several groups. Global features capture the overall features of object within a single vector. Local features capture the variations at boundary locations. Graph based features take into account the geometry of the object. Spatial features captures relationships between locations on the object. View based features take into account the visual similarity between views of an object. Generally most descriptors produce a histogram and comparison speed is proportional to the number of bins in the histogram. Some methods like the Reeb Graph produce much

Descriptor Name	Type	Retrieval Performance	Speed
Cord Histograms* Paquet and Rioux (1999b)	Global	Low->Medium	Fast
Colour Paquet and Rioux (1999b)	Global	Low	Fast
Shape Distributions* Osada et al. (2001)	Global	Medium	Fast
mD2* Ohbuchi et al. (2003b)	Global	Medium	Fast
AD and AAD Ohbuchi et al. (2003b)	Global	Medium	Fast
Parametrised Vectors Ohbuchi et al. (2002)	Global	Medium	Fast
Shape Histograms Ankerst et al. (1999)	Spatial	Medium	Fast
Shape Contexts Körtgen et al. (2003)	Local	Medium	Medium
Spherical Extent Heczko et al. (2001)	Spatial	Low	Fast
Complex Function Vranić and Saupe (2002)	Global	Medium	Fast
3D-DFT Vranić and Saupe (2001a)	Global	Medium	Fast
3D-Moments Saupe and Vranić (2001)	Global	Low	Fast
Spherical Harmonics Saupe and Vranić (2001)	Spatial	High	Fast
Area Volume* Tung and Schmitt (2004)	Global	Low	Fast
MODFD Ohbuchi et al. (2003b)	Global	Medium	Medium
Sphere Projection Leifman et al. (2003)	Global	Medium	Fast
3D-Hough Transform* Zaharia and Prêteux (2001b)	Global	Medium	Slow
3DSSD Zaharia and Prêteux (2001a)	Local	Medium	Medium
LFD Chen et al. (2003)	View	High	Slow
Reflective Symmetry Descriptor Kazhdan et al. (2002)	Global	Low	Medium
Octree Leifman et al. (2003)	Global	Low	Fast
Reeb Graph* Hilaga et al. (2001); Tung and Schmitt (2004)	Graph	Medium	Slow
EGI* Horn (1984)	Global	Medium	Slow

TABLE 2.1: Descriptor Properties

more complex feature vectors and have much greater computational requirements. Generation speed is much slower by comparison, however it is less important and generation of features can typically be done offline.

2.7 MPEG-7

MPEG-7 (Martínez, 2004) is a content description specification. It is known as Multimedia Content Description Interface and it provides a framework for describing multimedia content. It has three main elements. The descriptor, D, the description scheme, DS and the description definition language, DDL, defined in XML.

The descriptor is the feature representation. The description scheme specifies the structure of the descriptor(s) and the relationship between them. The DDL is the language used to specify the description scheme.

2.8 Distance Metrics

In order to establish the similarity (closeness) of two feature vectors in some feature space, a wide range of distance metrics have been presented in the literature.

A distance metric calculates the distance between two point sets in metric space. A distance metric satisfies the following properties (Iyer et al., 2005);

- $d(\boldsymbol{x}, \boldsymbol{y}) \geq 0$ (positivity),
- $d(\boldsymbol{x}, \boldsymbol{y}) = 0$ *iff* $\boldsymbol{x} = \boldsymbol{y}$ (identity),
- $d(\boldsymbol{x}, \boldsymbol{y}) = d(\boldsymbol{y}, \boldsymbol{x})$ (symmetry),
- $d(\boldsymbol{x}, \boldsymbol{y}) + d(\boldsymbol{y}, \boldsymbol{x}) \geq d(\boldsymbol{x}, \boldsymbol{z})$ (triangle inequality).

where $d(\boldsymbol{x}, \boldsymbol{y})$ is the distance between a vector $\boldsymbol{x}$ and a vector $\boldsymbol{y}$.

Typically if only one feature vector is being considered it does not matter what the exact score returned by the metric is, however the ordering of the results is important. In the following list the Minkowski norms, histogram intersection, Chi squared, Bhattacharyya, Kullback-Leibler and the Quadratic distance metrics are used within the later chapters of this thesis. The other distance metrics are presented for completeness.

- Minkowski Norms

The most commonly used metrics are the Minkowski norms (Equation 2.7). Typically the L_1 norm (the city-block distance) and the L_2 norm (the Euclidean distance) are used.

$$d(\boldsymbol{x}, \boldsymbol{y}) = \left(\sum_{i=1}^{N} |x_i - y_i|^L \right)^{1/L} \tag{2.7}$$

where L is the degree of the norm and N is the number of elements in the vectors. The norms are popular not only due to their simplicity and speed of calculation, but to the quality of results obtained given their simplicity.

- Histogram Intersection

The histogram intersection (Hetzel et al., 2001) (Equation 2.8) is another simple distance metric that is often used. For histograms normalised so that the sum of the bins is one, the distance is calculated as one minus the sum of the minimum values of corresponding bins between two histograms.

$$d(\boldsymbol{x}, \boldsymbol{y}) = 1 - \sum_{i=1}^{N} \min(x_i, y_i) \tag{2.8}$$

- Chi Squared

The χ^2 (chi squared) distance (Hetzel et al., 2001) (Equation 2.9 for comparing unknown distributions) is based on the χ^2 statistical test, however, the final score is not required for a distance calculation, only the χ^2-divergence.

$$d(\boldsymbol{x}, \boldsymbol{y}) = \sum_{i=1}^{N} \frac{(x_i - y_i)^2}{x_i + y_i} \tag{2.9}$$

- Bhattacharyya Distance

Other distance metrics include the Bhattacharyya distance (Thacker et al., 1997) (Equation 2.10 and 2.11), a statistical measure often used for comparing two probability density functions,

$$d(\boldsymbol{x}, \boldsymbol{y}) = \sum_{i=1}^{N} \sqrt{x_i} \sqrt{y_i} \tag{2.10}$$

$$d(\boldsymbol{x}, \boldsymbol{y}) = -\log \sum_{i=1}^{N} \sqrt{x_i}\sqrt{y_i} \tag{2.11}$$

- Kullback-Leibler

The Kullback-Leibler distance (Hetzel et al., 2001) is another measure often used for comparing probability density functions. See Equation 2.12 for the symmetrical version (also known as the Jeffrey's Divergence) and Equation 2.13 for the non-symmetric version (this is not strictly a distance metric although its often used) where ln is the natural logarithm and $log_2$is logarithm to the base of 2.

$$d(\boldsymbol{x}, \boldsymbol{y}) = \sum_{i=1}^{N} (x_i - y_i) \ln \frac{x_i}{y_i} \tag{2.12}$$

$$d(\boldsymbol{x}, \boldsymbol{y}) = \sum_{i=1}^{N} x_i \log_2 \left(\frac{x_i}{y_i} \right) \tag{2.13}$$

- Earth Mover's Distance

The Earth Mover's distance (Rubner et al., 1998) (Equation 2.14),

$$d(\boldsymbol{x}, \boldsymbol{y}) = \frac{\sum_{i \epsilon \imath} \sum_{j \epsilon \jmath} c_{ij} f_{ij}}{\sum_{j \epsilon \jmath} y_j} \tag{2.14}$$

where c_{ij} is the distance between two points, and f_{ij} is the set of flows that minimises the cost of $\sum_{i \epsilon \imath} \sum_{j \epsilon \jmath} c_{ij} f_{ij}$ subject to the following conditions:

$$\begin{array}{cc} f_{ij} \geq 0 & i \in I, j \in J \\ \sum_{i \in I} f_{ij} = y_i & j \in J \\ \sum_{j \in J} f_{ij} \leq x_i & i \in I \end{array}$$

where I is the set of indicies into vector $\boldsymbol{x}$ and J is the set of indicies into vector $\boldsymbol{y}$.

- Mahalanobis Distance

The Mahalanobis distance (Bishop, 1997a) (Equation 2.15),

$$d(\boldsymbol{x}, \boldsymbol{y}) = (x - y)^T C^{-1} (x - y) \tag{2.15}$$

where C is the covariance matrix. The Hausdorff distance (Huttenlocher et al., 1993) (Equation 2.16) is unique in regard to the other metrics listed here in that it does not require $\boldsymbol{x}$ and $\boldsymbol{y}$ to have the same number of elements. It compares points sets rather than vector to vector.

- Hausdorff Distance

The Hausdorff distance is recommended by Vranić and Saupe (2001b) as the distance metric to use when comparing structures such as an octree.

$$d(\boldsymbol{x}, \boldsymbol{y}) = \max_{x \epsilon X} \left\{ \min_{y \epsilon Y} \{d(x, y)\} \right\} \tag{2.16}$$

The quadratic distance (Ankerst et al., 1999) (Equation 2.17);

$$d_A(\boldsymbol{x}, \boldsymbol{y}) = \sqrt{(x-y) \cdot A \cdot (x-y)^T} \tag{2.17}$$

where A is the similarity matrix. The components of A, are calculated by $a_{ij} = e^{-\sigma \cdot d(i,j)}$ where $d(i,j)$ is a distance function between the i^{th} and j^{th} component of $\boldsymbol{x}$ and $\boldsymbol{y}$. The quadratic distance is a generalised case of the Euclidean distance which attempts to take into account the similarity or correlation between histogram bins.

- Quadratic Distance

The quadratic distance allows small variations in the histograms. E.g. due to numerical precision, a particular value may end up in one bin, or in the one next to it. According to Ankerst et al. (1999) varying σ had little significant effect on performance.

The choice of distance metric to use greatly depends upon application. For general usage, the Minkowski norms will often suffice, for applications where speed is preferred over accuracy, the L_1 norm or histogram intersection can be used. For applications where the different components cannot be assumed to be independent, a metric such as the Mahalanobis distance may be preferable.

2.9 Evaluation Techniques

In order to assess the relative performance of retrieval algorithms, a range of evaluation techniques can be used. The work in Järvelin and Kekäläinen (2000); van Rijsbergen (1975); Shilane et al. (2004) provides a range of criteria which can be used to evaluate the quality of a descriptor for retrieval purposes.

The Precision and Recall graphs, E-Measure, Nearest-Neighbour, First and Second Tier, DCG, and the Distance and Tier Images are all provided as part of the Princeton Shape Benchmark tools.

- Precision and Recall

The precision-recall graph is a commonly used method of evaluating the quality of a descriptor. Precision is defined as the proportion of relevant results out of the results returned (Equation 2.18). Recall is defined as the proportion of relevant results returned out of all the possible relevant results (Equation 2.19). Typically, one would expect that as recall increases, precision decreases.

$$Precision = \frac{\#Relevant\,Items\,Returned}{\#All\,Items\,Returned} \tag{2.18}$$

$$Recall = \frac{\#Relevant\,Item\,Returned}{\#All\,Relevant\,Items} \tag{2.19}$$

The basic precision-recall graph (showing precision against recall as the size of the returned set increases) is sometimes considered inadequate and the work by Huijsmans and Sebe (2001) suggests adding the precision-recall curve for a random retrieval and to take into account generality (the size of the class compared to the size of the database).

- Fallout

Related to these is fallout. Fallout is what is leftover, it is the proportion of irrelevant results out of the results returned.

$$Fallout = 1 - Precision \tag{2.20}$$

- Mean Average Precision

Another performance metric gaining attention is Mean Average Precision (MAP). MAP is the average precision of all relevant items returned. In Equation 2.21 the precision for each returned document, r, is calculated. The function *Relevance* returns 1 if the document is relevant, 0 otherwise. This results in the sum precision of relevant documents from the top N returned documents over all possible relevant documents.

$$MAP = \frac{\sum_{r=1}^{N} Precision(r)Relevance(r)}{Number\,of\,all\,relevant\,documents} \tag{2.21}$$

- The E-Measure

The E-Measure (Shilane et al., 2004) is one of several such criteria that combines precision (P) and recall (R) into a single value (See Equation 2.23). However Järvelin and Kekäläinen (2000) quotes this as the F-Measure, and the E-Measure as Equation 2.22.

$$\text{Järvelin's E } = \frac{b^2PR + PR}{b^2P + R} \tag{2.22}$$

$$\text{Shilane's E} = \frac{2}{\frac{1}{P} + \frac{1}{R}} \tag{2.23}$$

- Borko and Vickery Methods

Where b is a constant term, often 1. Other measures include the Borko (van Rijsbergen, 1975) method (simply $B = P + R - 1$) and the Vickery measure (van Rijsbergen, 1975) (Equation 2.24).

$$V = 1 - \frac{1}{2\left(\frac{1}{P}\right) + 2\left(\frac{1}{R}\right) - 3} \tag{2.24}$$

- Nearest Neighbour

The nearest neighbour criterion is the percentage of objects for which the nearest object is of the same class.

- First and Second Tier

The first and second tier criteria are the percentage of the first K elements that are of the same class, where K, for the first tier, is the size of the class. The second tier uses K as twice the size of the class. More specifically for a class C, $K = |C| - 1$ for the first tier and $K = 2 * (|C| - 1)$ for the second tier where $|C|$ is the size of class C (-1 to ignore the query object). The second tier is also known as the Bull-Eye percentage (Zaharia and Prêteux, 2001a).

- Discounted Cumulative Gain

The Discounted Cumulative Gain (DCG) is a measure that weights correct results returned earlier higher than those returned later within a ranked list. It is defined recursively in Equation 2.25 where G is a vector and G_i corresponds to the i^{th} element in the ranked list of results and has a value of 1 if the result is of the query class, or 0 otherwise.

$$DCG_i = \begin{cases} G_i, & i = 1 \\ DCG_{i-1} + G_i / \lg_2 i, & otherwise \end{cases} \quad (2.25)$$

For example, G can be $\langle 1, 0, 1, 1, 1, 0 \rangle$ which results in $1 + \frac{0}{lg_2 2} + \frac{1}{lg_2 3} + \frac{1}{lg_2 4} + \frac{1}{lg_2 5} + \frac{0}{lg_2 6}$ =2.56. The DCG is then normalised into the range 0.0 to 1.0 by dividing the result by the value computed if G was a vector of ones. In the example above, this would yield 0.649.

- Distance and Tier Images

Two other visual techniques are available the distance image and the tier image (Shilane et al., 2004). In both cases, an image is generated showing a matrix that compares each object against every other object and groups object according to class. This makes it easy to see class and inter-class relations. The distance image shows the distance or similarity between objects. Black pixels mark very similar objects, white pixels mark very dissimilar objects with grey values representing intermediate distances. Ideally each class would be a black box on the diagonal, and white otherwise indicating that objects in the same class were very similar, and other objects were very dissimilar. Black boxes between other classes indicate similarities between those classes.

The tier image is perhaps more useful than the distance image and shows nearest neighbour, first tier and second tier scores. The images are much "clearer" as they show the best matches for each model and ignore the other matches. This image shows the nearest neighbour (black) and the first (red) and second tier (blue) results for each object in the data set. White pixels mean objects are very dissimilar. The image diagonal should be black indicating that each object is matched best by itself. If the diagonal is not fully coloured with nearest-neighbour matches then this indicates a possible problem with the algorithm. Ideally all the coloured pixels would be within the class boundaries along the diagonal.

2.10 3-D Data sets

The literature has made use of a wide range of 3-D model data sets (Zaharia and Prêteux, 2001a; Hilaga et al., 2001; Leifman et al., 2003; Zaharia and Prêteux, 2002; Vranić, 2003; Veltkamp, 2001). These have a varying number of models ranging from about 100 to over 6,500. The number of classified models in each is typically much smaller. For example, the MPEG-7 data set (Zaharia and Prêteux, 2002) has 1,300 models but only 227 of them are classified. The classification data is required to be able to perform a good evaluation that is repeatable. The Princeton Shape Benchmark (PSB) data set (Shilane et al., 2004) has 1,814 classified models, equally split into a training and test

group and contains a large range of classes (161). The Viewpoint data set (Funkhouser et al., 2003) is the next largest data set, however it is not publicly available. There are several 3-D object repositories available on the World Wide Web such as (3D Cafe, No Year) containing a wide range of models. However these are not designed or intended as a benchmark data set and typically lack classification details, or contain only general indications of what the model is. Many of the web repositories are commercial entities with some sample objects free for use.

The PSB classification was manually created by partitioning a larger data set down to atomic concepts such as human and airplane. Further partitioning was done on geometric aspects such as human_arms_out. Any classes with less than four objects were removed from the data set (Shilane et al., 2004).

2.11 3-D Search Engines

There are numerous 3-D search engines available on the web. Typically these are the test systems documented in the literature (E.g. Tzovaras and Daras, 2004; Ansary et al., No Year; Suzuki, No Year; Corney, No Year; Antini, No Year; Funkhouser et al., No Year; Vranić, No Year). There are few production systems available. Typically these engines contain a fixed data set and only offer searching based on items already in the data set. Many engines offer several data sets. Typically, these are a custom data set and often the Princeton data set is also available. Some of the engines (such as CCCC and Princeton) allow the user to upload a query object and the Princeton system has a 2-D and 3-D sketch interface. Some of these engines allow searching using a number of different algorithms, but some only allow a single algorithm.

Typically all these engines only allow 3-D content-based searching. Princeton for example also allow a free text search on keywords when combined with a 2-D or 3-D sketch, but not when uploading a custom object.

2.12 Summary

In this chapter we have looked at a number of different 3-D descriptors, distance metrics and performance metrics. The majority of the descriptors are shape based, although there is a colour based technique and some are topology based. All techniques have shown good performance in their individual experiments. However there is no clear *best technique* due to the small number of comparative experiments with a large range of descriptors. There is also no standard data set used between the different descriptor experiments, although the Princeton Shape Benchmark provides the first step in this direction and many more recent works make use of it.

In the next chapter we will cover a range of popular classification techniques.

Chapter 3

Classification Background

3.1 Introduction

Classification is the act of forming a distribution into groups or *classes* according to some common criteria. This chapter begins with some terms and definitions related to classification. A similar topic is called *regression* which returns a real valued output rather than a class label. This topic is not within the scope of this thesis.

Generalisation

Generalisation is the ability for a classifier to correctly classify examples that it has not seen before. Typically generalisation is evaluated by setting aside some of the data set used to train a classifier for use as a test set later on.

Inputs, Outputs and Targets

Every classifier takes a set of input patterns and produces one or more outputs. Inputs can be anything from the values in a descriptor to items of metadata. Typically, input data needs to be continuous (each input will usually need to be between 0.0 and 1.0). A technique to convert discrete data to continuous data called one-of-N encoding is described later. The outputs are what the classifier decides the inputs represent, i.e. the class or label that the inputs belong to. Targets are the correct classifications for the input data. During the supervised learning (see below) of a classifier, they are used to calculate the error of the classifier and the results are fed back into the learning algorithm.

Supervised and unsupervised learning

Supervised learning is the term used to describe the training of a classifier with target data (class labels) available for the training set. The aim is to find the correct mapping between input data and the target data. Unsupervised learning does not use target data, and the goals of learning are more likely to be finding clusters in data or modelling distributions as opposed to finding a mapping.

The curse of dimensionality

Increasing the number of inputs is one way to improve the accuracy of a classifier. However, as the dimensionality of the inputs grows, the size of feature space can grow exponentially. Additionally the amount of training data required to accurately build the classifier grows exponentially too. This is because as feature space becomes bigger, the data becomes sparse and so requires more data to fill it up again. This phenomenon has been termed the curse of dimensionality. Techniques to reduce the dimensionality of the input data are often used to alleviate this problem.

Under and over fitting

Under fitting is used to describe a classifier that is too simple to properly represent the data it is modelling. An example is a classifier that can model straight lines, but is trying to represent a curve (such as the sine function). Over fitting is used to describe a classifier that has managed to properly model the input data, but has failed to properly represent the real model. An example again with the sine function, an over fitting classifier (such as a polynomial of high degree) would be able to model the points it has seen, but new points can be greatly removed from the real sine function. Figure 3.1 shows an example of under and over fitting the sine function. Some sample data points are also shown. The straight line represents a classifier under fitting the data. The very curvy line represents a classifier over fitting the data. While it goes through all the data points it still does a bad job of approximating the real function.

3.2 Pre-processing techniques

3.2.1 Input Normalisation

Many classifier techniques expect input patterns to have each component in the range [0.0, 1.0]. Therefore it is often necessary to pre-process inputs. How this is achieved is dependent on the type of data. One method is to use a sample data set and calculate the minimum and maximum values for each input and use that to apply a scaling factor.

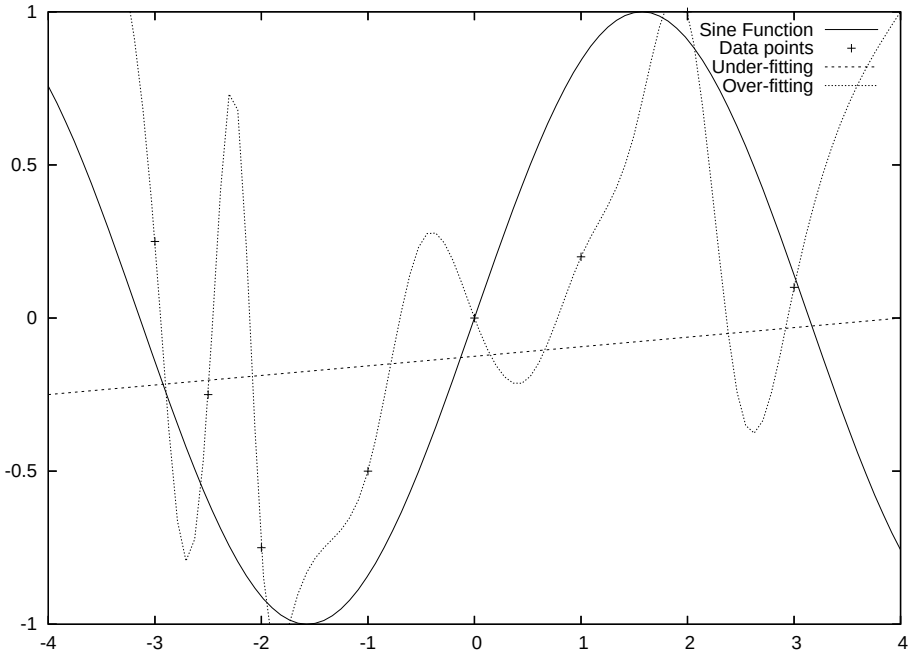


FIGURE 3.1: Under and Over fitting the Sine function

3.2.2 One-of-N Encoding

In some cases, inputs will be discrete values and not continuous. Some discrete variables such as age can easily be converted into a continuous variable. However, other variables do not have a standard numerical representation. The technique used in these situations is called One-of-N coding. We can create a separate input for each of the possible discrete values. We then assign a 1 to the input corresponding to the discrete value, and a 0 to the remainder. So if we wanted to represent the labels red, green and blue we would assign the inputs as (1,0,0), (0,1,0) and (0,0,1) respectively.

3.2.3 Missing data

A common problem in classification is an incomplete set of inputs. The simplest approach is just to ignore the missing values or set a default value. However, if these values are important to the classification process, then this is not always a satisfactory solution. The best solution is to base the missing values on the rest of the input data. A simple approach would be to copy the missing fields from the most similar data item, or to use the most frequently occurring value for that field.

3.2.4 Feature selection

Feature selection is a simple technique to reduce the number of inputs by discarding irrelevant or repeated data. The choice of the sub-set of the features can be done by

an exhaustive search of all possible sub-sets. This is guaranteed to find the optimal set, however there are usually some constraints so as to reduce computational complexity. For example 10 inputs gives 1024 possible subsets to search through, 100 inputs gives 10^{30} possible subsets which makes exhaustive searching impractical.

3.2.5 Dimensionality Reduction

Principal Components Analysis (PCA) can be used to reduce the number of inputs by combining them where possible to get smaller sets of input features. It does this by combining inputs that are similar. This technique only operates on the input data and does not use the target data so. This is a more powerful technique than feature selection as it combines inputs as opposed to discarding them. The main problem with this technique is that the data lost from the reduced dimensionality may be a critical factor in the classification process. This differs to the PCA used in 3-D indexing as here this technique is used to reduce the number of inputs, where as in 3-D indexing, it is used to transform an object into a common space.

3.2.6 Invariance

Sometimes it is desirable to build in invariance into a classifier system. A common example is translation invariance. There are several different techniques for doing this. The first way is to train the classifier by example. This has the disadvantage of requiring a much larger training set. The classifier will only have an approximate invariance, but it is relatively straight forward to implement. The second method involves pre-processing the input data to make it invariant before it even gets to the classifier directly. The third method is to build the invariance into the classifier. In the case of Multi-Layer Perceptron networks, careful design of the layers can apply invariance to the data as it propagates through the network.

3.3 Overview of standard classification techniques

The purpose of classification is to train a machine to be able to assign a correct label to a set of inputs. The complexity of this task increases as the number of inputs, or number of different classes, or even as the amount of data increases. There are several well known classification techniques which are described in the rest of this section. These techniques are documented in Bishop (1997b); Haykin (1999) unless otherwise noted.

3.3.1 *k* - Nearest Neighbour

The k-NN classifier is one of the more traditional classification schemes. It is very simple in nature and is well understood. The basic premise is to produce a ranked list of the nearest (or most similar) objects in the training set when compared to a query pattern in some metric space. This is identical to the process in CBR. The top k matches are then used to obtain a classification. Typically some sort of majority vote is used to determine the label assigned to the query object. This classifier requires no training, although the value of k needs to be determined somehow. If it is too small, the classifier becomes sensitive to noise and if it is too large the computational time increases and becomes biased towards the classes with the larger number of members. A commonly used version of the k-NN is the Nearest Neighbour (NN) classifier where k is equal to one. The disadvantage of this scheme is that the quality of results depends on the training set, and while larger sets may give better results, they also increase the computational cost. The k-NN algorithm is stable, i.e. small changes to the data set do not cause major changes in the classification results.

3.3.2 Multi-Layer Perceptron

The Multi-Layer Perceptron (MLP) is one of the more popular classification techniques. Jones (1990) showed that a MLP with just two layers using a sigmoid activation function can approximate any function to an arbitrary error. However, this does not mean that it is feasibly possible to do so. The main disadvantage with the MLP is that is suffers from the "curse of dimensionality" meaning that as the number of dimensions increases, the computational cost increases at an exponential rate quickly making more complex problems infeasible.

The MLP works by propagating an input pattern through a number of layers with varying numbers of nodes. Each node has a weight assigned to it and has some activation function assigned to it. The activation function of a MLP determines what sort of functions it can represent. If the activation function is linear, then the network is no more powerful than a single layer network (as a combination of linear transforms is another linear transform). Typically a sigmoid activation function is used which performs a non-linear mapping allowing much more powerful networks to be built. Typically MLPs are trained using the error back-propagation algorithm. This algorithm takes into account the individual weightings within the network and can choose the best change to make to the weights per iteration.

3.3.3 Radial Basis Function Networks

The Radial Basis Function (RBF) networks classifier is a technique that relies upon casting the classification problem into a much higher dimensional space than the input vector in order to increase the likelihood of creating a linearly separable problem. An RBF network consists of a number of input nodes, a hidden layer and an output layer. The hidden layer typically uses a Gaussian activation functions to perform a non-linear transform. This layer will also contain many more nodes than the input to cast into the higher dimensions. The output layer consists of a number of linear activation functions.

The RBF uses a randomly initialised set of weights meaning that each time it is trained, different results will occur (potentially better or worse). The training process should be able to reduce the effects of initial conditions if enough iterations are performed.

Training a RBF network is faster than training a MLP network. Training is split into two fast stages. The first stage uses an unsupervised method to determine the parameters of the basis functions. The final stage solves a linear problem, mapping the hidden layer to outputs.

3.3.4 Support Vector Machine

The Support Vector Machine (SVM) is a popular and powerful classification technique. It does not suffer from the curse of dimensionality that other classification techniques do and it is a kernel based technique. Due to the kernel nature of the support vector machine, different types of network can be built, such as polynomial learning machines, radial-basis function networks and two-layer perceptrons. An attribute particular to SVMs is that they can provide good generalisation performance even though they do not incorporate problem-domain knowledge.

The SVM is traditionally a binary classifier as this makes the maths much easier to solve. There has been no satisfactory method to produce a multi-class SVM, although there are several methods that work. Typically a number of binary SVMs are trained and the results are combined (Hsu and Lin, 2002). These methods significantly increase computation expense as the number of classes increases. The *one-versus-many* method trains one classifier for each class. Training examples are labelled with '1' if they are of the target class, or -'1' otherwise. The label associated with the classifier returning the largest positive distance from it's decision boundary is selected to make the classification. The *one-versus-one* method trains a classifier on every possible pairing of classes. The final classification is made by a voting process where each classifier can vote for one of it's two classes. The winning class is then used to make the classification. The third method, *DAG-SVM*, is similar to the one-versus-one method except a directed acyclic graph is constructed such that each classifier is a node in the tree and the leaves are the

resulting classes. This reduces the number of classifications required whilst keeping a similar level of performance.

3.3.5 *k*-Means Clustering

The k-Means is a simple clustering technique with similarities to the k-NN classifier technique. The k clusters are created at random positions in feature space. The k-Means training is an iterative process that updates the cluster positions. During each iteration, all training points are assigned to the nearest cluster. Then, each cluster's position is moved to the centre of all the objects that belong to it. The process is then repeated until the clusters become stationary, or the specified number of iterations is exceeded. The effectiveness of this algorithm depends on the choices of initial centre points, and the number of centre points, k. There has been a lot of work to effectively determine k and the position of the initial centre points, and some work to make k dynamic given an initial guess (clusters close together are combined, and clusters spread over a large area are split). A common choice for centre points is to randomly pick k examples and use them as the initial centre points.

3.3.6 Kohonen's Self Organising Map

Kohonen's Self Organising Map (SOM) transforms an input pattern of arbitrary dimension into a one- or two-dimensional discrete map and to perform this transformation in a topologically ordered fashion. The SOM algorithm is simple to implement, however it is very difficult to analyse mathematically. There are three essential processes called competition, co-operation, and synaptic adaptation. During competition, neurons in the network compute their respective clause of a discriminant function. This discriminant function provides the basis for competition among the neurons. The neuron with the largest value of discriminant function is declared the winner. During co-operation the winning neuron determines the spatial location of a topological neighbourhood of excited neurons, providing the basis form co-operation. During synaptic adaptation the excited neurons increase their individual values of the discriminant function in relation to the input pattern through suitable adjustments to their synaptic weights. Adjustments are made so that a similar pattern returns an enhanced discriminant value.

3.4 Performance Metrics

There are many ways of evaluating the performance of a classifier system. A commonly used statistic is *accuracy*. There are actually several versions of the accuracy statistic. The basic statistic just measures the percentage of correct classifications out of all the

classifications. We can also apply this to obtain an accuracy per class and per classification.

- Confusion Matrix

Typically statistics are calculated using a confusion matrix. This records the the true and predicted classification of each object. It is a NxN matrix where N is the number of classes. Sometimes a Nx(N+1) matrix is used when a classifier can reject a query pattern. A classifier can reject a query pattern when it is unable to produce a prediction with a high enough confidence value. The accuracy can be calculated as the sum of the diagonal over the total number of classifications made. For two class problems there are numerous statistics defined (see below). A multi-class confusion matrix can be converted into a two-class confusion matrix for a particular class by marking the required class as positive and all other classes negative.

The two class confusion matrix records four values. The True Positive (TP) value is the number of positive examples correctly classified. Likewise the True Negative (TN) value is the number of negative examples correctly classified. The False Negative (FN) value is the number of positive examples classified as negative and the False Positive (FP) value is the number of negative examples classified as positive.

The users accuracy (also known as precision; see Equation 3.1) is the number of correct classifications over all the objects classified as that class.

$$users\,accuracy, precision = \frac{TP}{TP + FP} \tag{3.1}$$

The producers accuracy (also known as recall and sensitivity; see Equation 3.2) is the number of correct classifications over all the objects of that class.

$$producers\,accuracy, recall, sensitivity = \frac{TP}{TP + FN} \tag{3.2}$$

Specificity (see Equation 3.3) measures the proportion of negative examples correctly classified. The higher the number of false positives, the lower the specificity.

$$specificity = \frac{TN}{FP + TN} \tag{3.3}$$

- Receiver Operating Characteristics Graphs

A Receiver Operating Characteristics (ROC) graph (Fawcett, 2006) is a visual tool to help evaluate classifier performance. A key feature is that it is invariant to class distribution,

however it is a two class tool rather than a multiple class tool. Multiple ROC graphs can be generated (one for each class), but this breaks the invariance to class distribution. The ROC graph plots true positive rate against false positive rate. In the ideal situation, a curve on the graph will start at 0,0, progress to 0,1 and finish at 1,1. The diagonal of the graph represents a random classifier. The area under the curve (AUC) can be calculated to allow a single value comparison between classifiers.

The above methods calculate the overall accuracy of a classifier, they do not gauge the accuracy of an individual classification. This is a harder task than calculating the overall accuracy of a classifier as it is dependent on the input pattern. Different classification techniques can give different outputs. Some techniques can output a single class label, where as others can output a ranked list. Some techniques can also output a numerical value that can be used to gauge the confidence of the classification (e.g. distance from decision boundary). If numerical guidance is available, then it is possible to map the value directly into a confidence value. However, for classifiers outputting only a label, alternative methods of estimating confidences are required.

- The a priori and a posterori methods

The work by Giacinto and Roli (1999) looks into several such metrics and highlights the *a priori* and the *a posterori* methods as good confidence estimators. These techniques make use of a validation set. If the k nearest objects in a validation set were correctly classified, then it is likely that the query object will also be correctly classified. The a priori method estimates the confidence without requiring the query to be classified. It simply bases the confidence on how many of the neighbouring objects were correctly classified. The a posterori method requires the query object to be classified first and then bases the confidence on how many of the neighbouring objects were correctly predicted that class.

Equation 3.4 shows the a priori confidence estimate for a given classifier. For each of the K objects, X_k, in the neighbourhood the probability of it being correctly classified, $P(\omega_i \mid X_k \in \omega_i)$ is calculated (where $i = 1, ..., M$, M being the number of classes and ω_i is the label for class i) and weight the result by W_k which is $1/d_k$ where d_k is the Euclidean distance between X_k and the query pattern. The sum of the correct predictions is then divided by the sum weighting of all K objects.

$$a\,priori\,confidence = \frac{\sum_{k=1}^{K} P(\omega_i \mid X_k \in \omega_i) \cdot W_k}{\sum_{k=1}^{K} W_k} \tag{3.4}$$

Equation 3.5 shows the a posterori confidence estimate for a given classifier predicting a label ω_i. For each of the K objects, X_k, in the neighbourhood the probability of it being correctly classified, $P(\omega_i \mid X_k)$ is calculated, with the label ω_i and weight the result by

W_k. The sum of the correct predictions is then divided by the sum weighting of all K objects that were predicted label ω_i.

$$a\,posteriori\,confidence = \frac{\sum_{X_k \in \omega_i} P(\omega_i \mid X_k) \cdot W_k}{\sum_{k=1}^{K} P(\omega_i \mid X_k) \cdot W_k} \tag{3.5}$$

For classifiers returning only a label, the probability of the classifier predicting the returned label is 1, and 0 for all other labels.

3.5 Classifier Training Schemes

Classifiers need to be "trained" to learn the features of the data sets they work with. It is also useful to know how well a classifier will work. Several different training schemes have been proposed to help get a good estimate of the generalisation ability of a classifier.

3.5.1 Split-Sample

A commonly used method for classifier training and validation is Split-Sample Validation (Weiss and Kulikowski, 1990). The data set is split into a training and validation set (often a 50%-50% or 75%-25% split). The classifier is trained using the training set, and validation is performed on the validation set. The greater the number of samples, the closer to the true error the estimate will be. This means that for small numbers of samples the estimate is likely to be inaccurate.

3.5.2 Cross-validation

There are several similar techniques that come under the cross-validation heading. Cross-validation has been used to select classifier training parameters (Haykin, 1999) and to just estimate the generalisation performance of a particular set of parameters. In this situation, training and testing data sets are produced. The training data set is further partitioned into an estimation and validation set. For each set of parameters, a classifier is trained using the estimation set and its performance evaluated using the validation set. The best performing set of parameters is then selected and it's performance is evaluated using the test data set to avoid problems of over-fitting the training data.

Alternatively cross-validation can be used to provide a better generalisation estimate than split-sample when small data sets are available. Unlike split-sample validation, k-folds cross-validation uses all the data in both the training and validation stages. The data set is split into k partitions and k classifiers are trained each using one of the k partitions as the validation set and the rest as the training set. The average error from

the k classifiers is then used as the error estimate. Typically a value of 10 is used for k. When k is equal to the number of samples, then the method is known as *leave-one-out* cross-validation. Cross-validation is more suited to smaller data sets where split-sample would be at a disadvantage. It is computationally expensive for larger data sets or high values of k.

3.5.3 Boot Strapping

Boot strapping is similar to cross-validation except that it uses sub-samples of the data set instead of sub-sets. A sub-sample is random sampling with replacement of the original data set allowing sub-samples to be of nearly any size as required. This is useful when data sets are unbalanced or too small.

3.6 Combining Classifiers

For some problems, a single classifier will never be able to achieve good results no matter how well it has been trained. However, the combination of several classifiers should be able to make up for the deficiencies in the base classifiers.

3.6.1 Classifier Ensembles

There are numerous techniques for combining classifiers. Some of the earlier work on combing classifiers was by Hansen and Salamon (1990). They created a combination of classifiers called an ensemble. This is a set of classifiers (called base classifiers) trained slightly differently from each other and then the results of all the classifiers are combined using a combination rule. Hansen and Salamon (1990) uses plurality and Majority Vote to combine the base classifier predictions. Further work by Kittler et al. (1996, 1998) defined a theoretical framework for classifier ensembles and derived several basic combination rules. These are the Product, Sum, Max, Min, Median and Majority Vote rules. Experimental work showed that the Sum rule gave the best performance. However, much of the experimentation by other authors has suggested that Majority Vote is generally the best rule for general purposes (see e.g. Duin and Tax (2000))

Research has shown that the base classifiers should make errors on different parts of feature space to give the best combination results (see e.g. Kuncheva et al. 2000). This is often achieved by altering training data, network parameters, network types and even the network architecture. Typically combining weak learners give better results as they are more likely to make different errors than those optimally trained. Schapire (1990) shows that classifiers need to perform slightly better than random guessing to lead to improved performance when combined. It was shown that if an infinite number

of classifiers were combined then they would give 100% accuracy. Similarly if a infinite number of classifier's whose accuracy was below random guessing, then accuracy would be 0%. Giacinto and Roli (2001) state however that combining higher performing, but less error diverse classifiers can still out-perform a collection of weaker but more diverse classifiers.

3.6.1.1 Combining Ranked Outputs

Typically the classifier will be combined based on a single label (e.g. Majority Vote rule) or on an output value (e.g. Sum rule). However it might be preferable to combine a ranked list of classifications from a classifier (Ho et al., 1994). It could be that the second or third classification in a list is the correct class instead of the first item. For example an object that lies on the decision boundary of two classes could go either way. In the case where one classifier in the group can only output a single class (or reduced list compared to others), then either all classifiers need to crop their ranked lists to one class (class set reduction) or first combine the larger ranked lists and then reduce (class set reordering). Both methods return a list of possible classes ranked in the order likelihood.

3.6.1.2 Estimating error diversity

The greater the error diversity of a set of classifiers, the greater the expected increase in accuracy gained by combining them. Here we describe three methods for estimating the error diversity of a pair of classifiers.

The first is the *within-set generalisation diversity* (GD) measure (Partridge and Yates, 1996) for a set of classifiers.

$$GD = 1 - \frac{p\,(2\,both\,fail)}{p\,(1\,fails)}$$

where;

$$p\,(2\,both\,fail) = \sum_{n=2}^{N} \frac{n}{N} \frac{n-1}{N-1} p_n$$

and;

$$p\,(1\,fails) = \sum_{n=1}^{N} \frac{n}{N} p_n$$

p_n is the probability that exactly n classifiers fail on a random test sample. It can be calculated as the average percentage of samples incorrectly classified by n classifiers. N is the total number of classifiers, $p\,(1\,fails)$ is the probability that one randomly selected classifier fails to classify a random test sample and $p\,(2\,both\,fail)$ is the probability that two randomly selected classifiers fail to correctly classify a test sample.

As an example, if there are two classifiers (A and B) and there are five test samples. Classifier A correctly classifies the first three samples and incorrectly classifies the final two. Classifier B correctly classifies the first and last samples, but incorrectly classifies the remaining three. This gives $p_1$of 0.25 and p_2 of 0.2. This results in a $p\,(2\,both\,fail)$ of 0.2, a $p\,(1\,fails)$ of 0.45 and a GD of 0.56.

The second used *Q statistics* to evaluate the diversity of two classifiers (Kuncheva et al., 2000).

$$Q_{i,k} = \frac{N^{11}N^{00} - N^{01}N^{10}}{N^{11}N^{00} + N^{01}N^{10}}$$

where N^{ab} is the number of elements z_j of Z for which $y_{j,i} = a$ and $y_{k,k} = b$. Z is the labelled data set and y_i is the output vector for classifier D_i such that if D_i correctly classified sample z_j correctly then $y_{j,i} = 1$ otherwise 0. Likewise where i is one classifier, k represents the second classifier. For statistically independent classifiers, $Q_{i,k} = 0$ otherwise it will vary between -1 for more errors on different objects and 1 for correctly classifying the same objects.

As an example, using the same two classifiers as before, $N^{11} = 5$, $N^{00} = 5$, $N^{01} = 4$ and $N^{10} = 6$. This results in a Q value of $\frac{1}{49}$.

The third method is called the *compound diversity* (CD) (Giacinto and Roli, 2001).

$$CD = 1 - prob\,(c_i\,fails,\, c_j\,fails)$$

By example using the same classifiers as before, $c_i\,fails$ is 0.4 and $c_j\,fails$ is 0.6, resulting in a CD of 1 - (0.4 * 0.6) = 0.76.

Experimentation by Roli et al. (2001) determined that none of the methods are particularly better than the others and uses a combination of them in their work.

3.6.1.3 Improving error diversity

Base classifiers to be combined should be different from each other in order to help improve error diversity. Duin (2002) lists six criteria, below, in descending order of importance. Duin also lists disadvantages to several of the "fixed" training rules (fixed

meaning that no training is performed to tweak the output). Duin also notes that for combining classifiers, the output needs to be normalised, e.g. for confidence measures.

Confidence estimates by a classifier may be inaccurate due to over training. Using confidence estimates in combining classifiers can be susceptible to incorrect confidences leading to bad classifiers dominating the result.

1. Different Initialisations: E.g. network rates
2. Different Parameter choices: e.g. number of neighbours
3. Different architectures: e.g. number of hidden nodes
4. Different classifiers: e.g. MLP or k-NN
5. Different training sets: e.g. sub samples of the same data set for each classifier
6. Different feature sets: e.g. Shape D2 and Cord Hist 1

A final note by Duin (2002) says that base classifiers should be properly trained and care should be taken so they are not over-trained. It is preferable that the base classifiers are weakly trained. The combining classifier can then be trained as normal. Duin suggests that the training set can be split into separate sets for the base classifiers and for the combining classifier to avoid the issue of over-training, but it is a less desirable approach.

3.6.1.4 Test and Select

The *test and select* methodology (Sharkey et al., 2000) aims to find the best combination of base classifiers for an ensemble. An alternative name to this approach is called *over-produce and choose* (Duin and Tax, 2000). Typically most ensemble approaches combine all the generated base classifiers. In the test and select method, the idea is that due to redundancy in the base classifiers, a smaller subset will be required and so combinations of the base classifiers for use in the ensemble are *tested* and the best one is *selected*. When the number of combinations are large, then it may not be feasible to test all combinations. Randomly selecting an acceptable number of combinations may be appropriate but does not guarantee the optimal combination. Exhaustive search will be adequate for a small number of base classifiers, but it will quickly become too computationally expensive as more base classifiers are used. This method does not require calculation of the diversity of a collection of classifiers, rather it works experimentally.

Roli et al. (2001) proposes a number of alternatives to exhaustive search that do not guarantee finding the optimal combination, but should find a near-optimal solution in an acceptable time. These are forward search, backward search and Tabu search. Forward search starts by calculating the performance of all base classifiers. The highest

is then combined with every other classifier in a two classifier ensemble. The highest performing pair is then taken forward into finding the highest performing triple, and so on. The search terminates when performance starts to decrease. That is, the accuracy of k classifiers is greater than the accuracy of k+1 classifiers. Backwards search is similar to forward search except that it begins with an ensemble of all classifiers and starts by removing one classifier from the ensemble. The Tabu search is a combination of both forward and backwards search. When k classifiers have been evaluated, ensembles of k+1 and k-1 classifiers are evaluated. Cyclic searching is not allowed (Classifiers created in the previous steps are not used in the following step). Instead of terminating when performance decreases, the process stops after a certain number of iterations. This is to avoid local minima conditions.

3.6.1.5 Boot strapping

Boot strapping is a technique used to increase the size of a data set by duplicating existing members zero or more times. Section 3.5.3 mentions boot strapping in the context of estimating generalisation, where as here it is mentioned in the context of improving performance.

3.6.1.6 Bagging

Breiman (1996) developed a technique called bootstrap aggregating or more commonly known as *bagging*. This technique creates multiple data sets drawn from an initial data set. Samples are drawn at random with replacement and a particular sample can appear multiple times or not at all in the new data set. The classification technique is trained on each of the data sets and the results are combined. If the classifiers offer a numerical output, then the result is averaged. If the output is a class label, then a voting process is used to determine the final value. Bagging is more useful for unstable classifiers and has been shown to increase the accuracy of a given unstable classification technique.

3.6.1.7 Boosting

Boosting is another technique for combining several versions of a classifier based on a given training set. It was pioneered by Schapire (1990) and then improved upon and the current technique is called ADABoost (Freund and Schapire, 1996). In ADABoost each item in the data set is assigned a weighting which represents the probability of it being selected to become part of the training set for the classifier. The training data set is created using sampling with replacement. Initially all items have equal weighting. Then a classifier it trained and tested on a sampled data set. Test items that were incorrectly classified have their weightings increased and then the next iteration begins. This results

in a set of classifiers each trained on a different sub set of the data. The whole set of classifiers is used when making a classification, with a weight assigned to the label given by each classifier such that classifiers with a lower error are awarded a higher weighting.

3.6.1.8 Dynamic Classifier Selection

Dynamic Classifier Selection (DCS) chooses the most appropriate classifier at run-time to give the highest confidence estimate for that object. A brief description of this technique is that the classifier with the highest confidence for the query object is selected to classify the query object.

Giacinto and Roli (1999) proposed a framework for classifier selection in which numerous methods can be used to determine the confidence of the base classifiers. The a priori method estimates the confidence as the number of correct classifications over all classifications in the locality. The a posterori method estimates the confidence as the number of correct classification for label w over all classifications for label w when the query is predicted to have label w. Both these methods weight the confidence by the distances of the neighbours in the locality. The locality is defined as the k nearest neighbours from a validation set. See Section 3.4 for a description of the a priori and a posterori methods.

The framework selects an appropriate classifier from a pool of possible classifiers. When presented with a query pattern, a confidence value is generated for each classifier. Those classifiers with confidence less than a reject threshold (e.g. 50%) are removed from the pool. In the next step, the difference between each classifier confidence to that of the classifier with the highest confidence is calculated. A threshold value is then used (0.1 is given as an example). If the distance between each classifier and the best classifier is greater than the threshold, the best classifier is used to classify the object. However, if some classifiers have a distance smaller than the threshold, then one of these (including the best classifier) is randomly picked as the one to classify the object.

The DCS framework algorithm has the following stages;

1. Compute confidence for each classifier, C_j for $j = 1, ..., K$
2. For each classifier, C_j , if $C_j <$ *reject threshold* then remove C_j from classifier pool
3. Set C_m as the classifier with the maximum confidence
4. For each classifier, C_j, compute the difference, d_j, in confidence between C_m and C_j
5. If all differences, d_j are greater than *selection threshold* then select classifier C_m else randomly select a classifier from those with d_j less than *selection threshold*.

The experimental work by Giacinto showed that DCS can outperform ensembles, but not always.

3.6.2 Mixture of Experts

The original Mixture of Experts (MoE) algorithm splits up a feature space into regions with a single expert assigned to each region. A gating network is then used to choose a mixture of experts to calculate the final classification. The MoE algorithm has since been improved upon and Jordan and Jacobs (1994) describe the Hierarchical Mixture of Experts (HME) algorithm. In this technique each expert is trained on all the data and the gating network is trained to work out which experts are good for which input patterns. The Hierarchical aspect comes from the ability to have several layers of gating networks. The HME is trained using the Expectation-Maximisation technique to simultaneously assign the weights to each expert and to the gating networks.

Tang et al. (2002) used a different approach where a SOM was used to partition the input space. A secondary clustering of the SOM nodes combines nodes that are within the same region.

3.6.3 Other Techniques

There are several other techniques such as Dempster-Shafer theory of evidence, Bayesian methods and Behaviour Knowledge spaces (see Impedovo and Salzo, 2000), however space is always a limiting factor and they will not be covered in this thesis.

3.7 Optimisation Techniques

Each classification scheme has a number of parameters which can be used to adjust the performance of a classifier. Typically these need to be set when specifying a classifier and it is difficult to determine the optimal values without prior knowledge as they are data set and requirements dependant. In some cases prior knowledge can be used to estimate "good" parameter values, however more typically there will be little prior knowledge.

3.7.1 Exhaustive Search

The most basic and reliable way to find the best parameters is to search through every single value and combination and select the best one. However, this is a very computationally expensive method and quickly becomes infeasible when working with continuous variables. It is however an option for limited, discrete variables, especially variables that

are labels rather than numeric. E.g. the distance metric type used is a discrete variable and an exhaustive search through each one is possible.

3.7.2 Cross-Validation

As described in Section 3.5.2 cross validation can be used to select training parameters.

3.7.3 Genetic Algorithms

A commonly used technique is Genetic Algorithms (GA) (Beasley et al., 1993). Originally developed in the 1960's GA's have typically been used in optimisation and machine learning problems. Genetic Algorithms have their roots in evolutionary principles where "offspring" are formed by combining the chromosomes or genes of "parents". An iterative process evolves a population to an optimal result. Each gene or chromosome encodes a set of parameters that form a solution to the problem in hand as a binary string. GAs have two basic operators. The first operator is the crossover operator. This takes two parent genes and chooses a split point in which the first part of one gene is combined with the second part of the other gene to make a new, offspring gene. Sometimes a probability is used to determine whether or not to crossover the parents. The other operator is the mutate operation. This causes random bits in the binary string to be flipped. Typically a low probability is used (e.g. $< 0.1\%$) so only small changes are present. This operator ensures that values not currently in the population have a chance to be evaluated.

Typically in each iteration, the genes are ranked in order of fitness (e.g. by performance). Then a certain portion of the population (e.g. the lower 50%) are replaced by the offspring from crossover using the remaining population. The mutate operator is applied to the offspring.

The population should eventually converge if enough iterations are performed. Gene convergence is typically when 95% of the population have the same value. When all genes have converged the population is said to have converged. Of course different thresholds may be more suitable in different situations.

3.7.4 Particle Swarm Optimisation

Particle Swarm Optimisation (PSO) Kennedy and Eberhart (1995, 1997) use a swarm of particles which represent points within parameter space. The algorithm is based on the simulation of birds flocking and such behaviour observed in the natural world where members of the group are actively seeking "good" areas and members will "gravitate" towards other members especially if they look like they have found a good area.

The PSO algorithm adjusts the trajectories of the particles within this space based on the particle's previous best performance and the previous performance of the "best" particle.

Each particle records its best position, and each particle has access to the global best position. During each iteration, the current performance of each particle's parameters is recorded and the best position is updated if applicable. Each particle then updates its position based on how far away it is from both its personal best, and the global best, with the aim of moving closer to both of these positions. A random factor is introduced to avoid particles directly homing in on the centre point between the global and personal best. The process is described by the following equations:

$$v_{id} = v_{id} + \phi\left(p_{id} - x_{id}\right) + \phi\left(p_{gd} - x_{id}\right)$$

$$x_{id} = x_{id} + v_{id}$$

where v_{id} is the velocity of a particle id and x_{id} is the current co-ordinates of particle id. p_{id} is the co-ordinate where particle id showed the best performance and p_{gd} is the co-ordinate of the best performance found so far. ϕ is a random positive number generated for each particle. Each iteration a new velocity is calculated and the particle position is modified accordingly. Alternatively the "global best" can be the neighbourhood of the particle, typically particle $id-1$, id and $id+1$.

In this form, floating point numbers (i.e. continuous data) are assumed, however Kennedy and Eberhart (1997) proposes a modification to the algorithm to work with binary data that allows discrete data to be used in the algorithm.

This technique however does not work with labelled variables such as different distance metrics. This is because the technique searches for peaks or troughs in parameter space (depending on whether we are looking for maxima or minima respectively) and there is no such relationship between labels.

3.7.5 Simulated Annealing

Simulated Annealing (SA) is a method based on Monte Carlo simulation and it was first used by Kirkpatrick et al. (1983) for optimisation. SA is analogous to physical annealing where a substance is melted and has its temperature lowered slowly until it reaches freezing point. In SA, a possible solution is called an atom. At each iteration, the atom is displaced by a small randomly determined amount. The energy, E, is computed for the new position or state and the change in energy, $\triangle E$, between the old state and new state is calculated. If the change in energy is less than or equal to zero, the new state

is accepted. Otherwise a randomly generated number is compared against the following probability to determine whether or not to accept the new state;

$$p\left(\triangle E\right) = \exp\left(- \triangle E / k_B T\right)$$

where k_B is the Boltzmann constant and T is the current temperature. T is decreased at each iteration until it reaches zero. As T decreases, so does the chance of accepting a worse solution over a better one. However this probability allows a solution to move out of a local minima position. The energy E can be calculated as the error for the solution. Multiple solutions can be considered at once, however they are independent of each other.

3.8 Summary

In this chapter we have covered a range of popular classification techniques and various techniques that can be used to improve the performance over an individual classifier either by combining several base classifiers, or by automatically finding the best training parameters for a classifier. The different techniques are good for different distributions of data sets although no one technique claims to be better than others in all situations.

The next chapter introduces the SCULPTEUR project and sets the underlying context for the work performed within this thesis.

Chapter 4

The SCULPTEUR Project and the Semantic Web

4.1 SCULPTEUR Introduction

The SCULPTEUR project (Addis et al., 2005b) was a three year European project with partners from both cultural heritage and technical backgrounds and took forward technology developed during the ARTISTE project (Lewis et al., 2003). The SCULPTEUR project aimed to develop a system to store, search and retrieve multimedia content and associated metadata that formed a museum or gallery's digital collection. It aimed to add support for 3-D multimedia objects to existing support for 2-D images. It also aimed to integrate up and coming Semantic Web technologies to provide enhanced search capabilities for the metadata. The ability to allow external systems to inter-operate with the SCULPTEUR system using existing standards where appropriate and to provide e-Learning capabilities were other aims. One of the more ambitious goals was to develop automatic techniques to add metadata to the system by creating classifiers trained on existing data (the Classifier Agent).

In this chapter, the development of the architecture within SCULPTEUR to facilitate content-based retrieval is described together with the development of the Classifier Agent. The SCULPTEUR system architecture is first described to show how each component fits into the overall system. This is followed by a section on the development of the content-based retrieval facilities and then a section on the innovation of a classifier agent. This chapter focuses on the experimental work and architecture design as used in the SCULPTEUR system. Chapter 5 and Chapter 6 give a much more in-depth study of these areas.

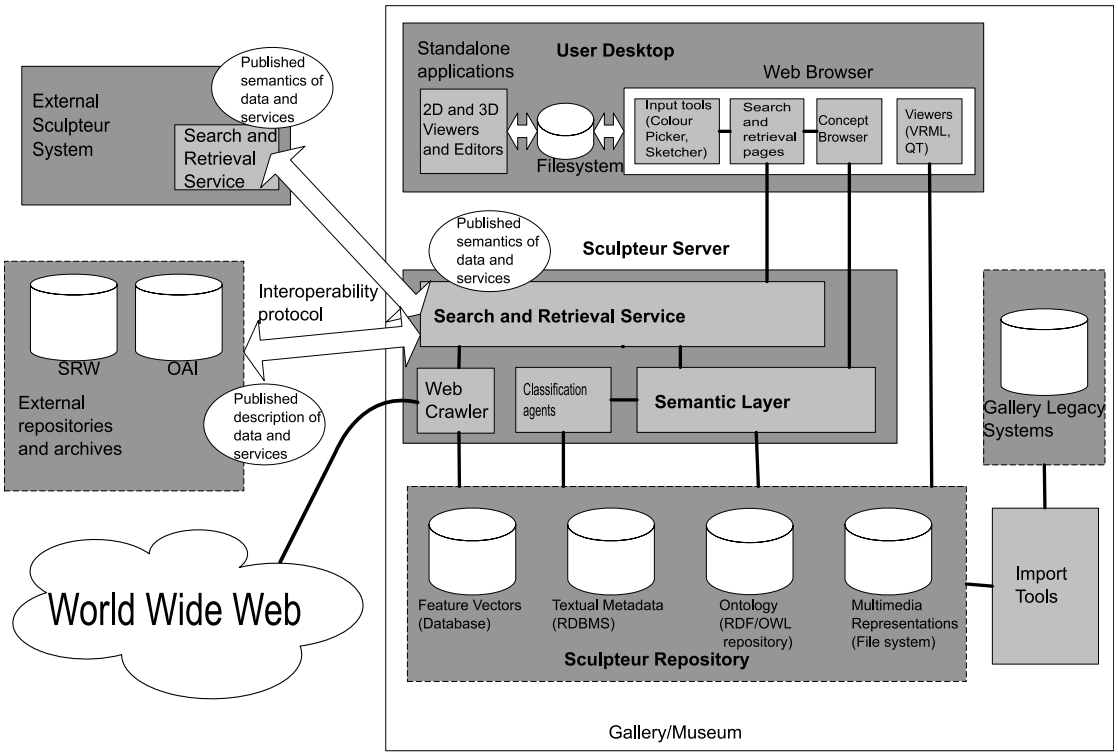


FIGURE 4.1: Architecture Diagram

4.2 SCULPTEUR System Overview

Figure 4.1 shows the SCULPTEUR architecture diagram. The core of the system was built around a Web Service implementing a Search and Retrieve interface based on the Z39.50 specification (SRW Editorial Board, 2004) called the SRW (Search and Retrieve Web service). A web application (called the WebApp) is built on top of the SRW and provides the primary user interface to the system. By using the WebApp users can search a museum's digital collection using a combination of keywords, concepts from the ontology and using content-based retrieval techniques. A concept browser allows users to browse or search through the ontology, with specific views developed for each gallery based on their requirements. The concept browser allows specifying parts of the metadata query through selecting specific concepts in the ontology. Figure 4.2 shows a view from the concept browser. The left-hand side of this figure shows "shortcuts" to key concepts within the ontology. The right-hand side shows the selected concept ("Object") and the relations to other concepts within the ontology.

Underlying the web service is a database storing all the metadata in the museum or galleries native database format. A mapping has been developed between the native database schema and the CIDOC Conceptual Reference Model (CRM) (Crofts et al., 2001) for each gallery. The CIDOC CRM is an ontology of cultural heritage information moving towards becoming an official ISO standard. It represents the concepts and relations covering, at a very high level, a large range of cultural heritage areas. The

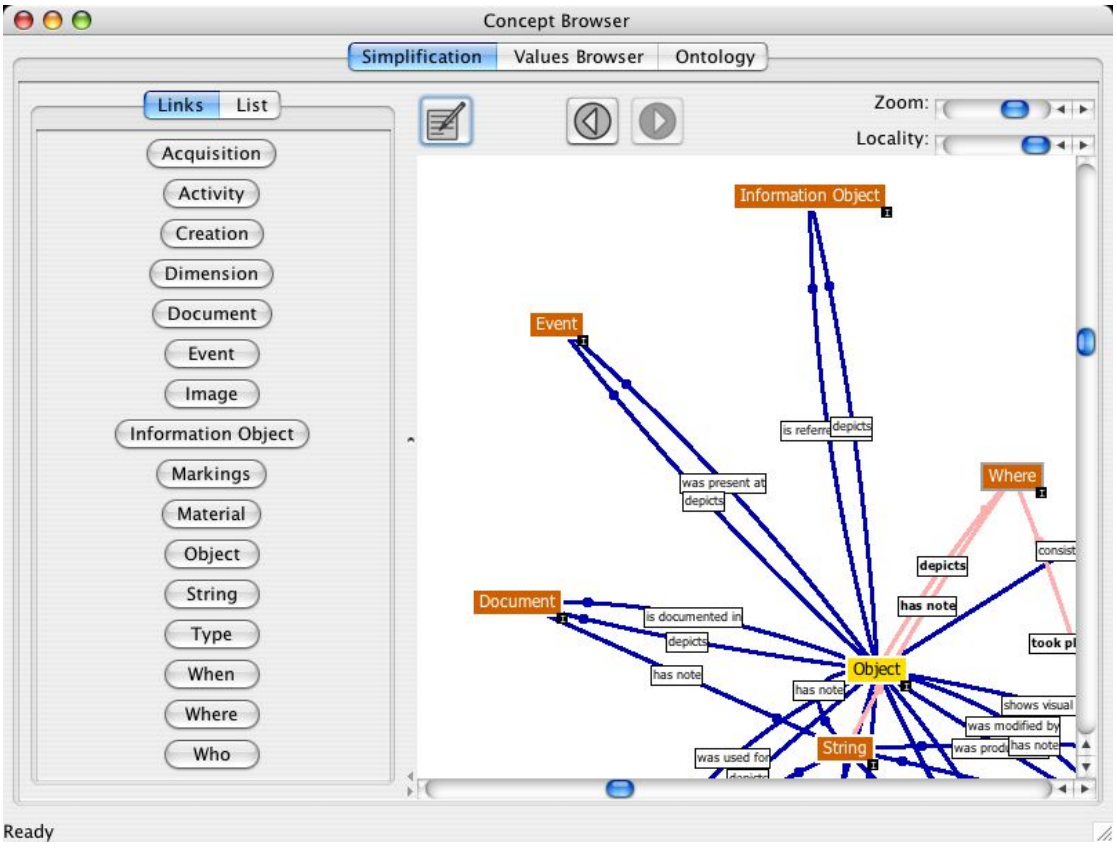


FIGURE 4.2: Example view from the Concept Browser

CRM does not cover all aspects of the users database schemas requiring gallery specific extensions to the CRM.

Content-Based Retrieval (CBR) in SCULPTEUR is implemented in the library FVS (Feature Vector Service; described later in this chapter) and is accessed through a MySQL UDF (User Defined Interface) interface to the library. Feature Vectors are stored as blobs (chunks of binary data) in the database allowing fast access by the MySQL module.

The WebApp, like any other interface to the SCULPTEUR system, uses the SRW to process its queries. Some functionality that can not be processed by CQL is facilitated by additional servlet functionality. Internally, the SRW converts the CQL query into the correct set of SQL statements for a given gallery to work on their database schema. This process also generates the relevant CBR SQL statements to be applied after the metadata query has taken place. On the assumption that a CBR query will always take longer than a metadata query, the metadata query is performed first and the reduced data set is then passed to the CBR query. This does of course make the assumption that all relevant objects have the correct metadata associated with them. This is really re-ranking the metadata query results using CBR similarity distances. However, this may not be appropriate in all situations.

FIGURE 4.3: Query Interface

Figure 4.3 shows the query interface in the WebApp. The query interface shows a large number of metadata fields key to a particular gallery's collection. A user may enter each field manually. Alternatively, by clicking on the magnifying icon next to a particular field, a list of all possible values is displayed, or if the field is already partially complete, a list of possible values beginning with the existing data is displayed. At the bottom of this form, the content-based retrieval part of the query can be formulated and will be describe in more detail below.

Figure 4.4 shows the interface for specifying a CBR query. When an image or 3D object query is selected, the user is prompted to upload a 2-D query image or 3-D object. Once uploaded a list of available CBR descriptors is displayed (3-D descriptors for 3-D objects, and 2-D descriptors for 2-D images). A preview is also displayed showing the user's query. If the query is a 2-D image, the user has the option of selecting a sub-image

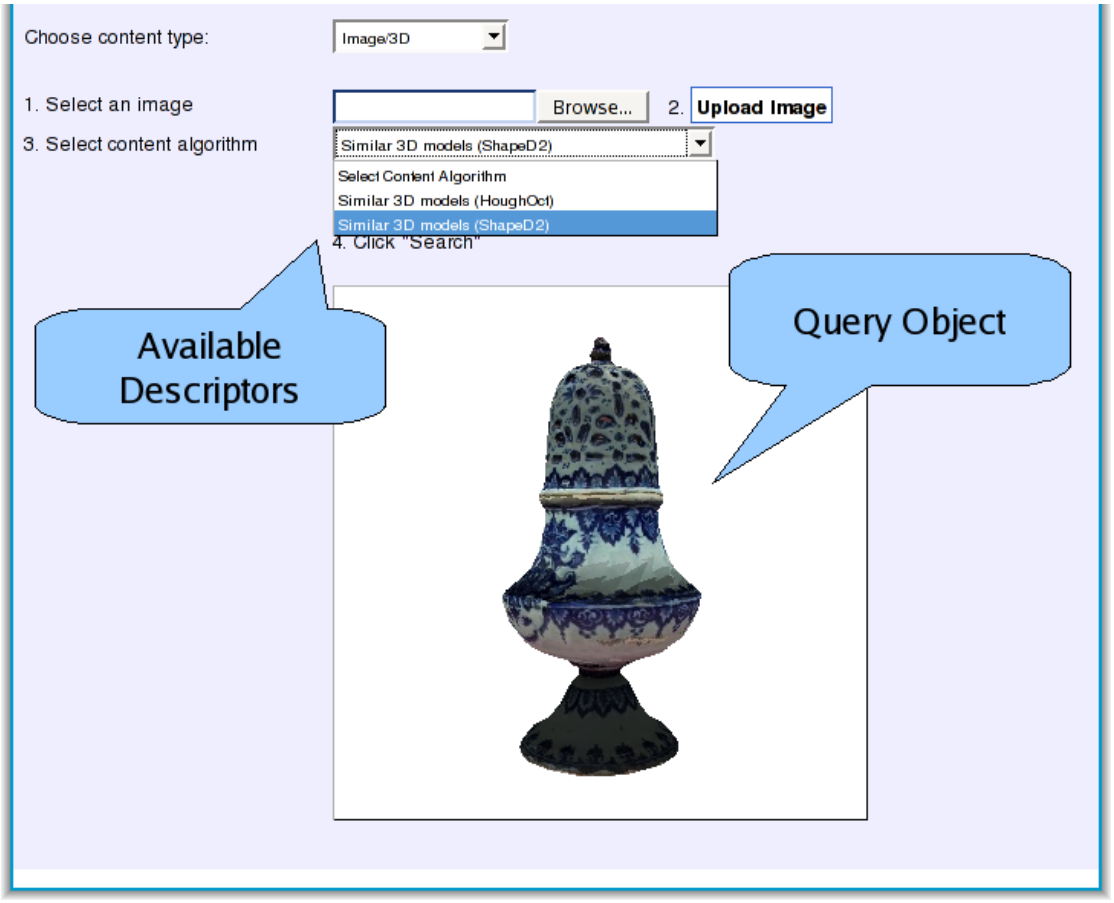


FIGURE 4.4: CBR with a query object uploaded

as the query. If it is a 3-D object, then the user's 3-D viewer is used if installed. Unlike for 2-D, no manipulation of the 3-D object for the query is possible.

Alternatively the colour picker tool allows a user to manually create a colour histogram (see Figure 4.5) to find similar 2-D images. The user can adjust the colours and the weightings for each colour in the histogram by simple controls in a Java applet.

Figure 4.6 shows the results of using the colour picker to choose a red colour and using the search term "chair". As can be seen, the combination of keyword and colour produces a page full of red chairs. The notable oddity is the orange coloured chair in third place. However the colour picker is finding images that contains some amount of the red component specified and does not look specifically for the amount of the specified colour.

The Classifier Agent runs as a separate entity communicating with a SCULPTEUR server to obtain objects and metadata for use in training data sets. It uses the feature vectors present in the system as inputs to classifiers.

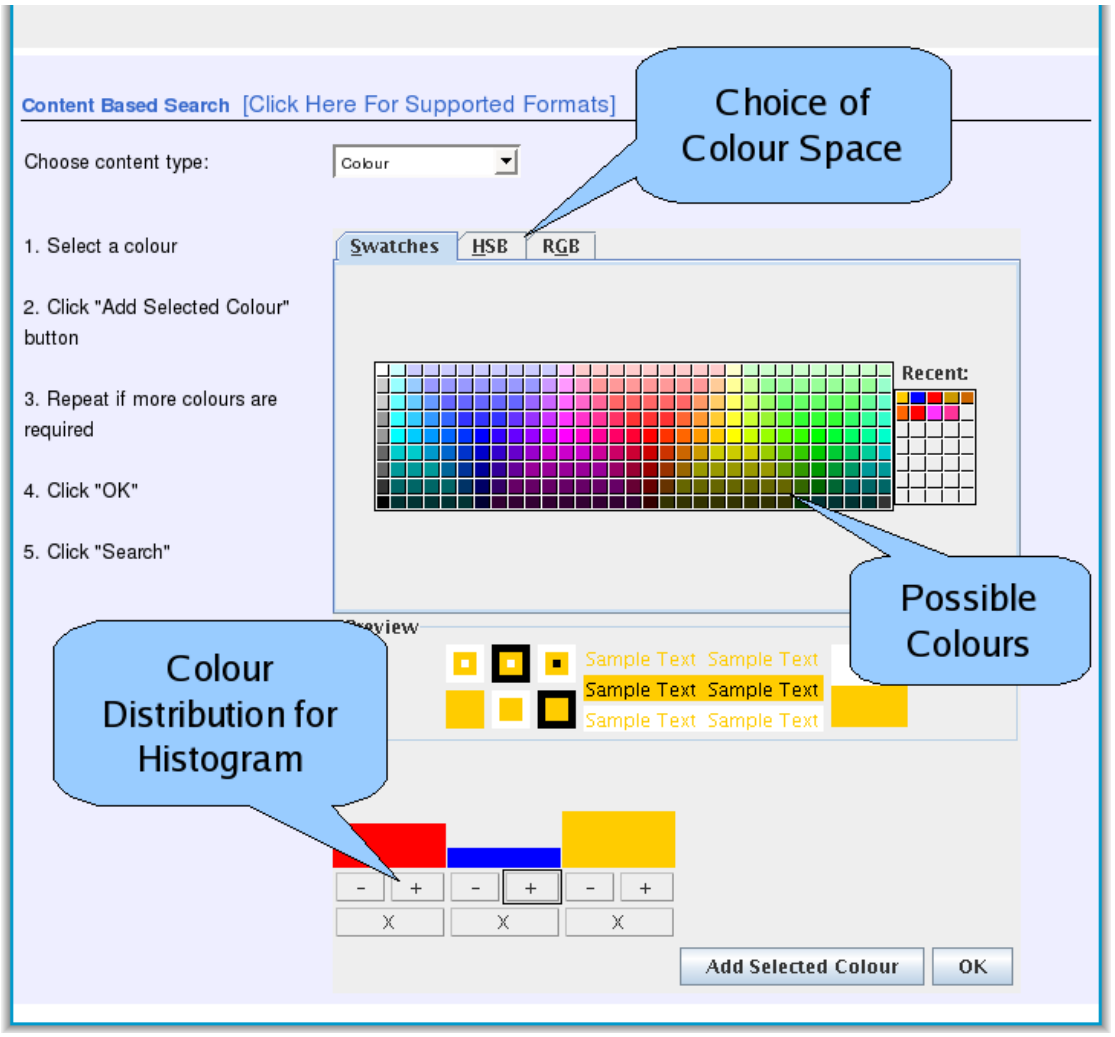


FIGURE 4.5: Colour Picker

4.3 Content-Based Retrieval

Content-Based Retrieval (CBR) in SCULPTEUR is just one component of the search and retrieve interface along with metadata and concept based search. Individually, each search method can produce reasonable results. Best results are obtained, however, through a combination of the different search methods. CBR in SCULPTEUR needs to be fast (potentially many thousands of images and objects to query), easy to use (user's will want it to "just work"), extensible (need to be able to add new CBR techniques easily), stable (the system will be deployed in a working environment) and reuse existing techniques developed from the previous ARTISTE project (Lewis et al., 2003). It also needs to be portable and work on at least Linux and Windows based platforms due to partner requirements. Minimising the number of dependencies (external software libraries) is also advantageous due to the wide range of target machines and platforms.

The result in SCULPTEUR is the development of the FVS library that is based loosely on the FVG (Feature Vector Generator) tool from ARTISTE and provides a MySQL

FIGURE 4.6: First page of results for query term "chair" and using the colour picker to select the colour red.

UDF interface for generating and comparing feature vectors. This section describes more details of the issues involved in integrating CBR into the SCULPTEUR system. Chapter 5 gives an in-depth evaluation of the 3-D algorithms.

FVS makes heavy use of classes and inheritance to simplify the addition of new descriptors and make the interfaces to the library much simpler and cleaner to use. Each descriptor is composed of two classes. The first is a FeatureAPI class which implements the generation and comparison functions for that algorithm. The second class is a FeatureVector class which stores the feature vector data and handles I/O (Data file reading and writing). These classes are sub-classed for each descriptor with commonly used functionality in the super-classes or in utility classes.

The FeatureAPI classes have an intermediate level of inheritance between the super FeatureAPI class and the descriptor's implementation that specifies whether the descriptor

is a 2-D descriptor or a 3-D descriptor as the type of input media is different for each type. Should another type of media be added to FVS (for example video), a new intermediate FeatureAPI would need to be defined. In each case, a new generate function is defined to accept video data sources, whereas the compare function is defined in the top level FeatureAPI class, remaining the same for every descriptor.

Feature vector I/O is also encapsulated in a class hierarchy. A top level class, called MemoryBlobWriter, defines the interface for reading and writing primitive data types. Two sub-classes exist; the first reads and writes endian independent binary data and the second reads and writes ASCII data (used for analysis/debugging). FeatureVector objects have a read and write function that take a MemoryBlobWriter. This allows multiple features to be saved in one file which is important for facilitating the multi-scale algorithms.

2-D images are loaded using the VIPS library which can potentially handle most image formats. The advantage of using VIPS over other image libraries are that it is designed to handle very large images efficiently. For 3-D, there is no such "wrapper" library and each 3-D model format needs to be implemented separately. FVS supports VRML through a modified version of the CyberX3D library (Konno, 2003). This library was chosen as it did not use C++ exceptions which is a requirement for using FVS in MySQL. It also supports a file format used by one of the project partners named the .TRI format (format unpublished). FVS also supports the .OFF file format (Object File Format) used in the Princeton Shape Benchmark (Shilane et al., 2004).

3-D objects are much more complex than the 2-D array of pixels an image is composed of. A 3-D model is typically composed of a set of connected polygons (faces) called a mesh. Each face is composed of a number of vertices which may or may not be shared with other faces. Each face can have a surface normal, as can each vertex (usually the average of the normals from the faces it belongs to). Each vertex can have one or more texture co-ordinates indicating how one or more texture maps (typically a 2-D image contained in a separate file) are mapped onto the model. FVS uses a custom data structure to store all this data for processing by the 3-D descriptors.

In SCULPTEUR metadata searching eventually resolved into SQL statements used in the MySQL database. To improve the speed of a CBR-based query, it was decided to develop a MySQL interface to the CBR techniques for direct incorporation into the SQL statements. While there have been no comparative studies on the retrieval speed of other architectures, discussions comparing SCULPTEUR to ARTISTE and SCULPTEUR to eChase (Sinclair et al., 2005a) revealed that the SCULPTEUR method is the faster of the three. ARTISTE called the FVG command line tool, while eChase separated the CBR to a separate system to facilitate more advanced CBR techniques. However, the additional overhead involved in both of these methods out-weighs their advantages. More

recently, eChase has moved to a hybrid approach, trying to allow near direct database access whilst keeping the CBR engine separate from the SRW.

A further requirement was the production of a thumbnail generator for automatically producing 2-D thumbnail images from the original 2-D and 3-D content. The user interface in the WebApp stayed fairly similar to that used in the ARTISTE system; a drop-down box with "user friendly" names for the descriptors. There are also some custom Java applets providing advanced functionality for cropping query images and using a colour picker applet to build a colour histogram for colour based queries.

4.3.1 Reuse of existing technology

In the ARTISTE project (Lewis et al., 2003), CBR was implemented in a tool called FVG (Feature Vector Generator). This was a command line program implementing the 2-D based algorithms. Each algorithm was written as a stand-alone component by different authors. This tool was run from the command line against images and feature vectors stored on the file system.

FVS is based upon the FVG tool, taking the 2-D descriptor and feature vector I/O code and re-writing it to fit into the FVS architecture. Large amounts of duplicated code was moved into super-classes or into utility classes. The FVS library took the FVG tool and developed it further for the SCULPTEUR project. It was re-written to allow the easy addition of 2-D and 3-D descriptors by leveraging C++ features such as classes, inheritance and templates. A MySQL UDF front-end was written as an alternative to the command line front-end to facilitate faster retrieval.

In FVG, each algorithm was implemented as a self contained unit. This however meant there was a lot of duplicated code with minor variations between them. Using sub classes allows nearly all of the higher level parts of the algorithm to be shared. (e.g. image loading, FV I/O and commonly used routines for image manipulation).

4.3.2 The Algorithms

FVS provides support for both 2-D and 3-D content-based retrieval. For 2-D CBR (see Lewis et al., 2003), RGB, L*a*b* and monochrome histogram matching is implemented along with the CCV (Colour Coherence Vector), PWT (Pyramidal Wavelet Transform) and QBF (Query-by-Fax) algorithms (Fauzi and Lewis, 2002). There is also a colour picker (allowing the user to manually specify the colour histogram for matching).

A multi-scale interface allows sub-image matching to be applied to any of the 2-D algorithms. The image is decomposed into a pyramid structure consisting of 64 by 64 pixel tiles, at different resolutions (the image is scaled down by a factor of two at each level).

At the top level, there is a single tile representing the whole image, and at the lowest level there are many sub-images tiled across the whole image. The sub-matching comparison finds the tile that gives the best match to the query and can return the position in the image in addition to the similarity distance.

For 3-D CBR, there is an Area-Volume ratio descriptor (Tung and Schmitt, 2004), the Cord Histograms (Paquet and Rioux, 1999b), Shape Distributions D2 (Osada et al., 2001), Modified Shape D2 (Ohbuchi et al., 2003a), augmented Multi-resolution Reeb Graph (Tung and Schmitt, 2004), Extended Gaussian Image (Horn, 1984) and 3-D Hough Transform (Zaharia and Prêteux, 2001b) descriptors. These are described in more detail in Chapter 2 and are evaluated in Chapter 5.

4.3.3 Ease of Use

Making CBR easy to use is a trade off between choosing the best "overall" settings and letting the user choose for themselves the best settings for their current task. Each CBR technique has different parameters that can effect how it generates a feature vector or how it compares a pair of feature vectors. Presenting all these options will confuse most users and experimenting with generation parameters can be computationally expensive if there is a large reference data set. However choosing a set of parameters to work well in all situations is very difficult. Those that work best overall, may not be suitable in all situations.

In FVS, some good default parameters were chosen and the ability to optionally specify custom ones was added to the MySQL and command line interfaces. However at the user interface level, only default parameters are used. Users wishing to be able to customise their requirements are able to do so by communicating directly with the FVS module or tool.

4.3.4 MySQL Module

The MySQL UDF module is the primary means for the SCULPTEUR system to use CBR techniques. It exports two functions, a generate and a compare function (to generate a feature vector from an image or object and to compare a pair of feature vectors respectively). Feature vectors are created using the generate function and are stored as blobs in the database. The compare function takes two such blobs and returns a similarity distance between them. Creating the MySQL module presented several challenges specific to using MySQL. The biggest issues were debugging problems with the module and handling the language interactions between the C based database application and the C++ based module.

Initial distribution of the module consisted of a collection of dynamically linked modules, however it soon became apparent that the differences between Linux distributions meant that a statically linked module containing all the required dependencies was required.

MySQL modules are required to be thread safe and cannot use threads themselves. This caused a big issue as VIPS and some of its dependencies use the Posix Threads library. This impacted on which libraries and which versions could be used with FVS.

Many C++ applications use exceptions as a method of reporting errors. When an error occurs, an exception detailing the error is *thrown*. However, the very act of throwing an exception causes MySQL to crash. This also had an impact in what libraries could be used with FVS, and how FVS was built itself.

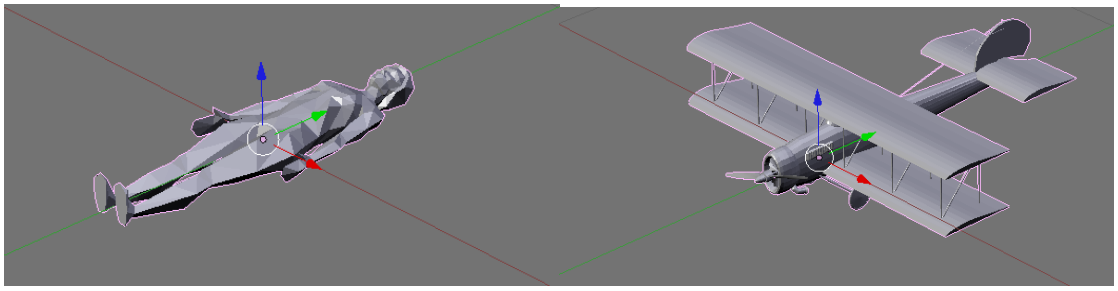
4.3.5 Thumbnail Generator

A cross platform thumbnail generator was required for both 2-D images and 3-D objects to show small representations of the real object on a web page of results. For 2-D images, the built-in VIPS functions for scaling were enough. 3-D, however, is much more challenging and little work has been done in this area. Typically, 3-D thumbnails are created manually by taking a screen shot from a 3-D viewer; a time consuming process.

To generate an image of a 3-D object, the mesh needs to be projected onto a 2-D plane. This process is known as *rendering*. There are two standard libraries that exist to render 3-D objects into a 2-D scene. The first is Direct 3-D, part of Microsoft's Direct X platform for using multimedia. It is widely used, however it is limited to Windows based platforms only. The second library is called Open GL and it is cross-platform. Open GL is maintained by a consortium of industrial partners that oversee the the specification of Open GL versions and of extensions. An extension is a particular feature that is not part of the current specification. New functionality can be added by vendors for immediate use long before it becomes part of the main specification.

Typically these libraries are used to render a 3-D scene into a 2-D window visible on a user's display. However, for our purposes, there may be no display (e.g. a headless server). Creating an Open GL graphics context with no screen requires platform specific extensions to Open GL. The Mesa 3D project provides a software implementation of Open GL (emulating functions typically performed in hardware) which includes a cross-platform method of creating an Open GL graphics context without requiring a display.

There are several features that need to be considered when producing a thumbnail image of a 3-D object. Unlike in 2-D, where the view is pre-defined, a 3-D object can be viewed from any orientation and at any distance from the camera (viewpoint). Ideally the object will fill the whole area of the thumbnail (or as much as possible) but be completely visible. For this, the bounding box (this is the smallest box that encloses the entire object) of the



(a) Human (b) Bi-Plane

FIGURE 4.7: Two objects in the same co-ordinate system, but require viewing from different angles.

object was computed and used to adjust the camera so that the bounding box was fully visible within the thumbnail area. This ensures that the object is within the viewable area.

The second issue is that of object orientation. An object could be represented in any orientation, however it is more likely to be axis aligned (i.e. along the x, y and z axes) but it is impossible to know for sure from the object alone. In the 3-D descriptors, Principal Components Analysis is used to determine the axis with the most variance and rotate the object such that this is the x-axis. This may not correspond with the primary viewing axis (that is the axis along which an observer would look to see the front of the object; see Figure 4.7). As a result, no additional rotation has been performed. It is assumed that all objects are in the left-handed co-ordinate system (y points upwards, x points to the right and z points towards the viewer), rotated about the x-axis such that the z axis points upwards.

4.3.6 Similarity Distance Normalisation

Similarity distance normalisation in ARTISTE took a sample data set and recorded all the distances for comparing each object with every other object. These distances were then used to plot a probability curve so that a score of 1.0 means there is 100% probability there was no better match and 0% means that there is 100% probability of getting a better match. The "control points" of the curve are then hard coded into a normalisation function for each algorithm on which this process was performed. While this process was sufficient for ARTISTE where there was a single data set. In SCULPTEUR there are several different data sets (one per gallery) and the existing normalisation did not work well in many cases.

The proposed approach for SCULPTEUR (which was not fully integrated into the system) was to store the score data in a histogram. The histogram is then used to determine a mapping using a log function between similarity distance and normalised score. This

data was stored as a blob (like feature vectors) and numerous normalisation blobs could be created for the same descriptor (for example one per data set).

4.3.7 Concluding Words

The architecture described here allows for very fast retrieval of objects based on their similarity. However, this is at the expense of robustness and flexibility. The tight coupling of CBR to the core SQL queries makes it hard to use algorithms that need more than just a pair of features for comparisons and return a single distance.

In eChase, CBR is removed from the core SQL queries and is instead accessed via a web service and the results added in to the final results table. This allows a much greater range of algorithms to be implemented and allows CBR to potentially be hosted on another machine. However, this does sacrifice a lot in query speed. A comparison of around 8000 objects can take nearly a minute in eChase, whereas this would have been a few seconds with SCULPTEUR.

4.4 Overview of the Classifier Agent

The Classifier Agent is one of the more ambitious goals of the SCULPTEUR project bringing together the fields of 3-D content-based retrieval, classification and semantic web technologies into a single system.

The aim of the Classifier Agent is to train classifiers using the existing feature vectors and metadata in the system as training data. It would then use these classifiers to classify objects (either new objects entering the system or existing objects with missing metadata) and add the classification to the system. The classification could either be some metadata field already in the database, or it could potentially determine that a new metadata field or concept in the ontology needed to be created. It would also have some of its functionality directly available to a user of the system should they have a specific task to complete.

Chapter 6 gives an in-depth review of the classification using 3-D CBR techniques as classifier inputs. In this chapter the focus is on the architecture as used in the SCULPTEUR project.

4.4.1 Architecture

The Classifier Agent is a user driven web application composed of a PHP user interface and C++ binaries providing the classification routines for speed. The user interface allows browsing of existing data sets and the creation of new data sets using the SRW

interface of the SCULPTEUR system to retrieve the set of objects specified by a query (either specific or general groupings of objects). It allows the creation of new classifiers either with manually or automatically specified training parameters and the subsequent classification of user uploaded objects with these classifiers. Classifiers can be tailored to generally classify between a large number of classes, or can be specialised to distinguish between a small number of classes by altering the training data set.

Two classification techniques are available in the agent, the k-NN classifier and a classifier that applies majority vote to winning clusters in the k-Means clustering technique. These techniques are well understood making it easier to understand why the agent is making the predictions it does.

4.4.1.1 3-D Object Data sets

A data set is a collection of objects and class labels that represent a problem to be solved. The problem could be as specific as "is this object a vase or a statue?", or it could be more general "what is this object?". Data sets can be manually created, however it is of more use to create a data set using the SCULPTEUR system, automatically obtaining the metadata for the objects.

A small Java program is used to communicate to the SRW to obtain a data set based upon a given query from the system. In the agent, the query is specially constructed to obtain labels for a single concept in the ontology due to the limited amount of metadata available for 3-D objects. However, much more complex queries can be formulated for use in a system containing more metadata. The SRW returns URLs to the 3-D object, feature vectors and thumbnails in addition to the requested metadata within its response. The required data can then be downloaded separately to complete the data set acquisition.

The system ontology contains a number of concepts or "classes" which can indicate object type such as vase, statue or tile, but they could also be artists names or periods of creation. These different types of class are not mutually exclusive, so if the descriptors support these different class types, then a query object may obtain several labels during the classification process, e.g. "type = vase" and "artist = Christopher Dresser". The agent is able to query the ontology and retrieve URLs pointing to 3-D objects and feature vectors through the SRW interface.

A training data set can be created by querying the system using the SRW interface to find specific or broad groupings of objects and metadata. Alternatively a data set can be manually created and presented to the system. Classifiers can be trained on a data set by manually specifying training parameters, or by using a technique to automatically determine the optimal parameters. Query objects can be passed to a classifier and the predicted label is presented to the user along with some statistics indicating the confidence of the classification.

QUERY := public_en.obj_number = "*" and public_en.pl_view = "*3D"

XPATH := /art_object | /art_object/object_name2 | /art_object/a_part |
/art_object/a_part/photo | /art_object/obj_number |
/art_object/a_part/photo/pl_short_caption |
/art_object/a_part/photo/representation

FIGURE 4.8: Example SRW Query to obtain all 3-D objects (Using VAM schema)

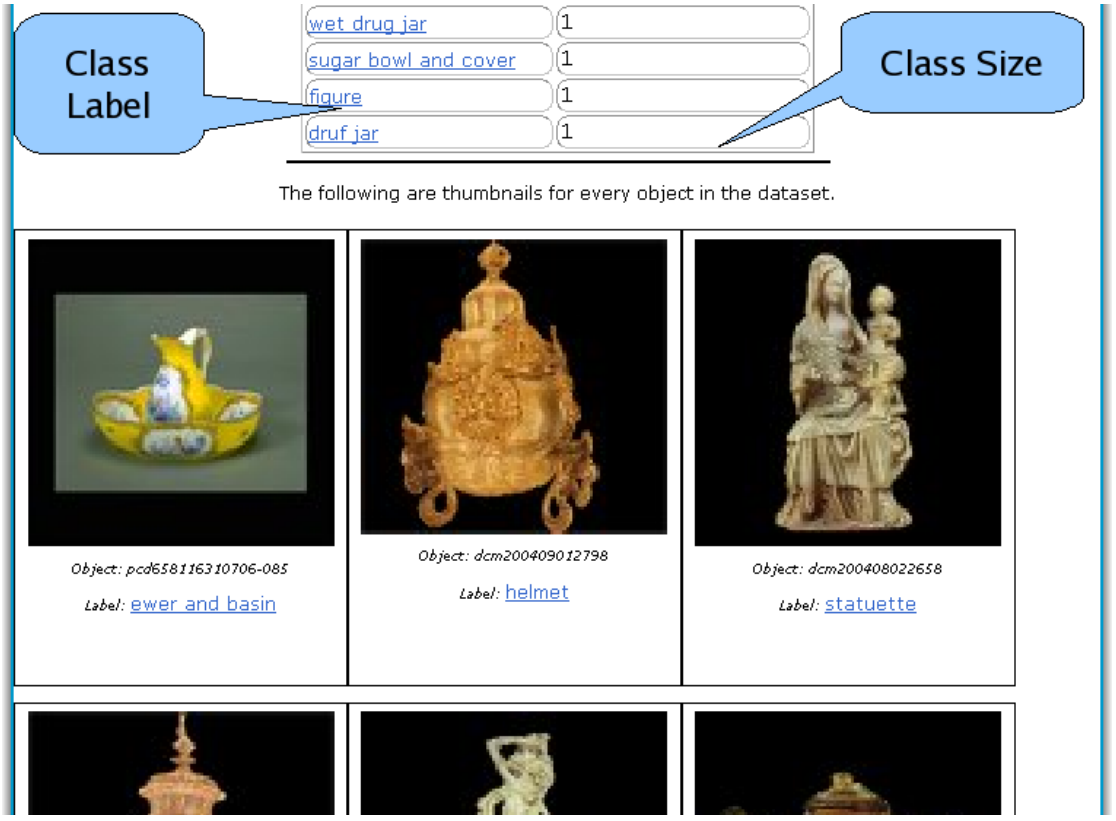


FIGURE 4.9: Data set Browser

Figure 4.9 shows a sample view from the data set browser. It shows the summary table for the number of objects in each class, followed by a thumbnail and other details for each object. Additionally, clicking on each label shows all the other objects with the same label.

4.4.1.2 Classifier Training

Each classification technique has a number of parameters that need to be specified. Both of the techniques used in the agent allow the choice of descriptor and distance metric to be selected in addition to those parameters specific to the technique. The k-NN classifiers allows the choice of k (the number of neighbours) to be selected and the k-Means classifier allows k (the number of clusters this time) and a cluster shift threshold to be specified.

To train a classifier, several things need to be taken into account. Firstly the properties of the data set need to be considered. These are the number of objects in the set, and the number of classes. This will typically have an effect on the time it takes to train a classifier and the number of classes determines the complexity of the problem. Typically low numbers of objects and high numbers of classes will lead to poor generalisation in classifiers whereas high numbers of objects and low numbers of classes should lead to high generalisation in the classifiers. The number of objects in each class determines how well the classifier will perform. If the classifier is good enough, i.e. shows high enough performance, it can be stored in the system for further use. If it is a particularly bad classifier, it can be discarded.

4.4.1.3 Classifier Optimisation

Classifier optimisation is the method of determining the best parameters for which to train a classifier with to maximise its accuracy. This can be done manually by trying all combinations of parameters and selecting the best set. This is often known as exhaustive search. Typically, however the number of possible combinations is far too large to actively search through each one. There are a number of techniques that exist to avoid searching all possible combinations by choosing the better combinations over the worse ones. These methods may not necessarily find the optimal solution, however they should find a near optimal solution.

The number and range of parameters makes manual or exhaustive search a long process. Particle Swarm Optimisation (PSO, Kennedy and Eberhart, 1995) allows the machine to find the optimal parameters for numerical problems without having to search through every possible combination. This is achieved by concentrating on parameters that give better results and ignoring those that give poorer results. The PSO cannot handle non-numerical parameters and in this work exhaustive search through the descriptor and distance metric combinations is performed before applying a PSO on the numerical classifier parameters. The larger the number of non-numerical parameters in the search space, the less useful this technique becomes.

This is described more fully in Chapters 3 and 6.

4.4.1.4 3-D Object Classification

Created classifiers are stored with a unique name selected by the user and are made available for use to all users of the system. Users may view a classifier to see what data set and training parameters it was created with and also the accuracy obtained through testing. This allows the user to select the classifier they think is most appropriate for classifying their object.

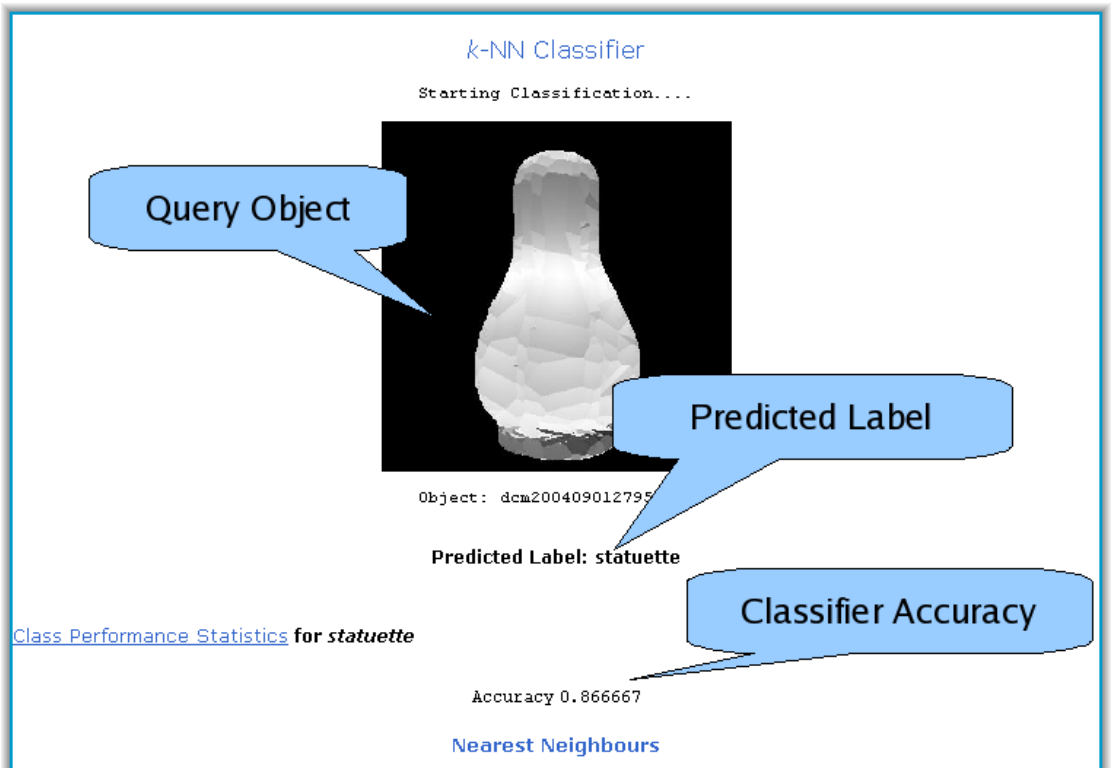


FIGURE 4.10: Classification Results

A user may upload a query 3-D object to the system for classification. A pre-created classifier can be selected by the user. A results page is presented to the user showing an automatically generated thumbnail of the query object, the predicted label and a confidence value of the correctness of the classification. This page also shows the k nearest objects used to classify the object for k-NN classifiers, or the objects in the nearest cluster for k-Means classifiers.

Figure 4.10 shows the results from a classification using a k-NN based classifier. The query object is displayed along with its predicted label. This is followed by the accuracy for the predicted label from the classifier (this is different from the accuracy of the classifier for all classes). Following this would be the nearest neighbours used to determine the predicted label. These would be displayed in much the same way as the data set browser.

4.4.2 Classifier Agent Evaluation

A prototype of the classifier agent was made available to users during formal evaluation of the SCULPTEUR system. Feedback from this evaluation (Coates, 2005) was then used to update the agent for the final version delivered for the project. Due to the small size of the data set at that time, the evaluation focused on the usability of the classifier agent rather than how well it worked in classifying user objects. Table 4.1 shows the

Class Name	Training	Testing
Mask	3	3
Misc	9	10
Statue	8	7
Tile	16	16
Tool	5	5
Vase	31	31

TABLE 4.1: The Evaluation data set

class sizes in the data set. Users were asked to create some classifiers based on the data sets available and choose the values of parameters. The optimisation technique was not available in the initial evaluation. The users were provided with the URL to the system and asked to create a few classifiers based on the training data available. This consisted of a small number of objects from the Victoria and Albert Museum (VAM) and GET-ENST.

4.4.2.1 Evaluation Feedback

The feedback from the evaluation highlighted that the initial version of the Classifier Agent was too complicated for typical users. Too much of the underlying technical details were exposed to the users and the interface assumed prior knowledge about classification techniques. The users evaluating the system had little or no prior knowledge about classification techniques and so found the initial system hard to use due to lack of understanding of the techniques presented. This resulted in producing a much more descriptive interface, complete with guides on how the classification techniques worked and what good default values for the different parameters would be in the final version of the agent.

Another issue highlighted that the range of statistics presented was confusing to the users as they had little idea of the meaning of each statistic and were more concerned on just knowing whether the classifier was good or not. In the final agent, only the accuracy statistic was shown as this gives a good overall indication of performance that is easy for users to understand.

These issues led to only a few classifiers being made during the evaluation and little experimentation appeared to have been performed in attempting to find good parameters for the classifiers. In the final version of the agent, the optimisation technique is available to users so they do not have to manually find good parameters.

4.4.3 Classifier Agent Discussion

As a proof of concept, the Classifier Agent worked quite well showing that it is indeed possible to use the 3-D descriptors of objects in the system as inputs to classifiers that are able to distinguish between broad classes. It also showed that it is possible to use optimisation techniques to find near optimal training parameters for these classifiers. However, further experimentation is required using larger data sets and more closely related classes. One concern is how well the current system will scale with increasing amounts of data and classes. The classification techniques used may not be able to cope well with more complex class boundaries and the optimisation techniques will take more than just a few minutes to complete.

Another area for future work is the improvement of the training data set creation. The query interface of the SCULPTEUR system could be used to define a training data set as the result of a query. A user would then need to select a metadata field as the class labels, or the user would need to manually classify the data if no suitable field existed.

As more and more classifiers are created, it will become harder for a user to select the appropriate classifier for their problem. One solution is to get the machine to select the best classifier itself. A technique for this task is called Dynamic Classifier Selection and it is investigated in Chapter 6.

4.5 Conclusions

In this chapter, an overview of the SCULPTEUR project and system has been presented. Two components of this system, content-based retrieval and the classifier agent have been described. These components provide some of the motivation and direction of the work in the next two chapters. Chapter 5 investigates how well the different 3-D descriptors perform in a more in depth analysis. Chapter 6 investigates the use of various classification techniques applied to the 3-D descriptors.

Chapter 5

3-D Content-Based Retrieval

5.1 Introduction

The previous chapter described the SCULPTEUR project and the content-based retrieval architecture. This chapter examines the performance of the 3-D descriptors used within the SCULPTEUR project.

The area of Content-Based Retrieval (CBR) started to take off during the 1990's with the start of wide spread World Wide Web (WWW) usage giving increasing access to large numbers of digital images (Smeulders et al., 2000). Textual retrieval for content soon became insufficient with the explosion in the number of images and poor annotation. The problem was down to keywords. When using text to locate documents, you can find those words in the documents. 2-D images and other forms of multimedia typically do not have words in them so a set of keywords are assigned to the image which can then be matched against the query terms. One problem lies in defining a suitable set of keywords that adequately describes the content and knowing what keywords are available to be able to appropriately define the query. The phrase "an image is worth a thousand words" is an apt description. Fixed vocabularies can help the user to know what terms are available, however, the vocabulary may not be descriptive enough. Textual querying is therefore not a suitable query mechanism for content retrieval; however, as much recent research has shown, it can complement a content-based query very well (Goodall et al., 2004b). This chapter however is concerned with content-based retrieval on its own.

For CBR there are two main questions that we would like to address;

- I want to find similar objects to mine, what should I use?
- I have an example of object type X and I want to find other examples of object type X, what should I use?

The first question is asking what is the best overall method to use. The second question is being more specific by defining the type of object that is required. This type of question requires more detailed understanding of the performance of the CBR techniques for different types of object.

How can we find out this information? Answering the first question is easier, we can just put all our descriptor and metric combinations through an evaluation and rank them according to some retrieval performance metric. The second question is slightly different in that we wish to optimise the retrieval by choosing the descriptor and distance metric combination that performs best on objects of type X. In this case, statistics need to be generated for each class per descriptor and distance metric combination which can easily generate an unmanagable amount of data. We could put all the descriptor and distance metric combinations through an evaluation and see which combinations are ranked highest and give that information back to the user. This is good for small numbers of classes, but becomes less useful for many classes (information overload). It is also advantageous to know what descriptors perform badly on a class. Often the best performing descriptor may not be appropriate to use due to computational cost, and so a faster one may be more suitable. It may also be useful to see which classes the descriptors are unable to distinguish between. This helps identify areas to target for future descriptors.

Chapter 2 reviewed a large range of 3-D content-based retrieval algorithms and distance metrics. The review highlighted several areas of possible research. Firstly, Shilane et al. (2004) is the only attempt at a comprehensive evaluation of different algorithms, however it does not cover all of the algorithms available. Secondly there is no general evaluation of different distance metrics except in papers such as Ankerst et al. (1999); Hetzel et al. (2001) where a few metrics are compared to see how they effect the performance of a particular algorithm. Typically the literature makes use of data sets built for experimental purposes and not data used in real production systems.

Here our first investigation is to compare and contrast the performance of a selection of these algorithms to identify their strengths and weaknesses. This will also highlight potential areas of further research to improve the existing algorithms or develop new ones. This chapter begins by explaining the problem of content-based retrieval and more specifically the problems involved in 3-D CBR. This is followed by a description of the different 3-D CBR algorithms, distance metrics used in this work and the techniques employed for evaluating retrieval performance. The performance of the different 3-D algorithms and distance metrics is then evaluated in terms of the Princeton Shape Benchmark (PSB) and with real museum objects. This chapter then ends with some concluding remarks.

(a) Fully Textured (b) Mesh

FIGURE 5.1: Example 3-D Object

5.1.1 The Content-Based Retrieval Problem

3-D Content-Based Retrieval is concerned with the retrieval of objects based upon their content. Typically this can be achieved by using an example to find "similar" objects, however it is also possible to define a query in terms of partial content (e.g. a sub-section of a model or image, or specific colours contained within the object). The definition of "similar" can be somewhat vague and is dependent on the task in hand, but often means similar shape, colour or texture. For example, to find vase shaped objects, the user can give the system an example vase model. Alternatively to find red objects, a user can specify the colour red using e.g. a colour palette and giving the system that instead. It is also possible to "sketch" the basic outline of an object although the quality of results depends upon the skill of the user in sketching a query (Min et al., 2003). Traditionally the area of CBR has been limited to 2-D images, but more recently has expanded into other areas of multimedia including 3-D objects. While the fundamental problem remains the same, each type of media brings certain advantages, but also a number of additional problems that need addressing. The rest of this chapter is concerned with the problem of 3-D CBR.

5.1.2 3-D CBR Problems

3-D objects have a number of useful properties that aid CBR. Unlike 2-D images, 3-D objects contain only the representation of the object and typically have no extra information such as background, occluding objects or varying lighting conditions to complicate things. Figure 5.1 shows an example 3-D object and the corresponding mesh representing

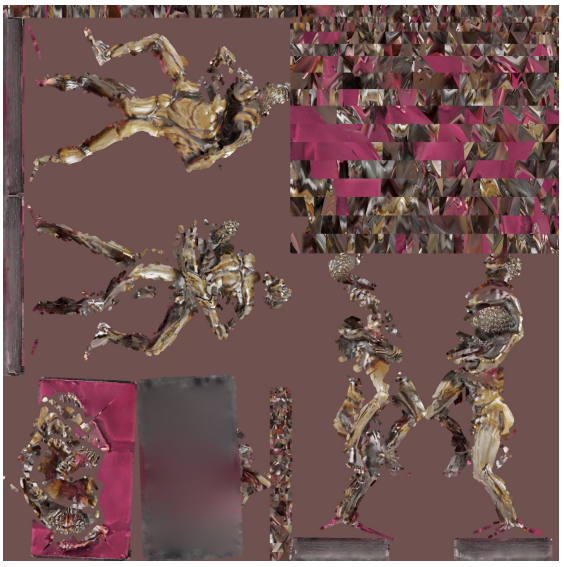


FIGURE 5.2: Example Texture Map

FIGURE 5.3: Untextured Object

the object. The representation of the object is explicit rather than implicit or embedded in the pixels as in the case for images. This simplifies the problem in many ways. However, there are several issues specific to 3-D CBR. Typically 3-D objects are represented as a mesh of (possibly) interconnected polygons, typically triangles. In the ideal case this will be a single closed mesh of triangles, i.e. every triangle edge connects to only one other triangle edge. Additionally all triangles will be consistently orientated, either all pointing inwards, or all pointing outwards (outwards will typically be the correct facing for viewing an object). When calculating the surface normal of a triangle, the triangle can be specified either with a clockwise or anti-clockwise ordering of the vertices. Choosing the wrong ordering means the normal will point in the opposite direction to

that intended (e.g. inwards instead of outwards). In some cases, especially from models obtained off the web, models may be composed of higher order polygons, multiple open meshes, and arbitrary triangle orientation (which may or may not be specified). Additionally the same object may be represented with a differing number of polygons or geometric transformations depending on how the model was created. These factors need to be taken into account either directly in the algorithm or as a pre-processing step. Failure to do so can lead to significantly reduced retrieval performance. Of course, in some cases pre-processing may remove the very feature that distinguishes between one class of objects and another. For example, something like a footstool and a table could differ only in scale. However, this requires all models to be acquired with a known scale, a detail which is often lacking.

Another characteristic of 3-D models is that colour information (typically a texture map; see Figure 5.2) may be stored in another location (i.e. another file) and can easily become separated and lost. Figure 5.3 shows the untextured version of the object in Figure 5.1. In some cases a model may not have any colour information at all, e.g. if it has been acquired by a laser scanner. This makes the application of colour based techniques limited.

In this chapter we will evaluate a number of descriptors and distance metrics in terms of their retrieval performance. Two data sets will be used. The first is a small data set composed of objects generated from real museum artifacts. The second is a much larger data set from the Princeton Shape Benchmark and is composed of objects obtained from the World Wide Web that will generally be of low quality. The following sections will give an overview of these descriptors, distance metrics and evaluation techniques along with more details of the two data sets used.

5.2 Description of Algorithms

A number of algorithms have been implemented for use in the SCULPTEUR project (Addis et al., 2005b) and are used within the work presented here. This section gives some more details of the algorithms implemented although Chapter 2 gives a more comprehensive overview. The majority of 3-D objects used in this work do not have any colour information associated with them, so only shape based descriptors have been used. Fourteen algorithms have been implemented. These are the Area Volume Ratio (Tung and Schmitt, 2004), the Cord Histograms (six versions) (Paquet and Rioux, 1999b), Extended Gaussian Image (EGI) (Horn, 1984) and 3-D Hough Transform (Zaharia and Prêteux, 2001b) (both are implemented using spherical and octagon decomposition methods as described below), the D2 Shape Distribution (Osada et al., 2001) and the modified D2 Shape Distribution (Ohbuchi et al., 2003a) and finally the augmented Multi-resolution Reeb Graph (MRG) (Tung and Schmitt, 2004).

Short Name	Descriptor Name
Area Volume	Area Volume
Cord Hist 1	Cord Histogram (Lengths)
Cord Hist 2	Cord Histogram (First Principal Axis)
Cord Hist 3	Cord Histogram (Second Principal Axis)
Cord Hist 4	Cord Histogram (Joint First/Second Principal Axes)
Cord Hist 5	Cord Histogram (Combined First/Second Principal Axes)
Cord Histogram	Combined Cord Histogram
EGI Oct	Extended Gaussian Image (Oct Method)
EGI Sphere	Extended Gaussian Image (Sphere Method)
Hough Oct	3-D Hough Transform (Oct Method)
Hough Sphere	3-D Hough Transform (Sphere Method)
MD2	Modified Shape D2
Shape D2	Shape Distributions (D2 variant)
MRG	Augmented Multi-resolution Reeb Graph

TABLE 5.1: Short names of descriptors

There are several desirable characteristics for a 3-D descriptor. It should be invariant to changes in rotation and position. Scale invariance can also be desirable. However, in some cases the original scale may be desirable if the objects are all captured in the same manner, or encode the scale somehow. Scaling methods can also scale the model such that the largest axis is in the range [0.0:1.0] and the other axes are scaled by the same factor, or each axis can be separately scaled such that they are all in the range [0.0:1.0]. It is also desirable for the descriptor to ignore how the model is composed; that is it should work on the surface of the model and not directly on the polygons composing it.

Table 5.1 lists the short names for the descriptors used in this work.

5.2.1 Area Volume Ratio

The Area to Volume ratio descriptor (Tung and Schmitt, 2004) calculates the ratio between the surface area and the volume of a 3-D object. See Section 2.6.1 in Chapter 2 for more details. The principal drawback of this method is that the triangles need to be orientated consistently for a correct volume calculation and a closed mesh is required.

The area volume ratio descriptor is likely to perform badly against the PSB models as they are more likely to have inconsistently orientated triangles and have holes in the mesh. Both of these conditions will result in an incorrect volume calculation. Typically the museum objects will have consistently orientated triangles due to the acquisition process, however some objects may contain holes in the mesh. Therefore varying, but typically low performance is expected, however it is fast to compute which may be advantageous in some circumstances.

5.2.2 Cord Histograms

The Cord Histograms were introduced by Paquet and Rioux (1999b) and are described in Section 2.6.2

The histograms are rotation and translation invariant but again normalisation for scale is required. Principal Components Analysis (PCA) is used as the normalisation step. This also adjusts the first principal axis to the x-axis, and the second to the y-axis making the angle based Cord histograms easier to calculate. Histograms with 16 bins have been used for the Cord Hist 1, 2 and 3 descriptors. The Cord Hist 4 descriptor has 32 (16 + 16) bins and the Cord Hist 5 descriptor has 256 (16 * 16) bins. The Combined Cord Histogram has a histogram size of 48 (16 + 16 + 16) bins.

5.2.3 Extended Gaussian Image

The Extended Gaussian Image (EGI) method is a way of indexing features. It is described in Section 2.6.14

Two methods are used to perform the indexing to the histogram. These are the Oct method and the Sphere method (Zaharia and Prêteux, 2001b, 2002). The Oct method subdivides an octahedron twice such that there are 128 faces. Each face has a surface normal and each quantised surface normal represents a bin for the histogram. The Sphere method uses spherical co-ordinates as the index into a bi-dimensional histogram. Each axis is quantised into five sections. The problem with the sphere method is that each bounded region can be a different size, with larger regions at the equator and smaller regions near the poles. Misalignment during PCA can cause this to be a problem.

The EGI Oct method has a histogram of 386 (128 * 3) bins and the EGI Sphere method has a histogram of 50 (5 * 5 * 2) bins.

5.2.4 Hough Transform

The 3-D Hough Transform developed by Zaharia and Prêteux (2002) takes its roots from the 2-D generalised Hough Transform (Ballard, 1981). See Section 2.6.15 for more details. Like with the EGI technique, the Oct and Sphere methods are employed in indexing surface normals in this implementation. The Hough Transform creates a table indexed by surface normal orientation (represented as spherical co-ordinates in the Sphere method or as a face index in the Oct method) and distance from centre of mass storing the surface area for each polygon in the mesh. Similarity matching is performed by comparing the tables treated as histograms. The true Hough Transform method creates an accumulator that gathers evidence for the existence of an object based on parameters

calculated from a reference. Peaks in the accumulator space identify possible matches. However, this is quite slow compared to matching just the histograms calculated here.

The Hough Sphere method has a histogram of 250 (5 * 5 * 10) bins and the Hough Oct method has a histogram of 1280 (10 * 128) bins.

5.2.5 Shape Distributions

The Shape Distributions (Osada et al., 2001) are a collection of descriptors that capture distributions of various features of the shape of an object. See Section 2.6.4. The work done by Osada et al. (2001) determined that the D2 variant performed best overall and hence this variant is used here. The Shape D2 descriptor captures the distribution of distances between random pairs of points on the shape surface. It is rotation and translation invariant and robust to changes in mesh resolution. However, it is not scale invariant and so requires some pre-processing. It is created using a large number of sample points (1024^2) recorded into 64 histogram bins as used in Shilane et al. (2004); Osada et al. (2001).

5.2.6 Modified Shape Distributions

Based upon the Shape D2 algorithm, the modified Shape D2 (MD2) (Ohbuchi et al., 2003a) has several modifications that aim to improve it and these are described in Section 2.6.5. The MD2 uses a 64 bin histogram and 1024 sample points.

5.2.7 Augmented Multi-resolution Reeb Graph

The augmented Multi-resolution Reeb Graph (MRG) (Tung and Schmitt, 2004) stores geometric attributes associated with nodes of the Reeb Graph (see Section 2.6.21). These geometric attributes are typically various 3-D descriptors applied to the section of mesh the node represents. In this implementation, the attributes are the value of *mu* (the function of the graph), surface area, volume, Cord Histograms (Cord Hist 1, 2 & 3), surface curvature (as in the 3-D Shape Spectrum Descriptor (Zaharia and Prêteux, 2001a)) and the 3-D Hough Transform.

Unlike the other methods, simple histogram matching will not suffice. At the lower level, histogram matching between geometric attributes is still used, however similarity between nodes is much more complicated and is based on graph matching.

Short Name	Distance Metric
City-block	City-block Distance, L_1 Norm
Euclidean	Euclidean Distance, L_2 Norm
Intersect	Histogram Intersection
Chi	χ^2 Distance
Bhattacharyya	Bhattacharyya Distance
Kullback	Kullback-Leibler (symmetric) Distance
Kullback-ns	Kullback-Leibler (non-symmetric) Distance
Quadratic	Quadratic Distance

TABLE 5.2: Short names of distance metrics

5.3 Description of Metrics

As part of the testing process a range of distance metrics have been implemented as additional query parameters. These are the L_1 (City-block) and L_2 (Euclidean) norms, the quadratic distance, the Kullback-Leibler distance (both symmetric and non-symmetric versions), the Bhattacharyya distance, the χ^2 distance, and the histogram intersection as described in Chapter 2. The quadratic distance is the only metric that allows some degree of parameter control. The choice of bin distance function, and the value of sigma are left to be determined by the application. It was decided to use the L_1 norm as the distance function (this seems to be the most intuitive as there is only a single pair of values). Sigma is used as 1.0 as Ankerst et al. (1999) reported no significant changes in performance in altering the value, which was confirmed in early experimentation.

The majority of literature uses the Euclidean distance to match feature vectors. The Bhattacharyya and Kullback-Leibler distances have been shown to give better results in some cases, but this is at the expense of speed.

Table 5.2 lists the short names for the distance metrics used in this work.

5.4 Methodology

The aim of this work is to determine what descriptors are good in general and in which situations an alternative descriptor and metric will be more useful. The first case will be the more commonly used scenario. The second case will be useful for more specific queries. The work by Shilane et al. (2004) has shown that there is no one descriptor that does well in any situation, so in some cases it makes sense to use one descriptor, in others it makes sense to use a different one.

The descriptors to be evaluated are the Shape D2, MD2, Cord Hist 1, Cord Hist 2, Cord Hist 3, Cord Hist 4, Cord Hist 5, Cord Histogram, Area Volume, EGI (Oct and

Sphere methods) and the Hough Transform (Oct and Sphere methods) and MRG descriptors. The evaluation then continues to compare the distance metrics side by side with the Shape D2 descriptor. These are Minkowski L_1 and L_2 norms, Bhattacharyya, χ^2, Kullback-Leibler (both symmetric and non-symmetric), histogram intersection and quadratic distances metrics. The individual class performance will then be examined.

To evaluate retrieval performance, the Princeton Shape Benchmark (PSB) framework has been used (Shilane et al., 2004). This framework has been chosen as it is the only known framework proposed in the literature for evaluating 3-D CBR. The provided data set allows comparisons to other studies based on it. The PSB paper (Shilane et al., 2004) gives a sample comparison of a set of 3-D descriptors. This is composed of a set of tools to generate the evaluation statistics, graphs and images. It also contains a reference data set of approximately 1,800 manually classified objects.

In addition to the PSB data set (which is composed of models obtained from the World Wide Web and will be typically of a low quality) a data set created from museum artifacts is also used. This is composed of higher quality models and will be more typical of the kind of data that will be used. However, due to the time and cost of acquiring 3-D models of artifacts, this data set is much smaller.

The evaluation procedure will make use of the evaluation criteria, precision-recall graphs, nearest-neighbour, first- and second-tier, E-Measure and DCG statistics to give a broad view of the abilities of the descriptors. The statistics are calculated for the whole data set and on a per-class basis. The statistics are normalised to the range [0.0:1.0] for presentation purposes. The tier image will help give an overview of the class-based performance of each descriptor and results will be backed up by the statistics for individual classes. The tier image (For example, see Figure 5.7) shows the nearest neighbour (black) and the first (red) and second tier (blue) results for each object in the data set. White pixels mean objects are very dissimilar. The image diagonal should be black indicating that each object is matched best with itself. If the diagonal is not fully coloured with nearest-neighbour matches then this indicates a possible problem with the algorithm. Ideally all the coloured pixels would be within the class boundaries along the diagonal. These techniques are described in more detail in Chapter 2.

As an alternative to the PSB statistics, the ratio of within-class and between-class variance is presented. Within-class variance is the variance of the members within a given class. The between-class variance is the variance of the mean of each class.It is expected that descriptors that give a large ratio are better at separating the classes and so will provide better retrieval performance than those with lower ratios.

Intuitively, we would expect that a descriptor that minimises the variance within classes and maximises the variance between classes would perform better than those descriptors that do not. In this work the ratio of between class variance to mean within class

Class Name	Size
Figurine	52
Head	8
Mould	15
Pot	57
Statue	45
Tile	28
Misc	15

TABLE 5.3: Museum Data set: Classes and sizes

variance is calculated (See Equation 5.1). The higher the value of the ratio, the better the expected performance.

$$Variance\,Ratio = \frac{\text{between class variance}}{mean(\text{within class variance})} \tag{5.1}$$

This ratio does not take into account the class sizes, i.e. classes with few members have equal weighting to classes with many members. Conceptually, this ratio turns out to be very similar to the process defined in ANOVA (ANalysis Of VAriance) although there are differences in the calculations of the between and within class values.

5.4.1 3-D Object Data sets

The PSB data set contains four different levels of classification. These are hierarchical in nature and have a number of empty classes that are parent types. Typically only the leaf classes contain the actual objects, although in some cases the parent also contain objects. There is a "base" classification containing about 160 non-empty classes and this is the classification used in this work. There are also another three coarser grained classification schemes, with the coarsest classification containing only four classes. Each of these classifications is split into a training and testing group of equal size. This split is aimed for use in a classification system. The descriptor comparison in Shilane et al. (2004) used the base test set (referred to as the PSB Data set in the rest of this chapter) of 907 models so for comparative purposes, this work will use the same. Table B.1 shows the class hierarchies (separated by '/') for each populated class and it's size. As can be seen the classes have a varying number of objects ranging from only four objects to fifty objects. All the PSB objects are represented in the OFF file format (Shilane et al., 2004) storing only vertex data.

The museum data set contains 210 objects manually classified into 7 classes. Table 5.3 shows the classes and their sizes. These objects are represented in VRML and .tri file

formats and range from those containing a few thousand polygons, to some containing many thousands of polygons. However they typically contain many more polygons than the models in the PSB data set.

The figurine and mould classes contain the mould used to create the original figurines. There are two different figurines class contains objects that are clay figurines of two different objects. The figurines and moulds differ mainly in size and level of detail. A mould creates a large number of figurines before it breaks (or becomes otherwise unusable) and so a new mould is created from an existing figurine. However as the figurine is made of clay, it shrinks while drying making it smaller than the original. The new mould is also able to capture less detail than the original. This means that newer generations of figurines are much smaller and have much less detail than older generations. The mould and figurine objects are also broken in many cases and the model is of only a part of the original. The tile class is composed of paving tiles, typically flat square shaped objects. The main detail in these objects lies within the texture map (which is not used in this work). The statue class contains various statue-like objects. The pot class contains various vase shaped objects. The head class contains models of head shaped objects. The misc class contains objects that did not readily fit into any of the other classes.

It is expected that the tile class will be easy for the descriptors as they are all basically the same shape. The figurine and mould class likewise. However, we may find that the figurine and mould class have more correspondence due to the moulds being used to create the figurine. The other classes have a much more diverse range of objects, however the pot and head classes should obtain good retrieval results as they are more homogeneous than the other classes such as the misc and statue classes.

The objects and classes within the two data sets are quite different. The PSB data set is made up of objects obtained from the WWW. This means that they are likely to be lower quality due to the way they have been created and to keep their file size down. This could mean that the meshes are of a low resolution. Other problems include incomplete meshes and unlabelled or inconsistently orientated polygons.

5.5 Evaluation Results

The evaluation begins first by evaluating the descriptors and distance metrics in the context of the PSB data set, and then in the context of the Museum data set. For each data set, the overall results are first presented followed by results of class by class analysis.

Descriptor	Ratio
Area Volume	0.6
Cord Hist 1	5.7
Cord Hist 2	11.2
Cord Hist 3	8.2
Cord Hist 4	19.4
Cord Hist 5	38.1
Cord Histogram	25.2
Shape D2	80.5
Modified Shape D2	80.7

TABLE 5.4: PSB Data set: Ratio of Between Class and Mean Within Class Variance

5.5.1 Princeton Shape Benchmark Data set Results

The PSB data set contain models obtained from the World Wide Web and as such are prone to problematic meshes. As a result, the EGI Oct and both Hough methods are missing from these results as the quality of the models adversely effected the descriptors such that meaningful results were unable to be obtained.

Table 5.4 shows the ratio of between class variance and mean within class variance. The higher the value, the better as it means that members of the same class are grouped tightly together, but different classes are spread apart. It can be seen that the ShapeD2 and MD2 both give a very high ratio compared to other descriptors. The Cord Hist 5 and Cord Histogram (and to some extent Cord Hist 4) also give higher ratios than other descriptors, but the ratio is much closer to them. We would expect the Shape D2 and MD2 to perform better than the other descriptors. We would also expect the Area Volume (with the smallest ratio) to perform worse than the other descriptors.

The first experiment compares the relative performance of the descriptors on the PSB data set. The Euclidean distance metric has been used as the distance metric. Table 5.5 shows the statistics (see Section 2.9 for descriptions of these) and Figure 5.4 shows the corresponding precision-recall curves. The best performing descriptors are the Shape D2 and the MD2 descriptors. The Combined Cord Histogram and Cord Hist 4 are the next best. The worst descriptors are the Cord Hist 1 and Area Volume descriptors, although they are still better than performing a random retrieval. The precision-recall graph show a similar ranking of results to the statistics.

Comparing these results to those in the Princeton Shape Benchmark (Shilane et al., 2004), we can see that the Shape D2 is one of the poorer performing descriptors in their comparison where as it is one of the better ones in our comparison. This means there is definite room for achieving greater retrieval performance with alternative descriptors. It is also worth noting that the two EGI based techniques perform better than the Shape D2

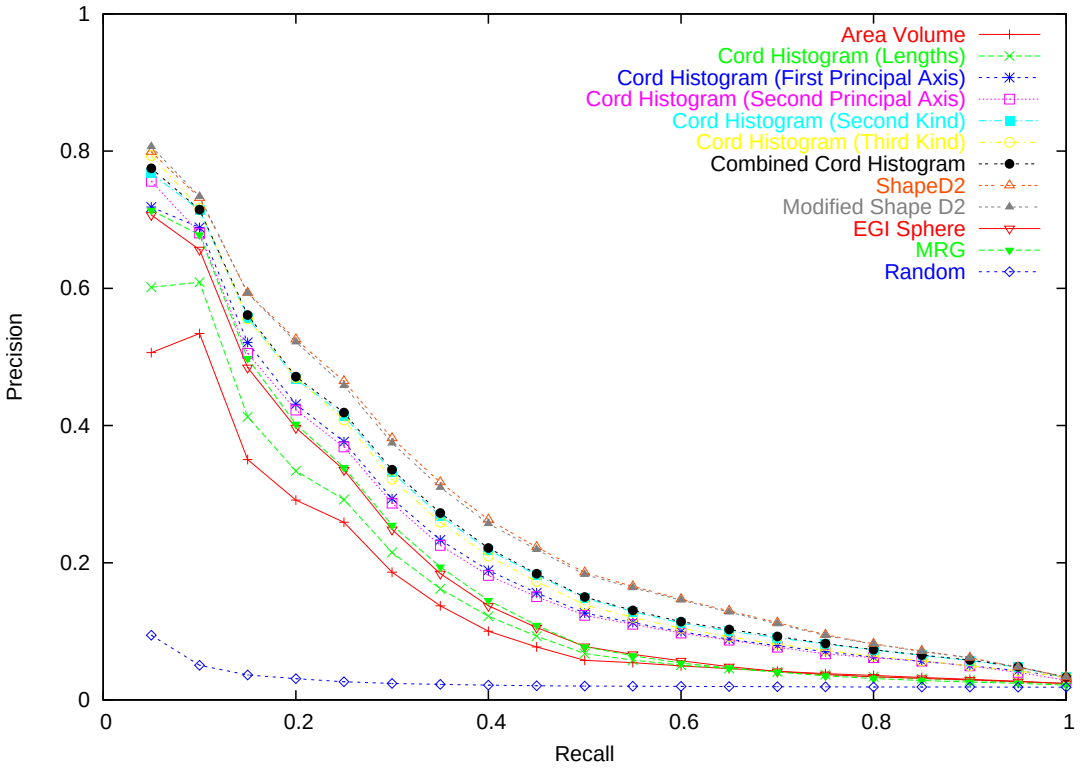


FIGURE 5.4: PSB Data set: Descriptor Comparison using the Euclidean Distance

Descriptor	Nearest Neigh-bour	First Tier	Second Tier	E-Measure	DCG
Area Volume	0.042	0.142	0.178	0.074	0.384
Cord Hist 1	0.131	0.162	0.200	0.088	0.399
Cord Hist 2	0.239	0.206	0.266	0.124	0.453
Cord Hist 3	0.202	0.205	0.261	0.124	0.446
Cord Hist 4	0.280	0.224	0.284	0.135	0.470
Cord Hist 5	0.282	0.221	0.277	0.132	0.463
Cord Histogram	0.290	0.225	0.287	0.136	0.471
EGI	0.181	0.186	0.224	0.102	0.418
MD2	0.341	0.242	0.325	0.154	0.492
ShapeD2	0.336	0.246	0.327	0.156	0.492
MRG	0.250	0.184	0.218	0.103	0.417
Random	0.019	0.018	0.034	0.018	0.307

TABLE 5.5: PSB Data set: Descriptor Performance using the Euclidean Distance

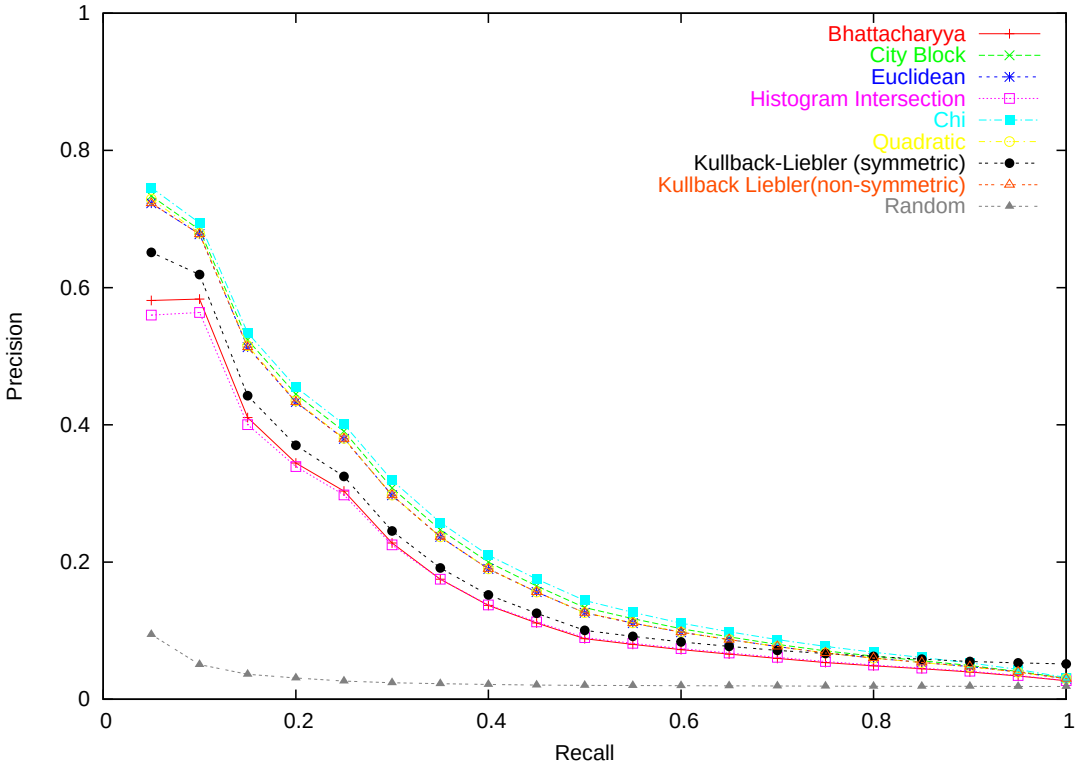


FIGURE 5.5: PSB Data set: Average Distance Metric Performance

Metric	Nearest Neigh-bour	First Tier	Second Tier	E-Measure	DCG
Bhattacharyya	0.120	0.160	0.199	0.087	0.400
Chi	0.224	0.199	0.250	0.117	0.440
City-block	0.215	0.193	0.244	0.114	0.435
Euclidean	0.205	0.189	0.238	0.110	0.431
Intersect	0.144	0.153	0.195	0.088	0.401
Kullback	0.156	0.168	0.211	0.091	0.401
Kullback-ns	0.205	0.189	0.238	0.110	0.431
Quadratic	0.207	0.189	0.239	0.111	0.431

TABLE 5.6: PSB Data set: Average Distance Metric Performance

in their comparison. One difference between implementations is that our EGI descriptors store surface area whereas the PSB implementations do not.

The second experiment compares the relative performance of the distance metrics on the PSB base test data set averaged over all descriptors. Table 5.6 shows the statistics and Figure 5.5 shows the corresponding precision-recall curves. It can be seen that the metrics fall into two groups. The higher performing group contains the Chi, City-block, Euclidean, Kullback-Leibler (non-symmetric) and Quadratic distances and the lower performing group contains the Bhattacharyya, Histogram Intersection and Kullback-Leibler

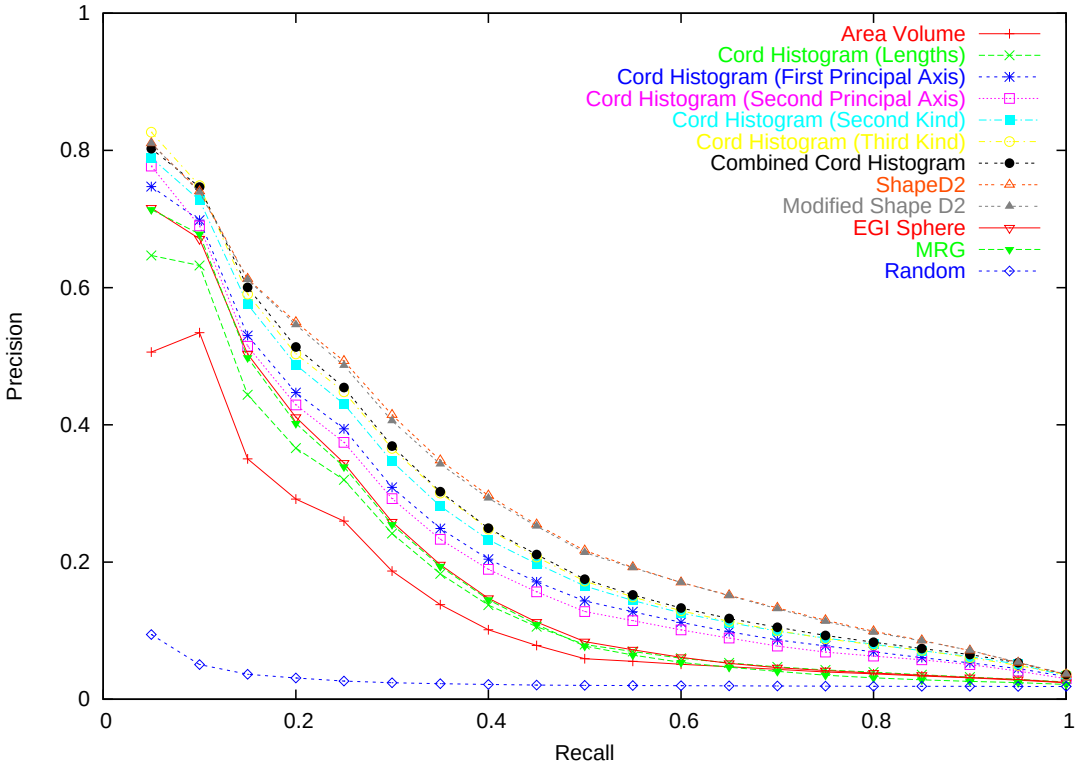


FIGURE 5.6: PSB Data set: Best Distance Metric for Descriptor

(symmetric) distances. The Chi distance gives the best average performance overall and the Bhattacharyya distance giving the lowest average performance overall.

Figure 5.6 shows the precision-recall curves and Table 5.7 shows the corresponding statistics for the best metric for each descriptor ranked on highest DCG score. In some cases multiple distance metrics scored exactly the same. The table shows that the Bhattacharyya and Chi distances are the best performers typically, with the City-block distance the only other metric and that appears only once. This contrasts with the previous results in Table 5.6 and Figure 5.5 where the Bhattacharyya distance was the worst overall distance metric. Although increases in performance can be observed compared to Table 5.5, the increase is only slight.

Table 5.8 shows the descriptor and metric combination that gave the list of classes the best performance (DCG). As can be seen, no one combination comes out on top for all cases. Most combinations are only good for a single class, and only a few combinations have a sizable number of good classes. The Shape D2, MD2 and MRG descriptors have more classes in general than the other Cords based descriptors. However, the Cords based descriptors have many classes spread out over the distance metrics. This suggests that for these descriptors, no one metric works bests, where as for the Shape D2 descriptor a smaller number of distance metrics have been best matches indicating that some metrics are better than others for this descriptor.

Descriptor	Metric	Nearest Neighbour	First Tier	Second Tier	E-Measure	DCG
Area Volume	Bhattacharyya / Chi	0.043	0.142	0.178	0.075	0.385
Cord Hist 1	Bhattacharyya	0.163	0.172	0.217	0.098	0.413
Cord Hist 2	Chi	0.246	0.217	0.278	0.131	0.465
Cord Hist 3	Cityblock	0.228	0.208	0.266	0.126	0.448
Cord Hist 4	Chi	0.303	0.234	0.297	0.142	0.477
Cord Hist 5	Chi	0.324	0.242	0.310	0.148	0.484
Cord Histogram	Chi	0.330	0.247	0.308	0.148	0.489
Shape D2	Chi	0.215	0.190	0.232	0.105	0.425
Modified Shape D2	Bhattacharyya	0.354	0.267	0.348	0.165	0.507
Shape D2	Bhattacharyya	0.352	0.269	0.348	0.166	0.509
MRG	N/A	0.251	0.185	0.219	0.103	0.417

TABLE 5.7: PSB Data set: Best Metric for Descriptor - Based on highest DCG

Figure 5.7 shows an example tier image for the Shape D2 descriptor using the Euclidean distance metric (see Section 2.9 for a description). We can immediately see the diagonal is a solid black line showing the nearest neighbour is correctly matched as itself. We can also see while the "human arms out" class and the "fighter jet" class both show good within class matches, there is also a large number of between class matching, indicating that the Shape D2 descriptor has trouble distinguishing between the two. There is a large number of matches outside the diagonal class boundaries, showing that there is a large amount of confusion between classes and that there is still room for improvement.

Table 5.9 shows the statistics for a few selected classes. There are far too many classes to present them all. The upper section of the table shows the statistics for some classes the the Shape D2 performed well on, and the lower half some classes the Shape D2 performed badly upon. A large difference in performance can be seen between those classes that obtain good retrieval performance and those classes that obtain poor retrieval performance. The higher performing classes are reflected in the Tier Image (Figure 5.7) as densely coloured class squares, where as the poorer classes are reflected by empty class squares.

Descriptor	Distance Metric	Classes
Area Volume	Kullback	Barren
Cord Hist 1	Bhattacharyya	Satellite, Hand, Shelves
Cord Hist 1	Chi	Mailbox
Cord Hist 1	Euclidean	monster_truck
Cord Hist 1	Intersect	Helicopter
Cord Hist 1	Kullback-ns	monster_truck
Cord Hist 1	Quadratic	Staircase
Cord Hist 2	Chi	Stealth_bomber
Cord Hist 2	City-block	Flying_bird, School_desk
Cord Hist 2	Euclidean	Jeep
Cord Hist 2	Intersect	Snake, Wheel, Gear
Cord Hist 2	Kullback-ns	Jeep
Cord Hist 3	Bhattacharyya	One_story_home
Cord Hist 3	Chi	Commercial
Cord Hist 3	City-block	Book
Cord Hist 3	Euclidean	Pail
Cord Hist 3	Intersect	Flying_saucer, Satellite_dish
Cord Hist 3	Kullback-ns	Pail
Cord Hist 3	Quadratic	Pail
Cord hist 4	Chi	Sink
Cord hist 4	City-block	Train_car
Cord Hist 5	Bhattacharyya	Vase
Cord Hist 5	Chi	Human, Fish, Axe, Face, Head
Cord Hist 5	City-block	Rectangular, hat
Cord Hist 5	Kullback	Biplane
Cord Hist 5	Quadratic	Billboard, Race_car
Cord Histogram	Chi	Enterprise_like, Skull, Skyscraper, Ship, Sedan, Semi
Cord Histogram	City-block	Knife, Single_leg
EGI Sphere	Bhattacharyya	Two_story_home
EGI Sphere	Chi	Bench
EGI Sphere	City-block	Church, Slot_machine
EGI Sphere	Kullback-ns	Tie_fighter
EGI Sphere	Quadratic	Ant
EGI-Sphere	Kullback	Hot_air_balloon, Standing_bird
MD2	Bhattacharyya	Chess_set, Human_arms_out, Sea_turtle, Computer_monitor, Door, Cabinet, Ladder
MD2	Chi	Chess_set,Gazebo, Submarine
MD2	City-block	Rabbit, Umbrella
MD2	Intersect	Rabbit, Umbrella
MD2	Kullback	Walking, One_peak_tent, Eyeglasses, Street_light
MD2	Kullback-ns	Butterfly
MD2	Quadratic	Barn
MRG	——–	Dog, Geographic_map, Glass_with_stem, Newtonian_toy, Potted_plant, Conical, Large_sail_boat
Shape D2	Bhattacharyya	Fighter_jet, Glider, Sword, city, Desk_chair, Handgun, Electric_guitar
Shape D2	Chi	Shovel, Desktop, Dining_chair
Shape D2	Intersect	Desktop, Dining_chair
Shape D2	Kullback	Hammer, Horse, Fire_place, Hourglass

TABLE 5.8: PSB Data set: Best descriptor and metric for classes

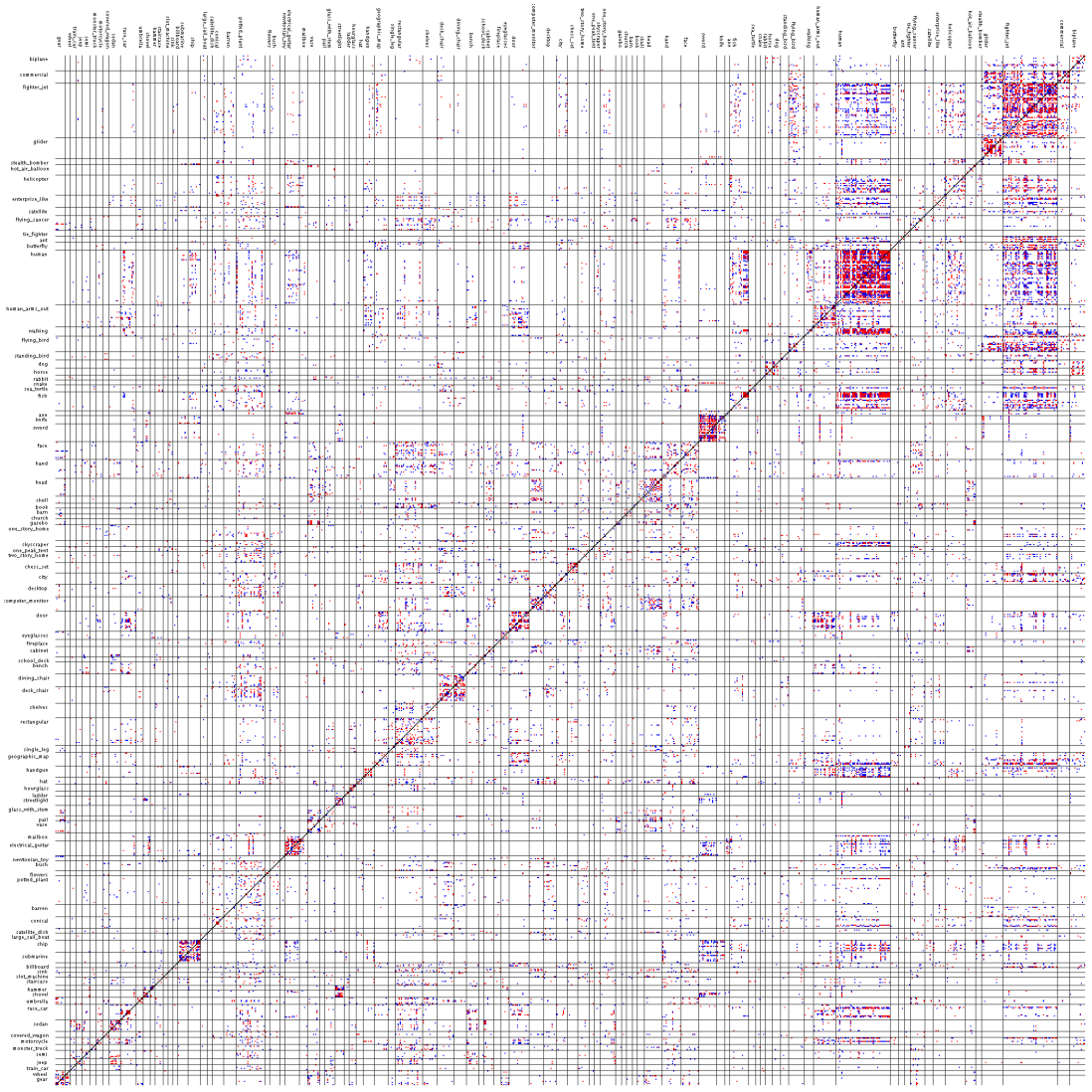


FIGURE 5.7: PSB Data set: Tier Image for Shape D2 using Euclidean Distance

5.5.2 PSB Results Commentary

The overall performance of these descriptors follow a similar ranking to that in Table 5.4. This gives a good indication that the ratio used can be used to predict the relative performance of descriptors without having to perform an exhaustive comparison. However, this is only an overall indication and other techniques still need to be used to analyse class by class performance.

5.5.3 Museum Data set Results

In the second part of this evaluation, the museum data is used to evaluate the performance of the retrieval algorithms. Both Hough methods and the EGI Oct method are

Class	Nearest Neighbour	First Tier	Second Tier	E - Measure	DCG
fighter jet	0.540	0.325	0.488	0.299	0.714
human	0.700	0.405	0.609	0.356	0.752
electrical guitar	0.692	0.404	0.526	0.311	0.700
sword	0.625	0.421	0.596	0.380	0.687
glider	0.842	0.327	0.415	0.280	0.670
human_arms_out	0.450	0.197	0.316	0.208	0.551
horse	0.333	0.200	0.300	0.108	0.414
knife	0.286	0.095	0.333	0.218	0.412
walking	0.000	0.054	0.107	0.058	0.276
tie_fighter	0.400	0.100	0.100	0.022	0.268
satellite	0.000	0.000	0.000	0.000	0.180
rabbit	0.000	0.000	0.000	0.029	0.163
satellite dish	0.000	0.000	0.000	0.000	0.135

TABLE 5.9: PSB Data set: Class Statistics for Shape D2 using Euclidean distance

Descriptor	Ratio
Area Volume	1.4
Cord Hist 1	6.3
Cord Hist 2	7.8
Cord Hist 3	5.5
Cord Hist 4	13.3
Cord Hist 5	23.5
Cord Histogram	19.6
EGI Oct	12.3
EGI Sphere	2.0
Hough Oct	40.7
Hough Sphere	13.1
Shape D2	31.4
MD2	31.4

TABLE 5.10: Museum Data set: Ratio of Between Class and Mean Within Class Variance

available for this data set.

Table 5.10 shows the ratios of the different descriptors histograms. The ratio for the MRG cannot be calculated as it is a variable length feature vector. The Hough Oct descriptor gives the highest ratio. Like the PSB data set, the Area Volume descriptor gives the lowest ratio. The Shape D2 and MD2 descriptors also give the high ratios.

Table 5.11 shows the statistics and Figure 5.8 shows the corresponding precision-recall curves. The MRG descriptor shows a clear improvement over the other descriptors in both the precision-recall curves and statistics. Surprisingly the next best descriptor is the Cord Hist 1, which while it has similar performance to many of the other descriptors for low recall values it manages to keep higher precision for higher recall, coming closer

Descriptor	Nearest Neighbour	First Tier	Second Tier	E-Measure	DCG
Area Volume	0.455	0.385	0.637	0.371	0.718
Cord Hist 1	0.705	0.492	0.656	0.462	0.780
Cord Hist 2	0.632	0.357	0.589	0.347	0.733
Cord Hist 3	0.655	0.398	0.588	0.384	0.734
Cord Hist 4	0.723	0.416	0.614	0.405	0.758
Cord Hist 5	0.718	0.421	0.620	0.407	0.754
Cord Histogram	0.727	0.419	0.616	0.408	0.760
EGI Oct	0.568	0.373	0.566	0.367	0.720
EGI Sphere	0.600	0.341	0.502	0.352	0.711
Hough Oct	0.623	0.361	0.544	0.346	0.711
Hough Sphere	0.345	0.293	0.503	0.271	0.663
MD2	0.800	0.401	0.596	0.395	0.754
Shape D2	0.759	0.400	0.595	0.395	0.754
MRG	0.805	0.534	0.699	0.520	0.818
Random	0.150	0.191	0.387	0.178	0.604

TABLE 5.11: Museum Data set: Descriptor Performance using the Euclidean Distance

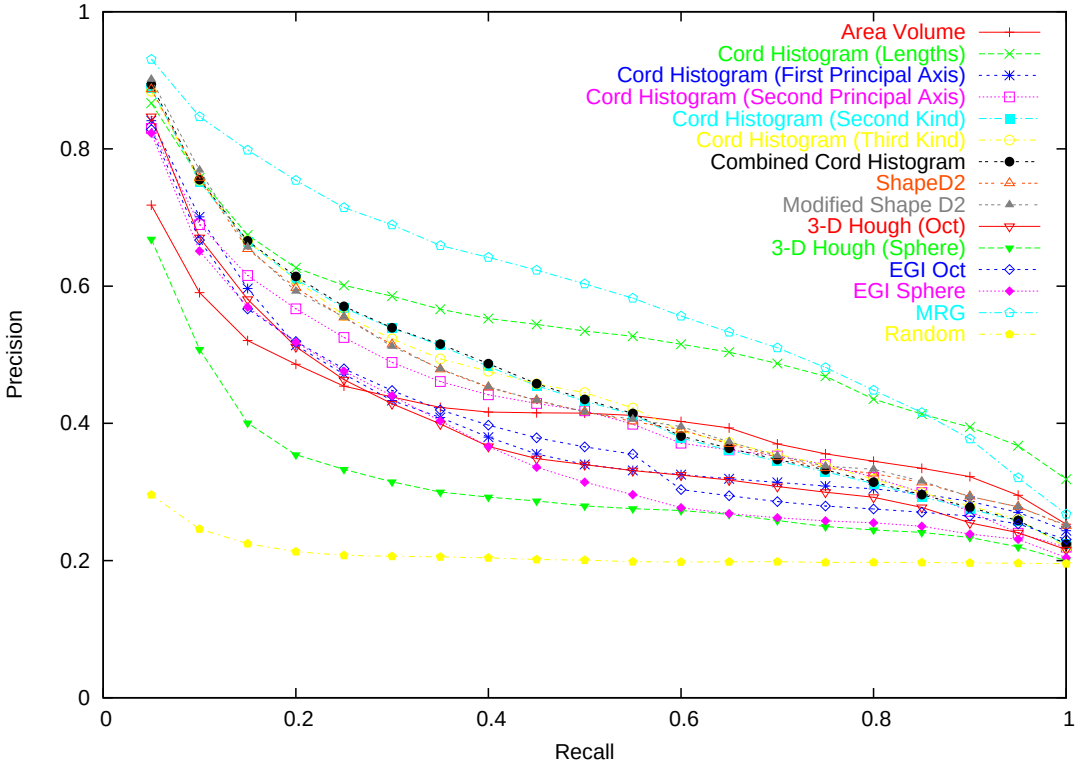


FIGURE 5.8: Museum Data set: Descriptor Comparison using the Euclidean Distance

Distance Metric	Nearest Neighbour	First Tier	Second Tier	E-Measure	DCG
Bhattacharyya	0.454	0.344	0.553	0.322	0.697
Chi	0.670	0.406	0.601	0.392	0.746
City-Block	0.664	0.397	0.590	0.386	0.743
Euclidean	0.655	0.389	0.582	0.378	0.736
Histogram Intersection	0.543	0.356	0.556	0.337	0.710
Kullback-Leibler (Symmetric)	0.551	0.332	0.538	0.316	0.700
Kullback-Leibler (Non-Symmetric)	0.655	0.389	0.582	0.378	0.736
Quadratic	0.656	0.388	0.582	0.377	0.735
Random	0.150	0.191	0.387	0.178	0.604

TABLE 5.12: Museum Data set: Average Distance Metric Performance

to the MRG performance. The worst algorithm is the Hough Sphere descriptor, although it is still much better than random. Also interestingly the Area Volume descriptor starts out on the lower side of the descriptor performance, but it keeps up its precision well as recall increases.

Table 5.12 shows the statistics averaged across all the descriptors (except the MRG descriptor (uses the Euclidean internally) and the Area Volume (only so many ways to compare two numbers)). Figure 5.9 shows the corresponding precision-recall curves. Chi seems to come out on top, closely followed by the City-block metric. The Bhattacharyya gives the worst performance. The Euclidean distance gives intermediate performance.

Table 5.13 shows the best distance metric for each descriptor ranked on DCG score first. Figure 5.10 shows the corresponding precision-recall curves. Again, like for the PSB data set the Chi distance does very well, however the Bhattacharyya distance only appears in a joint best position. Unlike the PSB, the City-block and Histogram Intersect distance appear on several results. Most notably, the Shape D2 and MD2 algorithms have these as the best metrics, whereas for the PSB it was the Chi distance.

Table 5.14 shows the best descriptor and metric(s) for each class based on DCG score. Compared to the variance table displayed earlier (Table 5.10), we can see that only the pot class has the best descriptor that gave the lowest variance. The lack of total correspondence is perhaps due to the different distance metrics altering the effects of the variance of the histograms, thus causing the descriptors to behave differently. As

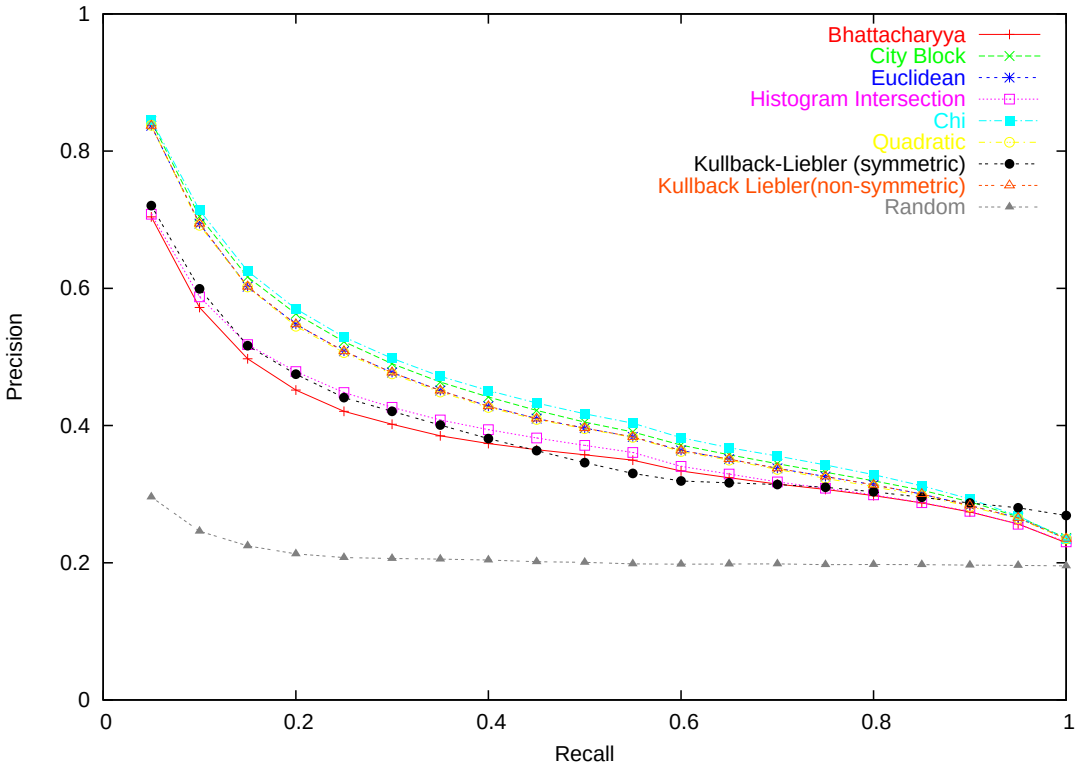


FIGURE 5.9: Museum Data set: Average Distance Metric Performance

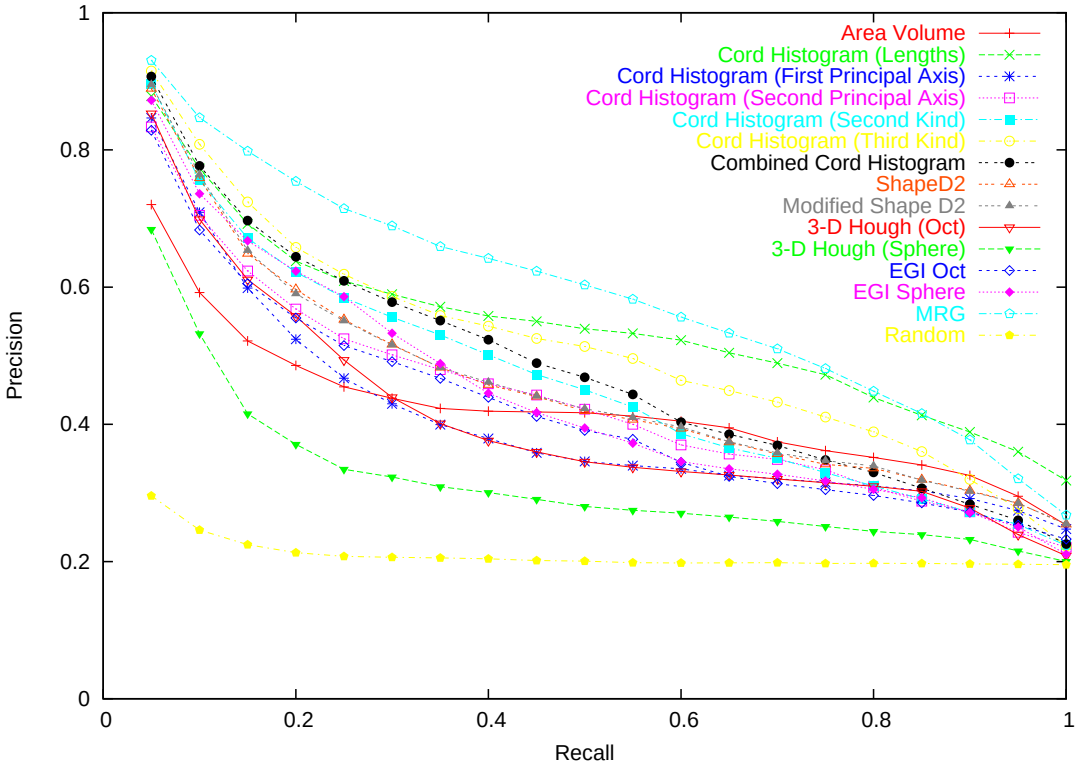


FIGURE 5.10: Museum Data set: Best Distance Metric for Descriptor

Descriptor	Metric	Nearest Neighbour	First Tier	Second Tier	E-Measure	DCG
Area Volume	Bhattacharyya / Chi / Kullback	0.455	0.386	0.638	0.371	0.719
Cord Hist 1	Chi	0.723	0.496	0.671	0.463	0.787
Cord Hist 2	Chi	0.636	0.361	0.593	0.344	0.737
Cord Hist 3	Chi	0.636	0.403	0.592	0.392	0.740
Cord Hist 4	Chi	0.732	0.427	0.619	0.415	0.761
Cord Hist 5	Chi	0.759	0.474	0.668	0.452	0.781
Cord Histogram	Chi	0.750	0.443	0.630	0.430	0.771
EGI Oct	City-Block	0.591	0.392	0.594	0.390	0.732
EGI Sphere	Chi	0.723	0.390	0.595	0.395	0.744
Hough Oct	City-block / Intersect	0.650	0.373	0.554	0.354	0.721
Hough Sphere	Chi	0.395	0.298	0.502	0.281	0.665
MD2	City-block / Intersect	0.764	0.404	0.602	0.397	0.761
Shape D2	City-block / Intersect	0.755	0.403	0.600	0.398	0.759
MRG	N/A	0.805	0.534	0.699	0.520	0.818

TABLE 5.13: Museum Data set: Best Metric for Descriptor - Based on highest DCG

expected, the misc class performs worse due to the diverse range of objects. Also, as expected the tile class performs well as the class consists of very similar objects.

This is also interesting as the MRG only appears once in this list despite being the best overall descriptor. The Hough Oct appears twice in this list despite giving a more average performance overall. In the tile class, we see Hough Oct which ranked tile in the middle in the feature vector variance was actually the best descriptor for it. It scores very high results with the Kullback metric, although all other results with different metrics are much worse.

Figure 5.11 shows an example tier image for the Shape D2 descriptor using the Euclidean distance. It can be seen that the figurine and mould classes showed a correspondence as would be expected as the mould is used to create the figurine.

Table 5.15 shows the statistics for each class. A large difference can be seen between the worst class performance (misc) and the best class performance (tile and mould).

Class	Descriptor	Metric	Nearest Neighbour	First Tier	Second Tier	E - Measure	DCG
Figurine	MRG	Bhattacharyya	0.885	0.578	0.800	0.629	0.897
Head	MD2	Intersect / City-Block	0.625	0.281	0.359	0.133	0.540
Misc	Hough Oct	Bhattacharyya	0.067	0.133	0.173	0.093	0.463
Mould	Cord Hist1	Intersect / City-Block	0.895	0.724	0.960	0.708	0.934
Pot	EGI Sphere	Kullback	0.911	0.625	0.905	0.615	0.880
Statue	Cord Histogram	City-block	0.321	0.279	0.403	0.277	0.633
Tile	Hough Oct	Kullback	1.000	1.000	1.000	0.524	1.000

TABLE 5.14: Museum Data set: Best Metric for Descriptor - Based on highest DCG

Class	Nearest Neighbour	First Tier	Second Tier	E - Measure	DCG
Tile	0.933	0.657	0.905	0.559	0.886
Figurine	0.808	0.463	0.732	0.392	0.818
Mould	0.965	0.530	0.761	0.480	0.886
Statue	0.464	0.167	0.271	0.169	0.569
Head	0.875	0.179	0.268	0.122	0.517
Pot	0.756	0.266	0.439	0.257	0.714
Misc	0.133	0.081	0.138	0.090	0.382

TABLE 5.15: Museum Data set: Class Statistics for Shape D2 using Euclidean distance

5.5.4 Museum Results Commentary

A surprising result is that of the Cord Hist 1 descriptor. It performed very well in many cases, however in Table 5.10 it achieved a low ratio. Of course the ratio does not take into account whether classes overlap each other in feature space, nor does it take into account how many objects are in each class. The use of distance metrics produced much larger differences in performance than was seen in the PSB data set.

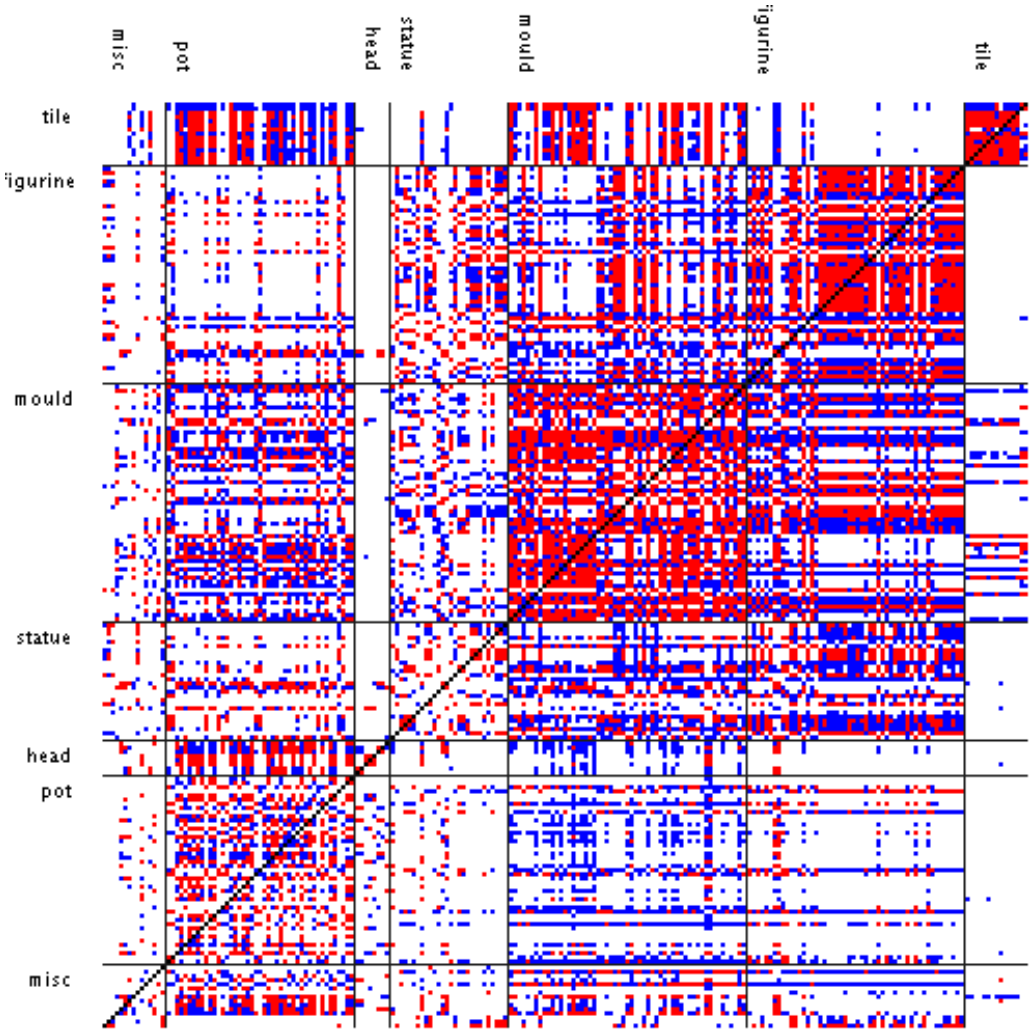


FIGURE 5.11: Museum Data set: Tier Image for Shape D2 using Euclidean Distance

5.6 Conclusions

We have seen that overall the Shape D2 and MD2 descriptors produce the best results, although on a class by class basis, other descriptors can give better performance. The use of alternative distance metrics to the Euclidean distance showed that better performance could be obtained, but again there was no consistently better metric, although there were definitely some poorer choices of metric for some of the descriptors. This implies some prior knowledge is required in order to obtain the best possible retrieval performance for a given query.

The differences between the museum and PSB data sets can be attributed to differences in quality of the models, but also due to the number of models and classes in those data sets. Smaller numbers of classes makes it much easier to correctly retrieve an object of

the same class so it is to be expected that the performance levels would be higher for the museum data set than with the PSB data set.

Overall there is room for improvement over the techniques presented here. Comparing our results to those in the PSB comparison shows that there are other more powerful techniques that could be used to improve retrieval performance. Indeed, the choice of descriptor alters the level of performance much more than changing the distance metric for a descriptor.

The next chapter describes the production of a classifier agent that uses the descriptors generated from the algorithms as inputs to the system.

Chapter 6

3-D Object Classification

6.1 Introduction

In this chapter, popular classification techniques are evaluated for their suitability for use in classifying 3-D objects based upon their 3-D shape descriptors. Classification techniques typically have large numbers of parameters to modify behaviour and resulting performance. However manually determining these parameters is a time consuming task and not one that is desirable to be repeated. Therefore an important aspect of this work is to minimise the amount of user intervention required in creating suitable classifier systems. Typically the more sophisticated the technique, the larger the range of parameters to set.

Chapter 4 described the work on developing a classifier agent to automatically classify unknown objects with the aim of filling in some of the missing metadata for the object. From this work a number of things need to be considered when making decisions about what to use when. Firstly, there are likely to be a large number of classes with few samples per class. Each object can have more than one class label (which may or may not be related to each other). There is potential for new "training data" to appear at regular intervals making existing classifiers obsolete. The metadata can be poor, unstructured and liable to errors (e.g. spelling mistakes in class names).

Two strategies are considered in this work. The first attempts to combine a number of classifiers in order to increase performance. The second attempts to find the optimal parameters to use with a single classifier.

6.2 Related Work

In Chapter 3 a large number of classification techniques were described. In this section, we will briefly re-cover the relevant areas for the work contained in this chapter.

Chapter 4 described the Classifier Agent developed as part of the SCULPTEUR project. This was more focused on user interface design and interaction with the SCULPTEUR system to obtain data rather than how well it actually performed. In this chapter several classification techniques are evaluated as are several methods to improve classification performance over using a single base classifier.

6.2.1 Classification Techniques

A number of classification techniques have been used in this work initially selected mainly because of their popularity in previous work. These are the k-Nearest Neighbour, Multi-Layer Perceptron, Radial-Basis Function Networks and the Support Vector Machine (See e.g. Bishop, 1997b; Haykin, 1999). These have been described in Chapter 3 and some more specific details have been added here where necessary.

6.2.1.1 k-Nearest Neighbour

The k-Nearest Neighbour (k-NN) is one of the simplest classifier techniques to understand and implement. It works by finding the k nearest objects in feature space and assigning a classification based upon the dominant label of those k objects; typically a majority vote is used. A commonly used version of the k-NN is the Nearest Neighbour (NN) classifier where k is equal to one. This classifier is fast to train (just need to store training data as reference feature points). Classification can be computationally expensive however as it is proportional to the number of reference features.

Unlike many other classification techniques, there is no random component to the initial conditions so the same classifier is always produced from the same initial parameters. This is advantageous as re-training will not give better or worse performance.

6.2.1.2 Multi-Layer Perceptron

The Multi-Layer Perceptron (MLP), described in Section 3.3.2, is a powerful and popular classification technique.

The MLP uses a randomly initialised set of weights meaning that each time it is trained, different results will occur (potentially better or worse). The training process should be able to reduce the effects of initial conditions if enough iterations are performed. Training of MLP's is typically achieved using some form of gradient decent function. In the space representing the weight vector and error value, the gradient of the error for a given set of weights is calculated. The weights are then adjusted to a new position in the direction of the lower error. This is applied iteratively until it is not possible to move to a point with lower error. However, this can lead to finding local minima rather than the

global minimum. Several methods attempt to avoid local minima whilst trying to find the global minimum quickly.

Each node in the MLP has a set of weights and a bias. The input vector is multiplied against the weight vector and has the bias added to it to produce an output value. During training, the bias is considered as an extra weight rather than a separate component. In the simplest case, the activation function can return the input value. In this form we have a *linear* activation function. However, non-linear activation functions produce more powerful classifiers. The *logistic sigmoid* activation function uses the exponent of the output value perform the mapping. Another function is the *softmax* function which is a generalisation of the logistic sigmoid function.

For each node, the output, y, is calculated as follows

$$y = g\left(w^T \mathbf{x} + w_0\right)$$

where w is the weight vector, w_0 is the bias and $\mathbf{x}$ is the input vector.

The linear activation function is simply;

$$g\left(a\right) = a$$

The logistic sigmoid activation function is;

$$g\left(a\right) = \frac{1}{1 + exp\left(-a\right)}$$

The softmax activation function is;

$$g\left(a_k\right) = \frac{exp\left(a_k\right)}{\sum_k^K exp\left(a_k\right)}$$

where a_k is the output of node k and K is the number of nodes.

The Hessian Matrix forms an important aspect of several MLP training algorithms. Each element is a second derivative of the error calculated for components of the weight vector. The matrix itself is of size $W \times W$ where W is the size of the weights vector.

There are many training methods and we describe three such methods here. Newton's method makes use of the Hessian Matrix to obtain the gradient at any given point. By taking the inverse of the Hessian matrix the weight vector representing the minimum error can be obtained. However, the calculation of the Hessian matrix is computationally costly and the method to calculate the weight vector of the minimum error is approximated

so requiring an iterative approach. The Quasi-Newton method approximates the inverse Hessian matrix over a number of steps reducing the computational cost.

The conjugate gradients (CG) method attempts to pick the best direction to start travelling immediately. Each step is taken in the direction orthogonal to the gradient until the next location where the gradient is again orthogonal to the search direction.

The scaled conjugate gradient (SCG) method is similar to the conjugate gradient method, but employs a faster function to estimate the Hessian matrix. The scaled part of the name comes from the scaling factor applied to the unit matrix which is added to the Hessian matrix to make sure it is positive definite and can be inverted.

The conjugate gradient methods are more computationally efficient than the Quasi-Newton method allowing higher dimensional problems to be solved at the expense of increased sensitivity to the line search accuracy.

6.2.1.3 Radial Basis Function Networks

The Radial Basis Function (RBF) networks classifier is another popular technique and is described in Section 3.3.3.

Three basis functions are used in the work in this chapter defined as $\phi(r)$. The Gaussian basis function is one of the most commonly used basis functions.

$$\phi(r) = exp\left(\frac{r^2}{2\sigma^2}\right)$$

where σ is the width of the basis function.

The Thin-Plate Spline (tps) is, in one-dimension, a piecewise-linear interpolation function;

$$\phi(r) = r^2 ln(r)$$

The $r^4 \log r$ basis function is;

$$\phi(r) = r^4 log(r)$$

6.2.1.4 Support Vector Machine

The support vector machine (SVM) is a more recent and powerful classification technique. It does not suffer from the curse of dimensionality allowing much more complex problems to be solved than with other techniques. Section 3.3.4 gives more details. The SVM uses

a randomly initialised set of weights meaning that each time it is trained, different results will occur (potentially better or worse). The training process should be able to reduce the effects of initial conditions if enough iterations are performed.

We use three kernel functions in this work, defined as $K(x, x')$ where x and x' are the kernel parameters. The spline kernel;

$$K(x, x') = \sum_{r=0}^{k} x^r x'^r + \sum_{s=1}^{N} (x - \tau_s)^k + (x' - \tau_s)^k$$

where k is the spline order, N is the number of knots located at τ_s.

The Polynomial kernel;

$$K(x, x') = (x^T x + 1)^p$$

where p is the degree of the polynomial.

The Exponential RBF kernel;

$$K(x, x') = exp\left(-\frac{\|x - x'\|^2}{2\sigma^2}\right)$$

Where σ is the width of the basis function.

6.2.2 Improving Performance

A number of techniques have been described in Chapter 3 for improving performance and are used in the work in this chapter. Classifier Ensembles are described in Section 3.6.1 and Dynamic Classifier Selection (DCS) in Section 3.6.1.8. Two optimisation techniques are used in this chapter. Particle Swarm Optimisation is described in Section 3.7.4 and Genetic Algorithms in Section 3.7.3.

6.2.3 Early Experimentation

The initial work in this area, undertaken as part of the classifier agent development (see Chapter 4), applied Support Vector Machines to the Shape D2 and Cord Hist 1 3-D shape descriptors. A binary SVM implementation was used (Gunn, 1997). The initial work created a classifier for each class in the data set. For each classifier, samples in the data set of the current class were labelled with '1' and all other samples were labelled with '-1'. Split-sample was used to train and evaluate each classifier individually. Naively perhaps,

Class Name	Training Size	Test Size
animals	11	10
bike	1	2
chairs	7	7
guns	3	2
helicopter	5	6
objects	10	9
other	11	11
people	16	17
planes	36	36
robot	15	16
shapes	2	2
space	3	3
spaceships	7	6
startrek	8	8
starwars	3	4
vases	14	13
vehicles	11	11
zepplins	3	3

TABLE 6.1: SVM Test Data set

the winning label was selected as the label from the classifier with the highest testing accuracy that gave a positive ('1') prediction. Here, we present the classifier results and of experimenting with more popular methods of obtaining multi-class results from binary classifiers. Here we present the results for using one-versus-all, one-versus-one and DAG-SVM combining methods (Hsu and Lin, 2002).

Table 6.1 shows the class names and sizes in the training and testing partitions. As can be seen there is a range of different classes. Some classes have very few objects (e.g. the class bike) and poorer performance can be expected for the objects in those classes. There are also a number of classes that could be considered very similar to each other (the space classes) however they are kept separate here.

Table 6.2 shows the performance of the SVM for the Shape D2 (upper half) and Cord Hist 1 (lower half) descriptors. The experiments use all three combination methods with a number of different kernels and kernel parameters (value of parameter is shown in parenthesis). It can be seen that the one-versus-all method gives the best performance, whilst the one-versus-one and DAG-SVM give lower, but similar performance. The Shape D2 results for both DAG-SVM and one-versus-one are notable as they are the same regardless of the kernel, however for Cord Hist 1, they are variable.

While Hsu and Lin (2002) suggested that the DAG-SVM and one-versus-one methods would give better results, it can clearly be seen that in this case one-versus-all gives the best results (approximately 30 % instead of approximately 15% accuracy). The data set used is not ideal causing these methods to perform badly. It is possible in the case of

SVM Kernel	One-versus-All	One-versus-One	DAG-SVM
Descriptor: Shape D2			
Spline	38.0%	13.9%	13.3%
Poly (1.0)	34.9%	13.9%	13.3%
Poly (2.0)	36.7%	13.9%	13.3%
RBF (0.5)	39.2%	13.9%	13.3%
RBF (1.0)	36.7%	13.9%	13.3%
RBF (2.0)	34.9%	13.9%	13.3%
Descriptor: Cord Hist 1			
Spline	31.9%	15.6%	15.7%
Poly (1.0)	29.5%	18.1%	16.9%
Poly (2.0)	28.9%	15.1%	14.5%
RBF (0.5)	33.7%	12.7%	13.9%
RBF (1.0)	31.3%	15.7%	15.1%
RBF (2.0)	29.5%	13.3%	13.9%

TABLE 6.2: SVM Results

Class Name	Training	Testing
Mask	3	3
Misc	9	10
Statue	8	7
Tile	16	16
Tool	5	5
Vase	31	31

TABLE 6.3: The PSO Data set

the DAG-SVM, that an alternative ordering of classifiers in the tree would give better results, but the optimal ordering would be different for each example presented.

The SVM work provided an insight into the problems associated with using CBR and classification techniques together. The data set was unbalanced, with some class having many members, and other classes with only a few members. Typically the classifiers performed badly on the classes with few members. Some of the classes were very mixed in terms of shape similarity. Classes with a diverse range of objects generally performed worse than those with more consistently shaped objects.

Our previous work in Goodall et al. (2005b) presented a small classifier system where users could train classifiers and manually choose the training parameters. These results showed that the users were not inclined to change the parameters too far from defaults to try and find optimal results. When applying a PSO augmented with exhaustive search, much higher performing classifiers resulted. However the PSO approach is limited in that it tends towards exhaustive searching due to the types of parameters we are using.

Table 6.3 shows the data set used in this work. It consists of the 3-D objects available from museums at that time. As can be seen it is a small data set with unbalanced classes.

Type	Descriptor	Metric	k	Accuracy
Manual	Area Volume	Euclidean	15	84.7%
Manual	Hough (Oct)	Euclidean	3	89.8%
Manual	Shape D2	Euclidean	15	87.9%
Manual	Cord Hist 1	Euclidean	15	70.3%
Automatic	Area Volume	City-Block	1	97.6%
Automatic	Shape D2	City-Block	1	98.1%
Automatic	Shape D2	Intersection	1	98.1%
Automatic	Cord Hist 1	Quadratic	1	96.8%

TABLE 6.4: PSO k-NN Results

Table 6.4 shows the results of the classifiers created during the SCULPTEUR evaluation (type manual; see Chapter 4) and those created using PSO to determine the parameters (type automatic). As can be seen high accuracy has been obtained, although the small number of objects and classes makes this work more proof-of–concept rather than giving useful results.

6.3 Experimentation

We wish to explore how well the CBR techniques described in Chapter 5 can be used with the classification techniques described in this chapter (See Table 5.1 in Chapter 5 for the full name of the descriptors, the short names are used here). In this work the Area Volume ratio descriptor, Cord Histograms, Shape D2 and Modified Shape D2 descriptors are used. The Multi-resolution Reeb Graph is not appropriate for use in this work as the feature vector can vary in length depending on the complexity of the model it is created from. The classification techniques used here typically require a fixed length input vector. The Extended Gaussian Image and 3-D Hough descriptors have also been left out as their large input vector size dramatically increase the computational time required to perform these classification techniques. Where appropriate, the Euclidean distance metric has been used.

This work will use the PSB data set and its four classification levels (Base, coarse 1, coarse 2 and coarse 3). As the PSB already contains a split into train and test groups, classifiers will be trained according to split-sample validation. Appendix B gives full details of classes and sizes for the PSB classifications.

Of note in the PSB classifications is that classes in the training data do not necessarily appear in the testing data and vice-versa. This means that obtaining 100% is impossible for the base and coarse 1 classifications. The reason for this according to Shilane et al. (2004) is to allow evaluation of the classifier when new classes are presented to it. Table 6.6 shows the highest accuracy that can be achieved. The Exact row shows the percentage of objects in the test set that have a corresponding exact class label in the

Class name	Training Size	Testing size
figurine	35	17
head	6	2
misc	10	5
mould	38	19
pot	30	15
statue	18	10
tile	10	5

TABLE 6.5: Museum data set

	Base Classification	Coarse 1 Classification	Coarse 2 Classification	Coarse 3 Classification
Exact	40.8%	91.4%	100%	100%
Hierarchy	90.3%	94.7%	100%	100%

TABLE 6.6: Maximum accuracy achievable in PSB classifications

training set. The Hierarchy row shows the percentage of objects in the test set that have a corresponding class label in their class hierarchy in the training set. As can be seen, the base exact percentage is the only one considerably effected by this with a maximum achievable accuracy of 40.8%.

In this work, a second smaller data set composed of 3-D models of museum artifacts is used for the more computationally expensive techniques. However, full validation of the technique would require a larger data set. To compare with the PSB data set, we have split the museum data set into a training and testing set. Due to the smaller size of the data set, we have used two thirds of the data in the training set and the remaining third in the testing set. Some of the classes are very small. See Table 6.5 for class size details. See Section 5.4.1 in Chapter 5 for a description of this data set.

Our testing platform is MATLAB (The MathWorks Inc., No Year); a powerful package for mathematical processing. Additionally, a toolbox called NetLab (Nabney, 2002) provides a large range of classification techniques based on Bishop (1997b). This allows rapid prototyping and testing of various classification techniques.

Three popular classification techniques are used within this work. These are the NN, MLP and RBF techniques. The NN is often used due to its simplicity making its results easy to interpret. The MLP is the typical choice in neural network literature. The RBF is a technique that's growing in popularity as it is much faster to train than the MLP networks while achieving comparable results. We chose not to use the SVM in this work.

We begin by presenting the *base* classifiers, that is, a single classifier of one of these techniques. We train a base classifier for each descriptor for each technique. These classifiers form both a baseline in performance for comparing other techniques against and they are the classifiers used by the combination techniques we will describe shortly.

The accuracy statistic has been used to determine how well a classifier performs. In this work the accuracy is given for both the exact label correctly predicted and for predicting a label that has a match within the class hierarchy. I.e. if the classifier predicted fighter jet and the actual object was a bi-plane, then it would score correctly as both objects are of the super-class air plane. It is also possible to apply this to obtain a per class accuracy.

In order to determine how much improvement in performance there is (or not) by using a particular combination technique, the base classifiers have been trained on parameters selected arbitrarily and checked to see that they give acceptable levels of performance. However, we have not explicitly tried to find the best parameters. This is to simulate a normal user trying to make a classifier without too much experimentation. In the nearest neighbour classifier we used the Euclidean distance metric. In the MLP classifiers, two hidden layers, a softmax activation function and the quasi-newton training method were used. In the RBF classifiers two hidden layers and the Gaussian activation function were used.

Due to the complexity of the data sets (multiple classes, wide range of different objects within a class) it is unlikely that a single classifier will be able to capture all the differences within the data set. A combination of classifiers is much more likely to be able to do much better as one classifier can make up for the weakness in another classifier.

Classifiers are combined using the popular classifier ensemble technique called Majority Vote and by using the Dynamic Classifier Selection (DCS) framework. While there are other popular classifier ensemble rules such as the sum and product rule, they are generally more suited to two class problems rather than multi class problems. The a priori and a posterori rules were used in the DCS framework. Experimentation with a neighbour size of 1 to 50 is performed and the highest accuracy obtained is presented. A separate comparison of neighbourhood size versus accuracy is given.

Finally an oracle is used to show the optimal results of combining the classifiers. If at least one of the classifiers makes a correct prediction for a given object, then that counts as a correct prediction for the oracle when calculating accuracy.

Classifiers are combined first by like type and then all classifiers are combined together. Ten instances of each of the MLP and RBF base classifiers (one for each set of parameters) were created the average performance is reported. The classifiers achieving the highest performance were used in the combination methods. This was to minimise the effects of the random weight initialisation for these techniques.

Parameter	Values
Descriptor	Area Volume, Cord Hist {1,2,3,4,5}, Cord Histogram, Shape D2, MD2
Metrics	Euclidean, City-Block, Intersect, Bhattacharyya, Bhattacharyya-log, Chi, Kullback, Kullback-ns, Quadratic,
k	1 to smallest class size

FIGURE 6.1: Nearest Neighbour parameters for GA

Parameter	Values
Descriptor	Area Volume, Cord Hist {1,2,3,4,5}, Cord Histogram, Shape D2, MD2
Hidden Nodes	1 to 16
Activation Functions	linear, logistic, softmax
Training Methods	conjugate-gradient(conjgrad), quasi newton (quasinew), scaled conjugate gradients(scg)

FIGURE 6.2: Multi-Layer Perceptron parameters for GA

Parameter	Values
Descriptor	Area Volume, Cord Hist {1,2,3,4,5}, Cord Histogram, Shape D2, MD2
Hidden Nodes	1 to 16
Activation Functions	Gaussian, tps, r4logr

FIGURE 6.3: Radial Basis Function Network parameters for GA

6.3.1 Optimisation

As an alternative to combining classifiers to improve performance, an attempt can be made to optimise the training parameters for a classifier to obtain the best possible performance out of it. Our earlier experimentation (Goodall et al., 2005b) used the Particle Swarm Optimisation coupled with exhaustive search to optimise classifier parameters. Exhaustive search was required to iterate through labelled parameters such as distance metric as these could not be encoded in the PSO directly. While improved performance was obtained, the use of exhaustive search dramatically increased computation time limiting the use of the technique. In this work, Genetic Algorithms (GA) are used to perform the optimisation as they can encode all the training parameters. We begin by using the GA to optimise the parameters for each classification technique, NN (see Table 6.1 for parameters), MLP (see Table 6.2 for parameters) and RBF (see Table 6.3 for parameters). The GA encode the descriptor in each case and the classification technique specific parameters.

As the number of values for training parameter does not necessarily require the full range of number provided by the bit range (e.g. the number 5 and 8 both require the same number of bits to represent them), we wrap the extra values back to the start of the

range. (e.g. if the bit string represents 8, but 5 was the maximum value, then we wrap the number at 5 which gives us 3 minus 8 modulo 5) This does however mean that values listed earlier in the parameter range have a higher chance of being selected for the initial population. With a large enough population this should not pose a problem.

For each possible solution, ten classifiers are constructed and the average accuracy is returned as the utility. This is to reduce the effect of the random weights initialisation in the MLP and RBF classifiers.

In the GA a population size of 20 was used with a maximum number of 50 iterations for the PSB data set, and for the Museum data set a population size of 50 was used. Cross-over was applied to replace the lower 50% of the population and a 0.1% chance of mutation for a bit in the chromosome string. The GA terminated when the population converges (the same set of parameters was specified by all individuals), or when the maximum number of training iterations had been reached.

Our initial experimentation with the GA used the test set to evaluate each individual solution. However, this makes the GA biased towards the test set and so the performance can be artificially higher then it should be. For the Museum data set we also created some un-biased classifiers. In this case we randomly selected 25% of the training data to validate the solution and used the rest to train the solution.

The previous results looked at methods of combining classifiers to improve performance. This section looks at improving individual classifier performance by attempting to determine the optimal training parameters. The Genetic Algorithms technique is used to encode the training parameters for the classifiers. In this case, the GA encodes the distance metric and the number of neighbours (k). The descriptor was hard coded to find the best parameters for the descriptor.

6.4 Results

The results sections begins by presenting the performance of the base classifiers along with the ensemble and DCS based performance.

6.4.1 Base Classifier Performance

Table 6.7 shows the performance for the Nearest Neighbour classifier. As expected, exact label matching is less accurate than hierarchical matching, and the higher the number of classes, the lower the overall performance. While generally low accuracy, it is still above the level of random classification (1 / number of classes). It can be seen that the Area Volume and Cord Hist 1 descriptors generally perform much worse than the other descriptors. The Shape D2 and MD2 give the best levels of performance, with the MD2

Descriptor	Base		Coarse 1		Coarse 2	Coarse 3	Museum
	Exact	Hierarchy	Exact	Hierarchy			
Area Volume	3.6%	8.6%	9.8%	12.1%	22.5%	56.2%	47.9%
Cord Hist 1	6.8%	12.5%	13.8%	17.4%	26.9%	52.1%	65.7%
Cord Hist 2	13.3%	21.7%	22.5%	25.8%	35.0%	58.8%	64.3%
Cord Hist 3	12.9%	21.2%	20.8%	24.9%	32.0%	60.4%	53.4%
Cord Hist 4	16.3%	25.8%	25.4%	29.4%	38.8%	60.3%	71.2%
Cord Hist 5	17.1%	26.2%	25.1%	29.3%	38.5%	61.4%	67.1%
Combined Cord Histogram	16.3%	26.1%	26.0%	29.7%	38.6%	58.9%	69.9%
Shape D2	17.5%	27.0%	27.5%	30.4%	38.8%	61.1%	72.6%
Modified Shape D2	17.2%	27.7%	28.2%	31.2%	39.5%	62.5%	78.1%
Oracle	29.9%	53.1%	54.0%	62.2%	82.4%	98.2%	93.2%
Majority Vote	1.7%	29.9%	3.6%	29.3%	41.6%	69.0%	76.7%
Dynamic Classifier Selection (a priori)	17.6%	28.3%	21.8%	32.1%	40.2%	71.8%	78.1%
Dynamic Classifier Selection (a posterori)	20.9%	31.5%	26.0%	36.3%	45.8%	77.8%	74.0%

TABLE 6.7: Nearest Neighbour Classifier Accuracy

giving over 5% extra for the Museum data set. It is also interesting to note that the Cord Hist 1 descriptor performs relatively much better on the museum data set when compared to the different PSB data sets.

The oracle shows around 30% accuracy for the base exact matching which means that the fine granularity of classes proves too complex for the capabilities of the base classifiers. At the other end of the scale, it gets 98.2% accuracy for the coarse 3 classification. Majority Vote struggles to perform well on the exact matching, giving results much lower than the base classifiers its composed of. Majority Vote gives better performance on the coarse 2 and coarse 3 classifications giving better results than its base classifiers. The a priori confidence estimate for DCS performs similarly to Majority Vote, but it also manages to work well on the exact label matching. For the museum and coarse 1 exact data sets it does not manage to do better than its base classifiers, but in all other cases it does. The a posterori confidence estimate for DCS shows the best combination results, out-performing both Majority Vote and a priori (except for the museum data set where a priori performs best). For coarse 1 exact, none of the combination methods do better than the base classifiers.

Table 6.8 shows the performance for MLP classifiers trained on the data. The upper

Descriptor	Base		Coarse 1		Coarse 2	Coarse 3	Museum
	Exact	Hierarchy	Exact	Hierarchy			
Average Accuracy							
Area Volume	9.2%	16.2%	20.9%	26.5%	31.6%	73.1%	55.1%
Cord Hist 1	7.4%	13.3%	18.2%	21.8%	27.9%	66.8%	64.4%
Cord Hist 2	10.7%	18.1%	19.5%	23.3%	31.8%	68.3%	53.0%
Cord Hist 3	10.8%	19.0%	20.9%	26.0%	28.4%	64.3%	53.6%
Cord Hist 4	9.4%	16.4%	17.8%	20.7%	32.8%	60.4%	57.0%
Cord Hist 5	8.5%	15.2%	15.3%	18.1%	27.9%	59.2%	56.0%
Combined Cord Histogram	9.8%	17.0%	20.3%	23.9%	31.6%	63.9%	56.2%
Shape D2	12.9%	21.1%	22.9%	25.5%	32.1%	65.4%	59.0%
Modified Shape D2	13.5%	22.9%	23.4%	25.9%	33.0%	66.6%	59.7%
Best Accuracy							
Area Volume	9.9%	17.3%	21.6%	27.0%	33.7%	74.4%	57.5%
Cord Hist 1	8.5%	14.4%	19.5%	23.5%	29.1%	67.5%	68.5%
Cord Hist 2	12.1%	19.5%	20.9%	24.7%	33.7%	69.7%	57.5%
Cord Hist 3	11.9%	20.2%	22.3%	27.5%	31.0%	67.6%	57.5%
Cord Hist 4	12.5%	22.7%	21.9%	26.5%	34.8%	63.7%	63.0%
Cord Hist 5	9.4%	17.5%	17.5%	20.1%	30.5%	65.9%	60.3%
Combined Cord Histogram	12.7%	21.9%	23.2%	28.4%	34.2%	71.1%	67.1%
Shape D2	14.9%	25.7%	25.0%	27.2%	33.3%	66.3%	61.6%
MD2	14.1%	23.0%	25.1%	27.8%	34.8%	69.5%	63.0%
Combined Classifiers							
Oracle	23.8%	46.2%	45.5%	59.6%	72.5%	95.3%	87.7%
Majority Vote	0.9%	25.4%	2.4%	26.7%	37.8%	70.6%	68.5%
Dynamic Classifier Selection (a priori)	12.0%	23.7%	20.5%	30.3%	38.5%	75.4%	74.0%
Dynamic Classifier Selection (a posterori)	14.1%	26.4%	22.9%	34.2%	42.2%	77.8%	74.0%

TABLE 6.8: MLP Classifier Accuracy

portion of the table shows the average accuracy for ten classifiers trained on the same parameters and the middle portion shows the accuracy of the best of those ten classifiers. We can see that there is a large difference between the average and best classifiers, in some cases nearly 10%. This indicates that the random weights have a large effect on the classifier and a larger number of training iterations may be helpful. Again we can see that as expected, the results for exact matching are much poorer than for hierarchical matching. The Area Volume ratio does quite well with the MLP classifiers and even shows the highest accuracy on the coarse 3 classification. In fact, in addition to the Area Volume, the Cord Hist 1 and Cord Hist 2 descriptors also give much better results in the coarse 3 than the other descriptors. The Area Volume and Cord Hist 2 descriptors are generally at the higher end for all the classifications. This is notable as the results in Table 6.7 show these descriptors as poor performers for the Nearest Neighbour classifier. Another point to note is that the Cord Hist 4 based classifiers are generally much lower performing than the other classifiers.

The oracle for the MLP shows poorer results than for the NN, indicating that these classifiers make much more similar errors than the NN does. This is reflected in the even poorer results for Majority Vote with only 0.9% accuracy for the base exact classification. The DCS results do not appear to suffer too much, although in several cases the base classifiers show better results.

Table 6.9 shows the performance of the RBF classifiers. The upper portion of the table shows the average accuracy for ten classifiers trained on the same parameters and the middle portion shows the accuracy of the best of those ten classifiers. We can see there is little difference between the average and best classifiers (only a couple of percent) indicating that the training process is good at reducing the effects of the random weights. It is immediately obvious that these classifiers generally perform badly compared to the NN and MLP classifiers, although it's coarse 3 results are generally much higher.

It can be seen that most descriptors give similar performance within the same data set, although the Area Volume and Cord Hist 1 descriptors do perform worse for the base and coarse 1 data sets. The museum data set shows the biggest variation between descriptors, with the Cord Hist 1 giving the highest performance. It can be seen that the a posterori combination method performs best in all cases, although it shows lower performance than some of the base classifiers for coarse 1 exact.

Table 6.10 shows the performance of combining all of the base classifiers. We can see that the performance is generally much higher when combining multiple types of classifier rather than combining the classifiers of the same type. The oracle shows nearly 100% accuracy for the coarse 3 classification. The Majority Vote did improve in some cases, for example, the NN Museum data set MV accuracy is higher. Despite these improvements, Majority Vote is still worse than the base classifiers is almost all cases. The DCS based classifiers did however show an improvement in performance in almost all cases. The a

Descriptor	Base		Coarse 1		Coarse 2	Coarse 3	Museum
	Exact	Hierarchy	Exact	Hierarchy			
Average Accuracy							
Area Volume	5.3%	9.5%	14.2%	17.0%	25.0%	70.1%	52.1%
Cord Hist 1	5.2%	8.8%	12.5%	15.9%	25.9%	69.6%	62.9%
Cord Hist 2	8.6%	14.9%	19.3%	22.6%	27.7%	68.9%	50.4%
Cord Hist 3	8.3%	14.2%	20.0%	24.2%	24.8%	69.8%	49.0%
Cord Hist 4	9.0%	15.4%	20.0%	23.9%	29.2%	70.9%	42.2%
Cord Hist 5	8.0%	13.8%	18.3%	22.2%	28.7%	69.6%	46.3%
Combined Cord Histogram	8.7%	14.9%	19.7%	23.7%	29.5%	70.8%	42.9%
Shape D2	6.2%	11.3%	13.8%	16.6%	24.4%	68.9%	57.1%
Modified Shape D2	6.1%	11.0%	12.8%	15.4%	24.3%	68.9%	57.1%
Best Accuracy							
Area Volume	6.6%	12.5%	15.3%	17.8%	28.0%	70.3%	54.8%
Cord Hist 1	6.0%	9.7%	13.5%	16.7%	27.3%	69.9%	64.4%
Cord Hist 2	9.7%	15.8%	20.6%	24.1%	28.4%	69.5%	52.1%
Cord Hist 3	9.3%	15.3%	21.1%	25.5%	25.6%	70.2%	52.1%
Cord Hist 4	9.7%	16.4%	20.9%	24.9%	31.0%	72.0%	43.8%
Cord Hist 5	8.9%	14.0%	19.6%	24.0%	29.7%	70.2%	47.9%
Combined Cord Histogram	9.7%	14.8%	20.7%	25.0%	32.2%	71.4%	46.6%
Shape D2	7.3%	12.3%	15.0%	17.4%	26.0%	68.9%	58.9%
Modified Shape D2	7.3%	11.9%	15.0%	18.1%	25.5%	68.9%	60.3%
Combined Classifiers							
Oracle	20.8%	38.3%	41.9%	53.5%	62.1%	81.5%	78.1%
Majority Vote	0.2%	15.0%	1.5%	20.5%	28.6%	69.7%	60.3%
Dynamic Classifier Selection (a priori)	8.3%	14.3%	13.2%	24.1%	32.6%	71.4%	65.8%
Dynamic Classifier Selection (a posterori)	11.5%	21.5%	18.0%	29.0%	37.4%	73.5%	68.5%

TABLE 6.9: RBF Classifier Accuracy

Descriptor	Base		Coarse 1		Coarse 2	Coarse 3	Museum
	Exact	Hierarchy	Exact	Hierarchy			
Oracle	34.5%	64.4%	64.8%	76.7%	90.3%	99.6%	97.3%
Majority Vote	2.4%	28.8%	1.9%	27.2%	40.7%	70.7%	75.3%
Dynamic Classifier Selection (a priori)	18.2%	28.4%	25.1%	33.6%	42.6%	77.6%	76.7%
Dynamic Classifier Selection (a posterori)	22.2%	33.4%	29.7%	38.8%	48.4%	78.7%	74.0%

TABLE 6.10: Combination of all classifiers accuracy

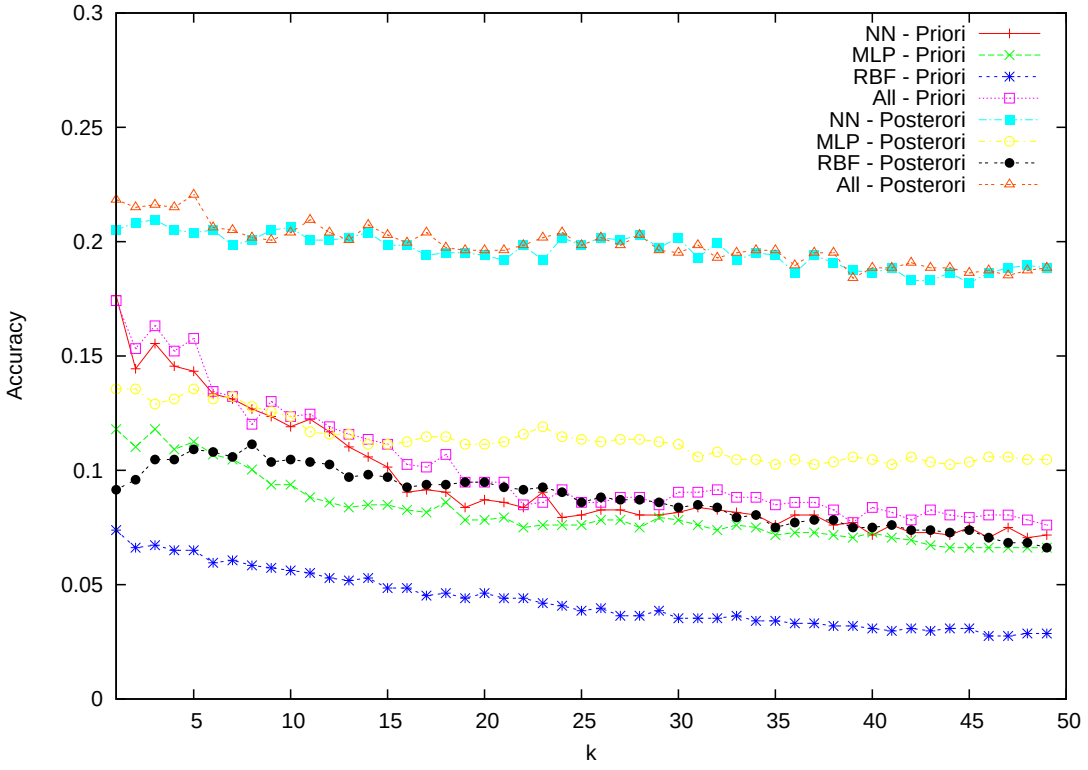


FIGURE 6.4: Neighbourhood size versus Accuracy

priori for the museum showed slightly lower accuracy than for the equivalent NN based classifier.

6.4.1.1 DCS: k versus Accuracy Evaluation

Figure 6.4 shows how the neighbourhood size, k, effects the overall accuracy for the DCS classifiers when considering the base classification and exact label matching. Note that the accuracy axis only goes up to 0.3, or 30%, A general trend is that as k increases, accuracy tends to decrease. The RBF posterori classifier increases as k increases until

	Descriptor	Metric	k	Exact	Hier
Rand-1	Shape D2	Kullback-ns	2	16.7%	26.9%
Rand-2	Cord Hist 3	Intersect	3	3.2%	6.1%
Rand-3	Cord Hist 4	Kullback	3	15.7%	25.1%
Rand-4	Cord Hist 1	Kullback	4	6.7%	13.2%
Rand-5	Cord Hist 4	Intersect	2	3.0%	5.3%
Rand-6	Shape D2	Kullback-ns	4	18.0%	29.7%
Rand-7	Cord Histogram	Quadratic	1	0.0%	2.6%
Rand-8	MD2	Quadratic	3	1.0%	4.2%
Rand-9	Cord Hist 5	Chi	4	16.8%	27.5%
Rand-10	Area Volume	Kullback-ns	4	5.4%	8.8%
Rand-11	Area Volume	Intersect	3	1.4%	11.1%
Rand-12	Cord Hist 4	Kullback	2	14.0%	24.1%
Rand-13	Cord Hist 3	intersect	2	3.0%	6.6%
Rand-14	Shape D2	Kullback	2	15.7%	26.2%
Rand-15	Cord Hist 1	Kullback-ns	4	7.2%	12.6%
Rand-16	CordHistogam	euclidean	1	16.3%	26.1%
Rand-17	Cord Hist 4	quadratic	3	0.1%	0.6%
Rand-18	MD2	quadratic	4	0.8%	4.9%
Rand-19	MD2	Bhattacharyya-log	1	18.0%	29.0%
Rand-20	Cord Hist 5	quadratic	2	0.6%	1.4%
Oracle				32.0%	60.8%
Majority Vote				3.3%	31.6%
a priori				17.5%	28.9%
a posterori				22.6%	35.2%

TABLE 6.11: Random - PSB Base - k-NN

little over k=5 where it starts to decrease again. Similar results were observed for other PSB classifications.

6.4.1.2 Random classifiers

Table 6.11 show 20 k-NN classifiers trained on the PSB Base data set and created with randomly selected parameters for descriptor, metric and k (limited to the range 1 : smallest class size approximately 4).

The randomly created classifiers show a large range of performance values. Very poor performance can be seen in some of the classifiers (0% in some cases) to those gaining high performance, similar to the best achieved in the NN base classifiers. The combination of classifiers gives slightly better results with the a posterori showing the greatest improvement in performance (approx 5% for both exact and hierarchical matching). This gives some good indication that with a reasonably sized sample, we could combine classifiers created with random parameters and obtain good results.

Descriptor	Metric	K	Exact Accuracy
Area Volume	Euclidean	22	10.4%
Cord Hist 1	Kullback-ns	25	16.5%
Cord Hist 2	Chi	4	22.8%
Cord Hist 3	Cityblock	1	25.3%
Cord Hist 4	Chi	1	29.3%
Cord Hist 5	Chi	1	30.9%
Cord Histogram	Chi	2	29.6%
Shape D2	Kullback-ns	1	25.7%
Modified Shape D2	Kullback-ns	1	29.9%

TABLE 6.12: (PSB) GA Results for k-NN

Classifier Type	Descriptor	Parameters	Accuracy	Iterations
NN	Cord Hist 4	Metric: City-Block K: 3	82.2%	31
MLP	Cord Histogram	Train: scg node: linear nhidden: 14	75.9%	50
RBF	Cord Hist 1	Func: Gaussian NHidden: 3	65.6%	35

TABLE 6.13: GA Results (Biased) - Museum Data set

6.4.2 Optimisation Techniques

In this section of results, Genetic Algorithms are applied to the problem of finding optimal training parameters for the classification techniques. Due to the computational requirements of the GA technique, the results are limited to using the PSB Base set for training a classifier for a particular descriptor and to the museum data set to find the optimal classifier for each classifier type.

Table 6.12 shows the results for using GA to select the optimal parameters for a k-NN classifier trained on the PSB base data set. Thees classifiers are created by using the test set to guide the GA and so are biased towards the test data set and can show artificially high accuracy. Comparing these results to Table 6.7 we can see that different parameters can be selected to greatly improve performance by adjusting the number of neighbours and the distance metric. We can also compare these scores to Table 6.10 where in most cases these classifiers beat the combination methods, but do not quite get as high as an optimal combination.

Table 6.13 and Table 6.14 show the results for applying the GA technique to the museum data set. Table 6.13 uses the test data set to guide the GA algorithm as to what is a good solution and what is a bad solution. This however makes it biased towards the test data set. Comparing to Tables 6.7, 6.8 and 6.9, we can see that the GA NN achieved best results, where as for MLP and RBF, the combined classifiers gave better results. This

Classifier Type	Descriptor	Parameters	Validation	Test	Iterations
NN	Cord Hist 4	Metric: Chi K: 1	74.3%	65.8%	42
MLP	Cord Histogram	Train: quaisnew node: softmax nhidden: 16	75.1%	72.6%	50
RBF	Cord Hist 1	Func: Gaussian NHidden: 3	60.3%	65.8%	50

TABLE 6.14: GA Results (Unbiased) - Museum Data set

suggests that the MLP and RBF classifiers would not show much increase in performance by further training. The NN classifier on the other hand show a dramatic increase in performance when k and the distance metric are altered.

Table 6.14 shows the results for using part of the training set to validate the solution. This means that the final classifier is not biased towards the test data set, however it does reduce the amount of data available to both train and validate the classifiers. Comparing to Tables 6.7, 6.8 and 6.9, we can see that the the combined classifiers gave better results. The MLP classifier gives the best results, with 72.6% accuracy, where as the NN and RBF both achieved 65.8%.

As can be see by comparing the two tables, the resulting descriptor is the same for each technique in both tables, however the technique's parameters have changed. It can also be seen that the biased results are higher than the unbiased ones (as would be expected). In fact the unbiased results are poorer than some base classifiers. Interestingly the RBF classifier results in the same classifier, although it takes longer to get there. In both cases the MLP took 50 training iterations, as did the RBF in the unbiased case. This means that they reached the maximum number of training iterations before the population converged. One reason for this could be that the value of one or more of the training parameters had little effect on the overall outcome of the classifier and so many different values exist in the population for that parameter.

The difference between the two tables shows that the NN classifier is a lot less able to generalise than the other techniques. The RBF classifier produced was the same in both instances suggesting that it would work just as well on another "test" data set where as the NN classifier could have drastically different results. This is likely to be due to the MLP and RBF classifiers being better able to model the class boundaries than the NN so objects that are not so near a neighbour will still be correctly classified.

6.5 Discussion

So why do the classifiers generally perform so badly on the PSB classifications, especially majority vote? As shown in Table 6.6 the highest PSB base-exact accuracy is 40%, but

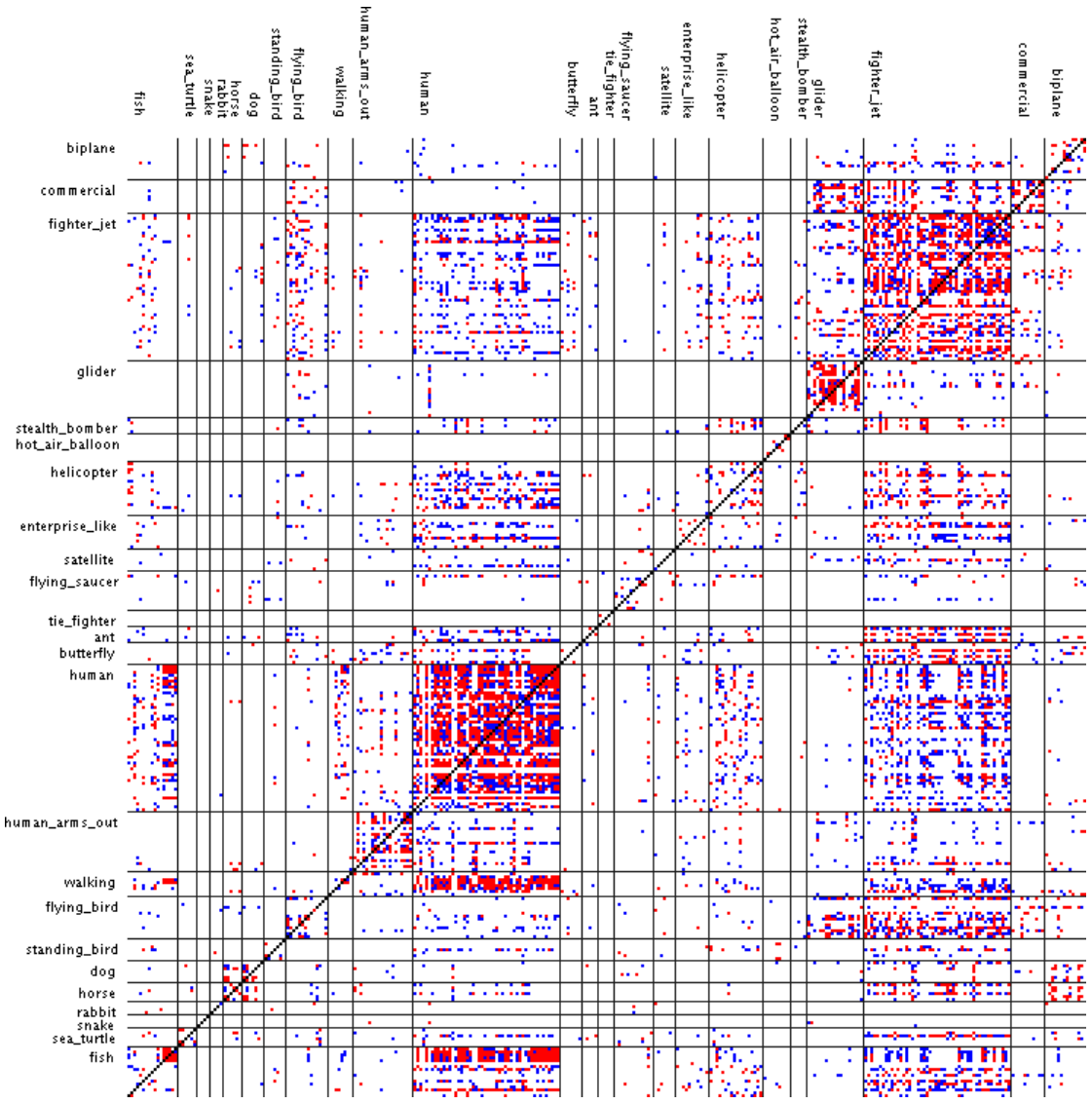
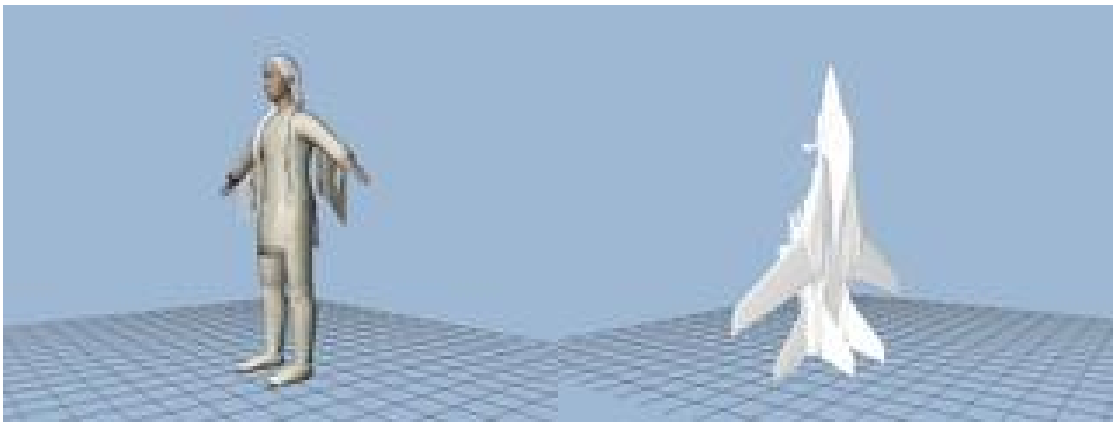


FIGURE 6.5: Tier Image for Shape D2

for the rest, much higher accuracy is possible. We shall turn to content-based retrieval performance metrics to help explain the results. The PSB proposed several statistics for use with 3-D CBR (Shilane et al., 2004). Of particular use in this instance is the tier image. This shows the objects that were the nearest neighbour (hopefully itself) and those within the first and second tier bands. The first and second tier criteria are the percentage of the first K elements in a ranked list from a retrieval that are of the same class as the query, where K, for the first tier, is the size of the class. The second tier uses K as twice the size of the class. More specifically for a class C, $K = |C| - 1$ for the first tier and $K = 2 * (|C| - 1)$ for the second tier where $|C|$ is the size of class C (-1 to ignore the query object). The tier image shows all objects that fall within these bands.

Figure 6.5 shows a subsection of the tier image for the Shape D2 descriptor on the test data set for the base classification. Ideally we expect all the pixels to be arranged along the diagonal such that the box bounding each class is full. However as can be seen, many



(a) Human (b) Fighter Jet

FIGURE 6.6: Two objects conceptually different, but similarly shaped

pixels are spread across the image showing that the Shape D2 cannot really distinguish between the classes very well. Of particular note is that the human and fighter jet classes show a strong correspondence meaning that many fighter jets will be classed as human and many humans will be classed as fighter jets. From this image it is easy to see why the classifiers were struggling with the base classification. The 3-D shape descriptors are not capable of discriminating between the full semantics implied by the class labels.

Figure 6.6 helps to illustrate this point. It shows a model of a human with the arms sticking out and it also shows a fighter jet. Notice how similar, in terms of the overall shape, the two models are. The descriptors we used in this work are all global descriptors and are unlikely to have picked up on the finer details of the models. The idea behind combining classifiers based on a range of descriptors is that between them they should be able to pick up on a lot of the finer details. One problem of course is that the descriptors are generally quite similar to each other. A more diverse range of descriptors would be expected to produce a more diverse set of classifiers. This could perhaps explain why the Majority Vote performed so badly. It is also likely that the large number of classes resulted in each classifier predicting a different label giving no majority. Increasing the number of classifiers should eventually start giving a consensus on the predicted label. In the worst case, you would need as many classifiers as you had classes plus one to ensure a majority however slim. However, as the number of classifiers in an ensemble increases, so do the computational requirements. The *test and select* methodology (Roli et al., 2001) may help by finding the best combination of base classifiers although this can be computationally expensive depending on the number of base classifiers involved. An alternative is to make use of the error diversity measures, (CD, GD and Q statistics; See Chapter 3 for more details) to discard classifiers that are very similar to another classifier. Of course can lead to a sub-optimal solution as two classifiers can be very similar, but the differences they do make could be significant.

The DCS based approach shows much better results, especially for the a posterori estimation method compared to Majority Vote. While DCS has been proposed as an alternative to ensembles as it does not require classifiers to be error diverse, it does help for the classifiers to have this property. Indeed, our results showed the best performance for the DCS techniques was when all the classifiers were combined. In some cases, however, the DCS techniques did not manage to improve performance over the base classifiers. We expect that this was caused by no classifier having a confidence much greater than the others leading to the random selection of several classifiers which may or may not predict the correct label. Perhaps a smaller threshold value would increase the overall accuracy of the DCS classifiers.

The base classifier results show that the Nearest Neighbour gives the strongest performance despite being the least sophisticated technique. This gives a good indication that this data set gives an *ill-posed* classification problem and it is likely that poor performance can be expected. The RBF results are surprisingly lower than the other classifiers except for coarse 3, where the RBF base classifiers showed higher accuracy. However in some cases the MLP and RBF classifiers performed better than their NN counterpart. The MLP and RBF results are highly dependent on the initialisation of the weights used within the network and the variation between best and average accuracies could be quite high in some cases. Allowing more training iterations would help at the cost of increased computation. The RBF classifiers seemed quite sensitive to the data and parameters they were given. Some combinations could lead to posterori probabilities of zero during training reducing the performance of the classifier.

The base classifiers showed that the individual classifiers were unable to cope with the more complex classifications. Even with the few classes in the coarse 3 data set, the base classifiers were unable to get above 74% accuracy. We can clearly see that the base classifiers are not useful for obtaining high quality results on their own. It is possible other techniques (e.g. SVM) may show better performance, but it is unlikely to boost the more complex classifications to more useful levels.

The differences between oracle performance and the performance of Majority Vote and Dynamic Classifier Selection indicates that a better combination technique could lead to higher performance. For the DCS classifiers, an alternative confidence estimator may yield better results, while for ensembles alternative combination rules may perform better. Higher oracle performance may be obtained by using alternative 3-D descriptors or classification techniques.

The use of GAs has shown to produce reasonable results. Better results were obtained when the test set was allowed to drive the selection process, however this results in biased classifiers. Better results could possibly be obtained by better ranking of the classifiers during training. In this work, each solution was created 10 times, each time using a random 25% to validate it. This is almost cross-validation, but does not make sure that

all of the data is used for testing at once. The high computational cost of this technique is very high and does not show improvement over the combined classifiers which are much less computationally intensive.

There are several possible uses of classification to improve CBR results. We can classify the object and then return only those objects in the data set with the same class label. In Chapter 5 we looked at the per-class statistics for a set of objects. By pre-calculating these statistics, we can classify the query object and discover it's class label. We can then find the best descriptor to use for that class. Finally, we could use the GA technique to help find the optimal parameters for a CBR technique, although of course this will be likely to make the algorithms tailored to a particular data set.

Recently Barutcuoglu and DeCoro (2006) has published a technique using Bayesian Aggregation to combine binary classifiers trained on a one-versus-all basis for each class in the hierarchy (including the parent classes). Each positive classifier is assigned a probability and it is organised in the class hierarchy. The Bayesian Aggregation process re-assigns the probabilities, reducing or removing completely false positives and increasing the correct classification. Of course, if the classifiers are very bad, then incorrect classifications can still be made. This seems like a better method than those experimented with for the SVM's as it attempts to remove the false positives which can effect the one-versus-one methods and it takes into account the class hierarchies which none of the SVM methods would do.

6.6 Summary

In this chapter we have investigated the use of using classification techniques using CBR feature vectors as inputs. We focused on using multi-class classifiers which work well with a few classes and uncomplicated class boundaries, but perform much worse as the number of classes increases and the classes become more mixed in feature space. The results indicate that the descriptors used are generally too similar or just not capable of distinguishing between the finer details and more powerful descriptors need to be investigated. However, the current descriptors are dissimilar enough to show the potential of combination techniques as small improvement were shown in our results. The use of binary classifiers can make it easier for the classifiers to distinguish between classes and the recent work by Barutcuoglu and DeCoro (2006) shows a powerful technique to obtain correct classifications. We have seen that generally the Nearest Neighbour classifier performs best, even though it has limitations in estimating class boundaries. It would have been expected that the MLP and RBF networks would be better estimators for the class boundaries. Perhaps the boundaries were too complex causing worse performance than the NN. This obviously brings into question the quality of the data sets. However,

real world data sets are unlikely to be perfect and the techniques used must be flexible enough to handle such problems.

Chapter 7

Semantic 3-D Object Annotation

7.1 Introduction

The previous chapters have explored various 3-D shape descriptors with their application to 3-D content-based retrieval and have investigated creating classifiers using the feature vectors as inputs. This work was originally undertaken in the context of a digital multimedia warehouse using advanced search techniques to access the data. In this chapter it will be shown how the more low level work described in the previous chapters can be applied to the bigger picture.

More specifically we study the problem of 3-D object annotation using Semantic Web technologies and classification techniques to help facilitate a possible solution. The use of annotations in a combined content-based retrieval has shown good improvements. However there exist many image and object collections that are un-annotated. Manually annotating small collections is a slow and mundane task which is prone to error, and is unfeasible on large collections. Therefore there is scope for developing systems which can automatically annotate objects. There has been a large amount of research in the area of 2-D image annotation (see e.g. Barnard et al. 2003; Duygulu et al. 2002) but there has been limited research in the 3-D annotation area (see e.g. García-Rojas et al., 2005).

We describe here how the classification techniques from the previous chapter can be used to annotate 3-D objects using concepts in an ontology. By adding such classifications to a semantic database, semantic web technologies can be used to derive new knowledge using the existing knowledge in the system and classifier predictions. This is essentially the task the classifier agent described in Chapter 4 was intended for.

In the previous chapter classification techniques were used to predict a single class label for a query object. While the classifier combination and optimisation techniques investigated did improve performance over a single classifier created with arbitrary parameters,

performance was still less than perfect and degrades dramatically as the number of classes under consideration increases and the number of samples in each class decreases. Any annotation system based on such classifiers therefore needs an element of confidence attached to any annotations it makes. An annotation made from a single classification should be treated with greater suspicion than an annotation made from many classifications agreeing upon the result.

Given a set of existing annotated content, we wish to annotate new content using the existing data as a reference. A single object may have more than one possible annotation. For example a blue coloured vase may have the annotations "blue", "vase", "pot" and "container". Additionally an annotation may apply to the whole object, or just part of the object (part correspondence (Barnard et al., 2003) is a much harder problem and not within the scope of this thesis). Typically a classifier is trained to distinguish between a set of class labels for a given concept. For example colour, shape or function. In the case of the blue vase, a classifier would typically need to be trained to distinguish between colours and another classifier would need to be trained to distinguish between shapes. However this depends on the features and class labels presented to the classifier. Other labels can be inferred; for example vase could imply container. Such terms could be obtained through a thesaurus or relations in an ontology. At this level, we treat a classifier as a black box; how it produces a classification is not important as long as it is correct (at least to within a given error). It is trained on sample data containing examples of the concepts to distinguish between and then it is used to classify query objects. Ideally the classifier system would take care of all the finer details; however as yet there is no standard methodology for doing so. The aim of this work is to investigate some possible methods. Feature vectors generated from the shape descriptors can be used as inputs to classifiers. Most of the descriptors produce fixed length normalised histograms which are almost perfect for inputs to classifiers. Some feature vectors are quite large however which can lead to problems in training certain classification schemes due to the large dimensionality.

This chapter begins with some background on annotation techniques and semantic web technologies. This is followed by a discussion on how these technologies can be combined with classification techniques to create the basis of a 3-D object annotation system. This chapter ends with some conclusions.

7.2 Some Background

There are many approaches to annotation and there are various levels of annotation. An object may have one or more annotations associated with it. These annotations may be associated with particular parts or regions of the object rather than the whole. In the

case of a 2-D image, annotations could point to the sun, sky or beach. For a 3-D object, annotations such as arm, leg or head could be applied to parts of a statue.

There are several approaches to automatically annotating objects. A simple approach would be to assign the annotation that most frequently occurs in a reference data set, but this method ignores any other knowledge about the object. Another method is to find the most similar object in the reference set and copy the annotations from that object to the query object (nearest neighbour classification). However it is possible the query object is very different to any in the reference set, yet it will still have annotations applied to it. Alternatively the correct annotations may be present across several similar objects instead of just one. In this situation complex class boundaries may not be fully represented by the reference objects; a more sophisticated approach is needed. Barnard et al. (2003) gives an overview of two general classes of annotation models, Multi-Model Hierarchical Aspect Models and Mixture of Multi-Model Latent Dirichlet Allocation. These models attempt to find a mapping between terms and regions. For example, the term sky may lead to mostly blue regions and mostly green regions may map to the term grass. Thus when searching for the term sky, unannotated images with blue regions can also be returned.

Duygulu et al. (2002) used machine learning techniques to associate a fixed number of terms to regions of an image. They treat the problem as one of language translation, where terms are one language and features in regions are another. Each region was determined by a segmentation algorithm and terms were associated to each region using Expectation Maximisation to *translate* regions to terms. Essentially the process finds similar regions in different images that contain the same terms. The probability of the term occurring for such regions increases with how strongly correlated the term and regions are.

Semantic Web technologies aim to impose a machine readable structure to any content published on the Web (or otherwise). For any individual item of data, a tag of some kind needs to be associated with it to say what the information is. Information is structured according to some schema or Ontology describing the concepts and relations between concepts. This allows any program that understands a particular schema to understand any data structured according to that schema. A schema or ontology can be described in several different formats. Typically RDF (Lassila and Swick, 1999) is used, however it is unable to model the full range of relationships that can be used to model an ontology. OWL, the Web Ontology Language (McGuiness and van Harmelen, 2004) is based upon RDF and provides a language to fully describe an ontology.

In an effort to help interoperability, there are several ontologies defined with the Dublin Core (DCMI Usage Board, 2004) being one of the more well known ones and often used as a basic level of interoperability between systems. This is an ontology to describe a resource. A resource can have such elements as creator, title, description and date.

```
<?xml version="1.0"?>
  <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
           xmlns:contact="http://www.w3.org/2000/10/swap/pim/contact#">
    <contact:Person rdf:about="http://www.w3.org/People/EM/contact#me">
      <contact:fullName>Eric Miller</contact:fullName>
      <contact:mailbox rdf:resource="mailto:em@w3.org"/>
      <contact:personalTitle>Dr.</contact:personalTitle>
  </contact:Person>
</rdf:RDF>
```

FIGURE 7.1: Dublin Core Example

Figure 7.1 shows an example taken from Manola and Miller (2004) describing contact details for a person.

Typically a group of experts in a particular domain will produce an ontology for that domain. In the case of SCULPTEUR, the CIDOC CRM (Crofts et al., 2001) was used as this describes the cultural heritage domain.

All of this structured content would be of little use if there was no way to search it, and several query languages have been developed. One of the more commonly used languages is RDQL (Seabourne, 2004), although there are several other competing languages offering more sophisticated querying. SeRQL is the query language used in Sesame (Aduna BV, No Year), a RDF database system. SPARQL (Prud'hommeaux and Seabourne, 2006) is the other language, a W3C recommendation.

Typically RDF is stored in a RDF database which provides one or more query languages to manage the data. Sesame (Aduna BV, No Year) has already been mentioned. This supports both RDQL and SeRQL and it also allows inference rules to be defined which are applied to data as it is imported into the database. Another database is the triplestore (Harris et al., No Year) supporting RDQL and SPARQL query languages. Jena (Hewlett-Packard Development Company, No Year) is another database supporting RDQL and SPARQL and it provides a customisable semantic reasoner which allows inferencing.

There has been some interest in using a Shape Ontology to describe a 3-D object from a collection of known "primitive" components. García-Rojas et al. (2005) use virtual humans as the basis for their work. They use this technique to identify parts of the human body. This technique however is directed towards a single class of objects (human shaped objects in this case) rather than a range of different objects.

7.3 Application

In Chapter 4 a classifier agent was described. The original aim of the agent was to help create metadata for new objects entering the SCULPTEUR system. By modelling

a prediction made by an arbitrary classifier for a given object as RDF, we can use Semantic Web technologies to make use of this prediction to populate the SCULPTEUR system. A very simple method would be to add the predicted class to the correct place in the system directly. However, this assumes perfect accuracy by the predicting classifier. As we have seen in Chapter 6 classifiers are far from perfect and different classifiers may make contradictory classifications. We need to store multiple classifications with a confidence value for how correct they are. Storing classifications in a semantic web database allows the use of semantic web inferencing technologies to be applied to create new knowledge about the object. This is useful when applying other existing knowledge about the object to the predictions, however, there is no real way to make use of the prediction confidences.

The basic building blocks for such a system can easily be implemented. Taking the PSB classifications as an example, we can easily represent them using SKOS (Miles and Brickley, 2005). SKOS allows simple knowledge structures to be defined. Of main interest is the definition of broader and narrower concepts which can be used to model the hierarchy of classes. Figure 7.2 shows how some of the classes can be represented using SKOS. It shows how the hierarchy from the root class, 0, to a leaf class such as F117 and biplane can be represented. Each object in the PSB data set can then be associated to one of these SKOS representations of a PSB class. Figure 7.3 shows how an object of class F117 can be represented. This represents the real class labels.

Representing a prediction is a bit more complicated as there are several things to model. A prediction is a class label, which should have some kind of confidence value associated with it. Each prediction is made by a classifier for a particular query object. A classifier can make predictions on multiple objects, and each object can have predictions from multiple classifiers. Figure 7.4 shows a possible representation in RDF.

7.4 Expanding Knowledge

These individual pieces don't offer much in the way of extra value. We are just storing the data we already have in a different way. However, once it is in such a format, existing Semantic Web technologies can be utilised to add to that knowledge. A basic method is to apply inference techniques to a predicted class label so that the parent class labels are also explicitly associated.

We can also exploit the fact that the PSB base classification contains very fine grained classes, and coarse 3 contains very coarse grained classes with the coarse 1 and 2 classifications somewhere in between. This means that in some or all cases, objects in a class in the base classification will all be in the same class in the coarse 1, 2 and 3 classification. However, the inverse is not true, that it objects in a class in coarse 1 are very likely to be in several classes in the base classification. This makes it easy to help infer relations

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:skos="http://www.w3c.org/2004/02/skos/core#">
  <skos:Concept
              rdf:about="http://www.ecs.soton.ac.uk/sg/psb#aircraft">
    <skos:broader
                rdf:resource="http://www.ecs.soton.ac.uk/sg/psb#0"/>
    <skos:narrower
           rdf:resource="http://www.ecs.soton.ac.uk/sg/psb#airplane"/>
  </skos:Concept>
  <skos:Concept
              rdf:about="http://www.ecs.soton.ac.uk/sg/psb#airplane">
    <skos:broader
           rdf:resource="http://www.ecs.soton.ac.uk/sg/psb#aircraft"/>
    <skos:narrower
              rdf:resource="http://www.ecs.soton.ac.uk/sg/psb#F117"/>
    <skos:narrower
            rdf:resource="http://www.ecs.soton.ac.uk/sg/psb#biplane"/>
  </skos:Concept>
  <skos:Concept rdf:about="http://www.ecs.soton.ac.uk/sg/psb#F117">
    <skos:broader
           rdf:resource="http://www.ecs.soton.ac.uk/sg/psb#airplane"/>
  </skos:Concept>
  <skos:Concept rdf:about="http://www.ecs.soton.ac.uk/sg/psb#biplane">
    <skos:broader
           rdf:resource="http://www.ecs.soton.ac.uk/sg/psb#airplane"/>
  </skos:Concept>
</rdf:RDF>
```

FIGURE 7.2: Representation of class hierarchy

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:skos="http://www.w3c.org/2004/02/skos/core#">
  <rdf:Description rdf:about="http://www.ecs.soton.ac.uk/sg/psb/1298">
    <skos:subject
              rdf:resource="http://www.ecs.soton.ac.uk/sg/psb#F117"/>
  </rdf:Description>
</rdf:RDF>
```

FIGURE 7.3: Representation of an object

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:prob="http://.../.#"
         xmlns:prediction="http://.../.#">
  <rdf:Description
            rdf:about="http://www.ecs.soton.ac.uk/sg/prediction#123">
    <prediction:classifier_id
         rdf:resource="http://www.ecs.soton.ac.uk/sg/classifier#111"/>
    <prediction:object_id
              rdf:resource="http://www.ecs.soton.ac.uk/sg/psb/1298"/>
    <prob:PriorProbObj rdf:ID="P(F117)">
      <prob:hasVariable>
        <rdf:Value>F117</rdf:Value>
      </prob:hasVariable>
      <prob:hasProbValue>0.4</prob:hasProbValue>
    </prob:PriorProbIbj>
  </rdf:Description>
</rdf:RDF>
```

FIGURE 7.4: Representation of a prediction

between the different classifications. More generally this is a form of ontology mapping which is generally a much harder problem.

Classification predictions could be combined using the techniques described in Chapter 6, however this would not add anything to the system; we may as well have just performed those operations on the classifiers directly.

Making use of probabilistic data in a semantic setting is a new research area requiring methodologies to both represent and interpret probabilities in RDF. While a new area for RDF, it is not a new research area in general. Current research has focused on defining a representation that is easy to convert to a Bayesian network representation allowing external, well established techniques to be applied to the data and the result imported back into the system (Ding and Peng, 2004). To see the advantages of such a scheme, Barutcuoglu and DeCoro (2006) have used Bayesian networks to improve the results of binary classifiers trained on the Princeton Shape Benchmark (PSB). While this work does not explicitly use Semantic Web techniques, it is using the Bayesian network to accomplish the same task.

7.5 Some Open Issues

There are several issues still to be addressed. The most important one is the distinction between manually assigned annotations and machine assigned annotations. Annotations with low confidence values should be kept separate from the real annotations to avoid polluting the system with bad annotations. However, as the confidence increases, the

higher the likelihood of an annotation being a true annotation. The question is, when is the confidence high enough? It is unlikely that 100% confidence will ever be reached as there will always be some element of doubt. A system operator could validate the proposed annotations which is a suitable proposal for limited numbers of annotations. However, when large numbers of annotations have been generated, automatically adding them is more desirable. The exact policy is really dependent upon the task. In some cases quantity may be more desirable than quality of annotations.

Another issue is when to decide how good a prediction is to make use of it. An object with a single prediction is not really a very reliable prediction, even with 100% confidence. Ideally several predictions should be collected together first for the same classification (e.g. PSB base classification).

7.6 Conclusions

In this chapter an outline for transforming the research into a Semantic Web setting and furthering the status of the classifier agent has been described. The techniques described here should easily be incorporated into an existing semantic system.

One of the original goals of the classifier agent was to automatically classify new objects entering the system and to automatically create the classifiers required to do this. It should be possible to drive the process of creating new classifiers in situations where the existing set of classifiers is insufficient to produce an annotation with a high enough level of confidence.

Chapter 8

Conclusions and Future Work

8.1 Conclusions

In this thesis we have investigated the use of 3-D shape descriptors for use in content-based retrieval, classification and annotation. Involvement with the SCULPTEUR project provided real world 3-D objects and users for testing in *out of the lab* situations. Two papers have been published; one is an analysis of a range of 3-D shape descriptors and the other is our initial investigations into classifier training parameter optimisation. One software package developed during the course of this thesis, FVS, has been released as open-source software and is available at `http://libfvs.sourceforge.net/`.

From the experimentation with 3-D CBR algorithms we have seen that there is no one descriptor that performs best in all situations. While some generally show higher performance than others overall, for specific cases other descriptors may perform much better. The same can be said for the different distance metrics, although they play a much smaller role in effecting performance compared to the choice of descriptor. It is also interesting to note that no one performance metric is best overall either. Some descriptors are very good at finding an object of the same class as the query object as the closest match but they are not so good at finding objects of the correct class for the other close matches. Such a descriptor would rank highly for the Nearest Neighbour statistic but lower for the first tier statistic. Another descriptor may never find an object of the correct class for the closest match but the other closer objects may well be of the correct class thus giving a low nearest neighbour ranking and a higher first tier score. Each descriptor and metric combination is sensitive to certain shape features while insensitive to other shape features. This is what makes them good for certain classes of object while poor for other classes. While it is unlikely that a descriptor will ever be sensitive to all shape features, it is much more likely that a subset of descriptors will be able to collaborate to achieve this goal.

As with the choice of descriptor and distance metric, the choice of performance metric is quite dependent upon the task in hand. Such analysis of the reference data sets and descriptors can provide useful information to a knowledgeable user to improve the quality of their search results. Current retrieval systems typically either have one specific descriptor or a small user-selectable range, but there no way to automatically pick the best descriptor for query object.

The classification work has shown that the descriptors used in this work are not discriminating enough for fine grained classifications, but reasonable results can be obtained with fewer classes. We have seen that the combination of classifiers can improve performance and with more discriminating descriptors and classification techniques, higher accuracy rates can be expected. While the combination of classifiers showed a small improvement in performance, it suggests that the current selection of descriptors are quite similar in discriminative capability. Dynamic classifier selection offers the best potential for improving the performance of a set of base classifiers over any individual performance while Majority Vote struggles with our data sets. It is interesting to note that most prior research involves distinguishing between two classes with many members in each whereas our work considered multiple classes with typically only a few members in each class. The Majority Vote classifier combination rule may not have had a strong majority with votes spread across a number of candidate classes. Such situations reduce the classification to a random assignment. A possible alternative is to reject the input pattern outright rather than assigning a weak classification.

The application of classifiers to annotate 3-D objects allows annotation of objects based upon content. This allows a machine to "learn" the features that represent a given annotation and apply that knowledge to new objects. The ability to learn annotations is entirely dependent upon the capability of the descriptors to capture the features that represent that feature. The use of probabilistic networks and standard Semantic Web reasoning tools should provide a much higher quality of annotations than by just using a single classifier on its own. By storing all the predictions made by classifiers, it allows for *evidence gathering* of the real class. By making use of the multiple PSB classifications, a range of applicable annotations can be applied to a single object. It should be noted that some annotations we wish to assign to an object cannot be represented visually and exist only to define a context. Such annotations are impossible to learn using visual features.

It is important to note that while the focus has been on 3-D objects, the techniques described in this thesis can be applied to any type of media for which a fixed length feature vector can be generated. By using techniques such as k-NN classifiers variable length feature vectors could also be used.

8.2 Future Work

The 3-D descriptor is the fundamental building block of the work in this thesis, however as we have seen, the descriptors used did not provide the discriminative capabilities to handle the fine grained classes. There are many other techniques that have not been investigated in this thesis and may help improve performance by either being substantially different to the existing descriptors or by having a much greater discriminative power. The current descriptors are all shaped based. Colour information has been ignored. Inclusion would make the descriptors very different, however the lack of colour information in the majority of 3-D models in our data sets makes the use of colour applicable to only a few objects.

Another limitation faced in this work is the small data set sizes. The PSB Base classification has an average class size of around ten members. This is very small for training a classifier, especially when there are nearly a hundred leaf classes to distinguish between.

We have only experimented with a few popular classification techniques. Further investigations with other techniques will be useful to establish whether any substantial increase in performance can be obtained. Alternative methods of combining classifiers may also yield improved results.

The complex class boundaries makes learning to discriminate between all classes in one go a difficult task to achieve, especially when some classes are very large and others are very small. Techniques for breaking the task into smaller, easier to solve problems may help improve results.

Chapter 7 suggested how 3-D objects could be annotated using classification techniques and how to potentially build upon that knowledge using semantic web techniques. It would be good to build such a system and experiment with methods of building upon the classifier predictions. The use of relevance feedback could be useful in training the system to "prefer" some annotations over others. An example here is the use of relevance feedback to adjust the confidence associated to a classifier. A classifier with a poor estimated confidence could make very good annotations in the view of a user. Likewise a classifier with a high confidence could make some very poor annotations in the view of a user. Of course annotations are often subjective and different people can make different annotations for the same object.

Manually creating classifiers may not provide for the whole range of difficult cases that may arise while annotating an object. Allowing the annotation system to drive the creation of new classifiers to handle difficult cases is an area that should help improve overall accuracy. For example, the manually created classifiers may predict with a high accuracy two mutually exclusive classes for set of objects. In this case it would be advantageous to generate some classifiers to distinguish between these two classes.

An interesting area to look at is the use of classification techniques to select the best descriptor to use in a CBR query. By using a classifier to determine the potential class of the query object, appropriate methods can be used to boost performance for objects of that class. Possibilities are to select the best descriptor for the query object class, or to return only objects of the predicted class (or at least make sure they are ranked higher).

Appendix A

Glossary

ARTISTE The predessesor project to SCULPTEUR that developed integrated content and metadata-based image retrieval across several major art galleries in Europe.

CBR Content-Based Retrieval.

CD Compound Diversity measure for comparing classifier diversity.

CIDOC CRM CIDOC Conceptual Reference Model. An ontology of cultural heritage information.

CQL Common Query Language.

CyberX3D A C++ VRML and X3D parser library.

DAG-SVM Directed Acyclic Graph Support Vector Machine. A multi-class SVM technique using binary SVMs.

DCG The Discounted Cumulative Gain.

DCS Dynamic Classifier Selection.

Descriptor A specific version of a content-based retrieval algorithm.

Distance Image A visual matrix representing the distance between every object in a data-set.

eChase A European project following on from the SCULPTEUR project.

EGI Extended Gaussian Image descriptor.

Feature Vector The output of a descriptor, typically a histogram.

First Tier A performance metric indicating how many of the top matches are of the correct class. The number of matches to consider is equal to the size of the class under consideration.

FVG The Feature Vector Generator software in ARTISTE.

FVS The Feature Vector software in SCULPTEUR. Derived from FVG.

GA The Genetic Algorithms optimisation technique.

GD Generalisation Diversity measure for comparing classifier diversity.

GET-ENST A partner in the SCULPTEUR project.

JSP Java Servlet Pages.

LGPL Lesser/Library GNU Public License.

MCS Multiple Classifier System.

MD2 Modified Shape D2 descriptor.

MLP Multi-Layer Perceptron.

MoE Mixture of Experts multiple classifier architecture.

MRG Multi-resolution Reeb Graph descriptor.

MV Majority Vote classifier combination rule.

MySQL An open-source database management system.

Nearest Neighbour The nearest neighbour is the closest object in feature space to a query. The nearest neighbour statistic is the proportion of objects for which the nearest neighbour was of the same class. The nearest neighbour classifier is a special case of the k-Nearest Neighbour classifier where k is equal to 1.

OWL Web Ontology Language. A language used to define an ontology.

PCA Principal Components Analysis.

Precision A performance metric indicating how relevant the matches are.

PSB The Princeton Shape Benchmark. A data set of around 1,800 objects classified into several groupings. The benchmark also provides a set of tools to evaluate performance.

PHP A scripting language used in web servers to provide dynamic content.

PSO The Particle Swarm Optimisation technique.

RBF Radial Basis Functions Network Classifier.

RDF Resource Description Framework.

RDQL RDF Data Query Language.

Recall A performance metric indicating the proportion of relevant items found.

ROC Receiver Operating Characteristics Graph.

SCG Scaled Conjugate Gradients. A training method for the MLP classifier.

SCULPTEUR A European project.

Second Tier A performance metric similar to the First Tier but with twice as many matches considered.

SeRQL Sesame RDF Query Language.

Sesame A RDF framework for storing and querying RDF.

SOM Kohonen's Self Organising Map.

SKOS Simple Knowledge Organisation Systems. A standard way to represent knowledge using RDF.

SPARQL Simple Protocol and RDF Query Language for querying RDF databases.

SQL Structured Query Language. Used to query databases such as MySQL.

SRW Search and Retrieve Web service.

SVM Support Vector Machine.

Tier Image A visual matrix showing which objects were counted as the nearest neighbour, first and second tier matches for every object.

UDF A User Defined Function in MySQL.

VIPS VASARI Image Processing System. Originally developed during the VASARI project. Used within FVS for 2-D image processing.

VRML The Virtual Reality Modelling Language.

WWW The World Wide Web.

X3D An XML representation of a 3-D object. Replaces VRML.

Appendix B

PSB Classifications

Class Name	Train	Test	Class Name	Train	Test
aircraft/airplane/F117	4	0	furniture/seat/chair/dining	11	11
aircraft/airplane/biplane	14	14	furniture/seat/chair/desk	0	15
aircraft/airplane/commercial	10	11	furniture/seat/char/stool	7	0
aircraft/airplane/fighter jet	50	50	furniture/seat/couch	15	0
aircraft/airplane/glider	0	19	furniture/shelves	13	13
aircraft/airplane/multi_fuselage	7	0	furniture/table/rectangular	26	25
aircraft/airplane/stealth bomber	0	5	furniture/table/round	12	0
aircraft/baloon vehicle/dirigible	7	0	furniture/table/round/single leg	0	6
aircraft/balloon vehicle/hot air balloon	0	9	furniture/table and chairs	5	0
aircraft/helicopter	17	18	geographical map	0	12
aircraft/spaceship/enterprise like	11	11	gun/handgun	10	10
aircraft/spaceship/satellite	0	7	gun/rifle	19	0
aircraft/spaceship/space shuttle	6	0	hat	0	6
aircraft/spaceship/flying saucer	0	13	hat/helmet	10	0
aircraft/spaceship/tie fighter	0	5	hourglass	0	6
aircraft/spaceship/x wing	5	0	ice cream	12	0
animal/arthropod/insect/ant	0	5	ladder	0	4
animal/arthropod/insect/bee	4	0	lamp/desk lamp	14	0
animal/arthropod/insect/butterfly	0	7	lamp/streetlight	0	8
animal/arthropod/spider	11	0	liquid container/bottle	12	0
animal/biped/human	50	50	liquid container/glass with stem	0	9
animal/biped/human/human arms out	21	20	liquid container/mug	7	0
animal/biped/human/walking	0	8	liquid container/pail	0	4
animal/biped/trex	6	0	liquid container/tank	5	0

Table B.1: PSB Dataset: Classes and sizes

Class Name	Train	Test	Class Name	Train	Test
animal/flying creature/bird/duck	5	0	liquid container/vase	11	11
animal/flying creature/bird/flying bird	0	14	mailbox	0	7
animal/flying creature/bird/standing bird	0	7	microchip	7	0
animal/quadruped/aptosaurus	4	0	microscope	5	0
animal/quadruped/dog	0	7	musical instrument/guitar/accoustic guitar	4	0
animal/quadruped/feline	6	0	musical instrument/guitar/electrical guitar	0	13
animal/quadruped/horse	0	6	musical instrument/piano	6	0
animal/quadruped/pig	4	0	newtonian toy	0	4
animal/quadruped/rabbit	0	4	phone handle	4	0
animal/snake	0	4	plant/bush	0	9
animal/underwater creature/dolphin	5	0	plant/flower with stem	15	0
animal/underwater creature/sea turtle	0	6	plant/flowers	0	4
animal/underwater creature/shark	7	0	plant/potted plant	25	26
animal/underwater creature/fish	0	17	plant/tree	17	0
blade/butcher knife	4	0	plant/tree/barren	11	11
blade/axe	0	4	plant/tree/conical	0	10
blade/knife	0	7	plant/tree/palm	10	0
blade/sword	15	16	satellite dish	0	4
body part/brain	7	0	sea vessel/sailboat	5	0
body part/face	17	16	sea vessel/sailboat/large sail boat	0	6
body part/hand	0	17	sea vessel/sailboat/sailboat with oars	4	0
body part/head	16	16	sea vessel/ship	10	11
body part/skeleton	5	0	sea vessel/submarine	0	9
body part/skull	0	6	shoe	8	0
body part/torso	4	0	sign/billboard	0	4
book	0	4	sign/street sign	12	0
bridge	10	0	sink	0	4
building/barn	0	5	skateboard	5	0
building/castle	7	0	slot machine	0	4
building/church	0	4	snowman	6	0
building/dome church	13	0	staircase	0	7

Table B.1: PSB Dataset: Classes and sizes

Class Name	Train	Test	Class Name	Train	Test
building/gazebo	0	5	swingset	4	0
building/lighthouse	5	0	tool/hammer	0	4
building/one story home	0	14	tool/screwdriver	5	0
building/roman building	12	0	tool/shovel	0	6
building/skyscraper	0	5	tool/wrench	4	0
building/tent/multiple peak	5	0	umbrella	0	6
building/tent/one peak tent	0	4	vehicle/car/antique car	5	0
building/two story home	11	10	vehicle/car/race car	0	14
chess piece	17	0	vehicle/car/sedan	10	10
chess set	0	9	vehicle/car/sports car	19	0
chest	7	0	vehicle/covered wagon	0	5
city	10	10	vehicle/cycle/bicycle	7	0
computer/laptop	4	0	vehicle/cycle/motorcycle	0	6
display device/computer monitor	0	13	vehicle/military tank	16	0
display device/tv	12	0	vehicle/monster truck	0	5
door	0	18	vehicle/pickup truck	8	0
door/double doors	10	0	vehicle/semi	0	7
eyeglasses	0	7	vehicle/suv	4	0
fantasy animal/dragon	6	0	vehicle/suv/jeep	0	5
fireplace	0	6	vehicle/train	7	0
furniture/bed	8	0	vehicle/train/train car	0	5
furniture/cabinet	0	9	watch	5	0
furniture/desk/desk with hutch	7	0	wheel	0	4
furniture/desk/school	0	4	wheel/gear	0	9
furniture/seat/bench	0	11	wheel/tire	4	0

Table B.1: PSB Dataset: Classes and sizes

Class Name	Train	Test	Class Name	Train	Test
aircraft/winged vehicle	107	135	furniture/seat	40	37
aircraft/ballon vehicle	7	9	furniture/shelves	13	13
aircraft/helicopter	17	18	furniture/table	43	35
animal/arthropod	15	12	geographic map	0	12
animal/biped/human	71	78	gun	29	0
animal/biped/trex	6	0	hat	10	6
animal/flying creature	5	21	ladder	0	4
animal/quadruped	14	17	lamp	14	8
animal/snake	0	4	liquid container	35	24
animal/underwater creature	12	23	mailbox	0	7
blade	19	0	musical instrument	10	13
body part/hand	0	17	plant	78	60
body part/head	40	38	satellite dish	0	4
body part/skeleton	5	0	sea vessel	19	26
body part/torso	4	0	shoe	8	0
bridge	10	0	sign	12	4
building	53	47	skateboard	0	4
chess piece	17	0	slot machine	0	4
chest	7	0	snowman	6	0
city	10	10	staircase	0	7
display device	16	24	swingset	4	0
door	10	18	handheld	40	83
fantasy animal/dragon	6	0	vehicle/car	63	51
fireplace	0	6	vehicle/cycle	7	6
furniture/bed	8	0	vehicle/train	7	0
furniture/cabinet	0	9	wheel	4	13

TABLE B.2: PSB Coarse 1 Data set: Classes and sizes

Class Name	Train	Test	Class Name	Train	Test
vehicle	230	245	building	53	47
animal	123	155	furniture	104	94
household	219	185	plant	78	60

TABLE B.3: PSB Coarse 2 Data set: Classes and sizes

Class Name	Train	Test	Class Name	Train	Test
natural	282	256	vehicle/car	80	0
man made	625	571			

TABLE B.4: PSB Coarse 3 Data set: Classes and sizes

Bibliography

3D Cafe. 3D cafe. online - www.3dcafe.com [Accessed 2004-09-01], No Year.

M. J. Addis, S. Goodall, P. H. Lewis, K. Martinez, P. A. S. Sinclair, F. Giorgini, C. Lahanier, J. Stevenson, M. Cappellini, and L. Serni. Searching and exploring multimedia museum collections over the web. In *Proceedings of EVA 2005*, Palazzo dei Congressi, Florence, Italy, 2005a.

M. J. Addis, K. Martinez, P. Lewis, J. Stevenson, and F. Giorgini. New ways to search, navigate and use multimedia museum collections over the web. In J. Trant and D. Bearman, editors, *Proceedings of Museums and the Web 2005*, pages 582–596, Vancouver, Canada, 2005b.

Matthew Addis, Mike Boniface, Simon Goodall, Paul Grimwood, Sanghee Kim, Paul Lewis, Kirk Martinez, and Alison Stevenson. SCULPTEUR: Towards a new paradigm for multimedia museum information handling. In *International Semantic Web Conference (ISWC 2003)*, pages 582–596, Florida, USA, October 2003a.

Matthew J. Addis, M. J. Boniface, Simon Goodall, Paul Grimwood, Sanghee Kim, Paul Lewis, Kirk Martinez, and Alison Stevenson. Integrated image content and metadata search and retrieval across multiple databases. In E. M. Bakker, T. S. Huang, and M. S. Lew, editors, *Second International Conference on Image and Video Retrieval 2003*, pages 91–100, Urbana-Champaign, Illinois, USA, 2003b.

Aduna BV. Sesame - storage and querying framework for RDF and RDF schema. online - http://www.openrdf.org [Accessed 2006-08-28], No Year.

Mihael Ankerst, Gabi Kastenmüller, Hans-Peter Kriegel, and Thomas Seidl. Nearest neighbor classification in 3D protein databases. In *7th International Conference on Intelligent Systems for Molecular Biology (ISMB'99)*, pages 34–43, Heidelberg, Germany, 1999. AAAI Press.

Tarik Filali Ansary, Mohamed Daoudi, and Jean-Philippe Vandeborre. Semantic 3D. online - http://www-rech.enic.fr:8080/3Dretrieval/ [Accessed 2006-07-23], No Year.

G. Antini. 3D CBR. online - http://delos.dsi.unifi.it:8080/CV/index.jsp [Accessed 2006-07-23], No Year.

Autodesk, Inc. 3-D studio max. online - http://www.autodesk.com/3dsmax/ [Accessed 2006-08-26], No Year.

D. Ayala, P. Brunet, R. Juan, and I. Navazo. Object representation by means of non-minimal division quadtrees and octrees. *ACM Transactions on Graphics*, 4(1):41–59, January 1985.

D. H. Ballard. Generalising the hough transform to find arbitrary shapes. *CVGIP*, 13: 111–122, 1981.

Kobus Barnard, Pinar Duygulu, Nando de Freitas, David Forsyth, David Blei, and Michael I. Jordan. Matching words and pictures. *Journal of Machine Learning Research*, 3:1107–1135, 2003.

Zafer Barutcuoglu and Christopher DeCoro. Hierarchical shape classification using bayesian aggregation. In *IEEE International Conference on Shape Modelling and Applications 2006 (SMI'06)*, page 44, Matsushima, Japan, June 2006. IEE.

David Beasley, David R. Bull, and Ralph R. Martin. An overview of genetic algorithms: Part 1, fundamentals. *University Computing*, 15(2):58–69, 1993.

Serge Belongie, Jitendra Malik, and Jan Puzicha. Shape matching and object recognition using shape contexts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(24):509–522, 2002.

Christopher M. Bishop. *Neural Networks for Pattern Recognition*, chapter 2, page 35. Oxford University Press, 4 edition, 1997a.

Christopher M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, 4 edition, 1997b.

Blender Foundation. Blender. online - http://www.blender3d.org/ [Accessed 2006-08-26], No Year.

Leo Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

Ding-Yun Chen, Ming Ouhyoung, Xiao-Pei Tian, and Yu-Te Shen. On visual similarity based 3D model retrieval. *Computer Graphics Forum*, pages 223–232, 2003.

Rachel Coates. Sculpteur second prototype evaluation (d3.4). Deliverable D3.4, February 2005.

J. R. Corney. Shape sifter. online - http://www.shapesearch.net/index.html [Accessed 2006-07-23], No Year.

N. Crofts, I. Dionissiadou, M. Doerr, and M. Stiff. Definition of the CIDOC object-orientated conceptual reference model, v.3.1, July 2001.

DCMI Usage Board. Dublin core vocabulary. DCMI recommendation, OCLC Research, 2004.

Zhongli Ding and Yun Peng. A probabilistic extension to ontology language OWL. In *Proceedings of the 37th Hawaii International Conference on System Sciences 2004*, page 40111.1. IEEE Computer Society, 2004.

Robert P. W. Duin. The combining classifier: to train or not to train? In *ICPR (2)*, pages 765–770, 2002.

Robert P. W. Duin and David M. J. Tax. Experiments wih classifier combining rules. In J Kittler and F. Roli, editors, *Multiple Classifier Systems: First International Workshop, MCS 2000*, pages 16–29, Cagliari, Italy, June 2000.

P. Duygulu, K. barnard, J. F. G. de Freitas, and D. A. Forsyth. Object recognition as machine translation: Learning a lexicon for a fixed image vocabulary. In *Seventh European Conference on Computer Vision*, pages 97–112, 2002.

M. F. A. Fauzi and P. H. Lewis. Query by fax for content-based image retrieval. In M. S. Lew, N.Sebe, and J. P. Eakins, editors, *Proceedings of International Conference on the Challenge of Image and Video Retrieval*, pages 91–99. Springer, 2002.

Tom Fawcett. An introduction to ROC analysis. *Pattern Recognition Letters*, 27:861–874, 2006.

Yoav Freund and Robert E. Schapire. Experiments with a new boosting algorithm. In *International Conference on Machine Learning*, pages 148–156, 1996.

Thomas Funkhouser, Patrick Min, Michael Kazhdan, Joyce Chen, Alex Halderman, and David Dobkin. Princeton 3D model search engine. online - http://shape.cs.princeton.edu/search.html [Accessed 2005-04-25], No Year.

Thomas Funkhouser, Patrick Min, Michael Kazhdan, Joyce Chen, Alex Halderman, David Dobkin, and David Jacobs. A search engine for 3D models. *ACM Transactions on Graphics*, 22(1):83–105, January 2003.

A. García-Rojas, D. Thalmann, and F. Vexo. An ontology of virtual humans: Incorporating semantics into human shapes. In *The 2nd European Workshop on the Integration of Knowledge, Semantics and Digial Media Technology (EWIMT 2005)*, pages 7–14, November 2005.

Giorgio Giacinto and Fabio Roli. Methods for dynamic classifier selection. In *10th International Conference of Image Analysis and Processing (ICIAP '99)*, pages 659–665, 1999.

Giorgio Giacinto and Fabio Roli. An approach to the automatic design of multiple classifier systems. *Pattern Recognition Letters*, 22:25–33, 2001.

S. Goodall, P. Lewis, K. Martinez, P. Sinclair, M. Addis, C. Lahanier, and J. Stevenson. Knowledge-based exploration of multimedia museum collections. In *Proceedings of European Workshop on the Integration of Knowledge, Semantics and Digital Media Technology (EWIMT)*, London, UK, 2004a.

Simon Goodall, Paul Lewis, and Kirk Martinez. 3-D shape descriptors and distance metrics for content-based artefact retrieval. In R. W. Lienhart and N. Babaguchi, editors, *Proceedings of Storage and Retrieval Methods and Applications for Multimedia 2005*, pages 87–97, San Jose, California, USA, January 2005a.

Simon Goodall, Paul. H. Lewis, and Kirk Martinez. Towards automatic classification of 3-D museum artifacts using ontological concepts. In W. K. Leow, M. S. Lew, T. S. Chua, M. Y. Chaisorn, and L. Bakker, editors, *Image and Video Retrieval: Fourth International Conference, (CIVR 2005)*, pages 435–444, Singapore, July 2005b.

Simon Goodall, Paul. H. Lewis, Kirk Martinez, Patrick A. S. Sinclair, Fabrizio Giorgini, M. J. Addis, M. J. Boniface, Christian Lahanier, and James Stevenson. SCULPTEUR: Multimedia retrieval for museums. In *Image and Video Retrieval: Third International Conference, (CIVR 2004)*, pages 638–646, Dublin, Ireland, July 2004b.

S. R. Gunn. Support vector machines for classification and regression. Technical report, Image Speech and Intelligent Systems Research Group, University of Southampton, 1997.

Lars Kai Hansen and Peter Salamon. Neural network ensembles. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(10):993–1001, 1990.

Steve Harris, Nick Gibins, Daniel Smith, and Phil Dawes. 3store. online - http://threestore.sourceforge.net/ [Accessed 2006-08-26], No Year.

Simon Haykin. *Neural Networks: A Comprehensive Foundation*. Prentice Hall, 1999. ISBN 0-13-273350-11.

Martin Heczko, Daniel A. Keim, Dietmar Saupe, and Dejan V. Vranić. Verfahren zur Ähnlichkeitssuche auf 3D-objekten. In A. Heuer, F. Leymann, and D. Priebe, editors, *Proceedings of the BTW 2001*, pages 384–401, Oldenburg, Germany, March 2001. Springer Verlag.

Günter Hetzel, Bastian Leibe, Paul Levi, and Bernt Schiele. 3D object recognition from range images using local feature histograms. In *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR'01)*, volume 2, pages 394–399, Kauai Island, Hawaii, December 2001.

Hewlett-Packard Development Company. Jena - a semantic web framework for java. online - http://jena.sourceforge.net [Accessed 2006-08-26], No Year.

Masaki Hilaga, Yoshihisa Shinagawa, Taku Kohmura, and Toshiyasu L. Kunii. Topology matching for fully automatic similarity estimation of 3D shape. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 203–212. ACM Press, 2001.

Tin Kam Ho, Jonathan J. Hull, and Sargur N. Srihari. Decision combination in multiple classifier systems. *IEEE Transactions on Pattern Analysis and machine Intelligence*, 16(1):66–75, January 1994.

B. K. P. Horn. Extended gaussian images. *Proceedings of the IEEE*, 72(12):1671–1686, December 1984.

Chih-Wei Hsu and Chih-Jen Lin. A comparison of methods for multiclass support vector machines. *IEEE Transactions on Neural Networks*, 13(2):415–425, 2002.

D. P. Huijsmans and Nicu Sebe. Extended performance graphs for cluster retrieval. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'01)*, pages 26–31, Kauai, Hawaii, December 2001.

D. P. Huttenlocher, G. Klanderman, and W. Rucklidge. Comparing images using the hausdorff distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(9):850–863, 1993.

ID Software. Quake 3: Arena. online - http://www.idsoftware.com/games/quake/quake3-arena/ [Accessed 2006-08-26], 1999.

S. Impedovo and A. Salzo. A new evaluation method for expert combination in multi-expert system designing. In J Kittler and F. Roli, editors, *Multiple Classifier Systems: First International Workshop, MCS 2000*, pages 230–239, Cagliari, Italy, June 2000.

Natraj Iyer, Subramaniam Jayanti, Kuiyang Lou, Yagnanarayanan Kalyanaraman, and Karthik Ramani. Three-dimensional shape searching: state-of-the-art review and future trends. *Computer-Aided Design*, 37:509–530, 2005.

Kalervo Järvelin and Jaana Kekäläinen. IR evaluation methods for retrieving highly relevant documents. In *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 41–48, Athens, Greece, 2000. ACM Press.

L. K. Jones. Constructive approximations for neural neworks by sigmoidal functions. *Proceedings of the IEEE*, 78(10):1586–1589, 1990.

Michael I. Jordan and Robert A. Jacobs. Hierarchical mixtures of experts and the EM algorithm. *Neural Computing*, 6(2):181–214, 1994.

S. B. Kang and K. Ikeuchi. The complex EGI: a new representation for 3D pose determination. *IEEE Transactions of Pattern Analysis and Machine Intelligence*, 15(7): 707–721, July 1993.

Michael Kazhdan, Bernard Chazelle, David Dobkin, Adam Finkelstein, and Thomas Funkhouser. A reflective symmetry descriptor. In *European Conference on Computer Vision (ECCV)*, May 2002.

J. Kennedy and R. C. Eberhart. A discrete binary version of the particle swarm algorithm. In *Proceedings of the 1997 Conference on Systems, Man and Cybernetics*, pages 4104–4109, IEEE Service Center, Piscataway, NJ, 1997.

K. Kennedy and R. C. Eberhart. Particle swarm optimization. In *Proceedings of IEEE International Conference on Neural Networks*, pages 1942–1948, Piscataway, NJ, 1995.

Sanghee Kim, Paul Lewis, Kirk Martinez, and Simon Goodall. Question answering towards automatic augmentations of ontology instances. In *Proceedings of the First European Semantic Web Symposium*, pages 152–166, Heraklion, Crete, May 2004. Springer.

S. Kirkpatrick, Jr. C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, May 1983.

J. Kittler, M Hatef, and R P W Duin. Combining classifiers. In *Proceedings of the 13th International Conference on Pattern Rocognition*, pages 897–901, 1996.

Josef Kittler, Mohamad Hatef, Robert P. W. Duin, and Jiri Matas. On combining classifiers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(3): 226–239, March 1998.

Satoshi Konno. Cyber X3D. online - http://www.cybergarage.org/vrml/cx3d/cx3dcc/ [Accessed 2005-05-06], 2003.

M. Körtgen, G.-J. Park, M. Novotni, and R. Klein. 3D shape matching with 3D shape contexts. In *The 7th Central European Seminar on Computer Graphics*, Budmerice, Slovakia, April 2003.

L. I. Kuncheva, C. J. Whitaker, C. A. Shipp, and R. P. W. Duin. Is independence good for combining classifiers? In *15th International Conference on Pattern Recognition (ICPR 2000)*, pages 2168–2171, 2000.

Ora Lassila and Ralph R. Swick. Resource description framework (RDF) model and syntax specification. online - http://www.w3c.org/TR/1999/REC-rdf-syntax-19990222, February 1999.

George Leifman, Sagi Katz, Ayellet Tal, and Ron Meir. Signatures of 3D models for retrieval. In *The 4th Israel-Korea Bi-National Conference on Geometric Modeling and Computer Graphics*, pages 159–163, February 2003.

Paul H. Lewis, Kirk Martinez, Fazly Salleh Abas, Mohammad Faizal Ahmad Fauzi, Stephen C. Y. Chan, Matthew J. Addis, Mike J. Boniface, Paul Grimwood, Alison

Stevenson, Christian Lahanier, and James Stevenson. An integrated content and metadata based retrieval system for art. *IEEE Transactions on Image Processing*, 13(3): 302–313, March 2003.

Frank Manola and Eric Miller. RDF primer. W3C recommendation, World Wide Web Consortium, 2004.

José M. Martínez. MPEG-7 overview. Technical Report ISO/IEC JTC1/SC29/WG11, INTERNATIONAL ORGANISATION FOR STANDARDISATION, 2004.

Deborah L. McGuiness and Frank van Harmelen. OWL web ontology language overview. W3C recommendation, World Wide Web Consortium, Februrary 2004.

Alistair Miles and Dan Brickley. Skos core vocabulary specification. W3C working draft, World Wide Web Consortium, November 2005.

Patrick Min, John A. Halderman, Michael Kazhdan, and Thomas A. Funkhouser. Early experiences with a 3D model search engine. In *Proceeding of the eighth international conference on 3D web technology*, pages 7–ff, Saint Malo, France, 2003. ACM Press.

Ian T. Nabney. *Netlab: Algorithms for Pattern Recognition*. Springer, 2002.

Ryutarou Ohbuchi, Takahiro Minamitani, and Tsuyoshi Takei. Shape-similarity search of 3D models by using enhanced shape functions. In *Theory and Practice of Computer Graphics 2003*, pages 97–104, University of Birmingham, UK, June 2003a.

Ryutarou Ohbuchi, Masatoshi Nakazawa, and Tsuyoshi Takei. Retrieving 3D shapes based on their appearance. In *Proceedings of the 5th ACM SIGMM international workshop on Multimedia information retrieval*, pages 39–45. ACM Press, 2003b.

Ryutarou Ohbuchi, Tomo Otagiri, Masatoshi Ibato, and Tsuyoshi Takei. Shape-similarity search of three-dimensional models using parameterized statistics. In *Pacific Conference on Computer Graphics and Applications*, pages 265–274, Beijing, China, October 2002.

Robert Osada, Thomas Funkhouser, Bernard Chazelle, and David Dobkin. Matching 3D models with shape distributions. In *Shape Modeling International*, pages 154–166, Genova, Italy, May 2001.

Eric Paquet and Marc Rioux. The MPEG-7 standard and the content-based management of three-dimensional data: A case study. In *IEEE International Conference on Multimedia Computing and Systems*, pages 9375–9380, Florence, Italy, June 1999a. IEEE.

Eric Paquet and Marc Rioux. Nefertiti: a query by content system for three-dimensional model and image databases management. *Image and Vision Computing*, 17(2):157–166, February 1999b.

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

D. Partridge and W. B. Yates. Engineering multiversion neural-net systems. *Neural Computation*, 8(4):869–893, May 1996.

Eric Prud'hommeaux and Andy Seabourne. SPARQL query language for rdf. W3C recommendation, World Wide Web Consortium, 2006.

Fabio Roli, Giorgio Giacinto, and Gianni Vernazza. Methods for designing multiple classifier systems. In Josef Kittler and Fabio Roli, editors, *Multiple Classifier Systems*, volume 2096, pages 78–87, Cambridge, UK, July 2-4 2001. Springer.

Yossi Rubner, Carlo Tomasi, and Leonidas J. Guibas. A metric for distributions with applications to image databases. In *Sixth International Conference on Computer Vision (ICCV'98)*, pages 59–66, Bombay, India, January 1998. IEEE.

Dietmar Saupe and Dejan V. Vranić. 3D model retrieval with spherical harmonics and moments. In B. Radig and S. Florczyk, editors, *Mustererkennung 2001 (DAGM 2001)*, pages 392–397, Munich, Germany, September 2001.

Robert E. Schapire. The strength of weak learnability. *Machine Learning*, 5:197–227, 1990.

Andy Seabourne. RDQL - a query language for RDF. W3C member submission, World Wide Web Consortium, 2004.

Amanda J. C. Sharkey, Noel E. Sharkey, Uwe Gerecke, and G. O. Chandroth. The "test and select" appraoch to ensemble combination. In J Kittler and F. Roli, editors, *Multiple Classifier Systems: First International Workshop, MCS 2000*, pages 30–44, Cagliari, Italy, June 2000.

Philip Shilane, Patrick Min, Michael Kazhdan, and Thomas Funkhouser. The princeton shape benchmark. In *Shape Modeling International (SMI04)*, pages 167–178, Genova, Italy, June 2004.

P. Sinclair, P. Lewis, K. Martinez, M. Addis, A. Pillinger, and D. Prideaux. eChase: Exploiting cultural heritage using the semantic web. In *Proceedings of the 4th International Semantic Web Conference (ISWV 2005)*, Galway, Ireland, November 2005a.

P. A. S. Sinclair, S. Goodall, P. H. Lewis, K. Martinez, and M. J. Addis. Concept browsing for multimedia retrieval in the SCULPTEUR project. In *Proceedings of The 2nd Annual European Semantic Web Conference*, Heraklion, Crete, 2005b.

Arnold W. M. Smeulders, Marcel Worring, Simone Santini, Amarnath Gupta, and Ramesh Jain. Content-based image retrieval at the end of the early years. *IEEE*

Transactions on Pattern Analysis and Machine Intelligence, 22(12):1349–1380, December 2000.

SRW Editorial Board. Srw: Search and retrieve web service. online - http://www.loc.gov/standards/sru/srw/index.html [Accessed 2006-08-26], 2004.

Motofumi T. Suzuki. Ogden. online - http://ship.nime.ac.jp/ motofumi/Ogden/ [Accessed 2006-07-23], No Year.

Bin Tang, Malcolm I Heywood, and Michael Shepherd. Input partitioning to mixture of experts. In *IEEE - INS International Joint Conference on Neural Networks*, pages 227–232, Honolulu, USA, May 2002.

Johan W. H. Tangelder and Remco C. Veltkamp. A survey of content based 3D shape retrieval methods. In *International Conference on Shape Modeling and Applications 2004*, pages 145–156, Genova, Italy, June 2004.

Neil A. Thacker, F. Ahearne, and P. I. Rockett. The bhattacharyya metric as an absolute similarity measure for frequency coded data. In *Statistical Techniques in Pattern Recognition (STIPR '97)*, pages 363–368, Prague, Czech Republic, June 1997.

The MathWorks Inc. Matlab. online - http://www.mathworks.com/products/matlab/ [Accessed 2004-09-01], No Year.

Tony Tung and Francis Schmitt. Augmented reeb graphs for content-based retrieval of 3D mesh models. In *International Conference on Shape Modeling and Applications 2004*, pages 157–166, Genova, Italy, June 2004.

Dimitrios Tzovaras and Petros Daras. 3D search. online - http://3d-search.iti.gr/3DSearch [Accessed 2006-07-24], 2004.

C. J. van Rijsbergen. *Information Retrieval*. Butterworths, 1975.

R. C. Veltkamp. Shape matching: Similarity measures and algorithms. In *Shape Modelling International (SMI 2001)*, pages 188–197, May 2001.

Dejan V. Vranić. An improvement of rotation invariant 3D shape descriptor based on functions on concentric spheres. *IEEE International Conference on Image Processing (ICIP 2003)*, 3:757–760, September 2003.

Dejan V. Vranić. Content-based classification of 3D-models by capturing spatial characteristics. online - http://merkur01.inf.uni-konstanz.cd/CCCC [Accessed 2005-04-25], No Year.

Dejan V. Vranić and Dietmar Saupe. 3D shape descriptor based on 3D fourier transform. In K. Fazekas, editor, *Proceedings of the EURASIP Conference on Digital Signal Processing for Multimedia Communications and Services (ECMCS 2001)*, pages 271–274, Budapest, Hungary, September 2001a. IEEE.

Dejan V. Vranić and Dietmar Saupe. A feature vector approach for retrieval of 3D objects in the context of MPEG-7. In V. Giagourta and M.G. Strintzis, editors, *International Conference on Augmented, Virtual Environments and Three-Dimensional Imaging*, pages 37–40, Mykonos, Greece, May 2001b.

Dejan V. Vranić and Dietmar Saupe. Description of 3D-shape using a complex function on the sphere. In *Proceedings of the IEEE International Conference on Multimedia and Expo (ICME 2002)*, pages 177–180, Lausanne, Switzerland, August 2002. IEEE.

Dejan V. Vranić, Dietmar Saupe, and J. Richter. Tools for 3D-object retrieval: Karhunen-loeve transform and spherical harmonics. In J.-L. Dugelay and K. Rose, editors, *Proceedings of the IEEE 2001 Workshop Multimedia Signal Processing*, pages 293–298, Cannes, France, October 2001. IEEE.

Web 3D Consortium. The virtual reality modeling language. online - http://www.web3d.org/x3d/specifications/vrml/ISO_IEC_14772-All/ [Accessed 2004-09-01], 1997.

Web 3D Consortium. Extensible 3D (X3D). online - http://www.web3d.org/x3d/specifications/ISO-IEC-19775-FDIS-X3dAbstractSpecification/ [Accessed 2004-09-01], 2004.

Sholom M. Weiss and Casimir A. Kulikowski. *Computer Systems That Learn: Classification and Prediction Methods from Statistics, Neural Nets, Machine Learning and Export Systems.* Morgan Kaufmann, 1990.

Titus Zaharia and Françoise Prêteux. 3d shape-based retrieval within the MPEG-7 framework. In *Proceedings SPIE Conference 4304 on Nonlinear Image Processing and Pattern Analysis XII*, pages 133–145, San Jose, CA, January 2001a. c.

Titus Zaharia and Françoise Prêteux. Hough transform-based 3D mesh retrieval. In *Proceedings SPIE Conference 4476 on Vision Geometry X*, pages 175–185, San Diego, CA, August 2001b.

Titus Zaharia and Françoise Prêteux. Shape-based retrieval of 3D mesh models. In *Proceedings 2002 IEEE International Conference on Multimedia and Expo (ICME'2002)*, pages 437–440, Lausanne, Switzerland, August 2002.

Cha Zhang and Tsuhan Chen. Efficient feature extraction for 2D/3D objects in mesh representation. In *IEEE International Conference on Image Processing (ICIP 2001)*, pages 935–938, Thessaloniki, Greece, October 2001.

D. S. Zhang and G. Lu. A comparative study of fourier descriptors for shape representation and retrieval. In *Proceedings of 5th Asian Conference on Computer Vision (ACCV)*, pages 646–651, Melbourne, Australia, January 2002a.

D. S. Zhang and G. Lu. An integrated approach to shape based image retrieval. In *Proceedings of 5th Asian Conference on Computer Vision (ACCV)*, pages 652–657, Melbourne, Australia, January 2002b.

D. S. Zhang and G. Lu. Shape-based image retrieval using generic fourier descriptor. *Signal Processing: Image Communication*, 17(10):825–848, November 2002c.