UNIVERSIDADE DA BEIRA INTERIOR

# Creating 3D Object Descriptors Using a Genetic Algorithm

**Dominik Węgrzyn**

Dissertação para obtenção do Grau de Mestre em

**Engenharia Informática**

(2º ciclo de estudos)

Orientador: Prof. Doutor Luís Alexandre

**Covilhã, Junho de 2013**

# Acknowledgements

I would like to express my gratitude to my supervisor Luís Alexandre for the useful comments, remarks and engagement through the learning process of this thesis.
Furthermore I would like to thank Paulo Fazendeiro for introducing me 3 years ago to the topic of genetic algorithms, which become my passion.
I would like to thank my loved parents Tomasz Węgrzyn and Malgorzata Węgrzyn, my girfriend Cláudia Saraiva and all my friends, who have supported me throughout the entire process, by keeping me harmonious and helping me putting pieces together. I will be grateful forever for your help and support.

# Abstract

In the technological world that we live in, the need for computer vision became almost as important as human vision. We are surrounded be all kinds of machines that need to have their own virtual eyes.
The most developed cars have software that can analyze traffic signs in order to warn the driver about the events on the road. When we send a space rover to other planet it is important that it can analyze the ground in order to avoid obstacles that would lead to its destruction.
There is still much work to be done in the field of computer vision with the view to improve the performance and speed of recognition tasks. There are many available descriptors used for 3D point cloud recognition and some of them are explained in this thesis.
The aim of this work is to design descriptors that can match correctly 3D point clouds. The idea is to use artificial intelligence, in the form of a GA to obtain optimized parameters for the descriptors. For this purpose the PCL [RC11] is used, which deals with the manipulation of 3D points data. The created descriptors are explained and experiments are done to illustrate their performance. The main conclusions are that there is still much work to be done in shape recognition. The descriptor developed in this thesis that use only color information is better than the descriptors that use only shape data. Although we have achieved descriptors with good performance in this thesis, there could be a way to improve them even more. As the descriptor that use only color data is better than the shape-only descriptors, we can expect that there is a better way to represent the shape of an object. Humans can recognize better objects by shape than by color, what makes us wonder if there is a way to improve the techniques used for shape description.

# Contents

# List of Figures

# List of Tables

# Acronyms

| | |
|---|---|
| GA | Genetic Algorithm |
| PCL | Point Cloud Library |
| PFH | Point Feature Histograms |
| FPFH | Fast Point Feature Histograms |
| SHOT | Signature of Histograms of Orientations |
| USC | Unique Shape Context |
| 3DSC | 3D Shape Context |
| ESF | Ensemble of Shape Functions |
| RIFT | Rotation Invariant Feature Transform |
| CVFH | Clustered Viewpoint Feature Histogram |
| PPF | Point Pair Feature |
| PCE | Principal Curvatures Estimation |
| VFH | Viewpoint Feature Histogram |
| HSV | Hue Saturation Value |

# Chapter 1

# Introduction

The possibilities to use the depth information that cheap 3D cameras like Xtion and the Kinect produce have increased exponentially the interest in using this 3D data for solving vision tasks. A 3D point cloud is a set of data points in a 3D coordinate system that can have additional information like color or normals.
A point cloud descriptor is a description of the 3D data in a point cloud. It describes elementary characteristics such as the shape and the color. Although we have many available descriptors, the ones with good performance tend to be slow, while the fast descriptors have bad performance. The aim of this thesis is to solve this problem by creating a fast descriptor, with similar performance to the best descriptors.
This project has two components. The first component is a genetic algorithm. Darwin [Dar59] taught us how the strongest and best members of the population can influence the faith of a civilization. Genetic algorithms are inspired in the theory of Darwin about the natural evolution. The genetic algorithms are a powerful tool that can find solutions for optimization problems, they are part of artificial intelligence. The second part of the project consists in recognizing objects in 3D point clouds. It is a big challenge to describe a point cloud in numbers that the computer can understand in order to choose correct matches for already known clouds. Descriptors are very important to distinguish shapes and colors in order to recognise the already known objects. There are many descriptors available for 3D object recognition and some of them [Ale12] are described in this thesis. To create a good descriptor we have to take in account a vast quantity of parameters and values that these parameters can take. It isn't easy to know exactly what values the parameters should have to generate good matches. In order to solve this problem we use the GA [Hol75], that can easily find near optimal solutions or even an optimal solution. The genetic algorithm generates new descriptors during the validation phase and then these descriptors are tested to get their performance. There is still much work to be done in this area. This makes this project very interesting because there are a lot of topics to learn and yet many to be discovered.

## 1.1 Objectives

The concept of optimization is associated to the determination of one or more possible solutions, which represent the extreme values of one or more fitness functions. Optimization techniques have a big relevance to the resolution of practical problems. When an optimization model represents one real system that involves just one function, it is called a mono objective problem. The problems that we are facing nowadays, in a technologically advanced world are complex. Sometimes the determination of optimal solutions may even be impossible and many times what we are looking for is just a good enough approximate solution.
The main goal of this project is to obtain descriptors for 3D point clouds, using a genetic algorithm to obtain the parameters that will be used to build histograms to represent the point clouds. The descriptors will be tested in a large dataset of point clouds in order to evaluate their performance and compare them with other already existing descriptors.

## 1.2 Organization of the Thesis

The rest of this thesis is divided into 5 chapters organized in the following way:

- ***Chapter 2 - State of the Art:*** This chapter describes the already existent descriptors that will serve as a base for this work.
- ***Chapter 3 - Genetic Algorithm:*** This chapter is dedicated to describe what are GA, how they function and how were used to optimize the parameters of the GA.
- ***Chapter 4 - Descriptor:*** This chapter describes what techniques were used to create the descriptors. In this chapter we have the explanation on how the shape and color information was used to create histograms capable of describing 3D point clouds.
- ***Chapter 5 - Experiments:*** This chapter explains the structure of 7 different descriptors created for this thesis. Every version is described and tested to obtain its accuracy.
- ***Chapter 6 - Conclusion:*** The last chapter contains the conclusions and some ideas for future work.

# Chapter 2

# State of the Art

The Point Cloud Library (PCL [RC11]) is a standalone, large scale, open project for 2D/3D image and point cloud processing. The PCL framework contains numerous state-of-the art algorithms. These algorithms can be used to extract keypoints and compute descriptors to recognize objects in the world based on their geometric appearance. As the goal of this thesis is to create descriptors for 3D point clouds, the next section describes the available descriptors in the PCL. This chapter is based on the paper [Ale12].

## 2.1 Available Descriptors

The 3D Shape Context (3DSC) descriptor was proposed in [AFM04]. It uses a spherical grid on each of the keypoints. The north pole of the grid is oriented as the surface normal at the keypoint. The grid is constituted by bins along the radial, azimuth and elevation dimensions. The divisions along the radial dimension are logarithmically spaced. Each bin makes a weighted count of the number of points that fall into it. The weights used are inversely proportional to the bin volume and the local point density. Since the axes tangent to the surface are placed randomly, there is the need to extract as many versions of this descriptor per database object as there are divisions along the azimuth direction.

The Unique Shape Context (USC) [usc10] descriptor is an upgrade of the 3DSC with the goal of avoiding the need to obtain as many versions of the descriptor as the number of azimuth bins. Consider a point $p$ with a spherical neighborhood of radius $r$. A weighted covariance matrix $M$ of the points in the neighborhood is computed as presented in equation 2.1.

$$M = \frac{1}{Z} \sum_{i:d_i \leq r} (r - d_i)(p_i - p)(p_i - p) \qquad (2.1)$$

The eigenvector decomposition of $M$ is used to obtain the 3 unit vectors of the local reference frame. The sign of the eigenvectors with the biggest and smallest eigenvalues is changed so that it is coherent with the majority of the vectors they represent. The sign of the third eigenvector is obtained from the other two considering that they must form an orthonormal base. The eigenvector with the smallest eigenvalue gives the normal direction. Apart from this reference frame determination process, the USC descriptor is obtained like the 3DSC.

The Rotation Invariant Feature Transform (RIFT) descriptor [SLP05] is a generalization of the SIFT descriptor [Low04]. A circular normalized patch is built at each input point. The circular patch is divided into 4 rings of equal width. For each ring, a histogram of gradient orientations with 8 bins is computed, thus producing a 32 value descriptor for each input point. The orientations of this histogram are obtained with reference to the radial outward at each point.

The Point Pair Feature (PPF) [BDI10] descriptor given two points $p_1$ and $p_2$ and their normals $n_1$ and $n_2$ is given by equation 2.2, where $\angle(a, b) \in [0, \pi]$ and represents the angle between $a$ and $b$ and $d = p_2 - p_1$.

$$PPF(p_1, p_2) = (\|d\|^2, \angle(n_1, d), \angle(n_2, d), \angle(n_1, n_2)) \qquad (2.2)$$

The PPF is found for all pairs of points. The distances are sampled in $d_{dist}$ steps and the angles in $d_{angle} = \frac{2\pi}{n_{angle}}$ steps and the vectors with the same discrete representation are grouped. Consider $O_m$ to be an object model and $I_s$ to be the input scene. A global model descriptor is a mapping from the sampled space to the model space. Consider that an arbitrary reference point $r_p \in I_s$ is chosen and assume it lies on a given object $O_m$. Then there is a point $o_p \in O_m$ that corresponds to $r_p$. If $o_p$ and its normal are aligned with $r_p$ and its normal, it is possible to align the model to the scene with a further rotation $\alpha$ around the aligned normal. The pair $(o_p, \alpha)$ is called the local coordinates of the model with reference to $r_p$. A point pair $(o_p, o_i) \in O_m$ is aligned to a scene pair $(r_p, r_i) \in I_s$ that has the same feature vector. Then the local model coordinates are transformed to scene coordinates. To find the local coordinates that maximize the number of scene points that lie on the model, a voting scheme is used. This optimal local coordinate system allows the recovery of the global object pose. The poses obtained by the voting scheme are clustered according to how similar their rotations and translations are. The final pose is the one that corresponds to the average pose of the cluster with the largest sum of votes obtained by its members in the voting scheme.

The Point Feature Histograms (PFH) [RRB08] descriptor's goal is to generalize both the surface normals and the curvature estimates. Given two points, $p$ and $q$, a fixed reference frame consisting of the three unit vectors $(u, v, w)$ is built centered on $p$ using the following procedure:

- The vector $u$ is the surface normal at $p$
- $v = u \times \frac{p-q}{d}$, where $d = \|p - q\|^2$
- $w = u \times v$

Using this reference frame, the difference between the normals at $p(n_p)$ and $q(n_q)$, can be represented by :

- $\alpha = \arccos(v \cdot n_q)$
- $\phi = \arccos(u \cdot (p - q)/d)$
- $\theta = \arctan(w \cdot n_p, u \cdot n_p)$

The angles $\alpha$,$\phi$,$\theta$ and the distance $d$ are computed for all pairs in the k-neighborhood of point $p$. In fact, usually the distance $d$ is dropped as it changes with the viewpoint, keeping only the 3 angles. The angles are inserted into an 125-bin histogram by considering that each of them can fall into 5 distinct bins, and the final histogram encodes in each bin a unique combination of the distinct values for each of the angles. One of these 125-bin histograms is produced for each input point. The PFHRGB is a version of PFH that includes color information, that includes three more histograms, one for the ratio between each color channel of $p$ and the same channel of $q$. These histograms are saved in same way as the 3 angles of PFH and hence produce another 125 float values, giving the total size of 250 bins for PFHRGB.

The Fast Point Feature Histograms (FPFH) [RRB09b] is a simplification of the PFH descriptor that reduce the computational complexity of the PPF algorithm from $O(nk^2)$ to $O(nk)$. The first step is to compute the histogram of the three angles between a point $p$ and its $k$-nearest neighbors (instead of all pairs of neighbors) in the same way as in PPF. This produces the Simplified Point Feature Histogram (SPFH). Then, for each point $p$, the values of the SPFH of its $k$ neighbors are weighted by their distance $w_i = d$ to $p$ to produce the FPFH at $p$ (equation 2.3).

$$FPFH(p) = \frac{1}{k} \cdot \sum_{i=1}^{k} \frac{SPFH(i)}{w_i} \qquad (2.3)$$

The three angles are stored in 11-bin histograms that are concatenated into a single 33-bin FPFH descriptor. In [RRB09a], the authors found that using a different weighting scheme improves the recognition rates (equation 2.4).

$$w_i = \sqrt{\exp \|d\|} \qquad (2.4)$$

The Viewpoint Feature Histogram (VFH [RBRH10]) adds viewpoint variance to the FPFH by using the viewpoint vector direction. It also produces only one descriptor for the input point cloud (it is a global descriptor). The process is the following:

- Find the input cloud centroid $c$
- For each point $p$ in the cloud, build the local reference frame $(u, v, w)$, using
  - $u = n_c$
  - $v = (p - c) \times u$
  - $w = u \times v$
- Find the angles $(\alpha, \phi, \theta)$ as in the PFH, using this reference frame.

Each of the three angles is binned into a 45-bin histogram. The angle $\beta = \frac{\arccos(n_p \cdot c)}{\|c\|}$ that is the central viewpoint direction translated to each normal makes with each point's normal is also encoded in a 128- bin histogram.

The Clustered Viewpoint Feature Histogram (CVFH) descriptor for a given point cloud dataset containing $XYZ$ data and normals, was proposed in [AAB11]. Stable regions are obtained by first removing the points with high curvature and then applying a smooth region growing algorithm. The CVFH is obtained using the following steps

- Determine the set $S$ of stable regions
- For each $s_i \in S$, find the centroid $c$ and its normal $n_c$
- Build a local reference frame $(u_i, v_i, w_i)$ like in the VHF but using $c$ and $n_c$ instead of the centroid and respective normal for the whole input cloud
- Find the histograms of the angles $(\alpha, \phi, \theta, \beta)$ as in VHF (the first 3 coded as 45-bin histograms and $\beta$ coded as a 128-bin histogram)
- Find the Shape distribution Component (SDC) (equation 2.5).

$$SDC = \frac{(c - p_i)^2}{\max{(c - p_i)^2}}, i = 1, ..., |S| \qquad (2.5)$$

The CVFH is given by the concatenated histograms $(\alpha, \phi, \theta, SDC, \beta)$ which is a 308-bin histogram.

The Signature of Histograms of Orientations (SHOT descriptor [FTS10]) is based on obtaining a repeatable local reference frame using the eigenvalue decomposition around an input point. Given this reference frame, a spherical grid centered on the point divides the neighborhood so that in each grid bin a weighted histogram of normals is obtained. The descriptor concatenates all such histograms into the final signature. It uses 9 values to encode the reference frame and the authors propose the use of 11 shape bins and 32 divisions of the spherical grid, which gives an additional 352 values. The descriptor is normalized to sum 1. There is also a color version (SHOTCOLOR) proposed in [sho11] that adds color information (based on the CIELab color space) to the SHOT descriptor resulting in a 1344 value descriptor (plus 9 values to describe the local reference frame).

The Ensemble of Shape Functions (ESF) is a global shape descriptor proposed in [WV11] consisting of 10 concatenated 64-bin histograms resulting in a single 640 value histogram for a given input point cloud. It is based on three shape functions [ROD01] describing distance, angle and area distributions. The descriptor also use an idea from [CYIR02] that is to classify each of the values into three classes based on where the connecting lines between points reside (on the object surface, off the surface and mixed (partly on and off)).

The Principal Curvatures Estimation (PCE) descriptor calculates the directions (eigenvectors) and magnitudes (eigenvalues) of principal surface curvatures on each keypoint. It produces a 5 value descriptor on each point (3 values for the principal curvature, which is the eigenvector with the largest eigenvalue, plus the largest and smallest eigenvalues. These are obtained using the cloud normals).

## 2.2 Performance of the Available Descriptors

Table 5.16 shows the performance of the descriptors mentioned above, that were extracted using the same point clouds as the used to evaluate the descriptors proposed in this thesis ([Ale12]).

Table 2.1: Available Descriptors Performance: test set errors.

| Descriptor | Object (%) | Category (%) |
|---|---|---|
| PFHRGB | 20.25 | 5.27 |
| SHOTCOLOR | 26.58 | 9.28 |
| PFH | 44.51 | 11.76 |
| FPFH | 52.74 | 13.92 |
| SHOT | 53.16 | 11.81 |
| USC | 54.22 | 17.97 |
| 3DSC | 59.49 | 21.94 |
| ESF | 62.66 | 18.35 |
| RIFT | 75.95 | 33.68 |
| CVFH | 80.59 | 48.95 |
| PPF | 82.91 | 47.68 |
| PCE | 83.55 | 52.74 |
| VFH | 94.73 | 78.27 |

The best descriptors (PFHRGB - 79.75% correct object matches and SHOTCOLOR - 74.42% correct object matches) are far better than the other descriptors. However these two descriptors are the only descriptors that use color information, the other use only shape data. Between the descriptors that use only shape information the best is PFH that matches correctly 56.49% of clouds. The PFHRGB descriptor has the best performance, however it is the slowest computationally. On the other hand the SHOTCOLOR descriptor is less accurate than PFHRGB, but in compensation is much faster. The goal of this thesis is to create a new descriptor, that uses shape and color information, that is even faster than the SHOTCOLOR but has at least a similar performance.

# Chapter 3

# Genetic Algorithm

## 3.1 Introduction

Evolution is a process where only some, the most capable individuals survive. Genetic algorithms can simulate this approach in both search or optimization problems. In the Nature, the organisms have specific characteristics that influence their capacity to survive and to reproduce. These characteristics are encoded in chromosomes. After the reproduction, the chromosomes of the offsprings have a combination of the information contained in its parent chromosomes. We hope that the offspring chromosomes get the best characteristics from its parents. The process of natural selection ensures that the best individuals have more opportunities to reproduce and this way pass their best chromosomes to the next generation. Occasionally the chromosomes are exposed to mutations that modify them randomly. The mutation can affect badly the capacities of individuals but can as well make them even better. Without mutation, the population of a certain species would tend to converge to a state where all individuals would have small differences between each other. The natural selection process can be seen as a search process in the space of the possible chromosome values. This search is guided by the fitness of the individuals that hold the chromosomes necessary to survive and reproduce. Figure 3.1 represents the scheme of a GA.

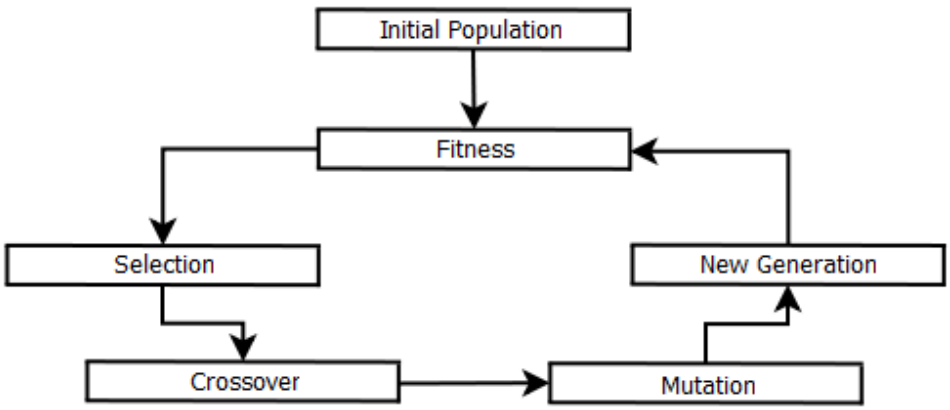


Figure 3.1: Genetic Algorithm

Elements that affect a GA:

1. The codification of the problem solutions as chromosomes
2. A function that evaluates the fitness of individuals
3. The initialization of the population
4. Selection operators
5. Reproduction operators

Genetic algorithms are classified as evolutionary algorithms, which use the same techniques as the natural evolution, such as selection, mutation and crossover to get the solution [Gwi06]. A GA uses a population of individuals, where each individual represents one possible solution for the problem. The chromosome of each individual is constituted by a set of genes and can be seen as a point in the space of search. In optimization terms, each gene represents one optimization parameter of the problem.
In this work a genetic algorithm [Hol75], tries to find the optimal parameters for a 3D cloud descriptor in the vast space of possibilities. The chromosomes encode some characteristics depending on the version (chapter 5). Some of the most used genes are the number of bins used to determine the size of the shape and color histograms and genes for radius, which is the neighbourhood around each keypoint of the original cloud inside which the points of the cloud are used to make shape and color histograms. The genetic algorithm in some versions also encodes weight parameters, that are used to give a certain weight to the shape histogram and color histogram distances, in order to get the final distance between two clouds. In other cases two different versions of descriptors are merged and for this purpose the chromosome has a weight for each descriptor. The chromosomes are binary encoded.

## 3.2 Initial Population

In order to start the evolutionary process we need an initial population. The normal way to initiate a population is to set random values for all genes. The size of the initial population has implications at two levels. A small population sometimes can't be representative of the entire search space. However in this case the computational effort for each generation is lower, the algorithm can take longer to converge to the optimal solution. A big population implies a bigger computational cost for each generation, but can converge using less generations. One possible solution is to use a small population and use a relatively high mutational ratio to assure that a bigger part of the search space of the problem is explored [Eng02].
In this work the initial population is set randomly. The population used was 6 and 10 chromosomes, depending on the version. Each chromosome holds genes that will be used to create the shape and the color histograms. All chromosomes with their genes values are evaluated using validation and training point clouds to establish the object error and the category error of each chromosome.

## 3.3 Fitness

The fitness function is probably the most important part of a GA. Its function consist in evaluate each chromosome as a real number. The fitness function tells us the quality of a chromosome, in other words we know how far he is from the optimal solution. In this project the object error is used as the fitness of the chromosome. The object error represents the percentage of correctly matched point clouds from the training set 1 among the point clouds from training set 2.

## 3.4 Selection

Natural selection is a process resulting in the survival of those individuals from a population of animals or plants that are best adapted to the prevailing environmental conditions. The survivors tend to produce more offsprings than those less well adapted species, so that the characteristics of the population change over time, thus accounting for the process of evolution [dic09]. This process is quite similar in GA. Due to the ability of selection, it is possible to keep the best chromosomes for future generations in order to discover the optimal solution. It also prevents eventual good chromosomes from being lost in the population. If we let only some individuals go to the next generation, it is possible to give more importance to the individuals that are close to the optimal solution. On the other hand, we need to choose which individuals will reproduce through crossover and mutation. This process of choosing the individuals that shall pass to the following generation and will reproduce is called selection [Eng02].
There are many methods used for selection. In this project the elitism selection and the roulette-wheel selection were used [Eng02]. Elitism consists in the selection of an individual or a group of individuals that will survive for the next generation without suffering crossover and mutation. In this case only the best chromosome of the population is selected. The roulette-wheel selection, also known as proportional selection is made according to the fitness of each individual. The probabilistic proportional distribution is calculated in equation 3.1, where $P(i)$ is the probability of the individual $i$ to be selected, and $N$ is the size of population.

$$P(i) = \frac{object_{error}(i)}{\sum_{k=1}^{N} object_{error}(k)} \qquad (3.1)$$

The probability of choosing an individual is bigger if its fitness is bigger. In order to select a chromosome using the roulette-wheel we need to follow the following steps [Eng02]:

1. Generate a random number $r$ inside the interval $(0, 1)$
2. $i = 1$
3. $s = P(i)$
4. while ($s < r$)
   (a) $i++$
   (b) $s+ = P(i)$
5. The selected chromosome is the $i$ that we obtain when we exit the while loop

## 3.5 Crossover

In GA, crossover is an operator used to obtain new genetical material using the parents genes. It is analogous to reproduction and biological crossover, upon which genetic algorithms are based. The purpose of reproduction is to obtain new individuals, using the selected ones. This way crossover is the process that involves the creation of a new individual using the genes of its parents [Eng02].
In order to make the crossover, new offsprings are generated in a number equal to $initial_{population} - 1$, using the selected chromosomes by the roulette-wheel selection. The other chromosome is the individual with best fitness encountered so far (elitism). Like in the selection, we also have many methods to make the crossover. In this project for every two selected chromosomes we generate a bit mask, with $0's$ and $1's$, that are set randomly. For all bits of the mask that are 1 we switch those bits between the two selected "parent" chromosomes making two new offspring chromosomes. This process is called uniform crossover [Eng02] and is illustrated in figure 3.2.

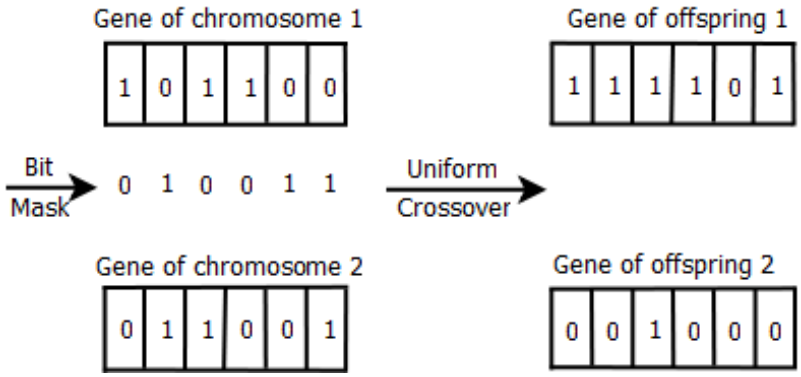


Figure 3.2: Uniform Crossover

## 3.6 Mutation

Mutation is also inspired the nature and it is a sudden departure from the parent type in one or more heritable characteristics, caused by a change in a gene or a chromosome [dic09]. In GA the mutation changes randomly the genes of the existing individuals to produce new individuals. The purpose of mutation is to keep the search space open to new possibilities, but should occur with a low probability. If the mutation happens with a high probability it changes a big part of the genetic information, and this is almost never a positive thing [Eng02].
Mutation is very important to introduce new genetic material in a random way. It makes possible to search a wide area of solutions in order to find the optimal solution. In this work the probability of mutation which is known as the mutational ratio ($P_m$) has big values at the beginning and then it decrease as the system evolves. This way at the beginning we expand our searches and later when the chromosomes start to converge to the optimal solution we avoid making big changes. The mutation used is known as in order mutation [Eng02].

This mutation consists in selecting two random positions from the chromosomes that may suffer from mutation and change its values (if the bit is 0 it turns to 1, otherwise it turns to 0). Figure 3.3 illustrates this process.

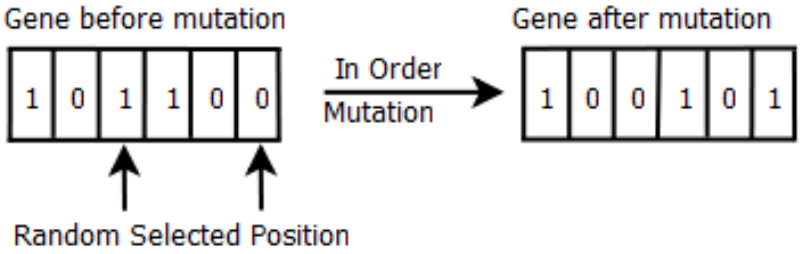


Figure 3.3: In-Order Mutation

## 3.7 Stopping Criterion

The GA needs to be able to stop the evolution when the goal is achieved. For this purpose we have the following convergence criterions [Eng02]:

1. Stop the evolution when a previously established maximal generation number is reached.
2. Stop when we find an individual with a sufficiently high fitness
3. Stop if the maximum fitness or the average doesn't change much during many generations.

The stopping criterions used in the genetic algorithm of this thesis are a mixture of the above. The stopping criterion is set to either the number of generations reaching 200 generations, or if no better solution in 40 generations in a row is found, or if a solution with 100% of correct object matches is found. After each generation we measure the validation error of the best chromosome. This way we avoid the overfitting of the descriptor, that could lead to the loss of the ability that the descriptor has to recognize point clouds. For this purpose we check in a validation subset of point clouds (apart from the clouds used by AG to determine the best parameters for the descriptor) for the validation error of the best chromosome. When this error begins to rise, we stop the AG and consider that we have found the best descriptor.

# Chapter 4

## Descriptor

In order to create a good 3D point cloud descriptor we need to represent the point cloud in a way that the machine can analyze and decide what is the best match for it. In this work the experiments have been done using shape and color information of each point cloud. In order to represent this information histograms were used. In statistics, a histogram is a graphical representation of the distribution of data. It is an estimate of the probability distribution of a random variable. A histogram is a representation of tabulated frequencies, shown as adjacent rectangles, erected over discrete intervals (bins), with an area equal to the frequency of the observations in the bin. Histograms are used in this work to record both angle and color occurrences. The histograms are also normalized displaying relative frequencies. This way the histograms show the proportion of keypoints that fall into each bin. The histograms are normalized to sum 1.
During the matching process, the point clouds with histograms that have the smallest distance between them are selected as the best match. The generic object recognition pipeline used in this project is presented in figure 4.1.

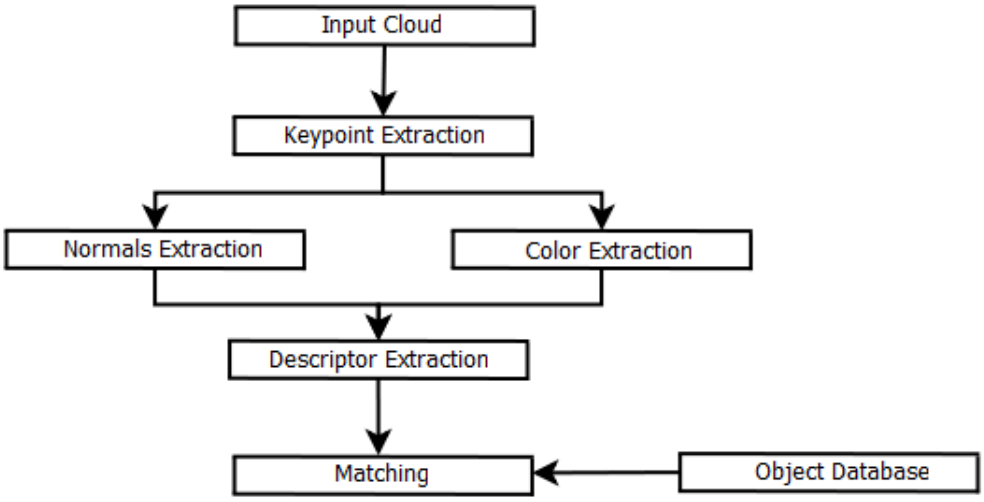


Figure 4.1: Object Recognition Pipeline

## 4.1 Keypoints

In order to create the shape histogram, the first step consists in finding the keypoints of each point cloud. The keypoints are extracted from the input clouds in order to reduce the processing cost. If we wanted to analyze all the points from a cloud it would take too much time to compute the descriptors. The keypoint cloud represents the input point cloud by containing only points considered to be important and that are used to describe the entire cloud. The keypoint cloud is obtained from the input cloud. For this purpose the input cloud was sub-sampled using VoxelGrid with leaf size with 2 $cm$ and all sub-sampled points become our keypoints.

VoxelGrid is used to downsample a point cloud. We reduce the number of points using a voxel grid, that can be seen as a set of tiny 3D boxes in space. Then, in each voxel all the present points will be approximated with their centroid. This approach is a bit slower than approximating them with the center of the voxel, but it represents the underlying surface more accurately.

After computing all keypoint clouds we need to calculate the normals of both, the input and the keypoint clouds. To compute normals of the keypoint cloud the search surface used is the input cloud and the search is made within radius with 1 $cm$. The search surface is used in order to have more points to calculate the normals of the keypoints. If we only wanted to use the keypoints to compute the normals, we would have lack of information. For the normals of the input cloud a 1 $cm$ radius is also used. Both normals are calculated using PCL OMP Normal Estimation. The normals are used to build the shape histogram used in the descriptors. Figure 4.2 presents a point cloud with its keypoint marked in black.

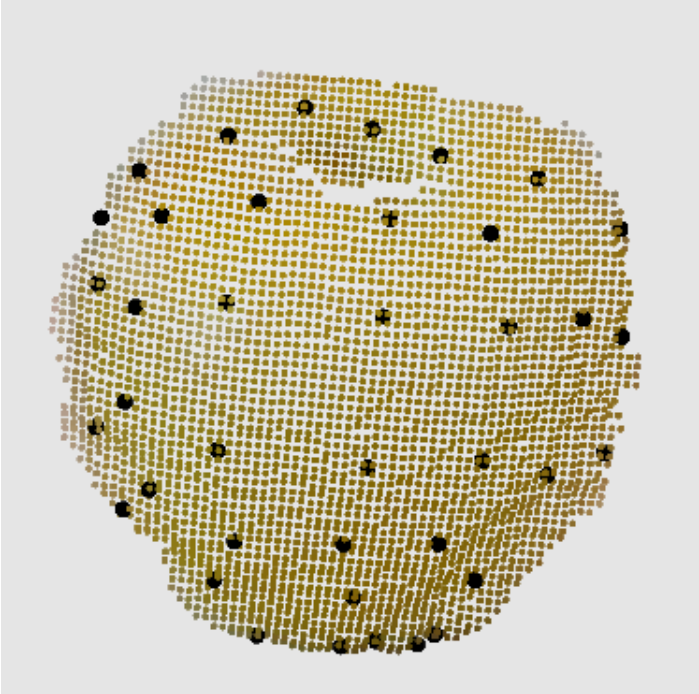

Figure 4.2: An example of keypoints (the black dots) extracted from a point cloud using the VoxelGrid approach.

## 4.2 Shape Histogram

After computing the keypoint clouds and all normals, for each keypoint we search all its neighbors inside a given $shape_{radius}$, which is one of the parameters optimized by the genetic algorithm. This search is done in the original input cloud and for this task the PCL utility KdTreeFLANN is used.

The next step consists in looking for each neighbor and finding the $angle_{degree}$ between the normal of this neighbor and the normal of the keypoint. Equation 4.1 shows how to calculate the $angle_{radian}$.

$$angle_{radian} = \arccos\left(\frac{Normals_{keypoint} \cdot Normals_{neighbor}}{||Normals_{keypoint}|| \cdot ||Normals_{neighbor}||}\right) \quad (4.1)$$

As the angle comes in radians, there is a need to transform it into degrees using the equation 4.2.

$$angle_{degree} = \frac{angle_{radian} \cdot 180}{\pi} \quad (4.2)$$

The selected $bin$ is calculated in equation 4.3. The $shape_{bins}$ is the total number of bins. This parameter is also a gene optimized by the GA.

$$bin = \frac{angle_{degree} \cdot shape_{bins}}{360} \quad (4.3)$$

After we have calculated all $angle_{degree}$ and their corresponding $bin$, we get an histogram, which has the number of $angle_{degree}$ that belongs to each $bin$. This number of $angle_{degree}$ in each bin is then normalized to sum 1. This shape histogram becomes one part of the final descriptor.
Using the same process from the first shape histogram, another shape histogram is created. Although the same process is used, now we have another bigger radius, $shape_{radius2}$ (this radius is also one gene from the chromosome of the GA population). Only the neighbors of the keypoint that are between the $shape_{radius}$ and $shape_{radius2}$ are used to make the histogram (this way we use a ring to obtain the second shape histogram [TFB12] as we can see in figure 4.3).

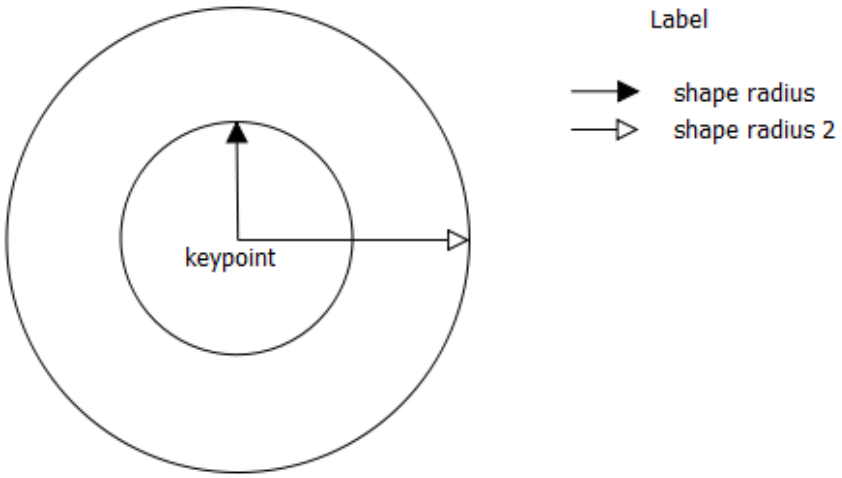


Figure 4.3: Ring

The advantages of using the ring is that it makes possible analyze two areas around the keypoint to create the histograms. The ring separates the shape information between points that are very close to the keypoint and the points that are far from them, making possible to obtain a better representation of the region around the keypoints.
Figure 4.4 presents an example of a shape histogram produced by $shape_{radius} = 1.3cm$ and fig.4.5 represents the ring histogram produced by $shape_{radius} = 3.6cm$ . The $shape_{bins} = 59$ for both histograms.

These histograms where taken from one of the descriptor versions made for this thesis where the normalization is made to sum 1 using the two histograms, this means that the sum of the relative frequencies from both histograms equals 1.

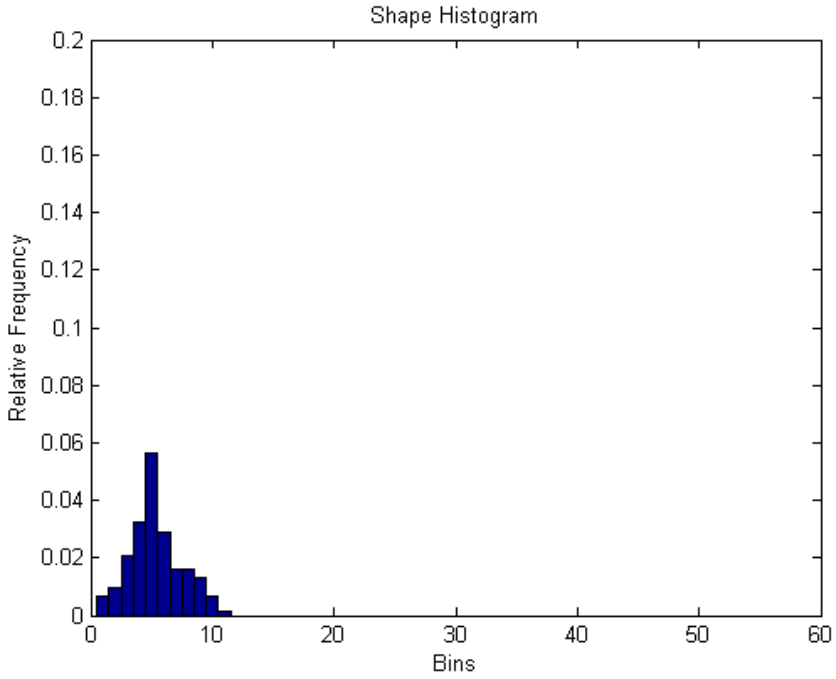


Figure 4.4: Shape Histogram 1

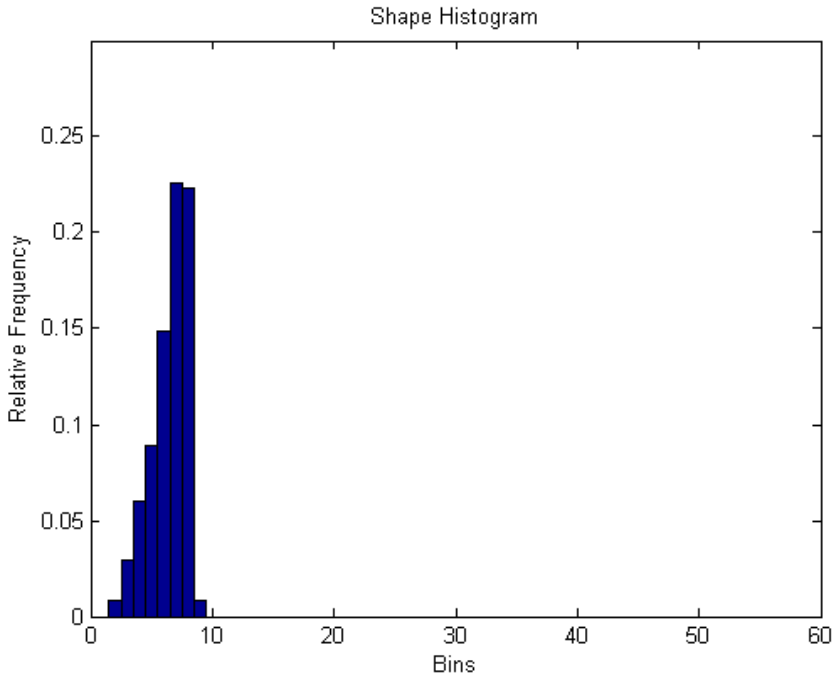


Figure 4.5: Shape Histogram 2

As we can see $\approx 20\%$ of angles fall to the histogram produced by the smaller $shape_{radius} = 1.3cm$ versus $\approx 80\%$ of angle that fall to the histogram produced by the bigger $shape_{radius} = 3.6cm$. The histogram shows that the area near the keypoint has small differences between the angles of the normals between

the keypoints and their neighbors. On the other side the region covered by the ring has bigger variations between the angles.

## 4.3 Color Histogram

After doing experiences with only shape histograms it become clear that it is too few information for the cloud descriptor. The next step in order to create the descriptor consists in adding color information. For this purpose the RGB(Red Green and Blue) colors are transformed into HSV(Hue Saturation and Value) colors, where hue is an angle from 0°to 360°and represents the color, while the saturation is measured in percentage and represents the greyscale. This model is used because with the HSV color space we can use only the H and S channels and obtain illumination invariance (other choices for the color space that have this property could have been used). Two histograms are created to add color information, one for the hue and other for saturation. In order to get the hue and saturation histograms, the process is similar to the shape histogram. In the case of the hue histogram, for each keypoint we search its neighbors inside a $hue_{radius}$ (which is a parameter from the GA) using the input cloud as the search surface (again using KdTreeFLANN). This way we get the $hue$ values of the neighbors. The next step consists in adding these hue values from all the neighbors of keypoints into the hue histogram.
The selected hue $bin_h$ is calculated using equation 4.4, where the $hue_{bins}$ is the total number of hue bins. This parameter is optimized by the GA.

$$bin_h = \frac{hue \cdot hue_{bins}}{360} \qquad (4.4)$$

After we have calculated all $bin_h$, we get an histogram, which has the number of $hue$ values that belong to each $bin_h$. The histogram is then normalized to sum 1. This hue histogram becomes another part of the descriptor.
For the selection of the $bin_s$ for the saturation histogram, the process is quite similar (equation 4.5), where $saturation_{bins}$ is the total number of saturation bins (generated by the GA) and the $saturation$ is extracted from the neighbors of the keypoint, in the same way as for hue and using $saturation_{radius}$ as the radius to search for neighbors.

$$bin_s = \frac{saturation \cdot saturation_{bins}}{100} \qquad (4.5)$$

After we have calculated all $bin_s$, we get the saturation histogram that is also normalized to sum 1. This saturation histogram becomes another part of the descriptor.

Figure 4.6 presents an example of a hue histogram produced by $hue_{radius} = 5cm$ and figure 4.7 is the saturation histogram produced by $saturation_{radius} = 5cm$ on the same keypoint. The $hue_{bins} = 32$ and $saturation_{bins} = 32$.

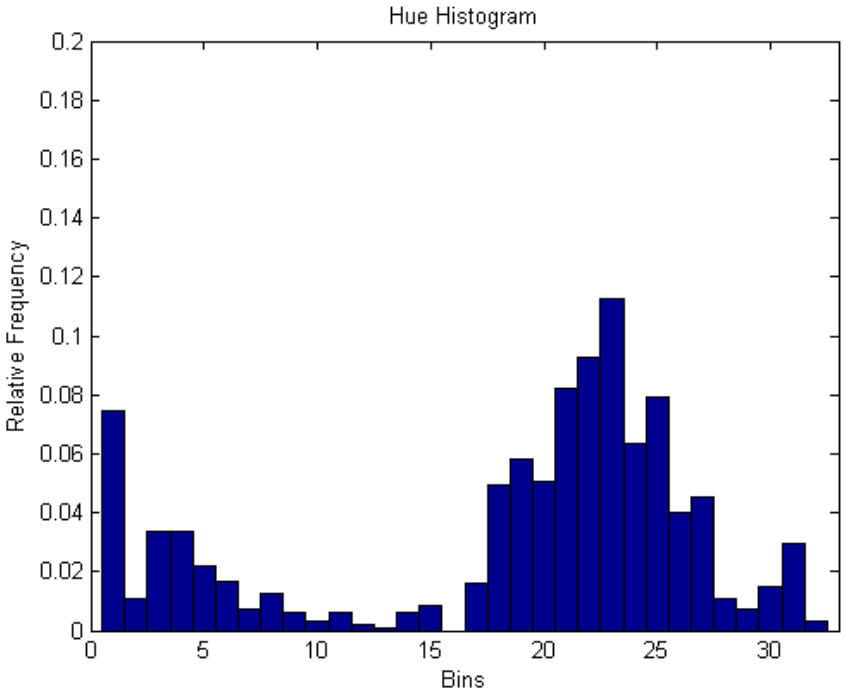


Figure 4.6: Hue Histogram

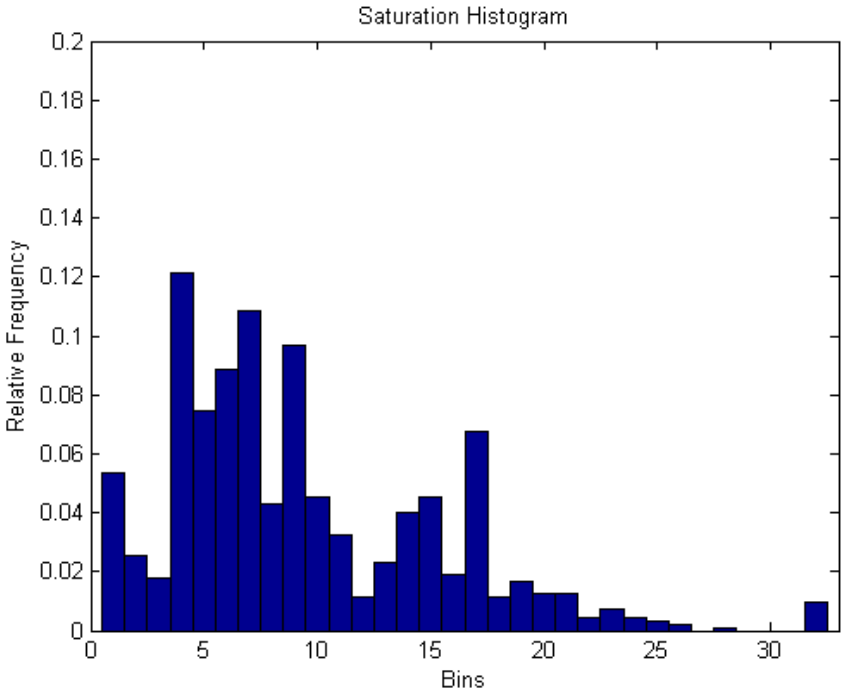


Figure 4.7: Saturation Histogram

In some versions (section 5.3 and 5.4) only this technique was used to create the hue and saturation histograms.
A further version was developed that has the same addiction that the shape histogram and uses the ring technique to create the histograms (section 5.7). In this case the GA has one more gene to optimize, which is the second color radius.
Other versions used a different technique to create the color histogram (section 5.5, the color part of section 5.6 which also use the ring technique). The idea is to merge the saturation and hue histograms in only one histogram. For this purpose, the GA optimizes the parameter $color_{divisions}$ and the number of bins for the color histogram is $(color_{divisions})^2$. Again we look for the neighbors of a keypoint for their hue and saturation values and select a $bin_h$ and a $bin_s$, using the same equations as above (equation 4.4 and 4.5), but this time the number of bins is $color_{divisions}$. Now to find the correct $color_{bin}$ to be incremented in the color histogram, we use the coordinates $(bin_h, bin_s)$ in the equation 4.6.

$$color_{bin} = bin_h \cdot 3 + bin_s \qquad (4.6)$$

Figure 4.8 represents an example of a color histogram produced by $color_{radius} = 1.3cm$ and figure 4.9 represents the respective ring histogram using $color_{radius} = 3.6cm$. The $color_{bins} = 36$. These histograms where taken from a version where the normalization is made to sum 1 using the two histograms, this means that the sum of the relative frequencies from both histograms equals 1.

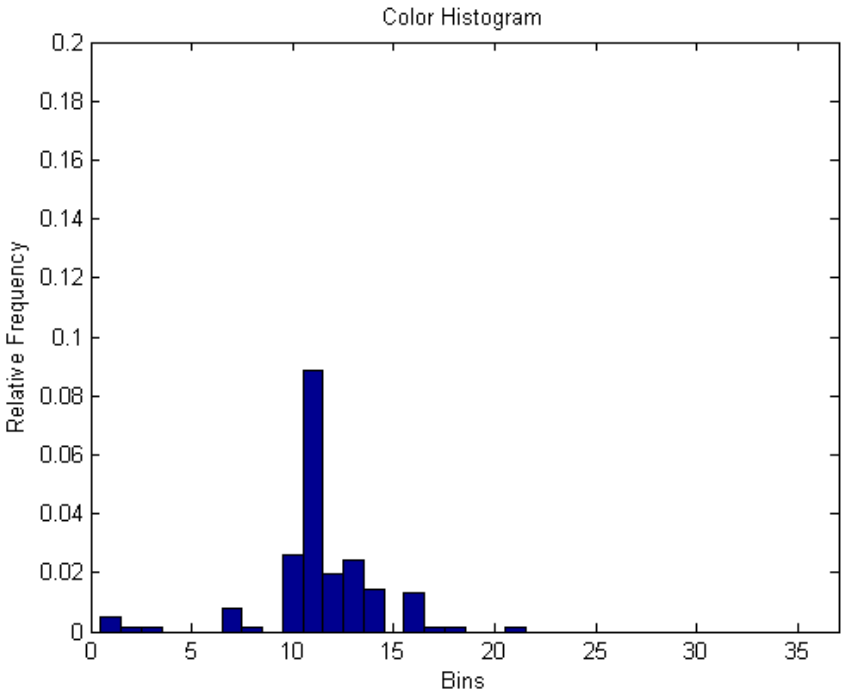


Figure 4.8: Color Histogram

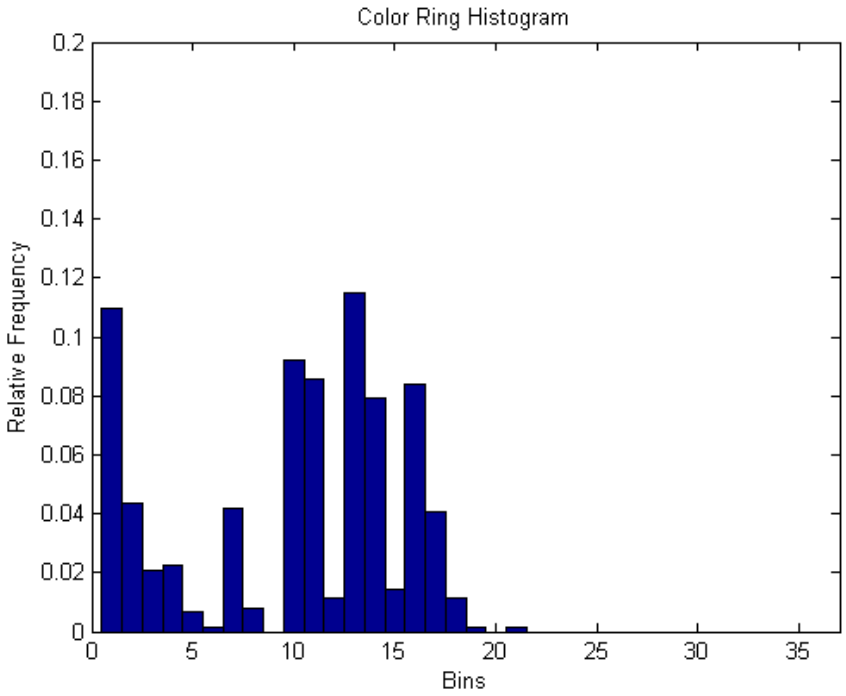


Figure 4.9: Ring Color Histogram

## 4.4 Distance Between Histograms

The way to find the distance between two clouds differs from version to version (chapter 5). In fact, only the final distance equation has changes from version to version, because each version has different structure and number of histograms. For the matching process we use all the histograms calculated so far. Before matching objects we compute the distances between each two histograms. In order to get the distances between shape histograms from two clouds first we compute the centroid of both histograms, then using the chi-squared distance [PHB97] (equation 4.7) we get the distance between the two centroid histograms. The $h_1$ and the $h_2$ are the two histograms between which we want to measure the distance.

$$d_{chisq} = \sum \frac{(h_1[i] - h_2[i])^2}{2(h_1[i] + h_2[i])} \qquad (4.7)$$

Then we calculate the standard deviation for both histograms (equation 4.8, $h[i]$ are the shape histogram values, while $c$ represents the centroid and $N$ is the number of keypoints) and we use the same process to get the distance between the standard deviation histograms as we used for the centroid histograms.

$$std = \sqrt{\frac{\sum (h[i] - c)^2}{N - 1}} \qquad (4.8)$$

Finally we sum the centroid distance with the standard deviation distance [Ale12] for the final shape distance. The same process is done for the second shape distance inside the ring and the distances between hue and saturation histograms. After getting all four distances between two clouds, we calculate the final distance between them, where we use a weight, represented by the parameter $w$ (equation 4.9, where $d_{sh}$ are the two shape distances, $d_{hu}$ is the hue distance and $d_{st}$ is the saturation distance). The parameter $w$ is adjusted by the GA.

$$d_{final} = (d_{sh1} + d_{sh2})w + (d_{hu} + d_{st})(1 - w) \qquad (4.9)$$

Some versions of the project are a hybrid from two different versions. In order to compute the distance between two clouds, we get the distance of each of the two versions and then each version distance is weighted. This weight is also optimized by a GA.

## 4.5 Matching

The next step is to find which test cloud fits best to each training cloud, this means the one with the smallest distance. The results of the matching of all test point clouds are used to produce the final result of the dataset.

# Chapter 5

# Experiments

## 5.1 Dataset

A subset of the large dataset of 3D point clouds [KLF11] was used to perform experiments. The clouds were divided into four subsets, constituted by two training subsets, one validation subset and one test set. The $test_{clouds}$ subset is composed by 475 different views of 10 objects, this subset is used after the best descriptor is generated by the GA. This set is used to calculate the test error of the descriptor. The training clouds subset contains 943 different views of the same 10 objects and is divided into the other 3 subsets. The $validation_{clouds}$ subset has 239 clouds (and represents $\frac{1}{4}$ of the training clouds). This subset is used to avoid the overfitting of the descriptors. After the test error for all chromosomes of a generation is computed, we calculate the validation error of the best chromosome of the generation. For this purpose we check how many clouds from the $validation_{clouds}$ are correctly matched, using the other 704 training clouds as the matching clouds. If the validation error from one generation is bigger than the previous generation validation error, than the GA stops and the best parameters for the descriptor are the ones from the previous generation. The other 704 training clouds are divided into $training_1$ and $training_2$ subsets. Those two training subsets were used by the GA to get the object error of the chromosomes ($training_1$ subset is formed by 352 training clouds and is used as testing clouds to calculate the object error and category error for each chromosome, while $training_2$ subset is constituted by the other 352 training clouds and works as the training subset for the GA). The genetic algorithm search for the best parameters to match each $training_1$ subset cloud with one $training_2$ subset cloud. After getting the best parameters (the ones that have the smallest errors), those parameters are used to find the best match for each cloud from the $test_{clouds}$ subset among the 943 training clouds subset, in order to get the test category and object errors of the descriptor. The category error represents the percentage of clouds matched to the correct category, for example the cloud apple_1_13 has a category correspondence with the cloud apple_4_27. In this thesis we use 10 categories which are: apple, ball, banana, bell_pepper, binder, bowl, calculator, camera, cap and cell_phone. The object error represents the percentage of cloud matched to the correct object. For this error a match of an object to a correct category isn't enough. This error is stricter and a correct match requires not only the correct category, but also the correct object, for example the cloud apple_1_13 has an object correspondence with the cloud apple_1_14. In this notation (apple_1_14), the name refers to the category, the first number is the object that belong to the category (apple) and the last number represents one of the views of the object.

Although the used dataset was always the same, many different structures were used to build the descriptor. Depending on the version different parameters were used as genes in the chromosomes of the population. The first versions served as a lesson to understand what is needed to create a descriptor and understand what parameters were more or less important. Those first versions didn't have good performance and there is no need to spent time on them. The next 7 sections explain the structure of the best versions and their performance.

## 5.2 Version 1

This 3D point cloud descriptor only uses shape information. For this purpose two shape histograms are created. Both histograms are build as described in section 4.2. In order to build this descriptor the GA optimized chromosomes containing 3 genes, representing parameters for the descriptor. The parameters are $Shape_{bins}$,$Shape_{radius1}$ and $Shape_{radius2}$. The GA has some restrictions, which are the intervals in which the parameters lie. The first parameter $Shape_{bins}$ is the number of bins that will hold the shape information. Its value is set between $[8; 64]$. The second parameter $Shape_{radius1}$ refers to the bigger radius inside which the search for the angles between each keypoint and its neighbors is done, in order to compute the $shape_{histogram2}$. Its value is set between $[2.1; 5.0]cm$. The $Shape_{radius1}$ is used as a ring in the $shape_{histogram2}$. The last parameter optimized by the GA is $Shape_{radius2}$, that is set between $[0.5; 2.0]cm$ and represents the smaller radius, used to obtain $shape_{histogram1}$. The number of bins for this descriptor is $Shape_{bins}$, so the descriptor has between 8 and 64 bins.

Table 5.1 shows the best two chromosomes optimized by the GA.

Table 5.1: Descriptor Parameters (version 1)

| Descriptor | $Shape_{bins}$ | $Shape_{radius1}$ (cm) | $Shape_{radius2}$ (cm) |
|---|---|---|---|
| a | 30 | 4.9 | 1.7 |
| b | 28 | 4.9 | 1.7 |

After both histograms are calculated, the distances between histograms are computed to do the final matches between the 475 test clouds and the 943 training clouds. For this descriptor no weights are needed, as it has only shape information and the distance between two clouds is given in equation 5.1, where $d_{sh1}$ and $d_{sh2}$ are the shape distances (one for each shape radius). Those distances are calculated as explained in chapter 4, section 4.4.

$$d_{final} = d_{sh1} + d_{sh2} \qquad (5.1)$$

After the matches are done, we check how many of the 475 test clouds were correctly matched. Table 5.2 represents the validation and test performance of the two best descriptors of this version.

Table 5.2: Descriptor Performance (version 1)

| Descriptor | $Validation_{ObjectError}(\%)$ | $Validation_{CategoryError}(\%)$ |
|---|---|---|
| a | 50.85 | 19.60 |
| b | 50.57 | 19.03 |
| **Descriptor** | $Test_{ObjectError}(\%)$ | $Test_{CategoryError}(\%)$ |
| a | 64.84 | 21.68 |
| b | 64.42 | 22.32 |

The best descriptor uses 28 bins to represent a cloud. This descriptor has an accuracy of 35.58% in matching the cloud to the correct object (from the 475 test clouds, only 169 were correctly matched) and 78.32% in matching the cloud to the correct category, this means that from the 475 clouds test set, 372 were matched to the correct category. As we can see the performance of this descriptor is not great and this descriptor is described in order to make comparisons to descriptors that use more information than just shape data.

## 5.3 Version 2

This 3D point cloud descriptor only uses color information. For this purpose four color histograms are created. Two histograms are for saturation and two histograms for, hue one using the ring and one using the smaller radius. All histograms are build as described in section 4.3. In order to build this descriptor the GA optimized chromosomes containing 3 genes, representing parameters for the descriptor. The parameters are $Color_{bins}$,$Color_{radius1}$ and $Color_{radius2}$. The GA has some restrictions, which are the intervals in which the parameters lie. The first parameter $Color_{bins}$ is the number of bins that will hold the color information. Its value is set between $[8; 64]$. The second parameter $Color_{radius1}$ refers to the bigger radius inside which the search for the hue and saturation values of the neighbors of each keypoint is done, in order to compute the $saturation_{histogram2}$ and $hue_{histogram2}$. Its value is set between $[2.1; 5.0]cm$. The $Color_{radius1}$ is used as a ring in both color histograms. The last parameter optimized by the GA is $Color_{radius2}$, that is set between $[0.5; 2.0]cm$ and represents the smaller radius, used to obtain $hue_{histogram1}$ and $saturation_{histogram1}$. The number of bins for this descriptor is $Color_{bins}$, so the descriptor has between 8 and 64 bins.
Table 5.3 shows the best two chromosomes optimized by the GA.

Table 5.3: Descriptor Parameters (version 2)

| Descriptor | $Color_{radius1}$ (cm) | $Color_{radius2}$ (cm) | $Color_{bins}$ |
|---|---|---|---|
| a | 3.1 | 1.8 | 41 |
| b | 3.5 | 0.7 | 39 |

After all histograms are calculated we compute the distances between histograms to do the final matches between the 475 test clouds and the 943 training clouds. For this descriptor no weights are needed, as it has only color information and the distance between two clouds is given in equation 5.2, where $d_{h1}$ and $d_{h2}$ are the hue distances (one for each hue radius) and $d_{s1}$ and $d_{s2}$ are the saturation distances. Those distances are calculated as explained in section 4.4.

$$d_{final} = d_{h1} + d_{h2} + d_{s1} + d_{s2} \qquad (5.2)$$

After the matches are done, we check how many of the 475 test clouds were correctly matched. Table 5.4 represents the validation and test performance of the two best descriptors of this version.

Table 5.4: Descriptor Performance (version 2)

| Descriptor | $Validation_{ObjectError}(\%)$ | $Validation_{CategoryError}(\%)$ |
|---|---|---|
| a | 8.80 | 3.97 |
| b | 9.09 | 4.54 |
| **Descriptor** | $Test_{ObjectError}(\%)$ | $Test_{CategoryError}(\%)$ |
| a | 42.31 | 25.47 |
| b | 41.89 | 25.05 |

The best descriptor uses 39 bins to represent a cloud. This descriptor has an accuracy of 58.11% in matching the cloud to the correct object (from the 475 test clouds, only 276 were correctly matched) and 74.95% in matching the cloud to the correct category, this means that from the 475 clouds test set, 356 were matched to the correct category. As we can see the performance of this descriptor, like the version 1, is not great and this descriptor is described in order to make comparisons to descriptors that use more information than just color. The version 2 (only color) in comparison to version 1 (only shape) has a better performance in object

recognition (22.53%), but is worse in category recognition (3.37%). Although the version 2 uses more bins than the shape version 1 (39 bins vs 28 bins), in terms of computational time the color version is faster than shape version, as the normals computation is a computationally heavy process.

## 5.4 Version 3

This 3D point cloud descriptor uses both shape and color information. For this purpose two shape histograms and two color histograms are created. Two histograms used are for shape, one histogram for saturation and one histogram for hue. In case of the shape one histogram uses the ring and one uses the smaller radius. All histograms are build as described in section 4.3 and 4.2. In order to build this descriptor the GA optimized chromosomes containing 6 genes, representing parameters for the descriptor. The parameters are $Shape_{bins}$,$Radius_1$ and $Radius_2$ used to build shape histograms, $Hue_{bins}$ and $Saturation_{bins}$ and that are used to compute the saturation and the hue histograms, and a weight $w$ used to weight the importance that the shape distances and the color distances have in the final distance between two point clouds. The GA has some restrictions, which are the intervals in which the parameters lie. The parameter $Shape_{bins}$ equals the number of bins that will hold the shape information. Its value is set between $[8; 64]$. Same way as $Shape_{bins}$, the $Hue_{bins}$ and $Saturation_{bins}$ is the number of bins that will hold respectively the hue and the saturation information and both values are set between $[8; 64]$. The parameter $Radius_1$ refers to the bigger radius inside which the search for the angles between each keypoint and its neighbors is done, in order to compute the $shape_{histogram2}$. Its value is set between $[2.1; 5.0]cm$. The $Radius_1$ is used as a ring in the $shape_{histogram2}$. The parameter $Radius_2$, that is set between $[0.5; 2.0]cm$ and represents the smaller radius, is used to obtain the $shape_{histogram1}$. The last parameter optimized by the GA is the weight $w$, that is set between $[0; 1.0]$, the $w$ is the weight of the shape distances, while $1 - w$ is the weight of the color distances. This descriptor has another parameter $Color_{radius}$ which is the radius inside which is done the search for the hue and saturation values of the neighbors in order to make the histograms. This value isn't optimized directly by the GA, but is always one of the radius ($Radius_1$ or $Radius_2$). This way we have only two histograms for color, one for hue and another for saturation (the ring technique is applied only to the shape information). The number of bins for this descriptor is $2 \cdot Shape_{bins} + Hue_{bins} + Saturation_{bins}$, so the descriptor has between 24 and 256 bins.
Table 5.5 shows the best two chromosomes optimized by the GA.

Table 5.5: Descriptor Parameters (version 3)

| Descriptor | $Shape_{bins}$ | $Shape_{radius1}$ (cm) | $Shape_{radius2}$ (cm) | |
|---|---|---|---|---|
| a | 49 | 4.4 | 1.2 | |
| b | 37 | 3.7 | 1.3 | |
| **Descriptor** | $Hue_{bins}$ | $Saturation_{bins}$ | $Color_{radius}$ (cm) | $weight_a$ |
| a | 63 | 63 | 1.2 | 0.4 |
| b | 62 | 50 | 1.3 | 0.42 |

After all histograms are calculated we compute the distances between histograms to do the final matches between the 475 test clouds and the 943 training clouds.

For this descriptor we use the weight $w$, as we need to know what is the importance of the shape and color information to get the best matches. Equation 5.3 shows how we compute the final distance ($d_{final}$) between two clouds, where $d_{sh1}$ and $d_{sh2}$ are the shape distances (one distance for each of the two radius), $d_h$ is the hue distance and $d_s$ is the saturation distance. Those distances are calculated as explained in chapter 4, section 4.4.

$$d_{final} = (d_{sh1} + d_{sh2})w + (d_h + d_s)(1 - w) \qquad (5.3)$$

After the matches are done, we check how many of the 475 test clouds were correctly matched. Table 5.6 represents the validation and test performance of the two best descriptors of this version.

Table 5.6: Descriptor Performance (version 3)

| Descriptor | $Validation_{ObjectError}(\%)$ | $Validation_{CategoryError}(\%)$ |
|---|---|---|
| a | 7.10 | 3.41 |
| b | 6.81 | 3.41 |
| Descriptor | $Test_{ObjectError}(\%)$ | $Test_{CategoryError}(\%)$ |
| a | 34.52 | 17.26 |
| b | 33.68 | 17.68 |

The best descriptor uses 186 bins to represent the point cloud. This descriptor has an accuracy of 66.32% in matching the cloud to the correct object (from the 475 test clouds 315 were correctly matched) and 82.32% in matching the cloud to the correct category, this means that from the 475 clouds test set, 391 were matched to the correct category. The version 1 used only shape information and had an accuracy of 35.58%. In version 2, that represent the point clouds using only color information the best descriptor had 59.11% correct matches. Version 3 that used both shape and color data to represent the points cloud has an improvement of 7.21% in comparison to the just color descriptor, what represents 34 clouds.

## 5.5 Version 4

This 3D point cloud descriptor uses both shape and color information. For this purpose one shape histogram and one color histogram are created. In case of the shape histogram we use the ring radius and the smaller radius. The shape histogram has a total of $2 * Shape_{bins}$, the first part of the histogram has data computed using $Radius_2$ and the second part uses $Radius_1$. Each part of the histogram is normalized to sum 1, what means that the entire histogram is normalized to sum 2. The same process is done for the color histogram, where the total of bins is equal to $2 \cdot (Color_{divisions})^2$ and again we use $Radius_2$ to compute the second part of the histogram and $Radius_1$ for the second part. The color histogram has also its parts normalized to sum 1. All histograms are build as described in section 4.3 and 4.2. In order to build this descriptor the GA optimized chromosomes containing 5 genes, representing parameters for the descriptor. The parameters are $Shape_{bins}$,$Radius_1$ and $Radius_2$ used to build shape histograms, $Color_{divisions}$ that are used to compute the color histograms (using also $Radius_1$ and $Radius_2$), and a weight $w$ used to weight the importance that the shape distances and the color distances have in the final distance between two point clouds. The GA has some restrictions, which are the intervals in which the parameters lie. The parameter $Shape_{bins}$ equals the number of bins that will hold the shape information. Its value is set between $[8; 64]$. The parameter $Color_{divisions}$ is used to calculate the number of bins that will hold color information. Its value is set between $[3; 8]$. In order to compute the number of bins we use the following equation 5.7.

$$Color_{bins} = (Color_{divisions})^2 \qquad (5.4)$$

The parameter $Radius_1$ refers to the bigger radius inside which the search for the angles between each keypoint and its neighbors is done, in order to compute the $shape_{histogram2}$. Its value is set between $[2.1; 5.0]cm$. The $Radius_1$ is used as a ring to obtain the second part of $shape_{histogram}$ and $color_{histogram}$. The parameter $Radius_2$, that is set between $[0.5; 2.0]cm$ represents the smaller radius and is used to obtain the first part of the $shape_{histogram}$ and $color_{histogram}$. The last parameter optimized by the GA is the weight $w$, that is set between $[0; 1.0]$, the $w$ is the weight of the shape distance, while $1 - w$ is the weight of the color distance. The number of bins for this descriptor is given in equation 5.5, so the descriptor has between 34 and 256 bins.

$$descriptor_{bins} = 2 \times Shape_{bins} + 2 \times Color^2_{divisions} \qquad (5.5)$$

Table 5.7 shows the best two chromosomes optimized by the GA.

Table 5.7: Descriptor Parameters (version 4)

| Descriptor | $Shape_{bins}$ | $Shape_{radius1}$ (cm) | $Shape_{radius2}$ (cm) | $Color_{bins}$ | $weight_a$ |
|---|---|---|---|---|---|
| a | 59 | 3.6 | 1.3 | 6 | 0.67 |
| b | 46 | 3.1 | 1.7 | 8 | 0.68 |

After all histograms are calculated we compute the distances between histograms to do the final matches between the 475 test clouds and the 943 training clouds. For this descriptor we use the weight $w$, as we need to know what is the importance of the shape and color information to get the best matches.

Equation 5.6 shows how we compute the final distance ($d_{final}$) between two clouds, where $d_{sh}$ is the shape distance between two histograms and $d_c$ is the color distance between two histograms. Those distances are calculated as explained in chapter 4, section 4.4.

$$d_{final} = (d_{sh})w + (d_c)(1 - w) \qquad (5.6)$$

After the matches are done, we check how many of the 475 test clouds were correctly matched. Table 5.8 represents the validation and test performance of the two best descriptors of this version.

Table 5.8: Descriptor Performance (version 4)

| Descriptor | $Validation_{ObjectError}(\%)$ | $Validation_{CategoryError}(\%)$ |
|---|---|---|
| a | 10.50 | 4.26 |
| b | 10.50 | 3.12 |
| **Descriptor** | $Test_{ObjectError}(\%)$ | $Test_{CategoryError}(\%)$ |
| a | 29.05 | 15.37 |
| b | 32.21 | 12.63 |

The best descriptor uses 154 bins to represent the point cloud. This descriptor has an accuracy of 70.95% in matching the cloud to the correct object (from the 475 test clouds 337 were correctly matched) and 84.63% in matching the cloud to the correct category, this means that from the 475 clouds test set, 402 were matched to the correct category. The shape information of this descriptor is the same that from version 3. This descriptors shape part differs from the shape part of version 3 as now all this information is just in one histogram, so we get different distances between shape histograms in each version. Even so, the main difference between those descriptors resides in the color part: version 3 uses one histogram for hue and one for saturation, while this version has only one histogram to record all color information using two radius (ring) and the data about hue and saturation is merged in the same histogram. The differences in the shape and color histograms mentioned above make an improvement of the object error of 4.63% in comparison to version 3, that represents 22 clouds.

## 5.6 Version 5

This 3D point cloud descriptor uses both shape and color information. For this purpose one histogram is created that hold both shape and color information of the cloud. In case of the shape part of the histogram we use the ring radius and the smaller radius. The shape part of the histogram has a total of $2 * Shape_{bins}$, the first part of the shape part of the histogram has data computed using $Radius_2$ and the second part uses $Radius_1$. Those two parts of the shape histogram are normalized to sum 1. The same process is done for the color part of the histogram (the color part is also divided in two parts), where the total of bins is equal to $2 \cdot Color^2_{divisions}$ and again we use $Radius_2$ to compute the second color part of the histogram and $Radius_1$ for the second part second color part of the histogram. The color histogram has also its two parts normalized to sum 1. The histogram is build as described in section 4.2 and 4.3. The descriptor from this version is very similar to the previous version 4. It has the same parameters, but we build only one histogram that has the entire information about shape and color. The descriptor is done with 4 genes optimized by the GA that represent the parameters for the descriptor. The parameters are $Shape_{bins}$,$Radius_1$ and $Radius_2$ used to build the shape part of the histograms and $Color_{divisions}$ that is used to compute the color part of the histogram (using also $Radius_1$ and $Radius_2$). There is no need for the weight $w$ in this case as only one histogram is used and this way we don't need to give weights for the shape/color distances. The GA has some restrictions, which are the intervals in which the parameters lie. The parameter $Shape_{bins}$ equals the number of bins that will hold the shape information. Its value is set between $[8; 64]$. The parameter $Color_{divisions}$ is used to calculate the number of bins that will hold color information. Its value is set between $[3; 8]$. In order to compute the number of bins we use the following equation 5.7.

$$Color_{bins} = (Color_{divisions})^2 \qquad (5.7)$$

The parameter $Radius_1$ refers to the bigger radius inside which the search for the angles between each keypoint and its neighbors is done, in order to compute the $shape_{histogram2}$, its value is set between $[2.1; 5.0]cm$. The $Radius_1$ is used as a ring to obtain the second part of $shape_{histogram}$ and $color_{histogram}$. The parameter $Radius_2$, that is set between $[0.5; 2.0]cm$ represents the smaller radius and is used to obtain the first part of the $shape_{histogram}$ and $color_{histogram}$. The number of bins for this descriptor is given in equation 5.8, so the descriptor has between 34 and 256 bins.

$$descriptor_{bins} = 2 \times Shape_{bins} + 2 \times Color^2_{divisions} \qquad (5.8)$$

Table 5.9 shows the best two chromosomes optimized by the GA.

Table 5.9: Descriptor Parameters (version 5)

| Descriptor | $Shape_{bins}$ | $Shape_{radius1}$ (cm) | $Shape_{radius2}$ (cm) | $Color_{bins}$ |
|---|---|---|---|---|
| a | 58 | 3.6 | 1.0 | 6 |
| b | 43 | 3.0 | 1.0 | 6 |

After the histogram is calculated we compute the distance between histograms to do the final matches between the 475 test clouds and the 943 training clouds. After the matches are finished, we check how many of the 475 test clouds were correctly matched.

Table 5.10 represents the validation and test performance of the two best descriptors of this version.

Table 5.10: Descriptor Performance (version 5)

| Descriptor | $Validation_{ObjectError}(\%)$ | $Validation_{CategoryError}(\%)$ |
|---|---|---|
| a | 9.09 | 3.69 |
| b | 9.38 | 4.26 |
| **Descriptor** | $Test_{ObjectError}(\%)$ | $Test_{CategoryError}(\%)$ |
| a | 32.42 | 18.74 |
| b | 32.00 | 18.74 |

The best descriptor uses 126 bins to represent the point cloud. This descriptor has an accuracy of 68% in matching the cloud to the correct object (from the 475 test clouds 323 were correctly matched) and 81.26% in matching the cloud to the correct category, this means that from the 475 clouds test set, 386 were matched to the correct category. This descriptor uses less bins (126) than the previous version (154) and has 2.95% less of accuracy.

## 5.7 Version 6

This 3D point cloud descriptor uses both shape and color information. For this purpose two shape histograms and four color histograms are created. Two histograms are for shape, two histograms for saturation and two histograms for hue, one using the ring and one using the smaller radius. All histograms are build as described in section 4.3 and 4.2. In order to build this descriptor the GA optimized chromosomes containing 7 genes, representing parameters for the descriptor. The parameters are $Shape_{bins}$,$Shape_{radius1}$ and $Shape_{radius2}$ used to build shape histograms, $Color_{bins}$,$Color_{radius1}$ and $Color_{radius2}$ used to compute saturation and hue histograms and a weight $w$ used to weight the importance that the shape distances and the color distances have in the final distance between two point clouds. The GA has some restrictions, which are the intervals in which the parameters lie. The parameter $Shape_{bins}$ equals the number of bins that will hold the shape information, its value is set between $[8; 64]$. The parameter $Shape_{radius1}$ refers to the bigger radius inside which the search for the angles between each keypoint and its neighbors is done, in order to compute the $shape_{histogram2}$. Its value is set between $[2.1; 5.0]cm$. The $Shape_{radius1}$ is used as a ring in the $shape_{histogram2}$. The parameter $Shape_{radius2}$, that is set between $[0.5; 2.0]cm$ and represents the smaller radius, used to obtain $shape_{histogram1}$. The parameter $Color_{bins}$ equals the number of bins that will hold the color information, its value is set between $[8; 64]$. The parameter $Color_{radius1}$ refers to the bigger radius inside which is done the search for the hue and saturation values of the neighbors of each keypoint, in order to compute the $saturation_{histogram2}$ and $hue_{histogram2}$. Its value is set between $[2.1; 5.0]cm$. The $Color_{radius1}$ is used as a ring in both color histograms. The parameter $Color_{radius2}$, that is set between $[0.5; 2.0]cm$ and represents the smaller radius, used to obtain $hue_{histogram1}$ and $saturation_{histogram1}$. The last parameter optimized by the GA is the weight $w$, that is set between $[0; 1.0]$, the $w$ is the weight of the shape distance, while $1 - w$ is the weight of the color distance. The number of bins for this descriptor was between 48 and 384 bins.

Table 5.11 shows the best two chromosomes optimized by the GA.

Table 5.11: Descriptor Parameters (version 6)

| Descriptor | $Shape_{bins}$ | $Shape_{radius1}$ (cm) | $Shape_{radius2}$ (cm) | $Color_{bins}$ | $weight_a$ |
|---|---|---|---|---|---|
| a | 49 | 4.0 | 0.6 | 14 | 0.62 |
| b | 21 | 3.6 | 1.9 | 41 | 0.69 |

After all histograms are calculated we compute the distances between histograms to do the final matches between the 475 test clouds and the 943 training clouds. For this descriptor we use the weight $w$ in order to weight the shape distance and the color distance for the final match. Equation 5.9 shows how the final distance between two clouds is calculated, where $d_{sh1}$ and $d_{sh2}$ are the shape distances, $d_{h1}$ and $d_{h2}$ are the hue distances and $d_{s1}$ and $d_{s2}$ are the saturation distances (one distance for each of the two radius). Those distances are calculated as explained in chapter 4, section 4.4. As we have two shape histograms, we make an average of those distances (by dividing the sum of the distances by 2). The same process is used for color distances, as we have 4 color descriptors, we divide the sum of all distances by 4.

$$d_{final} = \left(\frac{d_{sh1} + d_{sh2}}{2}\right) \times w + \left(\frac{d_{h1} + d_{h2} + d_{s1} + d_{s2}}{4}\right) \times (1 - w) \qquad (5.9)$$

After the matches are done, we check how many of the 475 test clouds were correctly matched. Table 5.12 represents the validation and test performance of the two best descriptors of this version.

Table 5.12: Descriptor Performance (version 6)

| Descriptor | $Validation_{ObjectError}(\%)$ | $Validation_{CategoryError}(\%)$ |
|---|---|---|
| a | 7.38 | 3.98 |
| b | 8.23 | 3.69 |

| Descriptor | $Test_{ObjectError}(\%)$ | $Test_{CategoryError}(\%)$ |
|---|---|---|
| a | 30.10 | 17.05 |
| b | 29.89 | 13.68 |

The best descriptor uses 206 bins (42 for shape information and 164 for color information) to represent the point cloud. This descriptor has an accuracy of 70.11% in matching the cloud to the correct object (from the 475 test clouds 333 were correctly matched) and 86.32% in matching the cloud to the correct category, this means that from the 475 clouds test set, 410 were matched to the correct category.

## 5.8 Version 7

Some experiments done for this work were hybrid versions. For this purpose this descriptor uses two different versions and the GA has one more gene, which is a weight for each version ($w_{version}$). This parameter is set in the interval $[0; 1.0]$. This weight is the only parameter for the GA, while the other parameters of the descriptor are set to the best parameters of the respective version. The best hybrid version was a mixture of version 3 (section 5.4) and version 4 (section 5.5). Table 5.13 shows the best parameters used in version 7 from the descriptor of version 3.

Table 5.13: Descriptor Parameters (version 3)

| Descriptor | $Shape_{bins}$ | $Shape_{radius1}$ (cm) | $Shape_{radius2}$ (cm) | |
|---|---|---|---|---|
| Version 3 | 37 | 3.7 | 1.3 | |
| Descriptor | $Hue_{bins}$ | $Saturation_{bins}$ | $Color_{radius}$ (cm) | $weight_a$ |
| Version 3 | 62 | 50 | 1.3 | 0.42 |

Table 5.14 shows the best parameters used in version 7 from the descriptor of version 4.

Table 5.14: Descriptor Parameters (version 4)

| Descriptor | $Shape_{bins}$ | $Shape_{radius1}$ (cm) | $Shape_{radius2}$ (cm) | $Color_{bins}$ | $weight_a$ |
|---|---|---|---|---|---|
| Version 4 | 59 | 3.6 | 1.3 | 6 | 0.67 |

The GA optimized $w_{version}$ and the optimal value for this parameter was 0.93. This means that the importance of version 3 is 93%, while the importance of version 4 is 7%. For this this descriptor, the part of version 3 has two shape histograms, one saturation and one hue histograms and four color histograms. The part of version 4 has one shape and one color histogram. After all histograms are calculated we compute the distances between histograms to do the final matches between the 475 test clouds and the 943 training clouds. Equation 5.10 shows how the final distance between two clouds is calculated, where $d_{version3}$ is calculated as explained in section 5.4 and $d_{version4}$ is calculated as explained in section 5.5. The weigh $w_{version}$ is optimized by the GA.

$$d_{version7} = d_{version3} \times w_{version} + d_{version4} \times (1 - w_{version}) \qquad (5.10)$$

After the matches are done, we check how many of the 475 test clouds were correctly matched. Table 5.15 represents the validation and test performance of the best descriptor of this version.

Table 5.15: Descriptor Performance (version 7)

| Descriptor | $Validation_{ObjectError}(\%)$ | $Validation_{CategoryError}(\%)$ |
|---|---|---|
| a | 8.23 | 3.41 |
| Descriptor | $Test_{ObjectError}(\%)$ | $Test_{CategoryError}(\%)$ |
| b | 27.16 | 14.74 |

The descriptor uses 340 bins to represent the point cloud. This descriptor has an accuracy of 72.84% in matching the cloud to the correct object (from the 475 test clouds, 346 were correctly matched) and 85.26% in matching the cloud to the correct category, this means that from the 475 test clouds set, 405 were matched to the correct category. This descriptor is the best descriptor created in this thesis. Although it has a better accuracy than version 4 improving 1.89%, it is much slower to compute.

## 5.9 Result Analysis

The following table has the performance of the descriptors mentioned in the chapter 2 and the descriptors implemented for this thesis. The last column refers to the time necessary to match the 475 test clouds using 943 point clouds as the training set.

Table 5.16: Descriptors Performance: test errors for the object and category recognition tasks.

| Descriptor | Object (%) | Category (%) | Time (s) |
|---|---|---|---|
| PFHRGB | 20.25 | 5.27 | 2992 |
| SHOTCOLOR | 26.58 | 9.28 | 178 |
| Version 7 | 27.16 | 14.74 | 1999 |
| Version 4 | 29.05 | 15.37 | 886 |
| Version 6 | 29.89 | 13.68 | 1265 |
| Version 5 | 32.00 | 18.74 | 948 |
| Version 3 | 33.68 | 17.68 | 1240 |
| Version 2 | 41.89 | 25.05 | 454 |
| PFH | 44.51 | 11.76 | 1668 |
| FPFH | 52.74 | 13.92 | 228 |
| SHOT | 53.16 | 11.81 | 76 |
| USC | 54.22 | 17.97 | 381 |
| 3DSC | 59.49 | 21.94 | 504 |
| ESF | 62.66 | 18.35 | 15 |
| Version 1 | 64.42 | 22.32 | 586 |
| RIFT | 75.95 | 33.68 | 14 |
| CVFH | 80.59 | 48.95 | 13 |
| PPF | 82.91 | 47.68 | 153 |
| PCE | 83.55 | 52.74 | 13 |
| VFH | 94.73 | 78.27 | 13 |

Among the implemented versions for this thesis, the fastest is the color-only Version 2. We can see that it is faster and more accurate than the shape-only version 1. Versions 3 and 6 have similar time of computation, however version 6 gives better results. The best two descriptors developed for this thesis (versions 7 and 4) have a difference in the performance of 2%. Even though the best version in terms of accuracy is version 7, version 4 might be preferable since it is much faster and is only slightly worst in terms of accuracy.
The computational times indicated for the available descriptors in the PCL were taken from the paper [Ale12]. The machine used to perform this task was more powerful than the one used for this thesis. The available descriptors are all paralyzed, while the versions in this thesis are only paralyzed during the normals computation. The descriptors from version 3 to 7 all use shape and color information so they can be compared to the shape and color versions (PFHRGB and SHOTCOLOR). The results from those versions are a little bit worse as these two descriptors, however they are faster than both existent color descriptors, especially when compared to the PFHRGB that has the best accuracy.
Version 1 is the only descriptor experimented that has only shape information and as we can see it is in the middle of the table between shape-only descriptors. It is interesting to notice that version 2 (that only uses color information) is better that any shape-only descriptor.

# Chapter 6

# Conclusions

## 6.1 Conclusion

In this thesis the goal was to study point cloud descriptors and create a new descriptor that was fast and accurate in matching point clouds. The idea was to use a genetic algorithm to find optimal parameters that were used to create the descriptor.
The main conclusions are:

- Usually the descriptors that use shape-only information have an inferior accuracy than descriptors that use color-only information.
- If we use both shape and color information we can see a big improvement in results when comparing to the descriptors that use color-only or shape-only data.
- Descriptors that use two radius (ring technique) have better performance than descriptors with only one radius.
- To represent color histograms, we had better results using one merged color histogram than one histogram for hue and one for saturation.
- The created descriptors offer good results when taking into account their computational time.
- There is a big difference between the accuracy in the correct matching of the object and the correct matching of the category, as expected.
- The best descriptor was a hybrid between version 3 and version 4, described in version 7.

Although the new descriptors have good performance, we wanted to have a better accuracy then the achieved in this thesis.

## 6.2 Future Work

The shape information of the descriptor is obtained using always the same techniques as explained in section 4.2. Shape information could be improved using different representation forms for the descriptor. We could use some techniques from the best shape-only descriptors like the 3 angles $(\alpha, \phi, \theta)$ from the very good shape-only descriptor PFH. Another possible way to improve the quality of the descriptors is to let the GA optimize not only the values of the parameters, but also the structure of the entire descriptor.
To allow anyone to benefit from this work, the source code of the descriptor is being prepared in order to be published online in the PCL.

# Bibliography


[AAB11] D. Gossow S. Gedikli R. Rusu M. Vincze A. Aldoma, N. Blodow and G. Bradski. Cad-model recognition and 6 dof pose estimation. In *in ICCV 2011, 3D Representation and Recognition*, 2011. 6

[AFM04] R. Kolluri T. Bulow A. Frome, D. Huber and J. Malik. Recognizing objects in range data using regional point descriptors. In *8th European Conference on Computer Vision*, 2004. 3

[Ale12] Luís A. Alexandre. 3D descriptors for object and category recognition: a comparative evaluation. In *Workshop on Color-Depth Camera Fusion in Robotics at the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Vilamoura, Portugal, October 2012. 1, 3, 7, 23, 36

[BDI10] N. Navab B. Drost, M. Ulrich and S. Ilic. Model globally, match locally: Efficient and robust 3d object recognition. In *Third IEEE Conference on Computer Vision and Pattern Recognition,CVPR 2010, San Francisco, CA, USA*, 2010. 4

[CYIR02] L. Sieger C. Y. Ip, D. Lapadat and W. C. Regli. Using shape distributions to compare solid models. In *in Symposium on Solid Modeling and Applications*, 2002. 6

[Dar59] Charles Darwin. *On the Origin of Species*. London: John Murray, 1859. 1

[dic09] *Collins English Dictionary*. Harper Collins, 2009. 11, 12

[Eng02] Andries P. Engelbrecht. *Computational Intelligence, An Introduction*. John Wiley & Sons, 2002. 10, 11, 12, 13

[FTS10] S. Salti F. Tombari and L. Di Stefano. Unique signatures of histograms for local surface description. In *in Proceedings of the 11th European conference on computer vision conference on Computer vision: Part III. Berlin, Heidelberg: Springer-Verlag*, 2010. 6

[Gwi06] Tomasz Dominik Gwiazda. *Genetic Algorithm Reference*. Paperback, 2006. 10

[Hol75] John Henry Holland. *Adaptation in Natural and Artificial Systems*. A Bradford Book, 1975. 1, 10

[KLF11] X. Ren K. Lai, L. Bo and D. Fox. A large-scale hierarchical multi-view rgb-d object dataset. In *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, 2011. 25

[Low04] D. G. Lowe. *Distinctive image features from scale invariant keypoints, Int. J. Comput. Vision, vol. 60, no. 2, pp. 91110.* 2004. 3

[PHB97] Jan Puzicha, Thomas Hofmann, and Joachim M. Buhmann. Non-parametric similarity measures for unsupervised texture segmentation and image retrieval. *2012 IEEE Conference on Computer Vision and Pattern Recognition*, 0:267–272, 1997. 22

[RBRH10] R. Thibaux R. B. Rusu, G. Bradski and J. Hsu. Fast 3d recognition and pose using the viewpoint feature histogram. In *in Proceedings of the 23rd IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Taipei, Taiwan*, 2010. 5

[RC11] R. B. Rusu and S. Cousins. 3d is here: Point cloud library (pcl). In *IEEE International Conference on Robotics and Automation (ICRA)*, 2011. v, 3

[ROD01] B. Chazelle R. Osada, T. A. Funkhouser and D. P. Dobkin. Matching 3d models with shape distributions. In *in International Conference on Shape Modeling and Applications, Genoa, Italy*, 2001. 6

[RRB08] Z. Marton R. Rusu, N. Blodow and M. Beetz. Aligning point cloud views using persistent feature histograms. In *International Conference on Intelligent Robots and Systems (IROS)*, 2008. 4

[RRB09a] A. Holzbach R. Rusu and M. Beetz. Detecting and segmenting objects for mobile manipulation. In *in S3DV Workshop of the 12th International Conference on Computer Vision (ICCV)*, 2009. 5

[RRB09b] N. Blodow R. Rusu and M. Beetz. Fast point feature histograms (fpfh) for 3d registration. In *International Conference on Robotics and Automation (ICRA)*, 2009. 5

[sho11] A combined texture-shape descriptor for enhanced 3d feature matching. In *in IEEE International Conference on Image Processing*, 2011. 6

[SLP05] C. Schmid S. Lazebnik and J. Ponce. A sparse texture representation using local affine regions. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2005. 3

[TFB12] Dominik A.Klein Dirk Schulz Torsten Fiolka, Jorg Stuckler and Sven Behnke. Place recognition using surface entropy features. In *in Proc. of IEEE ICRA Workshop on Semantic Perception, Mapping, and Exploration, Saint Paul, MN, USA*, 2012. 17

[usc10] Unique shape context for 3d data description. In *in Proceedings of the ACM workshop on 3D object retrieval, New York, NY*, 2010. 3

[WV11] W. Wohlkinger and M. Vincze. Ensemble of shape functions for 3d object classification. In *in Robotics and Biomimetics (ROBIO)*, 2011. 6