Robust Factorized Shape Descriptors

VICTORIA MATUTE ARRIBAS

KTH Computer Science and Communication

Master of Science Thesis
Stockholm, Sweden 2012
Robust Factorized Shape Descriptors

VICTORIA MATUTE ARRIBAS

DD221X, Master’s Thesis in Computer Science (30 ECTS credits)
Degree Progr. in Computer Science and Engineering 270 credits
Royal Institute of Technology year 2012
Supervisor at CSC was Carl Henrik Ek
Examiner was Danica Kragic

TRITA-CSC-E 2012:016
ISRN-KTH/CSC/E--12/016--SE
ISSN-1653-5715

Royal Institute of Technology
School of Computer Science and Communication
KTH CSC
SE-100 44 Stockholm, Sweden
URL: www.kth.se/csc
Abstract

Imagine a robot driving a car, it has GPS signals that tells it which path to follow. However, it should adapt to the traffic conditions in every place. It wants to go from Gothenburg to Stockholm. Traffic is chaotic that morning in Gothenburg. There are some traffic police leading traffic. The robot has learned all the traffic signs that the police uses to direct the traffic.

What should the robot do internally? Most of the signs are made with hands. The robot should look for something that looks like hands independently of the orientation of these hands. It will focus on the appearance of the objects.

Once the hands have been located, the robot should figure out what the police wants it to do. If the hand is vertical, the robot should stop. If the hand is horizontal to the left, it will have to turn left. In this case, the orientation of the hands matters but not the appearance.

The robot arrives to Stockholm. It wants to go to the city center. It finds some road signs with different directions depending on the destination. Stockholm has changed significantly during the last decade and there are both old and new road signs. The shape of the arrows are different for the new and old signs. Which patterns should the robot focus on now? Rotation or Appearance? Appearance is important for focusing only on arrows. However, the rotation of the arrow is more important in order to follow the right way. In this thesis, we present a factorized shape descriptor capable of separating appearance from orientation.

Keywords: shape, rotation, HOG, object recognition, SVM, KNN, Fourier, feature extraction.
Referat

Robust faktoriserad formdeskriptor


Vad ska göra roboten internt? De flesta av de signalerna är gjorda med händer. Roboten bör leta efter något som ser ut som händer oavsett inriktningen av dessa händer. Den skall därför fokusera på utseendet av objekten.


Acknowledgements

There is no single project you do alone. There is no single place in which you stay alone. Sometimes people around help you to find motivation, self-confidence, inspiration, warmness, curiosity, etc. All of them are needed to keep the track on what you are doing.

First and foremost I want to thank my thesis supervisor, Carl Henrik Ek. I feel Carl has been an ideal mentor, collaborator and guide. He has always been very supportive and he always believed in my capacity to go further every day. He influenced me not only in the content of the thesis but also the language in which it has been conveyed. Thanks to Carl for the motivation and the highly encouraging spirit he always has.

I would like to thank my office colleagues. Thanks to Virgile who guided me in every single step in this project and thanks to Ali who kept my mood up when I needed it the most. I would like also to thank CVAP Innebandy team and Bulle team for making CVAP lab a fun and interesting place to work.

I can never forget all my friends in Stockholm. First of all, David who helped me to improve my technical English. Florian, Thomas and Caroline who make me find energy to continue working every day. I also want to thank Hector, Dani, Jesus, Nacho, Manu, Isaac, Guillem, Clara and Breo who kept my mood up and always made me remembered the important aspects of life.

From Spain, special thanks to Alicia who is always there wherever I am. Thanks also to Alejandro.

I cannot end without thanking my family and foremost my parents. This is just the end of a 6-years path in which they spouse suffer the most. I could always find absolute confidence, support and company by my side. Nothing could be possible without them.
A conclusion is simply the place where someone got tired of thinking.

-Arthur Block
# Contents

## Acknowledgements

## List of Figures

<table>
<thead>
<tr>
<th>1 Introduction</th>
<th>1 Introduction</th>
<th>1 Introduction</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1. Context</td>
<td>1.2. Goals</td>
<td>1.3. Application</td>
</tr>
<tr>
<td>1.4. Thesis outline</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.4. Thesis outline</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>2 Background</th>
<th>2 Background</th>
<th>2 Background</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1. Features</td>
<td>2.1. Features</td>
<td>2.1. Features</td>
</tr>
<tr>
<td>2.1.1. HOG</td>
<td>2.1.2. SIFT</td>
<td>2.1.3. Shape Context</td>
</tr>
<tr>
<td>2.2. Fourier Transform</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.3. Cross correlation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.4. Classifier</td>
<td>2.4. Classifier</td>
<td>2.4. Classifier</td>
</tr>
<tr>
<td>2.4.1. KNN algorithm</td>
<td>2.4.2. SVM</td>
<td></td>
</tr>
<tr>
<td>2.5. Dataset</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.6. Labeling</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.7. Summary</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>3 Descriptor</th>
<th>3 Descriptor</th>
<th>3 Descriptor</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1. Overview of our descriptor</td>
<td>3.2. Appearance</td>
<td>3.2. Appearance</td>
</tr>
<tr>
<td>3.2.1. Fourier Transform</td>
<td>3.3. Rotation</td>
<td>3.4. Classification</td>
</tr>
<tr>
<td>3.4.1. Kernel</td>
<td>3.4.2. Labeling</td>
<td>3.4.2. Labeling</td>
</tr>
<tr>
<td>3.4.2. Labeling</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
3.5. Summary ................................................. 44

4 Experiments ............................................. 47
  4.1. Datasets ............................................. 47
  4.2. Distance Matrix .................................... 48
    4.2.1. Initial descriptor ............................... 49
    4.2.2. Proposed descriptor ............................ 49
    4.2.3. Comments ....................................... 49
  4.3. Initial descriptor ................................. 54
    4.3.1. Appearance labeling ........................... 54
    4.3.2. Rotation labeling .............................. 57
  4.4. Proposed Descriptor ............................... 57
    4.4.1. Appearance labeling ........................... 57
    4.4.2. Rotation labeling .............................. 58
  4.5. General results .................................... 59
  4.6. Summary ............................................. 60

5 Conclusions and Future Works ...................... 65
  5.1. Performance improvements ....................... 65
    5.1.1. Database ....................................... 65
    5.1.2. Translation ..................................... 67
    5.1.3. Higher number of cells ....................... 67
    5.1.4. Peaks in the rotation ......................... 68
  5.2. Quality improvements ............................. 68
  5.3. Other approach .................................... 71

Appendices ................................................. 72
  A Database .............................................. 73
  B Case Studies .......................................... 77

Bibliography .............................................. 81

List of Figures

1.1. Shape recognition .................................. 1
1.2. Different images of hands ......................... 2
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.3. Challenge of this project</td>
<td>3</td>
</tr>
<tr>
<td>1.4. Handwritten digits</td>
<td>4</td>
</tr>
<tr>
<td>1.5. System outline</td>
<td>6</td>
</tr>
<tr>
<td>2.1. Overview of the system</td>
<td>7</td>
</tr>
<tr>
<td>2.2. Gradients of a 2 dimensional function</td>
<td>9</td>
</tr>
<tr>
<td>2.3. Overview of the HOG feature extraction</td>
<td>10</td>
</tr>
<tr>
<td>2.4. HOG extraction</td>
<td>10</td>
</tr>
<tr>
<td>2.5. translated images</td>
<td>11</td>
</tr>
<tr>
<td>2.6. gradients</td>
<td>12</td>
</tr>
<tr>
<td>2.7. Hog of lines in Fig.(2.5)</td>
<td>12</td>
</tr>
<tr>
<td>2.8. translated images</td>
<td>13</td>
</tr>
<tr>
<td>2.9. gradients</td>
<td>13</td>
</tr>
<tr>
<td>2.10. Hog of lines in Fig.(2.8)</td>
<td>14</td>
</tr>
<tr>
<td>2.11. Different shapes for the HOG cells</td>
<td>14</td>
</tr>
<tr>
<td>2.12. Image gradient</td>
<td>14</td>
</tr>
<tr>
<td>2.13. SIFT extraction</td>
<td>15</td>
</tr>
<tr>
<td>2.14. Shape Context example</td>
<td>16</td>
</tr>
<tr>
<td>2.15. Shape Context.Edge points</td>
<td>16</td>
</tr>
<tr>
<td>2.16. log-polar histogram</td>
<td>17</td>
</tr>
<tr>
<td>2.17. Shape context histograms</td>
<td>17</td>
</tr>
<tr>
<td>2.18. Object recognition with SIFT</td>
<td>17</td>
</tr>
<tr>
<td>2.19. Histogram of two different images</td>
<td>19</td>
</tr>
<tr>
<td>2.20. Histogram of two rotated images</td>
<td>20</td>
</tr>
<tr>
<td>2.21. Magnitude of different length histograms but same shape</td>
<td>21</td>
</tr>
<tr>
<td>2.22. Magnitude of zero-ending histograms</td>
<td>22</td>
</tr>
<tr>
<td>2.23. Magnitude of repetitive signal</td>
<td>23</td>
</tr>
<tr>
<td>2.24. 2 functions. The red is the translated version of the green function</td>
<td>23</td>
</tr>
<tr>
<td>2.25. Cross correlation of functions in Fig.(2.24)</td>
<td>24</td>
</tr>
<tr>
<td>2.26. Cross correlation function</td>
<td>24</td>
</tr>
<tr>
<td>2.27. Cross correlation between two identical vectors</td>
<td>25</td>
</tr>
<tr>
<td>2.28. 1-sample rotated vectors cross correlation</td>
<td>25</td>
</tr>
<tr>
<td>2.29. 2-samples rotated vectors cross correlation</td>
<td>25</td>
</tr>
<tr>
<td>2.30. Feature extraction and classification problem</td>
<td>26</td>
</tr>
<tr>
<td>2.31. Transformation of the dataset</td>
<td>27</td>
</tr>
<tr>
<td>2.32. SupportVectorMachine</td>
<td>28</td>
</tr>
<tr>
<td>2.33. Kernel projection</td>
<td>28</td>
</tr>
<tr>
<td>2.34. 3 histograms of rotated images</td>
<td>29</td>
</tr>
<tr>
<td>2.35. distance between two points</td>
<td>31</td>
</tr>
<tr>
<td>3.1. Overview of the system</td>
<td>33</td>
</tr>
<tr>
<td>3.2. Is there only one way of describing this image?</td>
<td>34</td>
</tr>
<tr>
<td>3.3. Initial descriptor</td>
<td>35</td>
</tr>
<tr>
<td>3.4. Proposed descriptor</td>
<td>35</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

To recognize different objects and their different poses in an unknown environment, a robot needs to analyze the environment from sensory data. For this purpose, the robot can use images. An image is very descriptive and can be acquired at a rapid rate from inexpensive sensors.

Images are extremely high-dimensional objects. They contain a large amount of information: color, shape, texture, rotation, etc. Images are classified into different groups depending on this information. Detecting objects in an image is a challenging task owing to the different shapes and poses they can adopt.

Imagine that our robot wants to know what the traffic police in Fig.(1.1) is trying to signal. For that purpose we should consider the different images in Fig. (1.2). We should first recognize where the open hands of the police are. Regarded
CHAPTER 1. INTRODUCTION

as shapes, Fig. (1.2a), (1.2b), (1.2c) and (1.2d) are similar while (1.2e) is different. Fig. (1.2d) has not the shape of an open hand although it is the same object. Once we know where the open hands of the police are, we would like to know what traffic sign is doing. Regarded as rotation Fig. (1.2b) and (1.2d) and (1.2e) are similar and different to the others.

(a) Stop  
(b) Go to the right  
(c) Go to the left  
(d) What to do?  
(e) Turn right

Figure 1.2: Different images of hands

Our objective in this project is to factorize appearance and rotation of each image in such a way that classification could be easily done regarding these two characteristics or a combination of both. This is shown in Fig.(1.3). Possible applications are shape recognition or pose estimation. If we consider the traffic example, appearance classification returns images of hands and rotation classification returns the direction of this hands. However, if we encounter images such as 1.2d the human intuition gives a "turn right". In case that traffic is led by traffic police, we should obey them and not consider the traffic signs. We should make a difference between 1.2d and 1.2e. Is 1.2d really a hand? A combination of rotation and appearance is needed in this case.

The main field of our approach is feature extraction. It is one of the most important steps in image processing and computer vision. The aim is to remove the information in the images that are irrelevant for the task. In this case, appearance and rotation are relevant features. How to get appearance and rotation from an image? Using the gradients of an image for describing its shape is a good start. However, how could we get the rotation of an image? Are the gradients enough for describing a shape? We need to transform these gradients with the Fourier transform. This transformation is the main contribution of this project and it is explained in Chapter 3.

1.1. Context

Keeping in mind the importance of feature extraction, there is extensive literature on it. David G. Lowe, in his classical work Distinctive Image Features from Scale-Invariant Keypoints describes its approach to features that cover the image over the full range of scales and locations by using a Gaussian scale-space kernel [6]. Navneet Dalal and Bill Triggs study another approach for Human Detection in [5].
1.1. CONTEXT

The descriptor proposed is based on local histograms of image gradient orientations. The development of gradients features in our approach can be traced back to this work. In terms of shape recognition, Belongie, Malik and Puzicha [2] introduce a shape descriptor which describes the coarse distribution of the rest of the shape with respect to a given point on the shape, i.e. shape context. This is done for each of the points of the image.

Currently, most of the feature extraction task is done by using Histograms of Oriented Gradients (HOG), Scale Invariant Feature transform (SIFT), Speeded Up Robust Feature (SURF) [1] or shape contexts. The two main problems with these descriptors are their high dimensionality.

Images are easily affected by factors such as illumination, shadows or cluttered background. In order to reduce these effects a normalization equalization could be added to the HOG descriptor, as Navneet Dalal do in [4]. Other problems concerning images are occlusion and compression but they keep far from the main problem of this project.

The main information for describing an object is found in the boundary of the object. Imagine we have to recognize the letter 'o'. The boundaries consist of two concentric circles. A curve is represented parametrically as a function of arc length by the accumulated change in direction of the curve since the starting point. This function is expanded in a Fourier series. Since each boundary of 'o' can be decomposed into a set of closed curves, the Fourier transform becomes ideal for the description of these closed curves. This is done by Zahn and Roskies in [12] and Persoon and Fu in [9]. Concerning rotation properties, other transforms (such as Mellin transform) has been considered, as in [3].
Most of the studies in shape recognition is based on handwritten digits or letters for touch screens, as illustrated in Fig. (1.4).

Figure 1.4: Handwritten digits

An overview of all the descriptors depending on the interest region detector is introduced in [8].

1.2. Goals

The goal of this work is to create a new descriptor that could focus on different information of an image. The final features are factorized regarding rotation and appearance.

\[ \text{FinalFeature} = \alpha \text{appearance} + \beta \text{rotation} \text{ for } \beta = 1 - \alpha \]  \hspace{1cm} (1.1)

The main objective is to make a flexible feature extraction process where the descriptor extracts different kinds of features, i.e. appearance and rotation, and only a factor(\(\alpha\)) makes the features of the image being a combination of appearance and rotation depending on what we want to recognize from the images. This will help us to differ images (1.2d) and (1.2e) since (1.2d) is a combination of a hand of a police and a traffic signal.
1.3. APPLICATION

Once the descriptor is designed, it should be tested. There are different alternatives for classifying images. In this project the results achieved with K-nearest neighbour (KNN) algorithm and Support Vector Machine (SVM) will be studied.

The whole process of feature extraction and classification of this project is illustrated in Fig. (1.5)

1.3. Application

Feature extraction is used for a large range of computer vision tasks. The main application considered here is hand pose estimation [10]. Depending on the task, the estimation of the pose requires rotational dependence while we do not care about appearance since all the images are hands. When estimating the pose, we could search for exact poses or we could even define a margin in which we accept equal rotations of hands. This hand pose estimation could be used in traffic applications as exemplified throughout this thesis.

1.4. Thesis outline

The rest of this thesis is structured as follows:

Chapter 2 discusses the background and the underlying concepts that are most commonly used in this area.

Chapter 3 presents the descriptor designed in detail.

Chapter 4 introduces the results and comments in this project.

Chapter 5 presents the conclusions and the Future Works with some suggestions of possible improvements.
Figure 1.5: System outline
Chapter 2

Background

Now that we know the goals, applications and motivation behind our approach, we discuss in this chapter the state of the art on different aspects used in this descriptor and its application. Feature extraction problem has been widely studied since a few decades back and is still an active research topic. Fig.(2.1) shows an overview of a general image classification chain for an image. The main process is divided into three steps: feature extraction, transformation computation and application of the classifier.

What is the role of these three blocks? Considering the traffic example, we need to recognize every image as a hand or something else. The gradients of an image, the first block, give different information of the image than appearance and rotation separately. As a consequence we need to transform those features. Once we have the proper features, we need to test our approach. Are we outperforming the existing descriptors? A classifier helps to test our descriptor. We should also consider that supervised classification requires labeling of the images being classified.

We start by describing the different state-of-the-art descriptors in Sect. 2.1. Section 2.2 provides details of the transformation. A possible transformation is described in Sect. 2.3. Further sections describes all the necessary details for the classification. Section 2.4 describes SVM and Knn classifiers. It is also important to argue what dataset we use, described in Sect. 2.5. The final classification is discussed in Sect. 2.6.
2.1. Features

In computer vision, the concept of a feature is referred to as a piece of information which is relevant for solving a specific task. Features from images are the most important issue in this project. This project could be synthesized in the following questions: Which features from the images should we consider to make two images showing hands have the same features? Or which features from the images should we consider to make two images showing vertical and horizontal hands have different features?

There has been a lot of investigation on shape descriptors. We can consider two types of descriptors in this thesis: bag of words descriptors (i.e. SIFT and HOG) and shape contexts. Each approach has its advantages concerning computational cost and feature dimensionality. Furthermore, their different motivations gives intuitions in a possible and appropriate beginning for shape recognition features.

2.1.1. HOG

We need to describe a specific object in an image. We want to describe the shape of that object. If we need to describe the shape of that object, we only need to focus on the edges of that object. How to get these edges? For getting the edges of an object in an image we should look for the points in the image where brightness changes sharply. These sharp changes are measured with derivatives. The definition of a derivative is given in (2.1) When there is a sharp change in brightness, the derivative gets a high value. However, images are 2 dimensional, $x_1, x_2$. In this case we introduce the concept of gradient, (2.2). We would like to know not only how brightness changes are but also in which direction, as shown in Fig. (2.2). Now we have the description of object in the image $f(x_1, x_2)$ in a 2-dimensional function, the gradients of the image, $\nabla f(x_1, x_2)$. A more intuitive graphical representation of $\nabla f(x_1, x_2)$ is the histogram. It shows a visual impression of the distribution of data. At this point we have computed a Histogram of Oriented Gradients (HOG).

\[ f'(x) = \frac{\delta f}{\delta x} = \lim_{h \to 0} f(x) = \frac{f(x + h) - f(x)}{h} \quad (2.1) \]

\[ \nabla f = \left( \frac{\delta f}{\delta x_1}, \frac{\delta f}{\delta x_2} \right) \quad (2.2) \]

An overview of the typical HOG chain can be found in Fig.(2.3) [4] [5]. Here we describe each step of this chain:

- **Do the shadows belong to the shape of the object?** A gamma and color normalization is required in order to reduce the illumination effects in the image.
- **The main step, computation of the gradients.** Further methods also use second order image derivatives which are useful for
2.1. FEATURES

more complicated contours and silhouettes as it could be some structures in bicycles.

- **Division of the image.** The image window is divided into small spatial regions, called "cells". For each cell we accumulate a local 1-D histogram of gradient or edge orientations over all the pixels in the cell. This combined cell-level 1-D histogram forms the basic "orientation histogram" representation. Each orientation histogram divides the gradient angle range into a fixed number of predetermined bins. The gradient magnitudes of the pixels in the cell are used to vote into the orientation histogram.

- **Second Normalization.** In order to be more invariant to illumination, shadowing and edge contrast, another normalization could be done. In this case, we take local groups of cells (a block) that normalize their histograms. Typically each individual cell is shared between several blocks.

- **The final vector.** In the final step we collect the HOG descriptors from all the blocks and create a combined feature vector.

An illustration of HOG features is given in Fig.(2.4). In this example we are using 418 cells. The higher the number of cells (i.e the lower the size of the each cell), the more precise the histogram will be. In this example we are using 20 orientation bins, 20 different gradients. The higher the number of orientation bins in each cell, the more precise the descriptor will be.
CHAPTER 2. BACKGROUND

Figure 2.3: Overview of the HOG feature extraction

(a) Image of a giraffe
(b) HOG of giraffe

Figure 2.4: HOG of an image with bin size 5 and 20 orientation bins in each cell
2.1. FEATURES

HOG upholds invariance to geometric transformations except for object orientation. This is very important in object recognition since the objects could appear translated. Imagine images as shown in Fig.(2.5). The gradients of both lines are shown in Fig(2.6). They both have the same gradients and therefore they will have the same histogram of oriented gradients, Fig.(2.7). Imagine we rotate the given line, Fig. (2.8). Since they have different gradients, Fig.(2.9), they have different HOG features, Fig.(2.10).

![Figure 2.5: translated images](image)

Why the image should be divided into small rectangular regions, i.e rectangular cells? It depends on the shape of the object and where most of the information is located in the image. If most of the information is in the center, we use a circular cell. If the information is uniformly distributed, we use a rectangular cell. If the object is articulated, we use a Bar HOG. The most common shape used for cells in HOG descriptors is rectangular. However, a circular cell could also be applied as shown in Fig.(2.11) [4]. The circular HOG implies a bigger cell size when moving away from the center. This implies that more pixels are averaged in the outer cells than in the inner cells, so that the descriptor resolution decreases when moving away from the center. This could be useful when most of the information is in the center of the image. Apart from these three configurations, there is the Bar HOG. This uses oriented second derivatives, i.e bars, instead of first derivatives. This approach could be useful for articulated objects such as animals and human beings. In our case, hands could be better modeled by a bar HOG but our approach is done with R-HOG.

In overall, HOG captures local contour information, i.e. the edge or gradient structure that is highly characteristic of shape description. It allows to change the
precision and invariance of the information given by the gradients. This is due to
the controllable number bins for each orientation histograms, i.e for each cell, and
the cell size. The power and simplicity of HOG is the main reason why we use it in
this project.

2.1.2. SIFT

SIFT extracts keypoints of objects and shape matching is then identified by
consistent clusters of this keypoints [6]. The first stage is to determine the location
of these keypoints. Detecting locations that are invariant to scale change is done by
a Difference-of-Gaussian (DoG) function. The maxima and minima of this function
2.1. FEATURES

gives the most stable image features within different scales compared to other possible image functions such as the gradient (HOG). The convolution of the image with the DoG function is done at different scales. Once we have the keypoints locations, we compute the gradient magnitude and orientation at each image sample point in a region around the keypoint location. These are weighted by a Gaussian window as shown in Fig. (2.12)

It is proved that the number of keypoints rises with increased sampling of scales.
CHAPTER 2. BACKGROUND

Figure 2.10: Hog of lines in Fig.(2.8)

(a) R-HOG  
(b) C-HOG  
(c) Single centre C-HOG

Figure 2.11: Different shapes for the HOG cells

Figure 2.12: Image gradient
2.1. FEATURES

and the total number of correct matches also rises. However, the cost of computation also rises with this number. It gets good response when distortion, partial occlusions and illumination changes are involved. Fig.(2.13) shows an example of SIFT feature extraction.

![Image](a) Image ![Image](b) SIFT of the Image

Figure 2.13: (b) shows the initial 832 keypoints. Both images are courtesy of David Lowe

2.1.3. Shape Context

A different approach is shape context, a robust algorithm for finding correspondences between shapes [2]. Imagine we travel to London and Rome. We are only interested in the Olympic stadium of London and the coliseum of Rome, Fig.(2.14). We are traveling with some friends. We first visit the Olympic stadium of London. Each of our friends sit in a different place all around the stadium. Each of us have a different view of the stadium. Afterwards, we go to the coliseum of Rome. In the same way as in the stadium, each of our friends sit in the same place all around the stadium. How do we know if both monuments have same shape? Each of our friends have different view of the monuments. Both monuments would have the same shape if each our friends have the same view in the stadium than in the coliseum. This is what Shape Context descriptor does.

Fig. (2.15) and (2.17) show an example of shape context descriptors. The reference samples are ◦ , ◦ and ◌ which are edge sample points of the letter A . Each sample context is a log-polar histogram of the coordinates of the rest of the point set measured using the reference sample as the origin. The diagram of log-polar histogram bins used is shown in Fig.(2.16). The shape matching is done by finding similar shape context for each pair of points.

This gives a robust, compact and highly discriminative descriptor as well as invariant to several kinds of transformations such as occlusion. On the other hand, it is far too detailed descriptor by computing \( n - 1 \) vectors for each point on the contour of the shape being \( n \) the number of edge sample points.

The main difference with HOG is that the shape context does not compute any
mathematical formulae, i.e. gradients. Furthermore, it considers all the points in the shape of the object and not only keypoints, as SIFT does.

Figure 2.14: Shape Context example

(a) Shape 1 of letter A  (b) Shape 2 of letter A

Figure 2.15: Two different sampled edge points of letter A (courtesy of Serge Belongie)

**When to use SIFT, Shape Context or HOG?** The main application of SIFT is the localization of a specific object in a test image that contains many other objects. The features extracted are scale invariant so they are able to detect the object under changes in scale, as shown in Fig.(2.18). On the contrary, HOG and Shape Context are used for shape description. As an example, two circles of different size have the same gradients but the occurrences of each gradient are not the same (a scale difference) and therefore they do not have the same features. Considering the shape context, How many points do our two circles have? $p$, where $p$ is a large number. We should compute $p$ 2-dimensional histograms for each circle, being all the histograms very similar to each other. HOG computes a 1-dimensional histogram.
2.1. FEATURES

Figure 2.16: log-polar histogram bins (courtesy of Serge Belongie)

Figure 2.17: (a) and (b) are log-pol histograms very similar. However, there are totally different from (c) (courtesy of Serge Belongie)

Moreover, two different shapes could have the same keypoints and therefore same SIFT features. This way, the best option for shape description is HOG descriptor.

Figure 2.18: Object recognition with SIFT
2.2. Fourier Transform

Returning to the overview of our system, Fig.(2.1), we have computed the gradients of the image, a 1-D histogram of gradients (HOG). However, these are not the features we wish to have. We need a transformation of these features into the required ones.

Imagine we have two images of forks and their histograms as shown in Fig.(2.19). If we focus on their appearance we can label both images as forks. However, their histograms are not the same. They do not only represent the appearance of the image. We should find a transformation that make these histograms equal when sharpening appearance. If we focus on rotation we can label the first image as 0 degrees and the second as 90 degrees. Their histograms do not give any intuition when labeling the rotation of the image. However, if we translate the histogram of the horizontal fork to the right, Fig (2.20), the maximums of both histograms are placed in the same x-position. This way the histograms look more similar. It seems that the more we translate one histogram to make it look similar to the other, the more dissimilar rotation labeling they will have. In case that we translate the histogram of the horizontal fork to the left and not to the right, we should consider $2\pi$ periodicity of the gradients, as the maximums should appear in the right part of the histogram. We are looking for a transformation with two main properties. First, it should consider appearance and rotation of a picture separately and secondly, it should respect the $2\pi$ periodicity. This transformation is the Fourier transform.[12][9][3]

One dimensional Fourier transform will help us to get a histogram of the appearance and a histogram of the rotation of the image separately. The general definition of Fourier transform is defined as follows:

$$ F(u) = \frac{1}{M} \sum_{x=0}^{M-1} f(x)e^{-2j\pi ux/M} \text{ for } u = 0, 1, 2, ..., M - 1 \quad (2.3) $$

However, there is a more intuitive notation of the Fourier transform:

$$ F(u) = A(u)e^{j\psi(u)} \text{ for } u = 0, 1, 2, ..., M - 1 \quad (2.4) $$

The magnitude of $F(u)$ is $A(u)$ and the phase is $\psi(u)$. Once we know the definition of the Fourier transform we will see why it is interesting for this project. If we consider $f(x)$ as the histogram of the vertical fork of Fig(2.19) and $h(x)$ the histogram of the horizontal fork, the translation property of the Fourier transform implies same magnitude $A(u)$ for $f(x)$ and $h(x)$ but different phase $\psi(u)$. This way, same magnitude implies same appearance(both are forks) and different phase implies different rotation(vertical and horizontal forks).

The main motivation for using the Fourier transform is the translation property:
2.2. FOURIER TRANSFORM

Figure 2.19: Histogram of two different images. The upper histogram is $f(x)$ and the lower histogram is $h(x)$

For any real number $x_0$, if $h(x) = f(x-x_0)$, then $H(u) = F(u)e^{-2j\pi ux_0}$ (2.5)

This translation property with the polar notation is:

$$H(u) = F(u)e^{-2j\pi ux_0} = A(u)e^{j\psi(u)}e^{-2j\pi ux_0} = A(u)e^{j\psi(u)-2j\pi ux_0}$$ (2.6)

The magnitude of $H(u)$ is $A(u)$ and the phase is $\psi(u) - 2\pi ux_0$. The translation of $h(u)$ with respect to $f(u)$ is $x_0$. From here, we can get the intuition that the appearance factor will be related to the magnitude and the rotation factor to the
Imagine that we compute the HOG features of a same image with different number of bins. The more bins we have, the more accurate the histogram describes the actual edge-distribution in the image and the longer the histogram vector is. However, the magnitude $A(u)$ of the different histograms should be the same as they all come from the same image. The magnitude of the Fourier transform has some peculiarities. Two histogram vectors with the same shape but different length will not have the same magnitude, as shown in Fig. (2.21). There are two ways of making these histograms equal. One way is to include zeros at the end of the histogram. The magnitude will have the same shape as shown in Fig. (2.22). Another way is to remove all the zero samples of the histogram and repeat the sequence until a fixed histogram length is reached. If a sequence is a repeated version of a smaller sequence, they will both have the same magnitude, as shown in Fig. (2.23).
2.3. CROSS CORRELATION

Removing zeros from the histogram implies losing the votes of some gradients of the image. However, in this project it does not matter the value of each gradient as long as we obtain same magnitude for same appearance images.

What could be expected from the phase of two rotated functions? As stated in eq. (2.6), the difference of the two angles is $P(u) = 2\pi ux_0$. We should get the constant value $x_0$ from $P(u)$, i.e $x_0 = \frac{P(u)}{2\pi u}$.

2.3. Cross correlation

The Fourier transform seems to be a good transformation candidate. However, it is not very intuitive for rotation properties. There are more periodical tools that we could consider as it is the cross correlation. Cross correlation is a measure of similarity between two signals, very similar to convolution.

Imagine we are talking with a friend. Suddenly, a man approaches down the street. It is very far away and we cannot recognize him. It is just a man. Time goes by and the man gets closer to us. Now, the man’s face is familiar to us. We know him but we do not know why. Finally, the man greets us and suddenly we...
CHAPTER 2. BACKGROUND

Figure 2.22: Magnitude of different histograms. The difference is some zeros at the end

remember why we know him. He is our friend’s brother. Now that we can see them together we can recognize how similar they are. He talks with us and then he continue walking through the street. For a while, we will still remember him but when he disappears from the street we will once again forget him. Our friend is called $x$ and his brother is called $y$.

$$R_{xy}(m) = \begin{cases} \sum_{n=0}^{N-m-1} x_n y_n^* & \text{for } m \geq 0 \\ R_{yx}(-m)^* & \text{for } m < 0 \end{cases}$$

(2.7)

If we go into the mathematical formulas, we could get a nice interpretation. If we have two vectors $x$ and $y$ of length $N$, the formula (2.7) essentially slides the $x$ function along the $x$-axis, calculating the mean value of their product at each position. When the functions match in time $m$, i.e when they are similar, the value of $R_{xy}(m)$ is maximized. We can see an example with the following functions, Fig.(2.24). $y$ is a translated version of $x$. The cross correlation slides the vector $x$ over time, Fig.(2.25). It represents the common area (gray area) below both functions, Fig.(2.26). Since the red function, $y$ is translated one sample from the
2.3. CROSS CORRELATION

Figure 2.23: Magnitude of repetitive signal

vector $x$, the maximum of the cross correlation is moved one sample from the center. This is the same example as illustrated in Fig. (2.29), that we will explain later.

If we have two identical vectors $x = [1234]$ and $y = [1234]$, the cross correlation over time is (2.27). If we have two vectors such as $x = [1234]$ and $y = [2341]$, the cross correlation is (2.28). If we have two vectors such as $x = [1234]$ and $y = [3412]$, the cross correlation is (2.29). The maximum of the cross correlation stays in the center when the vectors are the same. Otherwise, the maximum moves as many samples as the vectors are translated. This is a simple way of guessing the rotation within images. We just need to compute the cross correlation of their histograms.

Figure 2.24: 2 functions. The red is the translated version of the green function
Cross correlation does not imply any transformation. It could be used for getting the rotation between two images but there is no way of getting the appearance similarity between two images.

### 2.4. Classifier

Finding correspondences between images could be divided into two steps [1] as shown in Fig(2.30). First, the descriptor is constructed by computing the gradients
2.4. CLASSIFIER

Figure 2.27: Cross correlation between two identical vectors

Figure 2.28: Cross correlation between a vector and a 1-sample rotated version

Figure 2.29: Cross correlation between a vector and a 2-sample rotated version
CHAPTER 2. BACKGROUND

of every pixel in a feature vector. This descriptor should be factorized in rotation and appearance by the Fourier transform. We are here now. Secondly, we match the descriptor vectors of different images to find similarities between them. This matching is also known as the classification problem. We are in the last block of Fig (2.1). It is a quality measure of our descriptor and we should consider three main issues: the dataset, the classifier and the labeling of the images.

![Figure 2.30: Feature extraction and classification problem](image)

For getting a general acquaintance of the performance of our system we build a classifier. There are many classifiers we could try: Knn, SVM, Perceptron, ANN or Naive Bayes Classifier. The KNN algorithm is amongst the simplest of all machine learning algorithms. It is a non-parametric algorithm, i.e it only depends on the data. SVM belongs to a family of generalized linear classifiers and can be interpreted as an extension of the perceptron (ANN). It is a parametric algorithm. The accuracy of a SVM model is largely dependent on the selection of the model parameters. We build the model with the training dataset and once we have this model, we test it with the testing dataset. On the other hand, in the KNN algorithm, we have to build the model each time we want to test the system with a testing sample. There are no parameters to support the whole training.

We try two options: SVM, for its versatility and KNN algorithm, for its simplicity. Both classifiers are explained below.

### 2.4.1. KNN algorithm

The KNN algorithm is the simplest amongst all the classifiers. For every test image, KNN classifier searches for the $k$ (in our case $k = 3$) nearest images in the training dataset. These $k$ nearest images are determined by a similarity measure between images. Each of these $k$ images belongs to a specific class. The majority class among the $k$ images is the estimated class.

**How to define the similarity between images?** This is called distance measure and is a similar conceptually to a kernel. Kernel is an algorithm for pattern analysis, i.e for general types of relations (clusters, correlations, principal components, classification) in data. Different types of Kernels are described in 2.4.2.
2.4. CLASSIFIER

2.4.2. SVM

Introduction

The SVM is a method used for classification and regression of data. We can distinguish two processes: training and testing. Given a set of labeled training examples, the SVM constructs a model that assigns new examples into one category or to the other. That model is made by the construction of a hyperplane or set of hyperplanes that separates the different classes.

The SVM becomes interesting when the sets of data are not linearly separable. For this reason, the original observed space is transformed in order to find a space where the sets are linearly separable, as illustrated in Fig.(2.31). SVM tries to simplify the classification problem after doing feature extraction. However, the main goal of this project is to simplify classification by a new feature extraction approach, as shown in Fig.(1.3). We will use SVM as another classification model to judge our new approach since our goal is to make the sets of data linearly separable before the classification process.

![Figure 2.31: Transformation of the dataset](image)

The decision boundary, the hyperplane, is placed to be as far as possible from the available data points. The location of the separating hyperplane involves a maximization problem. We want to maximize the distance of the decision boundary to any datapoint with the constraint that our decision boundary should be a linear function, as shown in Fig.(2.32).

For constructing the SVM these are the steps to follow [7] :

1. Choose a suitable Kernel function $\mathcal{K}$

   A Kernel function projects the input data into a new space where we hope the classification problem is linearly separable, as shown in Fig.(2.33). Choosing a Kernel function is the same as choosing the suitable new space you want to project your data so that it is linearly separable.

2. Solve the maximization problem.
3. Classify new data points $\mathbf{x}$ by the following via:

$$
\sum_{i} \alpha_i t_i \mathcal{K}(\mathbf{x}, \mathbf{x}_i) > 0
$$

(2.8)

t_i$ is the label of each training datapoint. The maximization problem computes $\alpha_i$ for each datapoint $\mathbf{x}_i$. $\mathbf{x}_i$ corresponding to $\alpha_i \neq 0$ are called the support vectors.

**Type of Kernel**

There are different ways of measuring the similarity between two vectors. The simplest method is the Euclidean distance.

**Euclidean distance** The Euclidean distance is as follows:

$$
d(a, b) = d(b, a) = \sqrt{\sum_{i=0}^{n} (h_{a,i} - h_{b,i})^2}
$$

(2.9)

$h_a$ and $h_b$ are the histogram of gradients (HOG) of image $a$ and $b$ respectively.
2.4. CLASSIFIER

However, there is a problem when using the euclidean distance on histogram of oriented gradients. In case that we have the following histograms, Fig.(5.2) the distance between all of them is the same. However, the rotation is not the same.

Figure 2.34: 3 histograms of rotated images

**Kernel functions**  An alternative is to use the following functions:

- **Linear Kernel**
  \[
  \mathcal{K}(\vec{x}, \vec{y}) = \vec{x}^T \cdot \vec{y} + 1 
  \]
  (2.10)

- **Polynomial kernel**
  \[
  \mathcal{K}(\vec{x}, \vec{y}) = (\vec{x}^T \cdot \vec{y} + 1)^p 
  \]
  (2.11)

The exponent \( p \) controls the degree of the polynomials. \( p = 2 \) will make quadratic shapes (ellipses, parabolas, hyperbolas).

- **RBF kernel**
CHAPTER 2. BACKGROUND

\[ \mathcal{K}(\vec{x}, \vec{y}) = e^{-\frac{(\vec{x} - \vec{y})^2}{2\sigma^2}} \]  \hspace{1cm} (2.12)

The parameter \( \sigma \) can be used to control the smoothness of the boundary.

- Sigmoid kernel

\[ \mathcal{K}(\vec{x}, \vec{y}) = \tanh(k \vec{x}^T \cdot \vec{y} - \delta) \]  \hspace{1cm} (2.13)

Parameters \( k \) and \( \delta \) need to be tuned to get the best performance.

**How to transform the original observed space? What is \( \Phi \)?** We should find a transformation \( \Phi(\vec{x}) \) of \( \vec{x} \) so that the sets are linearly separable.

If we define \( \vec{x} \) as :

\[ \vec{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \]  \hspace{1cm} (2.14)

and we define \( \Phi(\vec{x}) \) as :

\[ \Phi(\vec{x}) = \begin{bmatrix} x_1^3 \\ \sqrt{3}x_1^2x_2 \\ \sqrt{3}x_1x_2^2 \\ x_2^3 \end{bmatrix} \]  \hspace{1cm} (2.15)

We can show that \((\vec{x}^T \vec{y})^3 = \Phi(\vec{x})^T \Phi(\vec{y})\). This equality involves the Kernel definition:

\[ \mathcal{K}(\vec{x}, \vec{y}) = \Phi(\vec{x})^T \Phi(\vec{y}) = x_1^3y_1^3 + 3x_1^2y_1^2x_2y_2 + 3x_1y_1x_2^2y_2^2 + x_2^3y_2^3 = (x_1y_1 + x_2y_2)^3 = (\vec{x}^T \vec{y})^3 \]  \hspace{1cm} (2.16)

Kernel function defines the inner product in the feature space which is equivalently expressed as:

\[ \mathcal{K}(\vec{x}, \vec{y}) = \Phi(\vec{x})^T \Phi(\vec{y}) = \| \Phi(x) \| \| \Phi(y) \| \cos(\alpha) \]  \hspace{1cm} (2.17)

If we know \( \alpha \) and the length of the vectors, we know the distance between two points, Fig.(2.35). If we know this distance, the space is defined. We can conclude that distance between two points and Kernel are similar concepts.

2.5. Dataset

One of the most important aspects of a classifier is the training and testing data, i.e. the dataset.
2.6. LABELING

How big should be the training and testing set for same class? What is more important, training or testing? Imagine we want to learn all different kind of animals. At the end of the semester we will have an exam. The more animals we learn, the better results we have in the exam. Usually, the more data we have the more we can trust the results obtained. The teacher has 10 animals per specie. She gives us 7 to learn by hand. The remaining 3 will be asked for the exam. We study hard for the exam. We memorize each animal instead of analyzing the common features of each specie. We do the exam and we fail. Why? We were not able to recognize new animals. In case that the samples in the exam were the animals that we studied, we would have the maximum score. The main problem with the size of the training and testing set is overfitting. This problem arises when learning was performed too long. When overfitting occurs, the performance of the classifier on the training examples becomes better for bigger training set while the performance on testing data becomes worse.

When classifying with SVM, each sample could only be classified within two classes. Having this in mind, we can find 2 different patterns of the dataset: 2-class supervised classification (One against one) or multi-class supervised classification (One against many). "One against one" implies 2 classes (i.e. forks and butterflies) and every sample has its proper label (i.e. forks are class 1 and butterflies are class 2). However, "one against many" implies images from more than 2 classes (i.e. forks, butterflies, snakes, trees, etc) but we still have 2 main labels (i.e. forks are class 1, butterflies are class 2, snakes are class 2 and trees are also class 2). We should select a class as the interesting one, i.e. forks, and all the non-interesting classes (butterflies, snakes, trees, etc) merge to one same class label. It varies two facts: first, the labeling of all the images in our training and testing dataset and second, the total number of images. Intuitively, the "one against one" classification seems much weaker since it has less images. On the other hand, this classification is much easier since the two classes are well characterized, well cluttered.

For more information on what dataset or case studies we use you could read appendix A and appendix B respectively.

2.6. Labeling

In supervised learning, we have training examples and testing examples. A training example is an ordered pair $<x, y>$ where $x$ is an instance and $y$ is a label. A testing example is an instance $x$ with unknown label. The goal is to predict labels
It is supervised for the fact that someone, a supervisor, has provided explicit labels to the training examples.

2.7. Summary

This chapter gave an overview of the main concepts used for implementing our descriptor. There are two parts on this project: implementing the descriptor and testing the performance of our new descriptor. This is done by comparing with the basic HOG descriptor.

Now we have a basic notion of what we need to use in order to design our descriptor. Most of these tools were used in order to get the best performance of our descriptor. Concepts such as the Fourier transform and HOG descriptor are crucial for the final design of our descriptor. In the next chapter we present our novel approach of descriptor.
Chapter 3

Descriptor

Figure 3.1: Overview of the system

At this point, we are ready to make the robot drive on behalf of us. He is ready to translate the traffic police signals. Having the intuition and all the tools that we introduced in chapter 2, this chapter will explain how to construct the new descriptor.

Coming back to chapter 1, feature extraction is the starting point for many computer vision algorithms. Behind this feature extraction stands a descriptor; an image processing operation.

The main goal of this project is to provide a feature descriptor. Our approach is based on the HOG descriptor, 2.1.1. It is assumed that appearance and rotation can be characterized by intensity gradients or edge directions independently of the position of these gradients. Our final feature descriptor will depend on appearance and rotation of the image, (1.1). The Fourier transform is the perfect tool for factorizing the HOG descriptor into rotation and appearance descriptors, 2.2.

The main goal of a descriptor is to characterize an image in few words. An image has different ways of being characterized. Given an image of a beach, as shown in Fig.(3.2), a common descriptor is "blue, water, sun, sand". But this is not the only one. Another descriptor could be "towel, newspaper, book" or for a child could be "ball, sand castle, swimsuit, friends". There are different descriptions of the same image. In Section 2.1 we introduce the main descriptors used in computer vision. However we are looking further from this. We will implement two descriptors: appearance descriptor and rotation descriptor. Two images that represent objects with same shape should have same features, i.e. 1.2a and 1.2b should have the same
features. This is the appearance descriptor. On the other hand, two images with not exactly the same object but same rotation should have the same features, i.e. 1.2e and 1.2b should have the same features. This is the rotation descriptor.

![Figure 3.2: Is there only one way of describing this image?](image)

This chapter begins by giving an overview of the proposed descriptor (Fig. (3.1)) in Sect. 3.1. Section 3.2 introduces all the important properties concerning appearance. In the same manner, Sect. 3.3 provides further details for rotation. Regarding classification, there are some issues changed from the classical classifier, done in Section 3.4.

### 3.1. Overview of our descriptor

Our descriptor could be summarize in 3 steps:

1. Compute HOG on an image
2. Compute Fourier transform of the HOG
3. Compute magnitude and phase of the Fourier transform

In this project, we implement two descriptors: the initial descriptor and the proposed descriptor. The proposed descriptor is a improved version of the initial descriptor. The initial descriptor, shown in Fig. (3.3) removes the -90, 0, 90 and 180 gradients from the histogram. The reason is compression factor found in the images. The proposed descriptor shown in Fig. (3.4) includes some processing in
3.1. OVERVIEW OF OUR DESCRIPTOR

the histograms in order to improve the appearance descriptor. Although we implement different descriptors both have the three common basis described above.

Figure 3.3: Initial descriptor

Figure 3.4: Proposed descriptor

A gradient of a scalar field is a vector field that points in the direction of the greatest rate of increase of the scalar field, and whose magnitude is that rate of increase. In this case, the scalar field is the matrix that represents an image. Imagine our image is a black vertical line in white background. The gradient is shown in Fig.(3.5).

A proper HOG descriptor was implemented. Our HOG process is described in Fig.(3.6) and could be summarized in the following steps:

1. The image is divided in one cell
2. We compute the gradient of the image
3. We take out the edges effect, shown in Fig.(3.7) The edges will give a high 0, 90, 180 and -90 degrees which do not belong to the image.

Our HOG implementation has 2 changes from the original definition of HOG:

- It takes out the edges gradients of the image.
It divides the histogram into two parts: the 0, 90, -90, 180 gradients and the rest. This is due to a possible compression artifact in the images. As shown in Fig(3.8), it appears a noisy square grid in the image that makes 0, 90, -90 and 180 gradients being higher in the histogram.

Taking out this four gradients implies a loss of information in the histogram.
3.1. OVERVIEW OF OUR DESCRIPTOR

![Figure 3.7: Overview of our descriptor](image1)

![Figure 3.8: Compression artifact](image2)

Unless our HOG has a very high number of bins, each bar of the histogram accumulate the votes of more than one gradient, as we could see in 3.9. The higher the number of bins we have, the less information is lost by reducing the number of bins. This is because the portion of gradients for each bar in the HOG is lower. We should guess in which portion of gradients this four particular gradients are in order to take them out.

Since -90,0,90 and 180 gradients give much of the information in the image, i.e. hands or forks, it is important to keep them. The best option is to compute separately the similarity of the histogram from the similarity of these four gradients.

Once the HOG descriptor is computed, we apply the Fourier transform, its magnitude for getting the appearance descriptor and its phase for the rotation descriptor. So far, we have described the initial descriptor. However, we still do not have same features for same shape objects as 1.2a and 1.2b. The appearance descriptor of the initial descriptor should be changed. In the next section we
explain the main difference introduced in the proposed descriptor.

3.2. Appearance

We are looking for the hands of the traffic police, 1.1. As introduced in chapter 1, 1.2a and 1.2c should have the same features. This represents the appearance features. HOG does not give much intuition of the appearance of objects as in Fig.(2.19), both forks have different histograms. In the initial descriptor, we compute the magnitude of the HOG. This was motivated by the translated property of the Fourier transform, 2.6. However, \( h(x) \) and \( f(x) \) is not exactly the same for different number of bins and same image. We will motivate the different blocks introduced in the proposed descriptor.

3.2.1. Fourier Transform

Constant length of the histogram

Every image corresponding to the same shape-object should have the same appearance histogram(i.e magnitude of the histogram) independently of the number of bins. If we compute the HOG of an image with \( c \) bins, the histogram will have length \( c \). If we compute the HOG of an image with \( c + 1 \) bins, the histogram will have length \( c + 1 \). If we vary the number of bins, the histograms of the same image will have different lengths. The problem is that every sample in the histogram has effect in each of the samples of \( F(u) \), as we could check in (2.3). This leads to different appearance histogram for the same object depending on the number of bins.

For solving this problem, the length of the histogram is fixed to 500, no matter the number of bins is being used. This is done by introducing as many zeros as needed at the end, 2.22. However, we encounter a problem for translated histograms. For \( x = [123400000] \) and \( y = [342100000] \), the magnitude of the Fourier transform is not
the same. Therefore, we repeat the histogram as many times as needed for a length of 500. This is explained later in "Repetition of the histogram".

Repetition of the histogram

The magnitude of the Fourier transform of \( x = [1234] \) and \( y = [3421] \) is the same as the magnitude is invariant to translations in the histogram. This is the main reason why the Fourier transform is used in this descriptor. If we have two images of the same object but one rotated with respect to the other, i.e horizontal fork and vertical fork, the magnitude of the Fourier transform for both are the same. However, if we have \( x = [12340000] \) and \( y = [34210000] \), the magnitude of the Fourier transform is not the same. For solving this problem, instead of adding zeros we add a repetition of the original vector, i.e \( x = [123412341234] \) and \( y = [342134213421] \). This property is exemplified in figure (3.10).

![Figure 3.10: Magnitude of different histograms](image)

Taking out zeros in between

For a same image we expect always the same magnitude vector of the histogram, no matter the number of bins used. However, if we vary the number of bins used
in HOG, the magnitude vector of the histogram is not the same for a same image, as shown in Fig.(3.11). This implies different similarities with other images depending on the number of bins.

Is this problem not solved with the constant length of the histogram explained above? Now the problem is not the length of the histograms but the location of the zero samples in the histogram, as shown in Fig.(2.21). By increasing the number of bins, the histogram is more accurate to the real histogram. It is a sampling principle. However, for black and white images, histograms are very poor and by increasing the number of bins more zero samples are located between non-zero samples. This leads to different histograms. For solving this problem, the zero samples between the non-zero samples are taken away so the magnitude of the Fourier transform is identical for same objects.

Figure 3.11: Magnitude vector of histogram for same image and different number of bins

3.3. Rotation

In this case, we would like to have different features for vertical and horizontal hands. However, the features should be related with the rotation of the object in the image. HOG does not reflect the rotation of the image.
3.4. CLASSIFICATION

Our first approach for the rotation descriptor uses the phase of the Fourier transform. The phase of the Fourier transform isolates the information associated with the rotation of the image, as we saw in (2.6). This is done in the same way for the initial descriptor and the proposed descriptor. The only difference is the zero padding introduced in the proposed descriptor.

As we will see in our experiments in Chapter 4, the rotation descriptor does not achieve results. Our rotation descriptor is not robust among different number of bins. In order to improve the robustness of the rotation descriptor, we try other tools involving the rotation of the objects in the images. This is the cross correlation.

The cross correlation does not involve any transformation of the features, as stated in 2.3. It is a distance measure between two images and it is computed from the 1D histogram, HOG, Fig.(3.12).

![Figure 3.12: Cross correlation for Rotation descriptor](image)

3.4. Classification

In order to assess our descriptors, we apply them to a classifier. In order to classify each sample, we need to define two basic issues: similarity measure between images, the Kernel function, and how to label the training data.

3.4.1. Kernel

How to measure the similarity between images from the HOG features, appearance features or the rotation features? This similarity or distance is estimated by the Kernel function.
If we recall from (1.1), the final distance is:

\[
FinalDistance = \alpha \text{DistAppearance} + \beta \text{DistRotation}
\] (3.1)

\text{DistAppearance} \text{ and } \text{DistRotation} \text{ are the distance considering the appearance or the rotation features, respectively. We consider that } \beta = \alpha - 1 \text{ with } \alpha \text{ and } \beta \in [0, 1] \text{ values.}

We have different Kernels for HOG, appearance and rotation. The Kernels used are:

- \textbf{HOG} Linear Kernel (2.10)
- \textbf{Appearance} Linear Kernel (2.10)
- \textbf{Rotation} Slight variation of the Radial Basis Function kernel (RBF).

The bigger rotation there is, the less similar these 2 images are.

This function is limited to $[0, 1]$ values.

**Slight variation of RBF Kernel** When considering appearance, the higher the similarity factor the more similar. This value goes from 0 to 1. Rotation should have the same trend. We should make the rotation factor be decreasing and between $[0,1]$. The lower the angle between 2 images, the higher the similarity. For this, the exponential function is a well known decreasing function.

When referring to RBF Kernel, the normal RBF function is the one specified in (2.12). However, this is not the function that we use. We follow the intuitions from the phase of the Fourier transform.

The angle of the Fourier transform will be:

\[
\overrightarrow{x} = \frac{2\pi u x}{M} \text{ for } u = 0, 1, 2, ..., M - 1
\] (3.2)

\[
dif(\overrightarrow{x}, \overrightarrow{y}) = \frac{-2\pi u x}{M} + 2\pi u x_0
\] (3.3)

The difference that we want is $dif = x - x_0$ where $x$ is the corresponding bin from the histogram $f(x)$.

$x_0$ is the rotation in radians that we are looking for and not the whole $\overrightarrow{x}$. This is the main reason for using our own RBF and not the usual definition.

We will compute the rotation as follows:

\[
rot = \frac{\left| \frac{(x_1 - y_1)}{2\pi u_1} \right| + \left| \frac{(x_2 - y_2)}{2\pi u_2} \right| + \cdots + \left| \frac{(x_M - y_M)}{2\pi u_M} \right|}{M}
\] (3.4)

\[
\mathcal{K}(\overrightarrow{x}, \overrightarrow{y}) = e^{-rot}
\] (3.5)

where $\overrightarrow{x}$ and $\overrightarrow{y}$ are the angle of the Fourier transform of the histogram((2.3)).
3.4. CLASSIFICATION

**Why do we take the mean in (3.4)?** To find the rotation between two images, we should compare the phases of the Fourier transform of these two images ($\vec{x}$ and $\vec{y}$), as shown in ((2.6)). However, the angle of each image is a function (not a number) and the difference between the phases is not a constant function. At first, in order to get a numerical angle as the rotation of both images, we decide to take the difference of one random point of phases of the Fourier transform. Here we encounter 2 problems:

- This difference depends on the random point
- It is not a robust way of getting the rotation as it depends on a single point of the phase.

Finally, the Phase of the Fourier transform is better if we take the mean of the difference of the phases. This make it more robust than taking the maximum of the cross-correlation.

3.4.2. Labeling

Our system is supervised, i.e., the user has to label the training data and the testing data depending on the purpose of application. As an example,

- If we want appearance
  - class 1 butterflies
  - class 2 forks
- If we want rotation
  - class 1 vertical forks, vertical butterflies
  - class 2 horizontal forks, horizontal butterflies
  - class 3 diagonal forks, diagonal butterflies

The appearance labeling is easy. We focus on an object and all the pictures with that object are class 1 while the rest are class 2. However, it becomes difficult to label the images with respect to its rotation. Labeling is very important since the performance of our system depends directly on it. If the labeling is not correct, the system is predicting a different class than expected for a given image, so we can assume our system does not work well. However, it is only a problem with the labeling and not with the system.

**Rotation labeling**

We are looking for the angle in which the image keeps most of its information. It is difficult to get the rotation labeling of a circle, a star, a flower while it comes really easy to get the rotation of a stick, a fork or a tree. It becomes an easy issue by answering the following question: If we have to represent an image by a stick,
which direction would the stick have? That direction is the rotation label of the image.

We define 4 main orientations, 4 main labels, as shown in Fig.(3.13)

An example of rotation labeling is shown in Fig.(3.14)

![Figure 3.13: Rotation Labeling](image)

Figure 3.14: If we represent the bat as a stick, this stick will have a vertical orientation. It is labeled as class 2

![Figure 3.14: If we represent the bat as a stick, this stick will have a vertical orientation. It is labeled as class 2](image)

### 3.5. Summary

Designing the descriptor is the main goal of this project. In general terms, the descriptor is based on HOG. In order to get Appearance and Rotation features, we apply the Fourier transform. Depending on the compression artifact of the images, some changes should be done to the HOG. If our image has some compression artifacts, gradients 0, -90, 90 and 180 are separated from the general histogram. Furthermore, some changes should be done (i.e. fixed length, taking out zeros in
3.5. SUMMARY

between and repetition of the period) for making appearance features be the same for same appearance images.
At this point, we are ready to test our descriptor and compare with the performance of HOG.
Chapter 4

Experiments

Figure 4.1: Overview of the system

It is early in the morning. Our robot is prepared to travel from Gothenburg to Stockholm. It is going with a friend, robot HOG. They will both go in different cars and aim to meet in the center of Stockholm. They are not allowed to follow each other. However, we do not know if they will both manage to arrive in Stockholm. Our robot is ready for the trip but robot HOG could get lost. Of course, the descriptor described in Chapter 3 is the driving force of our robot. Once our descriptor is designed, we are ready for getting results and analyzing them to conclude i.e our descriptor results in an improvement. The implemented descriptor is tested with two different classifiers: SVM and KNN, as shown in Fig. (4.1).

We start by giving details of the different datasets used for testing, in Sect. 4.1. Sect. 4.2 shows the output of our similarity measure. Section 4.3 and Section 4.4 provides details of the results obtained with both descriptors. We provide a summary of both classifiers in Sect. 4.5.

4.1. Datasets

We test our descriptors with several different datasets.

- Sinusoids
- Images half white and half black
- Lines and 'L' images
CHAPTER 4. EXPERIMENTS

- Hand pose database
- Black and white images of different objects

The idea of the sinusoid functions is to test the Fourier properties in a function that we translate. Once we tried with functions we jump to images. The idea was to try our approach from images with very simple histograms of gradients to more elaborated. Some of the samples of the different datasets are shown in Fig. (4.2)

(a) Line database  (b) Hand pose dataset  (c) Black and white images

Figure 4.2: Different datasets

The hand pose database is a noisy database for two reasons: First, some compression artifacts makes the histogram have high -90, 0, 90 and 180 gradients, not corresponding this to the gradients of the image. Secondly, we encounter some shadows that could make the histogram not be a rotated version of other hand pose. This could be reduced by applying some normalization to the image before getting the gradients, as done in Fig.(2.3). These two problems disappear when using the black and white database.

Once we have the images, we will try our descriptors on them. Firstly, we present the results by distance matrices. This gives an overview of the performance for a given number of bins. Secondly, we present the results among different number of bins.

4.2. Distance Matrix

A distance matrix is a visualization tool that allows to get an overview of the performance of the descriptor. It compares the features of a set of images for a given kernel. Element $x_{i,j}$ of this matrix is the similarity between image $i$ and image $j$. This similarity is measured by a given kernel $x_{i,j} = \mathcal{K}(\vec{f}_i, \vec{h}_j)$ where $\vec{f}_i$ and $\vec{h}_j$ are the features of $i$ and $j$. The distance matrix is a gray-color matrix. White color means similar images while black means different images. Note that the distance matrix is symmetric.
4.2. DISTANCE MATRIX

In both classifiers SVM and KNN we use the same kernel/distance measure: Linear Kernel for appearance and HOG and our RBF for rotation, described in 3.4.1. This Kernel functions projects the features of the images into a new space so that the classification problem is linearly separable. We will introduce the distance matrix for the two descriptors described in chapter 3.

4.2.1. **Initial descriptor**

Fig. (4.4) shows the distance matrix for HOG, Appearance and Rotation. We use our RBF for rotation. The number of bins is 320. We have 8 classes in the dataset, Fig. (4.3). The descriptor used is described in Fig. (3.3).

![Figure 4.3: Dataset with 8 classes for initial descriptor](image)

4.2.2. **Proposed descriptor**

Fig. (4.6) shows the distance matrix for HOG, Appearance and Rotation. We use our RBF for rotation. The number of bins is 320. The dataset is illustrated in Fig. (4.5). The descriptor used is described in Fig. (3.4).

4.2.3. **Comments**

With these distance matrices we could value the descriptors by itself.

**HOG** We could see in Fig. (4.6a) and (4.4a) how forks are different from the rest of the images but similar within them, except the fork in horizontal position. Furthermore, fishes and snakes seem to be different to the rest but similar between them. This make sense as a fish and a snake look both as a stick.
CHAPTER 4. EXPERIMENTS

Figure 4.4: Distance matrix for initial descriptor and dataset 4.3
4.2. DISTANCE MATRIX

Figure 4.5: Dataset with 8 classes for the proposed descriptor

**Appearance** This is different from HOG distance matrix. Images with same object are very similar. This is illustrated by the white square around the main diagonal of Fig.(4.6b) and (4.4b). Contrary to HOG, forks are more similar to fishes, snakes and trees than to the others. This shows how appearance is only considered. Furthermore, if we look to the similarity within forks, we observe how they are all similar, no matter the orientation they have.

**Rotation** This is the most difficult case to get intuitions. We could observe in Fig.(4.6c) how vertical forks are similar to the vertical forks and not to the horizontal fork or slightly sloped fork(penultimate fork). For getting the rotation distance matrix in this case, we use the RBF that we propose in 3.4.

For HOG and Appearance we have the same result for the proposed descriptor than for initial descriptor. However, this is not the case for Rotation. We could observe how different Fig.(4.6c) and (4.4c) are. Fig.(4.6c) could get the similarities from the forks while Fig.(4.4c) does not. The rotation features from the proposed descriptor are better than from the initial descriptor.

Once we have compared the descriptors with the distance matrix, we will get a more specific quality measure of these descriptors by introducing them in classifiers. In this case, we will get results for different number of bins. This is a more complex system where its performance depends on several parameters: features, labeling, training set, testing set, iterations. We should take care of not overfitting our system. Our model should not describe the noise instead of the real data. The quality of our descriptor will be measured in terms of Error rate.
CHAPTER 4. EXPERIMENTS

(a) HOG

(b) Appearance

(c) Rotation

Figure 4.6: distance Matrix for proposed descriptor and dataset 4.5
4.2. DISTANCE MATRIX

![Graphs of fork, fish, snake, and tree shapes](4.7: HOG of similar shapes)

**Error Rate** The error rate showed in the graphics goes from 0 to 1 and is computed as:

\[ \text{ErrorRate} = 1 - \text{ClassificationRate} \quad (4.1) \]

where \( \text{ClassificationRate} \) is:

\[ \text{ClassificationRate} = \frac{tpr + tnr}{2} \quad (4.2) \]

true positive rate \( tpr = \frac{ntp}{n_{tp} + n_{fn}} \quad (4.3) \)

true negative rate \( tnr = \frac{ntn}{n_{tn} + n_{fp}} \quad (4.4) \)
CHAPTER 4. EXPERIMENTS

<table>
<thead>
<tr>
<th>Label</th>
<th>Predicted Class</th>
<th>Trues Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>true-positive(tp)</td>
<td>Positive</td>
<td>Positive</td>
</tr>
<tr>
<td>false-positive(fp)</td>
<td>Positive</td>
<td>Negative</td>
</tr>
<tr>
<td>true-negative(tn)</td>
<td>Negative</td>
<td>Negative</td>
</tr>
<tr>
<td>false-negative(fn)</td>
<td>Negative</td>
<td>Positive</td>
</tr>
</tbody>
</table>

We will try our descriptors with two classifiers: SVM and Knn. In Knn classifier we could see the evaluation for descriptors with $\alpha \in [0, 1]$, not only appearance and rotation descriptors. For values of $\alpha$ near to 1 our descriptor behaves similar to appearance descriptor and for values of $\alpha$ near to 0 it behaves as rotation descriptor.

Results with SVM classifier are obtained with the following dataset:

- dataset of black and white images.
- 8 different objects. 20 images of each object.
- Training set: 15 images of each object
- Testing set: 5 images of each object.
- **Appearance labeling** Target object(class 1): Star. The rest are class $-1$.
- **Rotation labeling** Target object(class 1): 135 degrees images, Fig.(3.14b). The rest are class $-1$.

Results with KNN with the following dataset:

- dataset of black and white images.
- 2 different objects(forks and butterflies). 20 images of each object.
- Training set: 10 images of each object
- Testing set: 10 images of each object.
- **Appearance labeling** Target object(class 1): Fork. The rest are class $-1$.
- **Rotation labeling** Target object(class 1): 135 degrees images, Fig.(3.14b). The rest are class $-1$.

The datasets used for both classifiers are different. Consequently, we can not compare results from both classifiers. We tried our descriptor with two classifiers and it outperforms independently of the classifier we use.

4.3. **Initial descriptor**

4.3.1. **Appearance labeling**

When labeling regarding appearance, we hope that appearance will have a lower error rate than HOG and even lower than rotation, as HOG is a halfway between
4.3. INITIAL DESCRIPTOR

appearance and rotation features. In this case, the appearance labeling will match with the prediction that the appearance descriptor does. The results for appearance labeling are shown in Fig.(4.8). We can see how we have different curves for different values of $\alpha$ in Knn classifier. The values for $\alpha$ correspond to the given distance formula (3.1). Appearance tends to zero error in the same way as $\alpha = 0.8$ and $\alpha = 0.6$. It appears to increase the error while $\alpha$ value decreases. Descriptors with $\alpha$ near to 1 behave as appearance while $\alpha$ near to 0 behave as rotation.

![Figure 4.8: Initial descriptor and appearance labeling](image)

Both for rotation descriptor and appearance descriptor, we observe peaks in different number of bins. However, HOG has not peaks. This is the main reason why we implement the proposed descriptor. Why these peaks appear? For answering this question, we go to the fundamentals of our descriptors, the histogram of gradients. These histograms for a same image and different number of bins are shown in Fig.(4.9). The histograms have the same distribution except for zero gradients in between non-zero gradients and the length of the histograms. If we look carefully at the results, Fig.(4.8), we observe a higher number of peaks in Fig.(4.8a) than in Fig.(4.8b). Why is this? One possible reason is the different range of bins in the plots for SVM and KNN. SVM goes from 30 to 240 and intervals of 30 while KNN goes from 80 to 320 and intervals of 80. We tried SVM classifier with different range of bins. Fig.(4.10) shows the performance of the descriptors for bins from 120 to 146 with intervals of one bin. We can observe a periodical behavior of the appearance descriptor and the rotational descriptor, with periods 16 and 8 respectively. Now we can explain why KNN in Fig.(4.8b) has lower number of peaks. The bins for the KNN classifier are 80, 160, 240, 320. All these numbers are multiples of 8 and 16. According to Fig.(4.10), appearance descriptor and rotation descriptor have the same value for multiples of 8 and 16 respectively. However, this periodicity does not come from the histograms of...
CHAPTER 4. EXPERIMENTS

(a) SVM

Figure 4.9: Histograms for same image and different number of bins

gradients as we observe in Fig.(4.10) how HOG is flat for any number of bins. Where does this periodicity come from? It should come from the Fourier transform.

Figure 4.10: Periodicity in appearance and rotation descriptors in the initial descriptor. Appearance labeling
4.4. PROPOSED DESCRIPTOR

4.3.2. Rotation labeling

In this case, we hope that the Rotation descriptor will have lower error rate than the HOG and even lower than using Appearance descriptor. The results for this initial descriptor are shown in Fig. (4.11). In this case, the rotation labeling will match with the prediction that the rotation descriptor does.

Figure 4.11: Initial descriptor and rotation labeling

4.4. Proposed Descriptor

4.4.1. Appearance labeling

If we ask the system to recognize forks within the following dataset Fig. (4.12), the output of the system is shown in Fig. (4.13). These results were obtained using the SVM classifier and 30 bins. The system gives good results as it returns the same forks as we ask for. However, there are three snakes that we did not expect. Snakes and forks are quite similar when referring to appearance. They both look like a stick and this explains why these snakes appear in the output.

As expected, the appearance descriptor has a lower error than HOG and rotation descriptor for appearance labeling. In Fig. (4.14) we can observe how these peaks disappear in the appearance descriptor. However, if we look to lower number of bins, Fig. (4.14), these peaks still appear even for HOG. The reason of these peaks is the instability of the histogram for a same image and low number of bins, shown in Fig. (4.16).

Why this instability of the histogram? There is a periodicity in the histograms which will have a more severe effect on the descriptor when using few bins. This could be explained with Fig. (4.17).

And why a constant value for higher number of bins? Expectations could be to have a lower error for higher number of bins. This is right if we have a more
challenging dataset. However, black and white dataset led to very easy histograms with few gradients as illustrated in Fig. (4.18).

4.4.2. Rotation labeling

If we ask the system to recognize 135 degrees objects, Fig.(3.14b), within the following dataset Fig.(4.19), the output of the system is shown in Fig.(4.20). This results are obtained using SVM classifier and 150 bins. The output is not what we expected because it gives objects that are not 135 degrees rotation. As expected for the initial descriptor, Fig.(4.21b) shows how rotation has lower error than HOG and even lower than Appearance. In this case, the rotation labeling will match with the prediction that the rotation descriptor does. However, we get different result in Fig.(4.21a). In this case, appearance is in the halfway between HOG and rotation.
4.5. GENERAL RESULTS

(a) SVM

Figure 4.14: Proposed descriptor and appearance labeling

(b) Knn

Figure 4.15: Proposed descriptor and appearance. Small number of bins

4.5. General results

Once we have analyzed different experiments, two questions arise:

- What connection has the results from the distance matrix and the classifiers?

As shown in the distance matrices, similarities with respect to appearance are better modeled with appearance features than with HOG or rotation. This gives a lower error rate when labeling the images concerning their appearance and higher error rate with rotation labeling, as the results of the classifiers show. We could also conclude from the distance matrix that the RBF Kernel used
for rotation features is very sensible. However, rotation labeling is also a weak point in classifiers whereas this is not shown in the distance matrix.

- Is our classifiers overfitted?
A good way to test if our classifiers are overfitted is to test our system with the own training set. As we can see in Fig.(4.22) the range of errors is quite big so the classifiers are not overfitted. This is the best result we could get from the classifiers as we are testing with the training set from which our classifiers have learned.

4.6. Summary

Finally, our robot and robot HOG met in Stockholm. But our robot was a bit impatient. It has been waiting for robot HOG for half a day. It seems that our robot took the right and straight way to Stockholm while robot HOG was lost in halfway and took some wrong roads in its way to Stockholm. Our factorized shape descriptor outperforms HOG descriptor. This is done by factorizing appearance and rotation of each image in such a way that classification could be easily done regarding these two characteristics.
Generally, concerning appearance labeling it seems that appearance descriptor has lower error than rotation labeling. It happens the contrary when we have rotation labeling. In all cases, HOG stays in the middle of both since it is a combination of both descriptors.
There are two main sensibilities when classifying: RBF Kernel and rotation labeling. RBF Kernel’s sensibility comes from the fact that the distance from the
4.6. SUMMARY

Figure 4.17: Periodicity of the histogram

nearest training point to the boundary is lower than for a linear kernel, as shown in Fig.(4.23). Actually, the maximization problem of the SVM involves maximizing this margin for a linear boundary.
Figure 4.18: Periodicity of the histogram for high number of bins
4.6. SUMMARY

Figure 4.19: Input of the system. Rotation labeling

Figure 4.20: Output of the system. Rotation labeling

Figure 4.21: Proposed descriptor and rotation

(a) SVM  
(b) Knn
CHAPTER 4. EXPERIMENTS

(a) SVM

(b) Knn

Figure 4.22: Proposed descriptor. Appearance labeling. Testing with training set

(a) Linear Kernel

(b) RBF Kernel

Figure 4.23: Different decision boundary
Chapter 5

Conclusions and Future Works

Our robot has arrived to Stockholm. It is not alone, it is with its robot friend HOG. Now that the trip is over, it is time to have a break. Both should consider their problems and the difficulties they encountered. Maybe robot HOG has some friends, robot SIFT, robot Shape Context that could help our robot to improve. Our robot stands for the factorized descriptor we propose in this project. The new descriptor gives an insight into the usage of the Fourier properties in feature extraction. It can be factorized into rotation or appearance. The descriptor could be summarize into 3 steps:

1. Compute HOG on an image
2. Compute Fourier transform of the HOG
3. Compute magnitude and phase of the Fourier transform

Once we have the results shown in chapter 4, we could suggest some improvements to the descriptor in terms of performance and quality. We start this chapter with some performance improvements, Section 5.1. Next, we propose some quality improvements in Section 5.2. To conclude we suggest other approaches similar to HOG descriptor, in Sect. 5.3.

5.1. Performance improvements

5.1.1. Database

It should be taken into account the different database this descriptor could be tested. In this case, it is considered black and white images, but more challenging datasets should be analyzed. These are the following (shown also in Fig. 5.2):

- Different objects How similar are 2 objects? When do they have the same shape?
CHAPTER 5. CONCLUSIONS AND FUTURE WORKS

- **Same object but different backgrounds** How can we focus on the object and not on the possible drawings of the background?

- **Same object but different poses** Should the different poses look the same even if it has different shapes?

![Different datasets](image)

Figure 5.1: Different datasets - (a) Different objects (b) Different backgrounds (c) Different poses

What results do we expect from these datasets? These are the distance matrix for the three different datasets in Fig.(5.2). Is it what we expect? What is it expected from $\alpha = 0.5$?

![Distance matrix](image)

Figure 5.2: Distance matrix of the challenging databases. The numbering of the images goes from left to right and from up to down: The image in the upper left corner is image 1 while the image in the down right corner is the last image. The numbering in the x-axes of the matrix goes from left to right and in the y-axes from up to down.
5.1. PERFORMANCE IMPROVEMENTS

5.1.2. Translation

Imagine we have two different images as Fig. (5.3).

![Translated images](image)

Figure 5.3: Translated images

The distance function of our descriptor is (3.1). However, translation could be also taken into account:

\[ \text{FinalDistance} = \alpha \text{DistAppearance} + \beta \text{DistRotation} + \gamma \text{DistTranslation} \quad (5.1) \]

Our descriptor is one cell mode. It is invariant to translation due to the descriptor will have the same gradients, even if the image is translated. However, dividing the image into different cells allows a local study of the image. If we look globally 2 objects, they could be totally different. However, by looking locally, more similarities could be found and a better recognition could be encountered. Imagine an image with the text "Hello" and another image with "Hurricane". Globally they are very different but if we look letter by letter, i.e., locally, both could be similar. They both have the letter H.

For more than one cell, DistTranslation is more difficult to compute. This is an idea of how it could be calculated. In this case, each cell has its own histogram. Then, we should compare, by a linear kernel, each histogram (each cell) with all the histograms (coming from the other image to compare) of the cells that are neighbors of that cell. We should keep track of the neighbor of each histogram that gives higher similarity. If this is done for all the cells and we keep track of the nearest neighbor, a direction should be majority for all of them. This will be the direction of the translation.

5.1.3. Higher number of cells

Our descriptor divides the image only in one cell and take the gradients of that cell. However, the image could be divided into different cells, as shown in Fig.(}
and could take the gradients of each of these cells. This allows a local study of the image and could solve scaling factors or segmentation of the image for comparing different objects. There are more steps in the usual HOG procedure that they could be also be implemented as the normalization of gamma and the normalization of contrast within the overlapping blocks of cells.

5.1.4. Peaks in the rotation

With our proposed descriptor we improve the results from the initial descriptor only for appearance. We still have some peaks for the rotation descriptor (red line in Fig. (4.14) and Fig. (4.21)). Why these peaks appear? Looking an answer to this rotation peaks, we faced another issue: the periodicity in the appearance and rotation descriptors, shown in Fig. (4.10). Why do they have this periodicity?

5.2. Quality improvements

Image quality is not optimal. Most of the images used in this project have noise and not plane background. Some questions arises regarding the pictures used:

- What happens if we have a very difficult background? Will the descriptors focus on the background instead on the image? As an example, we can consider the upp-right picture in Fig. (5.1b). In this case, we can use the cascade classifier proposed by Viola and Jones [11] in order to detect the goal object beforehand (the hand). Otherwise, a sliding window could be used to get the location of the hand.
- What happens if we have a rotated background instead of a rotated hand? Fig. (5.4)
5.2. QUALITY IMPROVEMENTS

- What happens if we change the colors of the hands? Is it the appearance the same? Is it the rotation correct? Should the color affect the gradients of the image?
- What happens if we encounter two different images (a fork and a snake, Fig. (5.5)) with the same shape?

![Figure 5.5: Rotated Background](image)

- We have done labeling for appearance and rotation. However, how could be the labeling for the different $\alpha$ not being 0 or 1?

Due to the image compression, -90, 0, 90 and 180 gradients are taken out from the histogram. However, these gradients become really important for comparing objects such as hands in vertical and horizontal pose. Where this problem comes from? Fig. (5.6) shows the image and a map of the gradients:

![Figure 5.6: Gradients map](image)

The legend is as follows:

- Gray = zero gradient (background has magnitude=0 so not in the histogram)
- Black = gradient different from -90, 0, 90, 180
CHAPTER 5. CONCLUSIONS AND FUTURE WORKS

- White=180 gradient.
- Gray different than the background= gradients -90,90

As we could observe in Fig. (5.6), all around the edges of the hand we have gray dots. This means that we have -90, 0 and 90 gradients. However, we also have gray dots in the inner part of the hand. This dots are due to the compression artifacts. The study of a single line could give us some more clues. Fig. (5.7) illustrates the different gradient map for a manual line and Fig. (5.8) illustrates a line coming from an image.

![Image](a) Image  ![Gradient mapping](b) Gradient mapping

Figure 5.7: Image Line

These two lines show how it should be the gradient image (Fig(5.8)) and how it looks like(Fig.(5.7)). This fits with the compression artifacts shown in Fig.(3.8)

![Image](a) Image  ![Gradient mapping](b) Gradient mapping

Figure 5.8: Manual line
5.3. OTHER APPROACH

5.3. Other approach

The same descriptor could be developed from a different descriptor than HOG. SIFT, Shape context or even SURF could be an alternative.
Appendix A

Database

- Black and white images
  These dataset is shown in Fig.(A.1) and(A.2).

  Figure A.1: 20 images of forks with different rotation

- Hand pose database
  These dataset is shown in Fig.(A.3).
Figure A.2: 20 images of butterflies with different rotation
Figure A.3: 20 images of rotated hands
Appendix B

Case Studies

These are the different datasets that we use in this project.

- SVM.
  Training dataset is shown in Fig. (B.1) and the testing dataset is shown in Fig. (B.2).

- KNN. 2 different objects
  Training dataset is shown in Fig. (B.3) and the testing dataset is shown in Fig. (B.4).

For target for appearance is the star. For rotation labeling is the 135 degrees images.
For target for appearance is the fork. For rotation labeling is the 135 degrees images.
Figure B.4: Testing dataset for KNN with 2 classes
Bibliography


