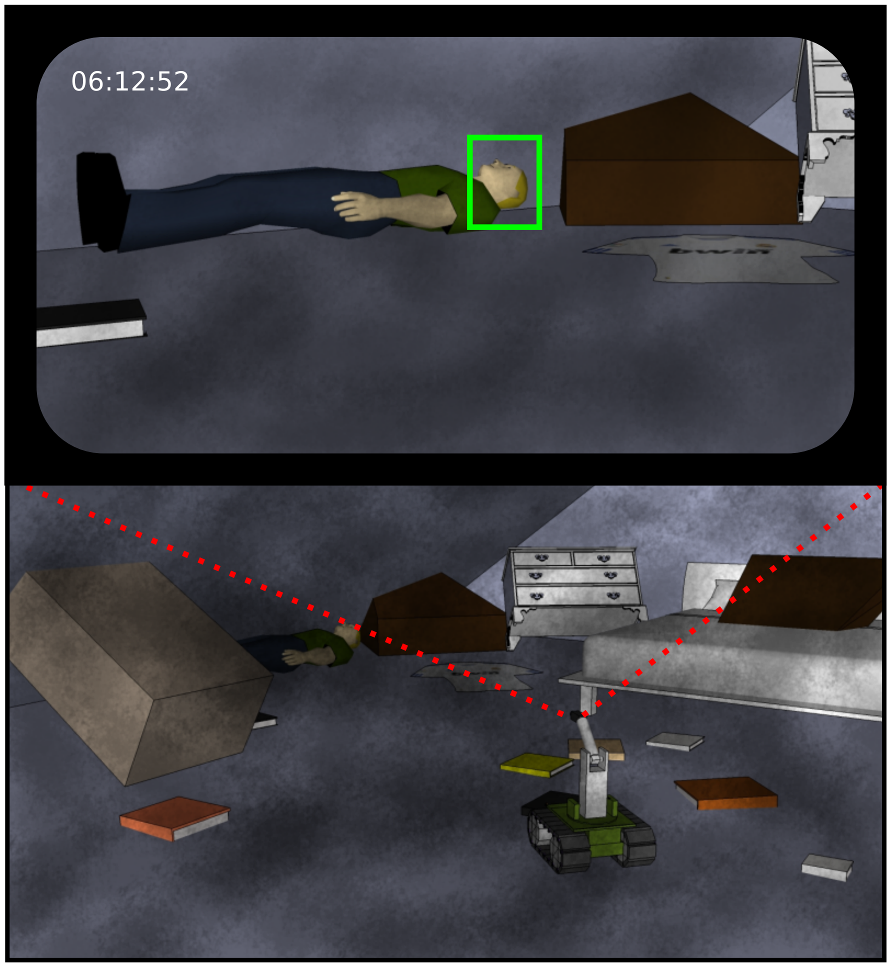
Visual Detection of Humans in a Disaster Scenario



Vision, Graphics & Interactive Systems

Long Master's Thesis
Fall Semester 2012 - Spring Semester 2013
Niels Gerlif Myrtue

Department of Electronic Systems
Aalborg University

AALBORG UNIVERSITY
Master's Thesis

Department of Electronic Systems
Frederik Bajers Vej 7
DK-9220 Aalborg Ø
http://es.aau.dk

Title:

Visual Detection of Humans in a Disaster Scenario

Theme:
Computer Vision

Project Period:
Fall Semester 2012 - Spring Semester 2013

Project Group:
1043

Participants:
Niels Gerlif Myrtue

Supervisors:
Thomas B. Moeslund
Arcot Sowmya

Copies: 3

Page Count: 73

Date of Completion:
June 6, 2013

Abstract:

This thesis describes how a system for detecting victims in a disaster scenario can be implemented in software using known methods from computer vision. The goal has been to create a system that can process video input from the camera on a robot. The first chapter provides background on disaster management worldwide and explores the cases where robots have been deployed for rescue work after a disaster has struck. To this day, robots have not played a central role in search and rescue operations after any large disaster.
The purpose of the human detection system is to work as a visual aid to a human robot operator. This can hopefully increase the chance of locating victims if the operator is burdened by mental fatigue as has been the case during robot operations.
The final detection system consists of 3 detection modules that all search for the human head or face in images. The first detector uses the HOG descriptor and an SVM classifier to determine if a human head is present within a smaller region of the image. The second detector uses the Viola-Jones object detection framework to detect human faces. The third detector uses template matching to search for the head using quarter circle shapes.
The combined detection system achieved a detection rate of up to 62 % on a varied set of image with humans. On a simulated disaster image set the detector has a detection rate of only 14 %.

AALBORG UNIVERSITET
Kandidatspeciale

Institut for Elektroniske Systemer
Frederik Bajers Vej 7
DK-9220 Aalborg Ø
http://es.aau.dk

Titel:

Visuel Detektion af Mennesker i et Katastrofescenarie

Tema:
Computer Vision

Projektperiode:
Efterårssemestret 2012 - Forårssemestret 2013

Project Group:
1043

Participants:
Niels Gerlif Myrtue

Vejledere:
Thomas B. Moeslund
Arcot Sowmya

Kopier: 3

Sidetal: 73

Afleveringsdato:
June 6, 2013

Abstract:

Dette speciale beskriver et system til at detektere tilskadekomne i et katastrofescenarie, som kan implementeres i software ved anvendelse af velkendte computer vision metoder. Målet har været at skabe et system som kan behandle video input fra kameraet på en robot. Det første kapitel leverer baggrund for katastrofehåndtering på verdensplan og undersøger de situationer hvor robotter har været anvendt i redningsarbejdet efter en katastrofe er indtruffet. Til dags dato har robotter endnu ikke spillet nogen central rolle i søgning- og redningsaktioner efter en større katastrofe. Formålet med detektionssystemet er at det skal kunne anvendes som et visuelt hjælpemiddel for en person som styrer robotten. Dette kan forhåbentlig forøge chancerne for at finde tilskadekomne i situationer hvor operatøren er belastet mentalt, hvilket har været tilfældet under aktioner med robotter.
Det endelige detektionssystem består af 3 detektionsmoduler som alle søger efter det menneskelige hoved eller ansigt i billeder. Den første detektor anvender HOG descriptor'en og en SVM classifier til at bestemme om en mindre billedregion indeholder et hoved. Den anden detektor anvender Viola-Jones object detection framework til at finde ansigter i billedet. Den tredje detektor anvender template matching til at søge efter hovedet ved anvendelse af kvartcirkelformer. Kombinationen af detektorer opnåede en detektionsrate på 62 % med et varieret billedsæt med mennesker. På et simuleret katastrofesæt opnåede detektorerne kun en detektionrate på 14 %.

Preface

This masters thesis has been produced by Niels Gerlif Myrtue, a student at the School of Information and Communication Technology (SICT) at Aalborg University. The thesis is the result of project work done at UNSW, Sydney, Australia as a part of the Practicum Exchange Program of the University. The project was proposed by professor Arcot Sowmya from the School of Computer Science and Engineering at UNSW who also served as the main supervisor during the project. Professor Thomas B. Moeslund from the department of Architecture, Design and Media Technology at AAU has served as secondary supervisor. The project work was done in the period from 15/11-2012 and until 06/06-2013. The majority of work has been done in Sydney, Australia.

The thesis consists of 4 chapters and an appendix. Appendix chapters are enumerated alphabetically. A DVD accompanies this thesis that contains the source code of software developed for the project, raw test results and a PDF version along with the source files for the thesis. Throughout the report, when material on the DVD is referenced, the path is indicated as a footnote such as this[1]. All literature used in the thesis is referenced numerically and listed at the end of the main thesis on page 73. An example of a reference could be [1]. The PDF version of this thesis allows the reader to click on any reference or citation to jump directly to it.

Niels Gerlif Myrtue

[1] \path\to\file

Contents

Chapter 1

Introduction

Large scale disasters, both natural and man-made, are an unfortunate reality for many people across the world. The situation in the wake of a disaster is often chaotic and disaster victims who have become trapped or otherwise incapacitated by the events surrounding them are for the most part fully reliant on the efforts of rescue workers. It is therefore vital to the lives of victims that the search and rescue operation happens as quickly and efficiently as possible. Advanced technology has so far only played a minor role when rescue workers attempt to locate and evacuate victims who have been trapped in damaged or collapsed buildings such as those shown in Figure 1.1.

Robots are believed to play a key role in rescue work in the future and have already been employed in certain cases to aid rescue workers. This thesis looks at these cases and discusses how the technology could be advanced further. One notable feature that has been lacking in robotic systems that have been used in disaster situations is the ability to analyze or visually enhance the video feed sent to the operator. Several advances have been made in computer processing capabilities and vision techniques that makes it viable to implement such a system that can aid the operator in locating human victims.

The purpose of this thesis is to show how a visual human detection system that uses images from a normal camera can be implemented in software. Furthermore it is explored how such a system might fit into the larger context of disaster management and more specifically how it can benefit search and rescue operations.

Chapter 2 provides the reader with background on the history and current state of emergency management. The United States and Japan are used as examples due to the high frequency of natural disasters that these countries experience. The chapter discusses the current technology, including disaster robots, that is employed and looks at methods for automatic detection of humans.

Chapter 3 explains the context of the system that has been implemented for this

Figure 1.1: Examples of disaster scenarios. (Left) Collapsed building after the Great Hanshin earthquake in 1995 [1]. (Right) debris after the attacks on the World Trade Center in 2001.

thesis and describes the different modules that form the human detector. Each module section also include results that were obtained from initial testing sessions.

Chapter 4 presents the final results that were obtained when the different detection modules were combined into one single human detector.

Appendix A provides the theoretical background of all methods that have been used to implement the detection system.

Chapter 2

Problem Analysis

This chapter contains an analysis of the problem being addressed in this thesis: locating disaster victims in a disaster scenario. The first section gives a short background on emergency management to put the topic into a historical context. The following sections look at the technology that is currently being used in relation with disasters. This includes the use of robots in urban search and rescue operations. The current state of robot technology is then discussed in light of this analysis.

What follows next is a description of the challenges that are faced when attempting to detect human victims in a disaster situation when using computer vision techniques. This section is followed by a literature survey that provides an overview of the methods that have been used to solve this problem.

2.1 Background of Emergency Management

Natural disasters have plagued life on earth for as long as life has existed. Since the beginning of recorded history, human writings speak of natural disasters in many forms, be it storms, earthquakes, wildfires or volcano eruptions. Human conflict has also been a major source of destruction, the power of which has only increased rapidly over the last two centuries. Focused war efforts have regretfully proven to be just as effective at destroying cities as any natural disaster, with the possible exception of a meteor strike. Acts of terrorism have also created several disaster scenarios, especially in recent years with the World Trade Center attacks being one grievous example.

Historically, emergency management wasn't seen in any structured form until relatively recent times.[2] Urbanisation is a likely cause of this, as the fast growth in population of larger cities also increased the need for more organised prevention and response to emergencies. An example of the former was seen after the Great Fire of London in 1666. In the aftermath, a law named *the first Rebuilding Act* was passed, and it stipulated the types of houses that were allowed to be built. One of the purposes of the law was to decrease the risk of another fire disaster.[3] In the United States, after World War II, emergency management began to move from the individual state level, and up to the federal level. This led to the establishment of the National Emergency Management Association in 1976, comprising representatives from every state, and to the Federal Emergency Management Agency (FEMA) in 1979. The purpose of FEMA was to support the states in times of emergency.

Another country that has been prone to natural disasters is Japan. The island lies near the borders of several continental plates and is situated in the volcanic zone *Pacific Ring of Fire.* As a consequence, earthquakes are frequent and sometimes extremely violent (Great Kantō, 1923; Great Hanshin, 1995 and Tōhoku, 2011 are a few examples of major earthquakes with high death tolls). Japan has also got one of the highest population densities in the world. As a result, Japan has developed some of the best (if not *the* best) emergency management structures in the world to deal with disaster prediction, prevention and response. To encourage and maintain public consciousness, the 1st of September was declared *Disaster Management Day* in memory of the Great Kantō earthquake and every year this day is traditionally used to educate the general populace about disasters,[4]. A detailed overview of the historic developments can be found in[5]. It's notable that the yearly budget of national emergency management in Japan, which was 1.2 trillion yen (∼150 billion USD) in 2010 is more than 10 times larger than that of FEMA for 2013 (∼14 billion USD) [5] [4], although the latter does not cover all the costs of damages which can be very high some years (estimated ∼130 billion USD as a result of Hurricane Katrina in 2005[6]).

The frequent nature of natural disasters in Japan means that research into this area receives a lot of attention and many resources from government and private corporations. Some contributions to this field thus originate from Japan. The following section will review the current state of technology used in emergency management.

2.2 Technology in Emergency Management

One very important aspect of emergency management is the ability to predict if and when a disaster will occur. Japan Meteorological Agency (JMA) is responsible for providing forecasts on a variety of natural phenomena including weather conditions and earthquake activity. This task is very complex and incorporates a multitude of data sources along with powerful computer processing facilities [7]. To communicate this knowledge effectively, an elaborate system exists that links JMA directly to other relevant government agencies, as well as meteorological agencies in neighboring countries. In America a similar agency exists, with the purpose of forecasting hurricane activity: The National Hurricane Center [8].

When a natural or manmade disaster occurs, a swift and organised response effort is vital to save the lives of victims. Evacuation needs to occur as quickly as possible and ideally before the disaster strikes. Major earthquakes, hurricanes and floods tend to leave behind them a vast area of damaged or collapsed buildings that may contain living victims in need of rescue. Urban Search And Rescue efforts, or more commonly just USAR, is therefore the next step after evacuation. The specifics of such an operation is entirely dependent on the nature of the disaster. However, general guidelines produced by the International Search and Rescue Advisory Group (INSARAG) do exist and are quite extensive [9].

During USAR operations, technology is used in many forms to aid the rescue workers. The INSARAG guideline lists the following modes of communication: satellite phone, VHF / UHF radio, internet access and cellular phone. While these channels are ubiquitous in today's society, their use in emergency situations have proven troublesome during larger disasters in the United States [10]. An example of a system that tries to mitigate the problems with traditional communication is the Danish SINE network that was rolled out nation-wide in 2010. The system allows all relevant emergency services to communicate at every level and runs separately from other communication channels [11].

The primary tools used by rescue workers are for excavation and tend to be low-tech in nature. Locating victims is largely a matter of performing exhaustive search of the premises, relying on the senses and experience of the human workers who are sometimes assisted by dogs. Thermal cameras can be used to detect body heat signatures of living victims. However, since a disaster site often has many non-human heat sources, their usefulness can be limited. Robots are believed to have great potential in helping rescue workers locate and help entrapped victims. Robots have so far not been widely used in real-life disaster situations but valuable information has been gained in the few cases where they have been deployed. The following section will explore these cases and the current state of disaster robots in general.

2.3 Disaster Robots During the World Trade Center Incident

From a practical point of view, robots have not yet shifted any considerable ground in emergency handling. Their use in real disaster situations has been limited, and in the cases where they were deployed, results have been mixed. The first real use of robots was during the response effort of the World Trade Center attacks on September 11th, 2001. Data collected from the incident has been analysed in detail [12]. Three different robots were deployed quickly after the attacks took place: Inuktun Microtracks, Inktun MicroVGTV and Foster-Miller Solem (seen in Figures 2.1, 2.2 and 2.3). The robots were delivered by the Center for Robotics Assisted Search and Rescue (CRASAR). Other robots were available, but for various reasons, these were not brought into the disaster site. None of the deployed robots was autonomous and therefore they required a human operator to control their movements. The robots had only a limited number of mounted sensors, including video camera for control and reconnaissance, speakers and microphone for 2-way audio communication and compass for determining orientation.

From the analysis [12], several findings are reported. Some of them are listed here:

- Lack of sleep caused a major degradation in the performance of the robot operators
- The deployed robots lacked proper sensor capabilities to help in maintaining a sense of orientation
- The robots had no image processing capabilities
- The robots relied entirely on the operator to distribute gathered information to other parties
- The robots had no awareness of their present state, the state of the surrounding environment or previous actions and events.
- Only the visual information feed was used by operators.

Other findings were related to the size and build quality of the robots, the training necessary to operate them effectively and structural organization needed to take full advantage of the robots at any time. Another interesting point was the unwillingness of the crew to accept and appreciate the importance of robots as tools. In the end, the crew chose robots that most closely resembled tools they were already familiar with, which also came down to simplicity. The authors note that to be effective, the interface of a robot should be as easy to work with as possible for an operator with only limited training.

Many of the problems noted appear to stem from the fact that the robots used were effectively just mobile cameras that had no autonomous functionality whatsoever. The fact that the search effort had to rely entirely on the vigilance of the human operator meant that mental fatigue caused by sleep deprivation and stress had a high impact on the outcome of the search.

2.4 Disaster Robots in Japan

In 2002 the NPO International Rescue System Institute was established in Japan. The institute is described as "*the industry-government-academia-civilian research organization to advance and diffuse high-technologies coping with disaster*" [14]. One of their research projects is directly related to disaster robotics. Their specific areas of operation are: Aero robots, On-Rubble robots and In-Rubble robots [15]. Two of their robots are shown in Figure 2.4. All robots are controlled wirelessly by a human operator and to make navigation easier, the research team has developed a method where a 3D model of the robot is superimposed on previously captured images from the mounted camera. This gives the operator a way of visualizing

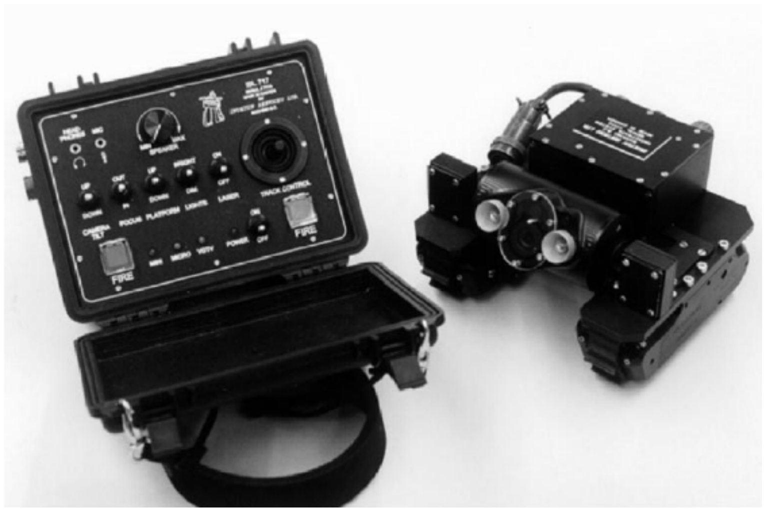


Figure 2.1: The Inuktun Microtracs robot [12]

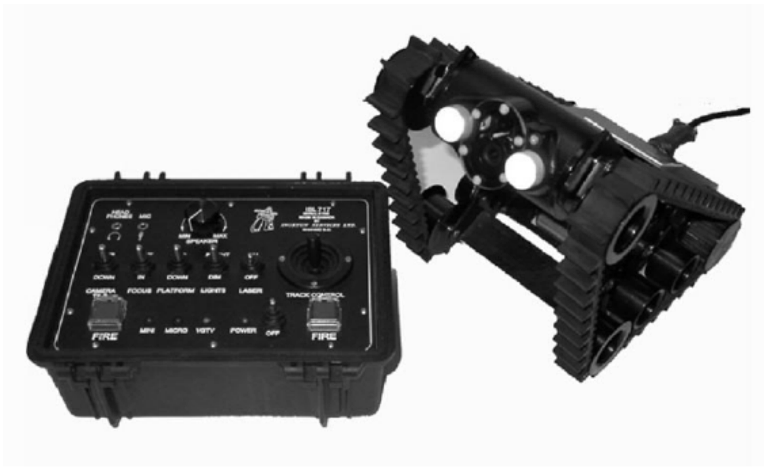


Figure 2.2: The Inuktun MicroVGTV robot [12]

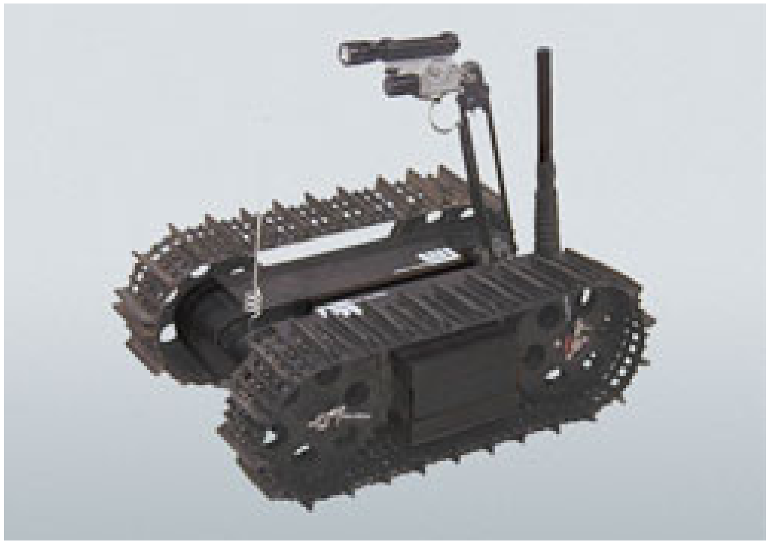


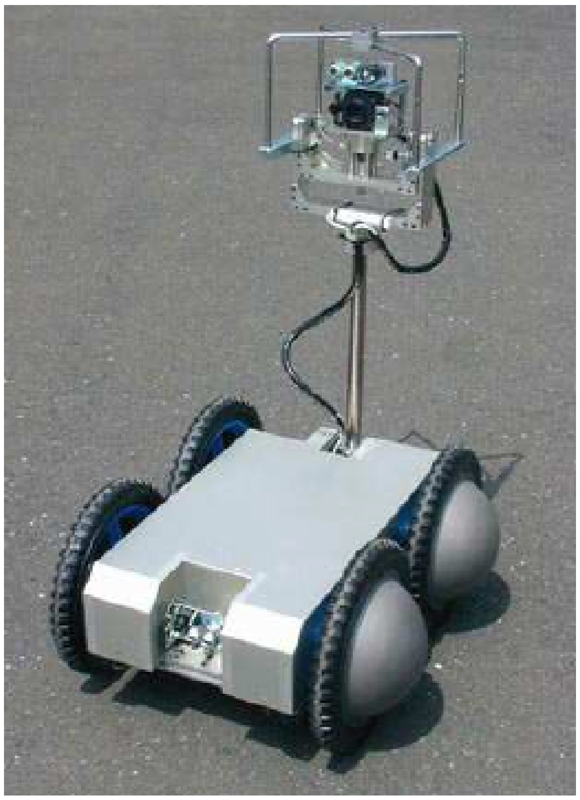
Figure 2.3: The Foster-Miller Solem robot [13]

the robot's position and orientation in the environment without the need of a full 3D model. In terms of autonomy, the authors [15] note that developing a fully autonomous robot is still too difficult and that a human controller is the most realistic solution.

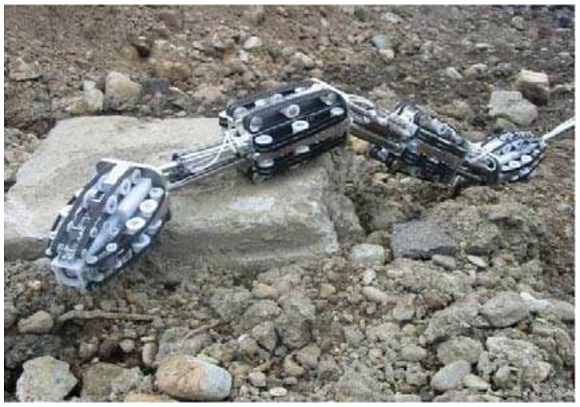
The recent Tōhoku earthquake and the resulting tsunami provided an opportunity, however unfortunate, to test out the robots in real life situations. Despite this, robots did not play a big role in the ensuing response effort [16]. At the disaster-struck Fukushima nuclear power plant, several robots were reportedly deployed [17] and they were mainly used to explore areas where the radiation levels were too high for humans [18].

2.5 Assessment of Current Disaster Robot Technology

Based on the previous analysis it is clear that disaster robot technology is a field that is being researched extensively and is showing a lot of promise. The performance of robots in recent disaster scenarios however has highlighted their shortcomings in USAR operations. A big problem is the logistics, as first responders often do not have the time or manpower to carry heavy robot equipment with them to the disaster site. Rescue workers favor small and light robots over larger ones [12]. In terms of technology, much of the current research effort is spent on making robots as mobile as possible. Autonomous operation appears to have low priority in projects that have ties to real life disaster situations.



(a) FUMA robot [15]



(b) Moira. A snakelike robot [15]

Figure 2.4: Examples of robots developed in the IRS project

Despite this, many of the problems reported may be mitigated by making the robots more autonomous. For instance, making the robots spatially aware of their surroundings would help the operator and perhaps reduce the number of times that the robot gets stuck. Adding image processing and computer vision capabilities could also reduce the psychological strain on the human operator. The ideal robot would be able explore an area entirely by itself and human intervention would only be needed if a victim was discovered.

Disaster robot research is a field where testing out solutions in real life situations is very difficult if not impossible for the most part. As a result, most authors are forced to test their methods on staged data sets that only resemble real situations to a certain degree. The RoboCup initiative, which was originally about designing robots that could play soccer, now also has a project called RoboCup Rescue [19]. Two leagues exist: Rescue Robot League, which focuses on building durable robot platforms that meet the demands of a disaster site and Rescue Simulation League which is focuses on planning and information sharing among multiple robots taking part in a rescue operation [20].

A notable outcome of the RoboCup Rescue initiative has been the development of the USARsim software framework [21]. This framework allows robot designers to test some of their methods in a virtual environment that attempts to simulate a real disaster scenario. As noted [12], the simulation environment is not able to reproduce all elements of a USAR situation. Still, given that the simulation is accurate in terms of visual representation and physical behavior of objects in the world, it is very conceivable that the framework can be used to test and validate methods in relation to image processing and robot movement.

2.6 Challenges in Disaster Victim Detection

A potential disaster scenario can be anything from a large open area to just a small confined space. This presents a huge challenge to the construction and maneuverability of the robot itself and also to any visual victim detection system on board. Assuming that the robot is not inhibited by debris and can move around freely, the vision system would still face a number of challenges that make victim detection difficult.

In general human detection, it is often possible to make simplifying assumptions about the appearance of a person which in turn makes it possible to reduce the complexity of the detection system . Such assumptions could be that the person is standing upright, is facing the camera and that the body is not covered by other objects in the image. In a disaster scenario, it's difficult to make any such general assumptions. It could be argued that a victim will likely be sitting or lying down, but this cannot be assumed. To reliably detect victims in a disaster scenario the following circumstances must be accounted for:

(i) the victim could have virtually any body pose

(ii) the victim may not be fully visible in the image due to debris or irregular lighting

(iii) the victim may be missing one or more limbs

Besides the state of the victim, other factors can affect the search effort such as unpredictable camera movements, a dusty atmosphere and possibly fires in the vicinity of the robot. Obviously, if the conditions are bad enough, it will not be possible to detect anything, regardless of how good the computer vision method is. A reasonable minimum working assumption is therefore that a victim in the image is at worst still detectable through human inspection.

The technical challenges involved in actual victim detection are just one side of the coin. Training, testing and validating a proposed detection system properly requires video footage from a real or simulated disaster scenario. Real footage from rescue operations probably exists, but is not publicly available. A simulated scenario in some form is therefore the only viable option in practice and since it is virtually impossible to simulate all aspects of a disaster scenario, compromises are unavoidable.

Despite the difficulties involved in testing and validation, much work is still being done to develop methods to identify victims in a disaster situation. Some of the methods that have been proposed to deal with victim detection are discussed next.

2.7 Disaster Victim Detection Methods

Many papers have been published, that deal specifically with the problem of detecting victims in a disaster scenario. However, compared to the more general topic of human detection, the disaster scenario can only be considered a small niche and it appears that a large portion of the research is related to the Robocup initiative. Generally it also appears that effort is usually divided over all aspects of an autonomous robot system, meaning that the computer vision tasks involved do not always receive full attention. Instead, authors tend to focus on robot control and mapping of the disaster environment autonomously [22, 23].

Besides a regular camera, many proposed systems use a variety of sensors to detect humans. This includes sound and distance [24] and most commonly a thermal camera [25, 24, 22, 26].

The environment in a disaster scenario presents a number of challenges to any CV algorithm. Despite the obstacles, a number of well known computer vision methods have been used for detecting disaster victims. One example is the use of HOG descriptors as a feature to identify a victim in an image [27, 25, 28]. Several different classifiers have been used such as SVM, K-nearest neighbors and AdaBoost. Neural networks have also been used in a similar fashion [29, 30], for instance, where a classifier is trained to recognize regions in an image that contain certain body parts or the entire body of a human. Another approach has been to search for the contour of the human body [31, 32, 24] or to search for skin colored pixels in the image [32, 24].

Results vary and in general it is very difficult to compare performance due to the differences in equipment, setup and test scenarios. It is often unclear how well a given method generalizes beyond the data sets used for testing. Most methods that have been proposed are general in nature and therefore not limited to the disaster scenario. What makes the disaster scenario special is that it's not generally possible to make assumptions about how humans will appear in the image. The following section provides a short survey of human detection methods described in the literature. This list includes the methods that address the disaster victims detection problem.

2.8 Human Detection Methods

As mentioned in previous sections, the problem of detecting victims in a disaster situation is just a small niche of the more general and larger field of autonomous detection of humans in images and video. This section will review some of the methods that have successfully been used to detect humans in images. The review includes the methods listed in the previous section. Beyond a description of all the methods is a short discussion on their applicability in the disaster scenario. Only methods that apply to single visible-light cameras are considered.

It is very difficult to create a classifier that can detect the human body in all situations. This is due to the huge amount of variation possible due to lighting, clothing, hair style, articulation of limbs and parts obscured by other objects. This fact is reflected in the literature since most approaches either involve detecting certain parts of the human body or limiting the search to specific poses only. A very common technique is to use a combination of different classifiers. The following sections describe some of the common methods that have been used to detect humans in images and video. The list includes the methods mentioned in the previous section that have also been used specifically for detecting disaster victims. At the end of each description is a short discussion of the applicability of the method in the disaster scenario.

2.8.1 Motion Based Segmentation

Motion based segmentation is a very simple way to separate moving objects from the background in a video sequence. In its most basic form it involves calculating the arithmetic difference between every pixel value in two consecutive frames. If pixels in one part of the frame change quickly it can indicate that an object has moved. The region can then be analyzed further. This approach has been used [33, 34, 35].

Simple motion based segmentation works well in situations where the camera is stationary for an extended period of time and the target is moving. However, in

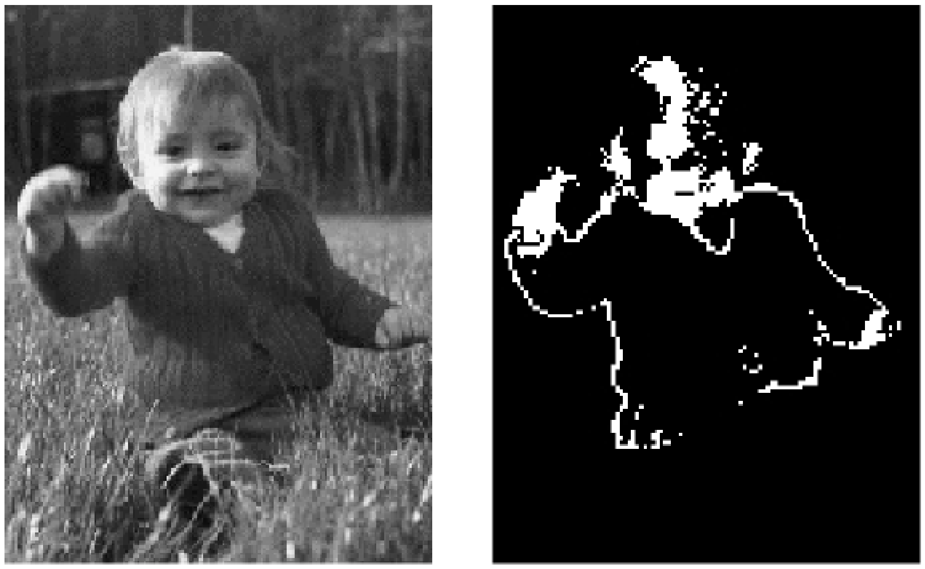


Figure 2.5: Example of skin color based segmentation [34]

a disaster scenario these conditions are not fulfilled since the robot carrying the camera is likely to be moving while the victim is lying still. A different way to exploit the motion however is to calculate optical flow between different frames [36]. This approach exploits the parallax effect whereby an object closer to the observer appears to move faster than one further away.

2.8.2 Skin Color Based Segmentation

One characteristic feature of the human body is the appearance of the skin. Many detection systems have been proposed that use this as a way to determine regions of interest in the image [35, 34, 37]. An example of an image that has been segmented by use of skin color can be seen in Figure 2.5. This method is useful since the skin on the head and face is often visible on humans in images. Modeling skin color is also fairly simple and usually involves training a classifier using images with skin parts manually labeled. The model can be parametric, e.g. a multivariate, Gaussian distribution or non-parametric, where the trained colors are placed in a histogram and perhaps sorted into bins. The histogram is then normalized to yield a probability of a given pixel being skin.

The biggest problem with skin color based detection is that human skin can appear in a very wide range of colors in an uncontrolled environment such as a disaster scenario. The method can, however, be used as a supplement to other methods for detecting parts of the body.

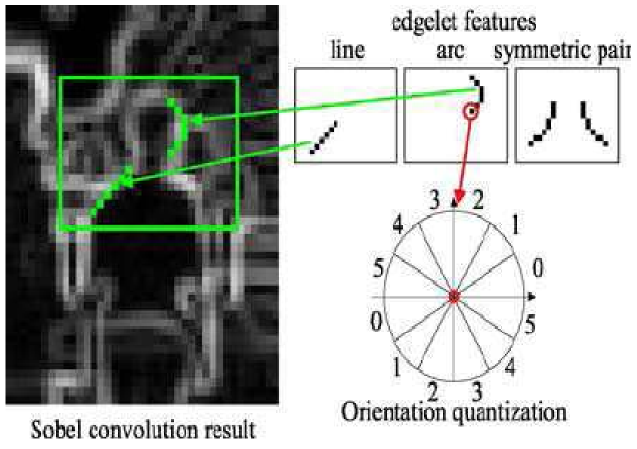


Figure 2.6: Example of an edgelet [38]

2.8.3 Shape or Contour Based Detection

As mentioned earlier, the shape of the human body can vary enough to make it virtually impossible to detect in all situations. Much research has been done where only parts of the body are being detected. Popular parts to search for are the head and shoulders [38, 39, 40] since these do not exhibit much variation even when the rest of the body is highly articulated. Another approach is to use a hierarchy of different shapes that match different parts of the body [41]. The case described in the paper is simplified somewhat since people in the images are standing upright some distance away from the camera.

There are many different ways to analyze the shapes or contours that show up in an image. A common characteristic seen in many of the shape based detection methods is that the input image is preprocessed to obtain an edge map that is then searched for a particular shape. Template matching has been used for human detection [42, 41] and it involves specifying a simple binary pattern of edge pixels to search for in the edge map. The algorithm then scans the map for occurrences of this pattern. A slightly different approach has been used [38] where the notion of an *edgelet* is introduced which is similar to a regular template but contains orientation information for each pixel as well. An example is shown in Figure 2.6. The method involves using multiple edgelets as weak classifiers that when put together form a stronger boosted classifier.

Another interesting approach that involves shape [40] use a hierarchical structure for decomposing the head-shoulder contour into smaller segments. Each level in the hierarchy is subdivided into 3 smaller segments and the bottom level contains primitives calculated from the image. The authors introduce a primitive feature called Oriented Integration of Gradients (OIG). Matching two shapes is done using

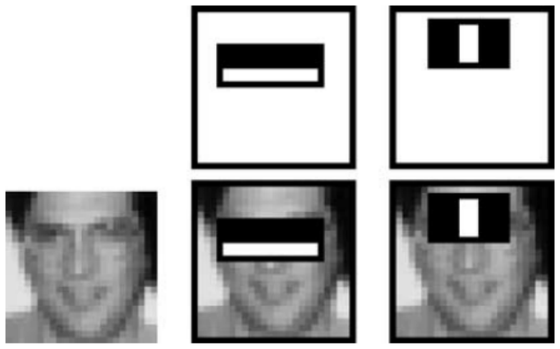


Figure 2.7: Example of Haar-like features used in face detection [45]

a Hough voting system.

The HOG descriptor [43] has also been widely used for detecting humans in images [44, 27, 25, 38]. The HOG descriptor is composed of normalized gradients obtained from the image and is normally stored in 8x8 cells. The cells are also sorted into overlapping blocks. The use of normalized gradients makes the descriptor very robust against changes in light and color.

In relation to victim detection, a shape based approach seems very viable. By searching for different parts of the human body separately it is possible to account for cases where certain parts are occluded in the image. Also, as long as the shape can be discerned in the image, it does not change based on external factors such as the surrounding conditions and image quality.

2.8.4 Viola-Jones Object Detection Framework

The Viola-Jones object detection framework [45] makes use of Haar-like features to classify objects. Features are defined by small regions in a detection window such as shown in Figure 2.7. The sum of all pixel values in the black and white regions are subtracted and the difference indicates whether there is an overall difference in intensity. It turns out that this approach is very effective at detecting human faces. The Viola-Jones framework sets up a cascade of classifiers that each use one Haar-like feature. By also making use of integral images the detection is very fast. The classifiers are trained using AdaBoost. The framework is very widely used for detecting faces [46, 47].

The Viola-Jones framework could work well in detecting victims in a disaster situation. While there is no guarantee that a victim will have a visible face, the detector could at least work as a supplement to other detectors.

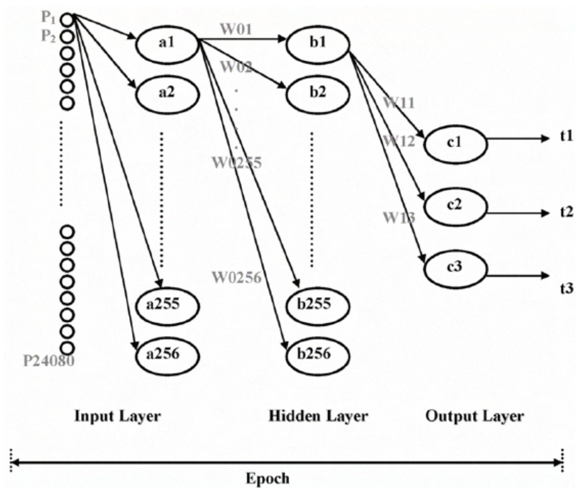


Figure 2.8: Example of an artificial neural network used in[30]

2.8.5 Artifical Neural Network Based Detection

Artificial Neural Networks (ANN) are used in a wide range of fields due to their versatility. Applications include distribution estimation, function regression and also human detection [29, 34, 30, 48]. ANNs consist of a possibly large number of basis functions that have weighted connections to each other. Often the network is divided into 3 layers: input, hidden and output layers. The network is trained by adjusting the weights in order to obtain a desired output from a given input. In the case where the input is an image, each pixel can feed into a separate input node, or as shown in Figure 2.8 into all nodes in the input layer. In one example a neural network has been trained to detect human hands, feet or entire bodies lying on the ground [30]. Another approach has been to use ANN's to detect faces in images [34, 48]. An ANN has also been used to determine whether a certain part of an image contains an orange bag [29]. The bag was used to simulate human skin.

ANNs appear to work well in cases where the object to be detected has a certain structure which is the case for both the human face and to a lesser extent the head-shoulder shape. Using the raw image as input however does not seem like the best approach since the classifier will be very sensitive to changes in color and lighting. With an edge map, however, ANN could probably be used for detecting disaster victims.

2.9 Chapter Summary and Conclusion

This chapter has provided the reader with some historical background on disaster management. Due to the major implications of natural disasters, several countries including the United States of America and Japan are spending large amounts of money every year on emergency prevention and response. Robot technology is being developed in the hope that robots can aid rescue workers in the wake of a disaster and thus improve the chances of saving the lives of victims. So far robots have not played a central role in any rescue operation and in the few cases where robots were employed, they did not possess any significant form of intelligent system that could aid the operator in locating victims. Based on the experience that has been gained from real applications, it seems that many problems can be addressed by making disaster robots more autonomous. The disaster scenario poses several challenges to a human detection algorithm since it is difficult to make general assumptions about the state of disaster victims.

A review of the literature that relates to human detection shows that several fast and reliable methods have been proposed that could feasibly be implemented on a robot platform and used to detect victims in a disaster situation. The following chapter describes how such a system can be implemented in software and which methods that have been chosen for this work.

Chapter 3

Human Detection System

This chapter describes how a human detection system could fit into an existing robotic system and how it could be used to aid the operator and make the search effort more efficient. The first sections define the purpose and context of the system and look at a major area of robot research that deals with searching and mapping an unknown area. The following section explains what methods have been chosen for use in the detection system and why. A full overview of the detection system is provided and the remaining sections provide details about the 4 major detection modules that make up the full human detection system.

3.1 Purpose and Context of Detection System

Based on the analysis in the previous chapter it is apparent that an autonomous disaster victim detection system will not work in isolation but will be a part of a larger and possibly quite complex system that involves both hardware and humans. Regardless of how advanced a robotic solution is, the actual hardware still needs to be operated by humans. This fact needs to be kept in mind when designing a victim detection system.

The work done for this thesis has been focused on the software implementation of a human detection system that only uses input from an ordinary camera. The design of a fully autonomous robotic system is thus beyond the scope of this work. Still, it is useful to consider how a visual victim detection system would fit into a larger robotic system and how it can affect the functionality as a whole.

3.2 Simultaneous Localization and Mapping (SLAM)

A very popular area of research within automated robotics is Simultaneous Localization and Mapping, often abbreviated as SLAM. This field involves designing

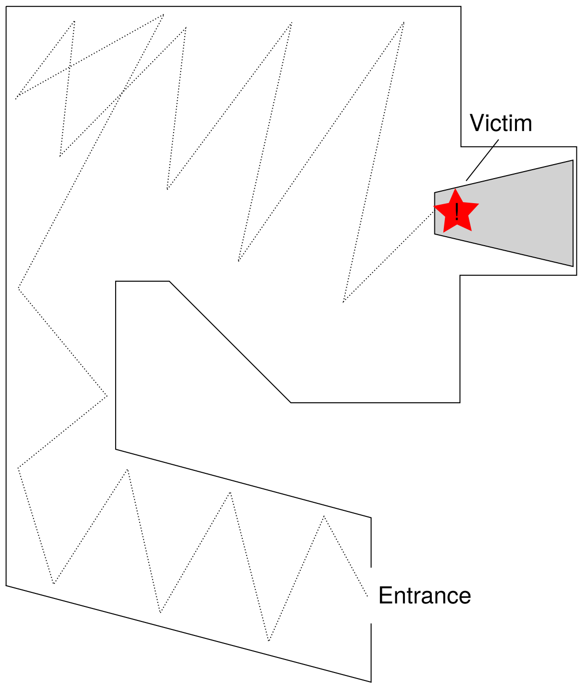


Figure 3.1: Illustration of how a map generated by a disaster robot could look. The dotted lines indicate the route the robot has taken in mapping the area. The position of a victim is indicated by the position and viewing angle of the robot at the time of detection.

algorithms that allow a robot to map an unknown environment by using a variety of sensors such as rangefinders, gyroscopes and accelerometers. Such a system has numerous uses that could include mapping a disaster area and locating human victims. The question is how the victim information should be incorporated into the mapping process.

The presence of a victim detection system could be entirely transparent to the mapping process or it could be used actively to guide the robot towards areas of interest. The mapping information could be combined with the detection system by recording the position and orientation of the robot when a victim is detected. This would aid rescue workers in locating and evacuating victims.

Returning to the points brought up in section 2.5, a disaster robot should be simple to use by rescue workers or it might not be used at all. A big part of this requirement comes down to ease of deployment. However, once the robot is running it must also be able to transmit information about victims in a meaningful way. If the rescue workers are unable to quickly interpret the data they receive then the usefulness of the robotic system is greatly diminished.

A simple 2D map with the locations of detected victims clearly marked as illustrated in Figure 3.1 would probably be the most efficient way of guiding rescue workers. If a robot could search and map an unknown disaster area and produce such a map autonomously, it could be of great help to the rescue operation. Unfortunately it is extremely difficult if not impossible to create such a perfect system today. No software algorithm is capable of identifying humans with the same accuracy as another human and there's a high risk that a fully autonomous system would at some point fail to detect the presence of a human victim that a human operator would have spotted.

One solution to this problem could be to retain the autonomous movement and victim search operation of the robot and then present any potential victims that are detected to a human operator for verification. Without the human verification element, the victim detection algorithm would need to have a very high detection accuracy to be useful, meaning a high detection rate and a very low probability of false positives. Otherwise the output map would likely be dominated by false positives and would thus be of little help. With a human to verify the output of the victim detector, false positives would not be a big issue. Since there is always a trade-off between detection rate and false positives, the algorithm could therefore be tuned to improve the detection rate.

The map illustrated in Figure 3.1 is a simplified scenario that is probably not representative of a real disaster scenario. The robots that were employed during the 9/11 response were mainly used to search areas that were impossible to enter by humans. Realistically, a disaster robot will thus be moving in a 3D space rather than on a 2D plane. This complicates the mapping process greatly and means that it likely wont be possible to determine the position of the robot relative to the entrance point with high accuracy. This is a problem regardless of whether the robot has victim detecting capabilities or not and it can well be argued that even an inaccurate estimate of where the victim is located is better than no estimate at all.

With a victim detection system as a visual aid, it is conceivable that a single human operator could monitor the activity of several robots at the same time, paying special attention to the areas in the video streams that are highlighted by the detection algorithm.

3.3 Project Delimitation

As mentioned earlier, the focus of this thesis is on creating a detection system that relies only on input from an ordinary visible-light camera. There are a couple of reasons for this choice. First of all, the use of an ordinary camera allows for easier testing and thus faster development since solutions do not require special hardware to run and since many different data sets for testing are available online. Furthermore ordinary cameras are cheaper than thermal ones, meaning that more robots could be built for the same price. Finally, by using an ordinary camera, the detection system could potentially be used in other scenarios than a disaster setting. The problem has received much attention in the scientific literature and is well known for being difficult to solve. For this reason it also has a certain appeal from an academic perspective.

The system developed for this thesis has not been made to run on an actual mobile platform and should thus be considered more as a proof of concept rather than a final solution. The constraints inherent to such a platform have however been kept in mind throughout the whole project. Processing time is perhaps the largest constraint imposed on the system since processing must happen in real time. Ideally the system should be able to process frames from the camera as they arrive, meaning around 10 frames per second. While less could probably still work, the frame rate should stay high enough that the video feed is perceived by the operator as continuous.

The majority of code produced for this thesis has been written in Python which is an interpreted language and therefore quite slow to execute. Certain time critical parts have been implemented to run faster, either using C++ or CUDA or by employing optimized toolkits such as the OpenCV library. Still, the system is not considered as optimized. The real time requirement has not been satisfied, however it is not considered to be technically unrealistic. The possibilities for implementing the system to perform image analysis in real time are discussed in section 3.12.

3.4 Choice of Methods

As described in the previous chapter, several methods have been proposed in the literature for locating humans in a disaster scenario and in general. The appearance of the human body in an image is generally very difficult to model due to the large variation possible due to articulation of body limbs. Since the camera is assumed to be moving and there is thus no simple way to segment the image such that the body is isolated from the background, searching for the entire body does not seem like the best option. A more appealing approach is to search for certain parts that can be modeled in a simpler way.

Since the ultimate goal is a system that can process images in real-time, the detector should be fast. This criterion is difficult to quantify since the speed of an algorithm depends on many things, including the efficiency of the implementation and the power of the hardware it runs on. Most code that has been written for this thesis has not been optimized and can only be considered of prototype quality. For this reason an algorithm is considered fast if the processing time for an image can be measured in seconds rather than tens of seconds or minutes.

Another criterion is that the detector should be robust against differences in body pose and orientation in the image. This follows from the discussion in section 2.6. Finally, if the detector requires training, the performance must generalize well to other data sets. This is a vital quality for the detector to possess since data sets from real disaster scenarios are not available. This criterion is considered to be satisfied if a detector performs well on a wide range of images captured in different external environments. In practice it is unlikely that any method can live up to all criteria. By using multiple detectors however the weaknesses of one detector can hopefully be mitigated by the others.

For the detection system designed in this thesis 4 methods were chosen: a head/face detector based on the HOG descriptor and an SVM classifier, a Viola-Jones based face detector, a head detector based on template matching and a contour detector for analyzing the shape of detected objects in the image. These 4 methods or method combinations form the basis of 4 detection modules. The theory behind all of these methods is described in appendix A.

All methods attempt to detect the human head or the face. This choice was made since the head and the face are some of the most recognizable parts of the human body. The face has a very distinct visual structure that is similar between humans and is largely unaffected by the state of rest of the body. This makes it relatively easy to detect in images, assuming that the person is facing the camera. While there is no guarantee that this will be the case for a victim in a disaster scenario, the presence of a face is a very strong indicator that a person is in the image. As such, it can be a very useful part of larger detection system.

The shape of the human head is another feature that is easily recognizable and isn't affected by articulation. The fact that the shape is very close to round can be exploited to create a fast head detector that is robust to different body poses and rotations of the head. The HOG descriptor was chosen as a feature due to its popularity for object detection.

Each of these 3 modules produce a location in image that is detected as a head or face. If more than one module detects the same area, it is considered a valid match. Otherwise the area is analyzed by the contour analysis module. This module will attempt to verify if the object in the area has the shape of a human head.

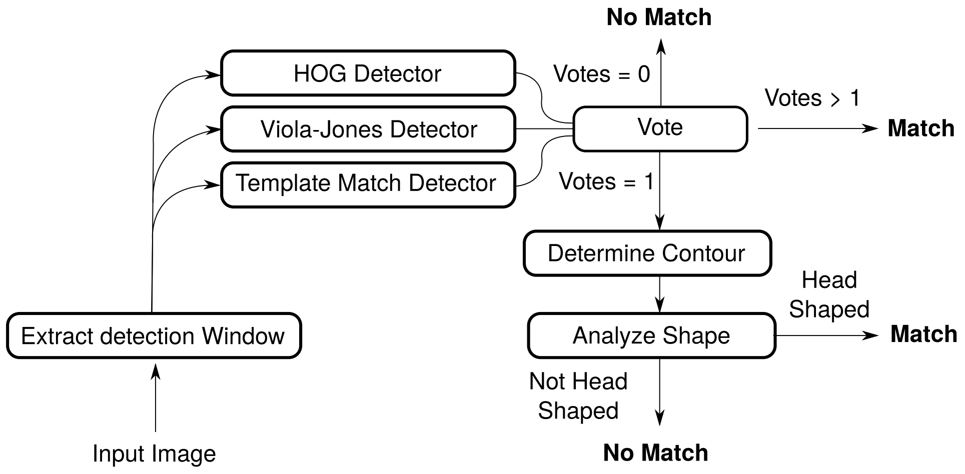


Figure 3.2: Flowchart showing the processing and detection steps involved in detecting humans in an image

All detection modules except the Viola-Jones detector focus on the shape of the head. One advantage to this approach is that the detectors should generalize well as the appearance of the shape of an object in an image is not affected much by the setting or the image quality. A downside could be that the human head is not the only round object that can appear in an image. Still, the shape is not too common and there is thus a good chance that a detected round object is actually a head.

A flowchart of the detection system designed for this thesis is shown in Figure 3.2. It should be noted that only 2 of the 4 detectors are robust against rotation in their implemented form. The HOG and Viola-Jones detectors require the head of the person to appear upright in the image. This can be solved by running the detectors on rotated version of the image at the cost of longer computation times. All steps in detection system are described in the following subsections.

3.4.1 Input Image

The input image is a color image in the RGB format. Resolution is not constrained but configuration of later steps may depend on it.

3.4.2 Extract Detection Window

A small region of the image is extracted. This is called the detection window. In the actual implementation, this step is done separately for each detector.

3.4.3 HOG Detector

This detector computes a HOG descriptor from the detection window and uses an SVM classifier to determine if the region contains a human head. The detector uses a linear SVM classifier that has been trained with human heads at different poses. The HOG detector and the training process is described in section 3.7.

3.4.4 Viola-Jones Detector

This detector uses the Viola-Jones object detection framework to determine if the detection window contains a face. The framework is described in more detail in section 3.8.

3.4.5 Template Match Detector

The template matching detector scans an edge map obtained from the input image in search of round shapes. The detector is described in chapter 3.9

3.4.6 Detector Vote

This is the decision function that decides if an image region contains a human head/face or not. It gets a vote from each detector and validates a given region if 2 or more detectors agree.

3.4.7 Determine Contour

This step will determine an enclosing contour around the center of the detection window using Graphcut and Grabcut methods.

3.4.8 Analyze Shape

This final step will analyze the contour obtained from the previous step and determine if it has the shape of a human head or the head and shoulders. If this is the case, the window is considered a match. Otherwise it is discarded.

3.5 Software Implementation

The detection system has largely been implemented in Python with small parts written in C++ or CUDA. Certain methods have been fully implemented for this work such as fixation based segmentation and the min-cut/max-flow algorithm (see section A.5) as well as the template matching detector. For other methods, library implementations have been used. These are the HOG Descriptor, SVM classifier, Viola-Jones detection framework, GrabCut and the Pb contour detector.

Several small tools have been created to make training, testing and validating the different detectors as easy as possible. All produced code, including these tools has been included on the DVD[1].

3.6 Evaluation Methodology

In order to evaluate the methods described in this thesis, a variety of image data sets were used. Where training data has been needed it was selected specifically for the method in question. For initial testing and tuning of each detection module, the VOC2012 [49] test set was used. From this set about 200 images containing humans were selected and used for testing. All humans in the images were manually labeled using a simple Python script. The labeling process is described in appendix B.

For the final system test a more realistic image set was used that tries to simulate some of the conditions in a disaster scenario. The test set was produced at the UNSW Robotics lab and has been used with permission in this work. Furthermore, an additional set of images were selected at random from the VOC2012 set to use for validation. The final test of the system is described in chapter 4. The following sections describe the implementation of the 4 detection modules of the system.

3.7 HOG Based Head Detector

This section describes the implementation of a human head detector that is based on the HOG descriptor. This includes how the detector has been trained and which data sets were used. The last subsection shows the results that were obtained on a labeled image set.

In order to train and test the implemented HOG detector, a framework has been created using Python and various open source libraries. The framework makes it easy to specify training data and settings and to test the trained classifier. The framework is included on the DVD along with instruction on how to use it[2].The detection system uses OpenCV to calculate the HOG descriptor from a detection window. Default parameters have been used and they can be seen in Table 3.1. The detector uses a linear SVM classifier to determine if a window contains a human head. The SVM classifier has been trained using the Scikit library for Python. The theory of the HOG descriptor and the SVM classifier is described in appendix A.1 and A.1.1.

By substituting the parameters from Table 3.1 into (A.1) we get:

[1]/code
[2]\code

Parameter	Value
Cell Size	8x8
Block Size	16x16
Block Stride	8
Orientation Bins	9

Table 3.1: Parameters used for calculating HOG descriptors

$$\text{nFeatures} = \text{floor}\left(\frac{\text{Win Height}}{8} - 1\right) \cdot \text{floor}\left(\frac{\text{Win Width}}{8} - 1\right) \cdot 36 \tag{3.1}$$

3.7.1 Training the Human Head Detector

In order to train the SVM classifier, images representing both classes are needed. In our case, this means a positive and negative training set. Since we are interested detecting faces, the positive training set should contain a variety of human faces in different poses. The negative training set can be anything except human faces. Choosing the optimal sets of images for training has largely been a process of trial and error.

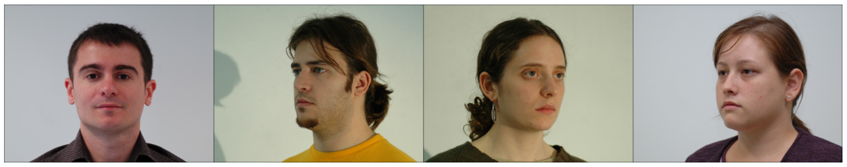
Many data sets have been created for other research projects for the purpose of face recognition/detection and are available online. A detection window size of 168x192 pixels was chosen. The choice of detection window size was somewhat arbitrary and the rationale was mainly that the aspect ratio corresponded well to the size of a human head. Inserting the window dimensions into (3.1) the number of elements in the feature vector is 16560. The following 2 sets were used as positives for training the HOG detector.

3.7.2 GTAV Face Database

This set has been obtained with permission from the Audio Visual Technologies Group at the Polytechnical University of Catalonia[3]. The database contains around 1700 images featuring 44 individuals, both male and female, photographed from different viewpoints. Images are normalized but often with significant deviation. A few examples are shown in Figure 3.3a. Cropping this image set was challenging due to the changing position of the subject relative to the camera.

To isolate the face a script was created that traces lines from the left, top and right side of the image until it detects large changes in pixel values. These positions are then used to determine the location and dimensions of the head. To define the bottom of the head, a constant value is added from the top of the located head. The aspect ratio of the region is then calculated and if it is close to $\frac{192}{168}$ the region

[3]F.Tarrés, A. Rama, "GTAV Face Database" available at
http://gps-tsc.upc.es/GTAV/ResearchAreas/UPCFaceDatabase/GTAVFaceDatabase.htm


(a) Examples from the data set

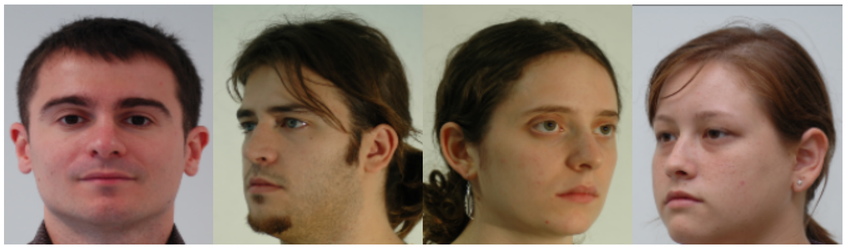

(b) Same images where the faces have been isolated

Figure 3.3: Examples from the GTAV data set [50]

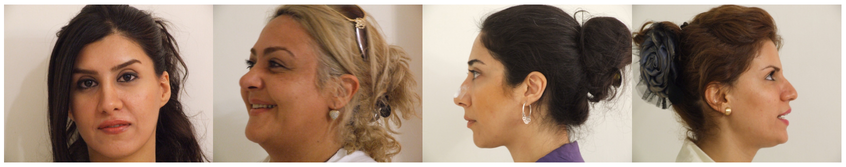
is resized to 168x192 pixels and saved. If the aspect ratios deviate too much, the image is discarded. The automated cropping script resulted in around 660 useful images. Examples are shown in Figure 3.3b. The resulting set differs slightly from the Iranian women set, described next, since more of the head shape is visible in the images.

3.7.3 Iranian Women Set

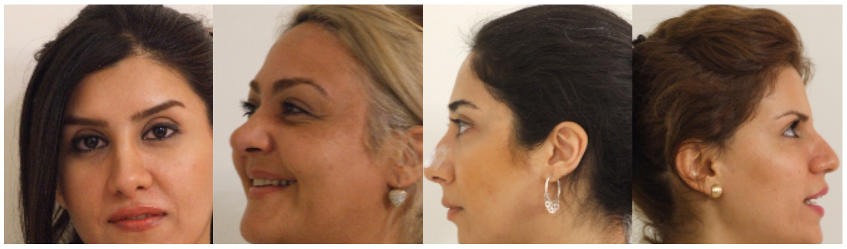
This set is a part of the Psychological Image Collection at Stirling (PICS) [51]. The set consists of 369 images featuring 34 women. The images are largely normalized with the subject faces appearing in the center of every image which allows for easy preprocessing. Furthermore, all subjects have been photographed from 5 different viewpoints ranging from profile to frontal shots. A few examples are shown in Figure 3.4a. All images were cropped to a size of 168x192 pixels using an automated script. Different hairstyles often make the heads appear very wide in the image. The approach used for the GTAV set to isolate the head therefore did not work well for this image set. Instead the script would trace a line from only one side, depending on the orientation of the face, and add a constant offset to this position. The height and vertical starting point of the region was set to a constant value for all images.

Beyond the positive image set, a negative set was also necessary. Images from the following databases have been used as negatives.

3.7.4 The Berkeley Segmentation Dataset and Benchmark



(a) Examples from the data set



(b) Same images where the faces have been isolated

Figure 3.4: Examples from the Iranian women data set [51]

The Berkeley Segmentation Dataset and Benchmark [52] is a collection of images that have been created for testing the performance of different boundary detection methods. However it serves well as a negative training set since it contains many images of non-human objects and scenery. Examples can be seen in Figure 3.5

3.7.5 Urtho's Database of Negatives

The Urtho database[4] contains around 3100 images with no visible humans. The images are black and white and feature a large variety of objects and scenery. Examples can be seen in Figure 3.6. The Urtho data set was used as negative samples during training.

3.7.6 Pascal VOC 2012 Database

[4]Originally obtained from http://face.urtho.net/. The original site is no longer available but an archived snapshot can be found here: http://web.archive.org/web/20091016021019/http://face.urtho.net/. The image set can be obtained from here: https://code.google.com/p/tutorial-haartraining/

Figure 3.5: Examples from the Berkeley image data set [52]

Figure 3.6: Examples from the Urtho negative image set [53]

Figure 3.7: Examples from the Pascal VOC2012 image data set [54]

The Pascal VOC 2012[54] image database is a large collection of images that are used as benchmarks for visual object classification. The database contains nearly 20000 images. Every image is accompanied by an annotation file that specifies the objects that are visible in the image and where they are located. For human subjects, the file also specifies the positions of visible body parts such as heads or hands.

This data set has been used both for negatives as well as a validation set to test the head detection.

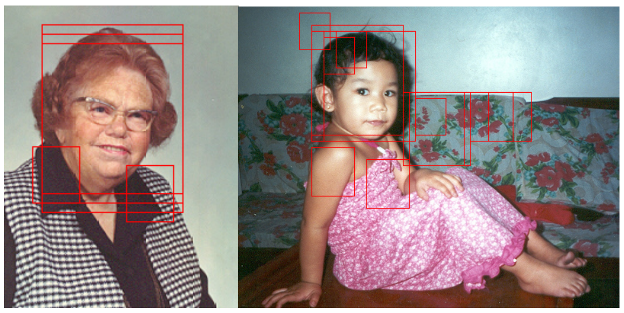
3.7.7 Training Procedure

Several classifiers were trained to test different combinations of image sets. In order to classify human heads correctly at different scales, all images in the data set were resized and a separate classifier was trained for every scale. Henceforth in this chapter this ensemble of multiscale classifiers are referred to as just one single classifier. The scales used for training were [1.0, 0.75, 0.50, 0.33, 0.25].

A random selection of around 350 images from the Berkeley and VOC2012 databases were used as negatives. Every image in the negative set contributed several samples since the resolution is larger than the 168x192 detection window used for matching. Negative samples were extracted from the images in a sliding window manner using a stride of 50 pixels. A large stride was necessary to limit memory usage during training. Unfortunately this also meant that the negatives were not utilized completely. To solve this problem each classifier was trained in several steps as follows:

1. the classifier is trained using the base set of positives and negatives
2. The set of negatives are scanned for matches using the trained classifier and a stride of 10 pixels
3. All windows classified as positive matches from the negative set that are saved
4. The classifier is retrained using an augmented set of negatives that include the saved match windows obtained in 3

3.7.8 Merging Overlapping Match Windows


(a) Examples of match detection windows emerging close to each other

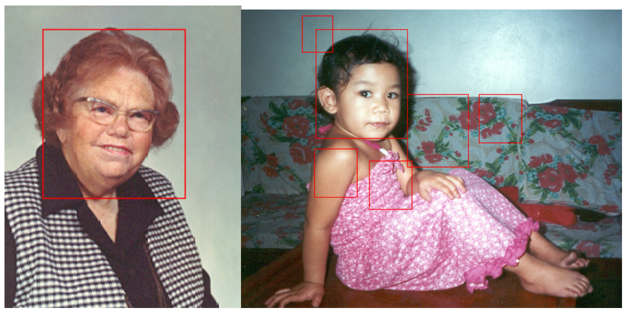

(b) Result of merging the overlapping match windows

Figure 3.8: Examples of match detection windows before and after merge operation

From initial testing it was apparent that the classifier would often classify several windows in an area as positives. Two examples of this phenomenon are shown in Figure 3.8a. The problem can be solved by merging match windows that overlap by a certain amount. A set of rules were defined to merge overlapping windows.

The rules were obtained from simple reasoning based on observed patterns of overlapping windows around heads in the images. The rules are illustrated in Figure 3.9 and listed here:

- If a window contains another window of smaller scale entirely within its area, the small scale window is discarded.
- If a window overlaps with another window that is more than 1 scale step smaller (e.g. a scale 1.0 window overlapping a scale 0.33) by more than 50 % of the smallest window area, the small window is discarded.
- If a window overlaps with one or more windows that are of same scale or one step below and the windows overlap pairwise by more than 50 % of the area of the smallest window in that pair, then all windows are merged into one window. The resulting window is assigned the largest scale of its constituents. The center of the new window is a weighted average of all window centers weighted by their relative scales.

Applying these rules gave good results. Two examples are shown in Figure 3.8b.

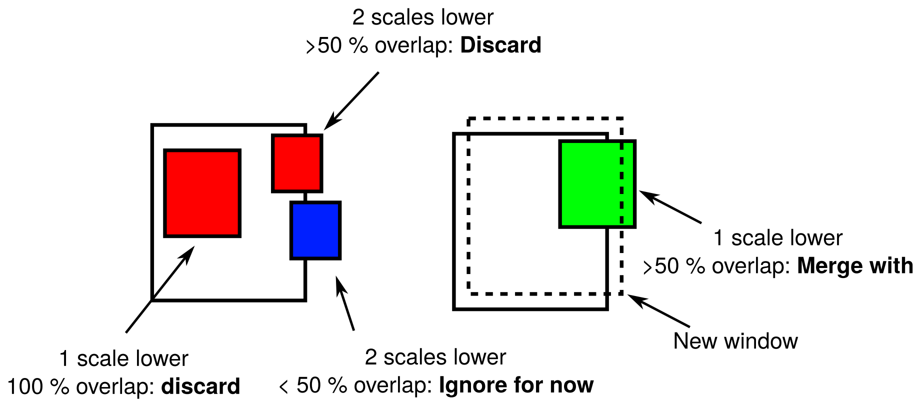

2 scales lower
>50 % overlap: **Discard**
1 scale lower
>50 % overlap: **Merge with**
New window
1 scale lower
100 % overlap: **discard**
2 scales lower
< 50 % overlap: **Ignore for now**

Figure 3.9: Illustration of the rules for merging match windows

3.7.9 Evaluating Classifier Performance

In order to validate the performance of each classifier, a set of 390 images from the VOC2012 database were picked, that all contain visible human heads. The set was picked at random and contains humans in many different poses. All images were labeled manually. For every image the position of the head and an approximate scale was specified. This was done using a simple Python script that is described in appendix B. The image set was classified using the test framework and the location and scale of match windows were saved for every image. The classified matches were then compared with the labels. For a match to be considered a true positive it had to satisfy the following criteria:

G	GTAV
B	Berkeley
V	VOC2012
MN	Matches from negatives
U	Urtho

Table 3.2: List of data sets and their abbreviations

- The center of the match window must lie within the labeled region.
- The scale of the match window must be within 1 scale step of the labeled region
- The labeled region must not already be tied to another match window

All windows that did not satisfy these criteria were considered false positives. Labeled windows that were not tied to a match window were marked as false negatives.

Several tests were carried out to gauge the performance of the HOG detector when trained with different data sets.

3.7.10 Test and Results

Much informal testing was done initially to gauge how well a HOG based head detector could work. As results turned out to be promising a more elaborate framework was created and more formal tests were designed. Only the formal tests are described herein.

The testing process was largely a matter of trying out different combinations of training data and observing the results. Due to the enormous amount of variation possible in a 168x192 window, it is impossible to create a negative data set that can represent all possible non-human-head samples. Several different sets were tested and their results compared. The full list of tests performed along with their results can be found in Table 3.3. The abbreviations used are explained in Table 3.2. Different combinations of positive and negative data sets were tested and the performance of each combination is shown both with and without retraining.

It is apparent that retraining the classifier using the matching negatives results in a substantial decrease in false positives and a relatively smaller decrease in detection rate. Deciding which combination of data sets performs the best turns out to be difficult since there is a clear trade-off between detection rate and false positives. Two figures are of importance: the detection rate and the ratio between true and false positives. The highest detection rate of 81 % was obtained with the GTAV and Iranian Women sets as positives and the Berkeley and VOC2012 as negatives. Unfortunately this combination also has a very high number of false positives. The best compromise between detection rate and TP/FP ratio appears to be a combination of all the data sets which yields a 55 % detection rate and around 3

#	Positives	Negatives	TP	FP	FN	Detection Rate	TP/FP
3	G	B + V	444	2732	241	0.65	0.16
0	G	B + V + MN	168	207	326	0.34	0.81
5	G + IW	B + V	554	6634	133	0.81	0.08
2	G + IW	B + V + MN	426	1879	260	0.62	0.22
4	G + IW	B + V + U	530	4077	157	0.77	0.13
1	G + IW	B + V + U + MN	369	1184	301	0.55	0.31

Table 3.3: Results of tests performed using the implemented HOG detector. TP = True positives; FP = False positives; FN = False negatives. A total of 685 labeled heads are visible in the test set.

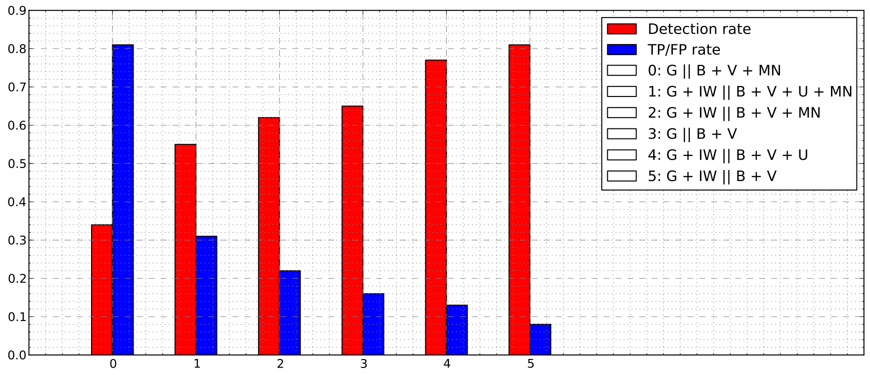


Figure 3.10: The results obtained by using different combinations of data sets for training the SVM classifier. The combinations are indicated as positives || negatives.

false positives per true positive. A few output images from this classifier are shown in Figure 3.11.

3.8 Viola-Jones Face Detector

This section briefly describes the implementation of a face detector that is based on the Viola-Jones framework and how it was tested. This detection module uses the OpenCV library to detect faces in images. The library contains a pre-trained face classifier that contains 19 stages in a cascade. Since a library implementation was used, this module does not have any notable implementation details. The theory behind the Viola-Jones framework is described in appendix A.2.

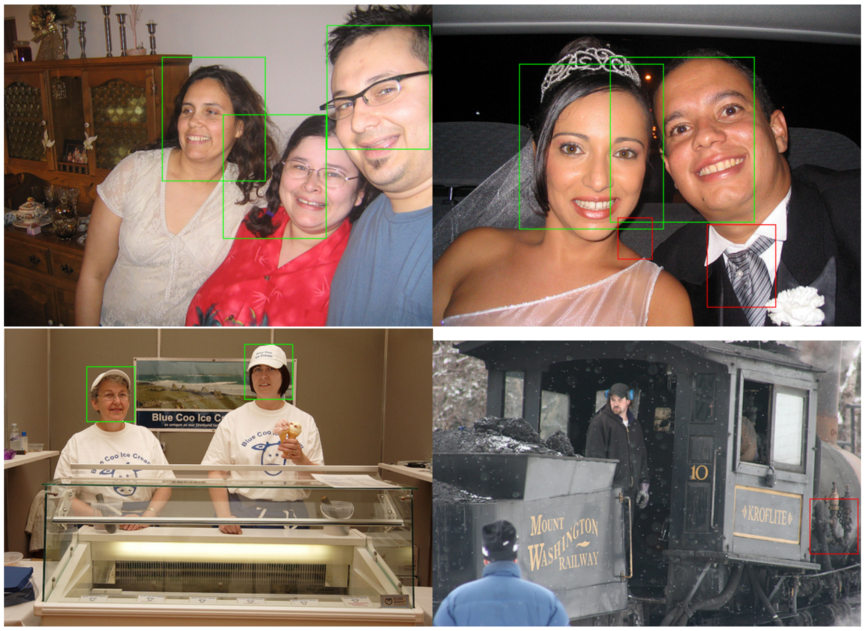


Figure 3.11: Examples of output images from the HOG detector where all data sets were used for training. Valid and invalid matches are indicated by green and red rectangle respectively.

3.8.1 Tests and Results

The Viola-Jones face detector was tested on the same image data set as was used for the HOG head detector. The set consists of 390 images containing humans that have been labeled manually. Using the pretrained classifier from OpenCV, a detection rate of 44 % was achieved with 73 false positives. The TP/FP ratio was 3.97. A few examples of the results obtained are shown in Figure 3.12.

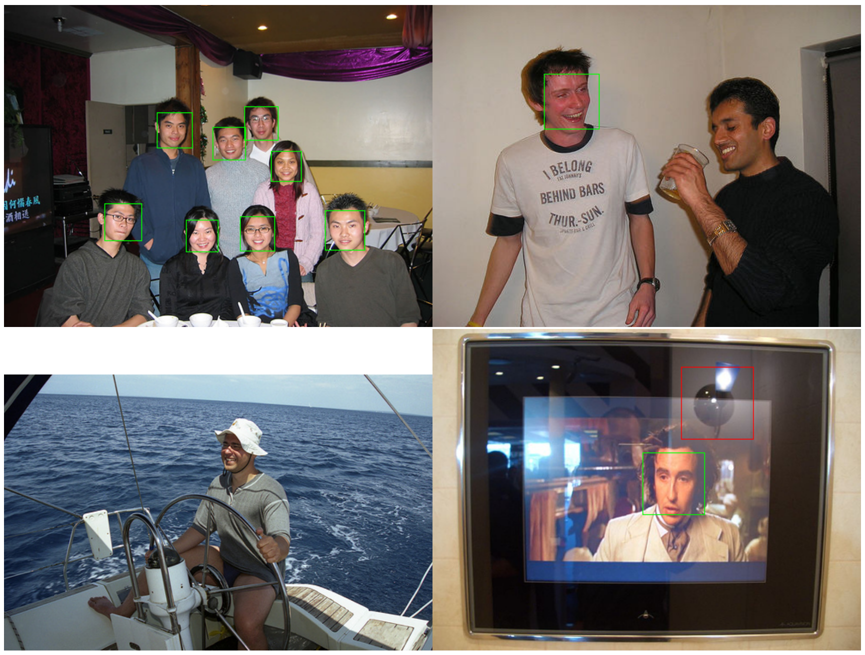


Figure 3.12: Results obtained from the Viola Jones face detector

3.9 Template Matching Head Detector

This section explains how a detector based on template matching can be used to detect humans in an image. The theory behind template matching is described in further detail in appendix A.4. The idea is to try and identify shapes that are characteristic of the human contour. The contour of a human body can have endless forms due to variation in pose and articulation of limbs. It is therefore extremely difficult to find any single shape that will work reliably with all body configurations. This problem is complicated even further if only a part of the body is visible in the image.

In many cases it is safe to assume that the person will have an upright pose and this will in turn simplify the detection process greatly. However, in a disaster scenario such an assumption would limit the search space too much since a victim will likely not appear upright in the image. For a detection method to be useful it will need to be invariant to, or at least robust against, rotation. With template matching this is fulfilled by rotating the template or the image during detection. Furthermore the template can be used at different scales to detect humans of different sizes in the image.

Rotating and scaling the template unfortunately adds greatly to the computation time. Combined with the fact that it's very difficult to define a general shape that fits the human body under all conditions, it does not seem viable to search for the entire body in one step.

A more appealing approach is to look for simpler shapes that are likely to show up on the contour of the body. A good shape to look for is that of the head since it is not affected much by pose or articulation. Furthermore, it has a round shape which can be exploited to vastly reduce the search space. The contour of a human head is not a perfect circle and will therefore generally not match completely with a perfectly circular template. In fact, it is usually closer in appearance to an ellipse. However, there seems to be a high probability that a half-circle or quarter-circle can be fitted very well. A list of circular shapes as well as their advantages and disadvantages in template matching is shown in Table 3.5.

Assuming that an ellipse is the best model for the shape of a human head, the straight forward approach is to just search the image for ellipse shaped contours at different scales and rotations. The results obtained using this approach as well as its inherent problems are described in section 3.9.4. Before any template matching can be performed however, the image needs to be preprocessed. This stage is described in the following.

3.9.1 Contour Map Generation and Preprocessing

Since we are interested in locating objects that exhibit a certain shape, an edge map obtained from the image is the best place to search. While any edge detection method can be used, the Pb detector produced the best results in several test images

Shape	Pros	Cons
Circle	Completely rotation invariant and therefore fast to search for	Will generally not find a perfect match on the head contour
Ellipse	Close to the actual shape of the head. Will provide the orientation of the head if fitted properly. Symmetric	Not rotation invariant (although symmetric) and thus slow to search for exhaustively
Half-circle	Good chance of finding a match on the head contour	Not rotation invariant
Quarter-circle	Small and fast to evaluate. At least one of the four quadrants should find a match within the head	Not rotation invariant. Shape is ubiquitous in images

Table 3.5: List of circular shapes along with their pros and cons

(see appendix A.3.3). The details of this method are described in section A.3.4. The Pb detector produces a function $P_b(x, y, \theta)$ that in turn gives the probability that a given pixel in the image lies on an object boundary with orientation θ. This means that object contours are stretched out spatially as shown in Figure 3.13b. By applying non-maximum suppression, only the pixels that lie on a local maximum in the boundary probability map are retained. This step is similar to the one used in the Canny edge detector which is described in appendix A.3.3. The result of applying non-maximum suppression is shown in Figure 3.13c.

While the Pb detector does a good job of finding the true contours of objects, it cannot completely eliminate the effects of internal edges and large texture gradients. The output will inevitably contain pixels that do not correspond to an actual contour in the original image. Fortunately these pixels tend to have lower probability values than the "true" contour pixels, and some of them can thus be filtered away by applying a simple threshold operation. Choosing a fixed threshold value that works well for any image is generally not possible so instead the threshold is set to 25 % of the largest probability value in the non-max suppressed image. The value was determined through experimentation and was found to give good results. The result of the threshold operation is shown in Figure 3.14.

3.9.2 Locating the Human Head Using Circular Shapes

The most obvious choice of a simple shape to match the human head is an ellipse. Performing an exhaustive search using an ellipse template is very expensive computationally since the algorithm needs to search the image at all positions, scales and rotations. However, the results can be viewed as an upper bound for how well

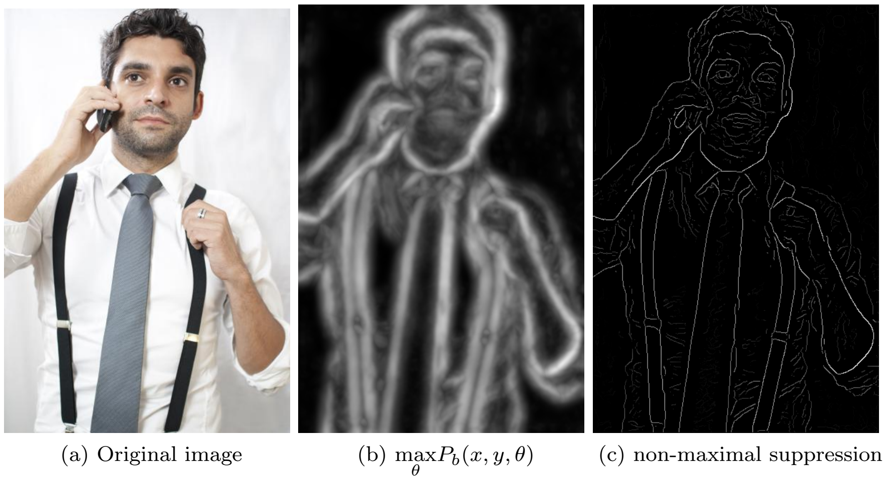

(a) Original image (b) $\max_\theta P_b(x, y, \theta)$ (c) non-maximal suppression

Figure 3.13: Results of applying the Pb detector to an image

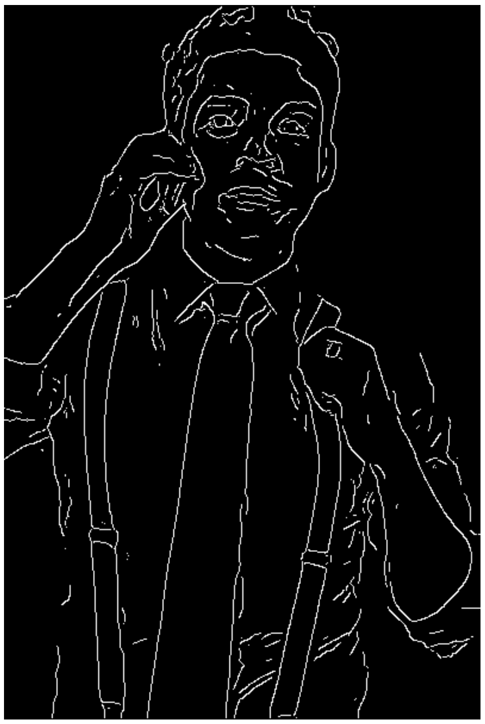


Figure 3.14: Result of applying a threshold operation on the non-max suppressed image. The threshold value is set to 25 % of the largest probability value in the image.

the template can be used to locate heads in a given image.

A small framework has been created using Python, OpenCV and various other libraries that can load an input image, preprocess it and perform template matching using a template of the form seen in equation (A.15). Template matching is performed as described in these steps:

1. Define scale and rotation ranges for template search.
2. Define a stride for the sliding window (number of pixels to skip with each move).
3. Create an empty buffer to contain the distances at all pixel location and at all scales and rotations.
4. For each scale in the scale range:
    (a) For each rotation in the rotation range:
        i. Scale and rotate the template according to the current configuration.
        ii. For each valid window region in the image:
            A. Calculate the distance between the template and the window using Hausdorff distance or Chamfer distance.
            B. Save the distance in the response buffer at the center point of the template.
5. Define threshold for distance to be detected as a match.
6. Locate all points in the distance buffer that are below the threshold.

This algorithm contains several parameters that must be specified and tuned to obtain the best performance while minimizing computational cost. These are the sliding window stride, the range of template scales, the range of rotations and threshold value for when a distance measure is small enough to be considered a match.

Determining good values for these parameters has been done through experimentation. The range of template scales needed is largely dependent on the image resolution. For the image set used in the following tests, a range of 10 scales between 1 and $\frac{1}{5}$ and a start radius of 100 pixels was found to give good results. This corresponds to a minimum head size of 40 pixels from top to bottom. At smaller scales the search algorithm starts to exhibit problems due to a small number of pixels in the template. This results in matches that have a low distance but do not actually exhibit the shape of an ellipses. This problem can be partially mitigated by multiplying the calculated distance by a factor that is inversely proportional to the number of pixels in the template. While this yields more accurate shape matches in some cases, it didn't improve performance noticeably and was therefore not used in the following tests. However, the interested reader can find more details in appendix C.

The range of rotations needed depends on the template and has therefore been determined on a per test basis. In the initial tests, every image was searched using a range of thresholds. In later tests a single value was chosen based on the observed performance.

3.9.3 Including Edge Orientation

A simple extension to the template matching framework is to include orientation information in the template [55, 38]. The Pb contour detector calculates responses for 8 different orientations between 0 and π so the orientation information in the template must be quantized into 8 orientation bins. The similarity measure also needs to be extended to include the orientation. This can be done as follows:

$$\mathrm{H_{mod}}(\mathrm{T}, \mathrm{I_{DT}}) = \max_{t \in \mathrm{T}} \mathrm{I_{DT}}\,(t) + \lambda \min\left(\left|I_\theta\,(t) - t_\theta\right|,\ 8 - \left|I_\theta\,(t) - t_\theta\right|\right)$$

$$\mathrm{D_C}(\mathrm{T}, \mathrm{I_{DT}}) = \frac{1}{N} \sum_{t \in \mathrm{T}} \left[\mathrm{I_{DT}}\,(t) + \lambda \min\left(\left|I_\theta\,(t) - t_\theta\right|,\ 8 - \left|I_\theta\,(t) - t_\theta\right|\right)\right]$$

For the Hausdorff and Chamfer distances respectively where

$I_\theta\,(t)$	is the edge orientation at coordinate t or the orientation of edge that is closest to t
t_θ	is the orientation of the element in the template
λ	is a penalty factor for differences in orientation between the template and the image window

$I_\theta\,(t)$ is essentially a modified distance transform where the orientation of the closest edge is retained instead of the distance.

The following subsections describe how tests were performed and what results were obtained. The purpose was to find the best circular shape to match the contour of a human head. All tests were performed on a set of 10 images from the VOC2012 database [54]. All images contain 1 or more persons with visible heads.

3.9.4 Tests and Results

For the initial tests, the parameters for the template matching algorithm were set as shown in Table 3.6. The shapes used were those listed in Table 3.5. To show the performance and allow for comparison between the different shapes, the results obtained using 4 of the 10 images are included herein. The 4 images are shown in Figure 3.15. Results from some of the tests have been included on the DVD that

Stride	1
Threshold	Varying
Largest radius	100 pixels
Scale range	10 scales in range $\left[1.0 - \frac{1}{5}\right]$
Rotation range	
ellipse	16 angles in range$[0 - \pi]$
Circle	1 angle
Half-circle	32 angles in range $[0 - 2\pi]$
Quarter-circle	4 angles in range $[0 - 2\pi]$

Table 3.6: Parameters for the initial template matching tests

Figure 3.15: 3 of the images used to test the template matching algorithm

accompany this thesis[5]. Every shape was matched against the images using both the Chamfer (A.21) and the Hausdorff (A.20) distance.

To analyze the performance of each shape, all heads in the 10 images were labeled using a simple Python script which can be found on the DVD[6]. These labels were then used to determine the accuracy of each shape. The lowest threshold was chosen where all heads in the images were detected[7]. The results for all shapes are shown in Figure 3.16a through 3.19a. Both the Chamfer and Hausdorff distance were used, however, in some cases the Hausdorff distance failed to detect all heads except at unreasonably high thresholds. In these cases the Hausdorff results have been left out. All figures show the threshold used. A match is considered a false positive if the center of the shape does not lie within the region of a label window. Details on how images have been labeled can be found in appendix B. It should be noted that no trimming of overlapping shapes has been performed. The exact number of FP's are therefore not useful by themselves and only serve as a way to compare the performance of the different shapes and similarity measures.

From these results. It is apparent that all circular shapes can be used to locate the head, to some extent. The Chamfer and Hausdorff distances gave different results depending on shapes used. Interestingly, the ellipse shape had fewer false positives when using the Hausdorff distance.

Another thing worth noting is that the ellipse and circle shapes require a higher threshold to be specified before they can match all heads. This is not too surprising since these shapes contain more pixels and are therefore unlikely to match any head exactly. Furthermore, some heads in the images were not detected at all by these two shapes.

The images only give a rough indication of how well the shapes and measures perform at various thresholds. More insight can be gained by plotting receiver operating characterics (ROC) for each test. These are shown in Figure 3.20. It is apparent that the quarter-circle and half-circle shapes outperform the other shapes. The Chamfer distance works best for these two shapes. The quarter-circle performs best and is also the fastest shape to evaluate of the two. It is therefore the best candidate for detecting heads. In most tests, the Chamfer distance gave the best results and is therefore the preferred similarity measure.

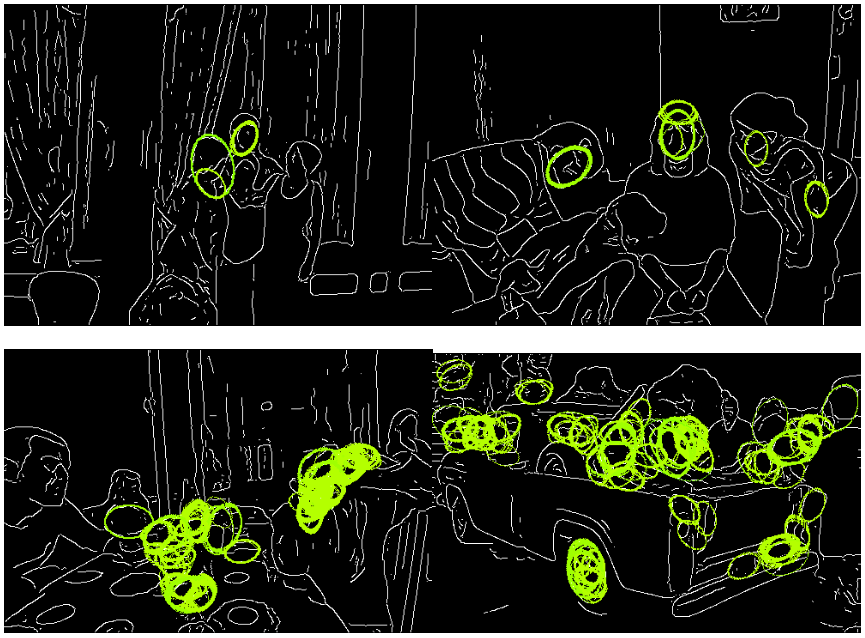
Including edge orientation when matching produced the results shown in Figure 3.21 for a quarter circle template. The additional orientation term did not improve results over the basic matching algorithm.

Even at low thresholds, all shapes produced many false positives. The next section describes how this problem can be mitigated.

[5]\test_results

[6]\code

[7]Allowed exceptions were faces that were either too small to fit the template or only partially inside the image frame. Also in a few cases the threshold had to be set so high that the results were practically useless.


(a) Chamfer distance threshold of 1.2

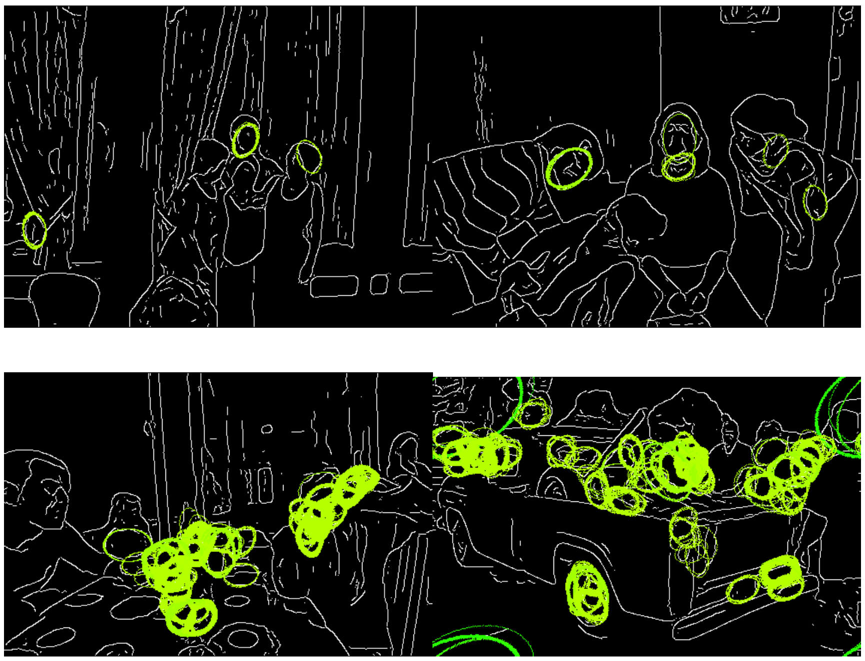

(b) Hausdorff distance threshold of 3.2

Figure 3.16: Results obtained using an ellipse shape. Note that the method fails to locate some heads using the Hausdorff distance.

3.9.5 Reducing False Positives and Duplicates

By inspecting the results visually, it is apparent that the majority of matches form groups in close proximity of each other. This suggests that a dense search across the image is not necessary for achieving good results. If a match is registered it is reasonable to assume that any further matches of shapes with the same orientation in that neighborhood will be due to the same object contour (ideally a human head). Even if this is not the case it won't necessarily hurt the performance, e.g. if one shape matches the head contour and another matches that of the face, the first match would still be valid and the second would be redundant. Redundant matches can be removed either on-the-fly by defining small areas around matches that the scanning algorithm will then ignore, or as a postprocessing step where the distances between matches are compared and redundant ones are removed. Both approaches should yield the same result. While the first is potentially faster, it is also more complex to implement. For the purpose of testing, the second approach has therefore been used.

The test results for the circular shapes showed that the quarter-circle is the best choice for locating human heads. Despite this, the high number of false positives observed during testing means that one single match is not enough to consider a contour to be a part of a human head. If two or more matches of different orientation are close together, the chance of a true positive will be higher. Certain constraints must be imposed on the configuration, i.e. the quarter circles must have different orientations and they must be close w.r.t. their center coordinates, as shown in figure 3.22.

The optimal proximity had to be determined experimentally. Figure 3.23a shows results using different proximity thresholds. For this test a larger image set consisting of 76 images was used with a response threshold of 0.6. The results can be seen in Figure 3.23a. Since a larger data set was used for testing the number of false positives is not directly comparable with previous results shown.

The ROC curve in 3.23a does not indicate a single optimal value for the threshold. The number of false positives drops faster than the detection rate for threshold between 0 and 11 pixels. Figure 3.23b shows that the ratio between true and false positives is highest at a threshold of 3 pixels and it drops quickly as the threshold is increased. The best compromise between the two figures seems to be around a threshold of 11 pixels.

By using a quarter-circle template and a threshold of 0.5 on the full VOC2012 test set, a detection rate of 83 % was achieved with a TP/FP ratio of 0.02. The high number of false negatives means that the template match detector is not very useful by itself. However, it should work with other detectors.

3.10 Contour Shape Analysis

Once a point in the image has been identified as a potential face or an entire head, it is useful to determine the entire region of that object. First of all, finding the shape

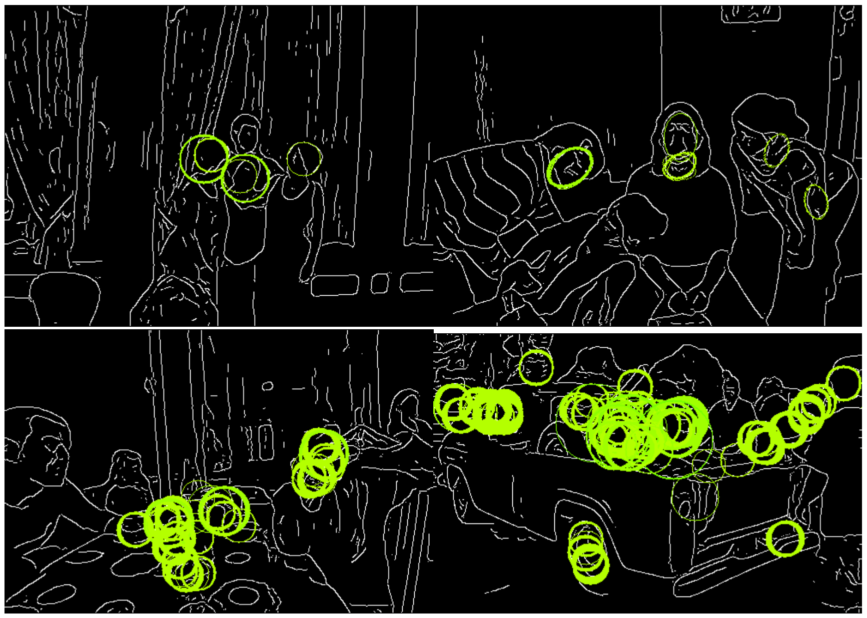
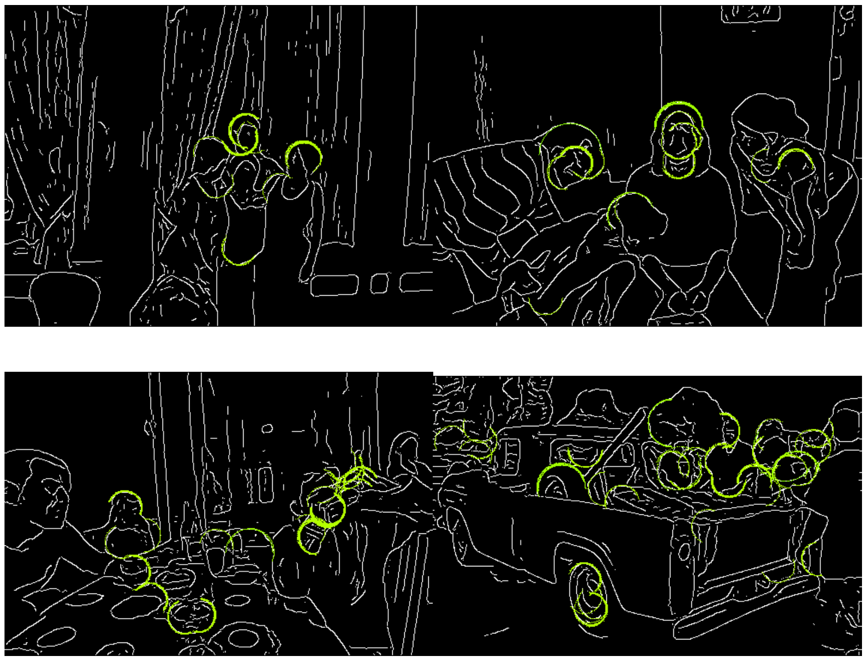
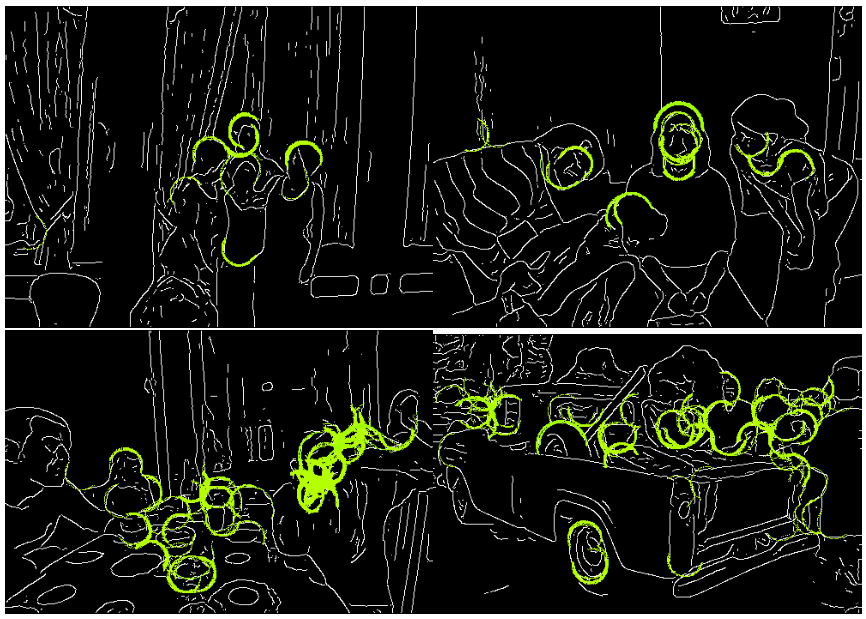


Figure 3.17: Results obtained using a circle shape with a threshold shape of 1.5

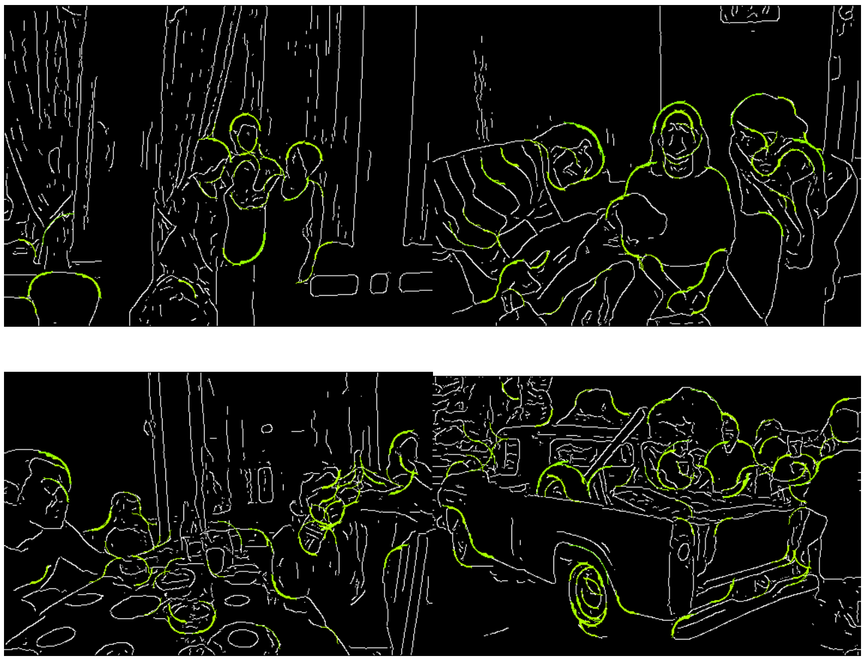


(a) Chamfer distance with a threshold of 0.8



(b) Hausdorff distance with a threshold of 2.3

Figure 3.18: Results obtained using a half-circle shape


(a) Chamfer distance at a threshold of 0.6

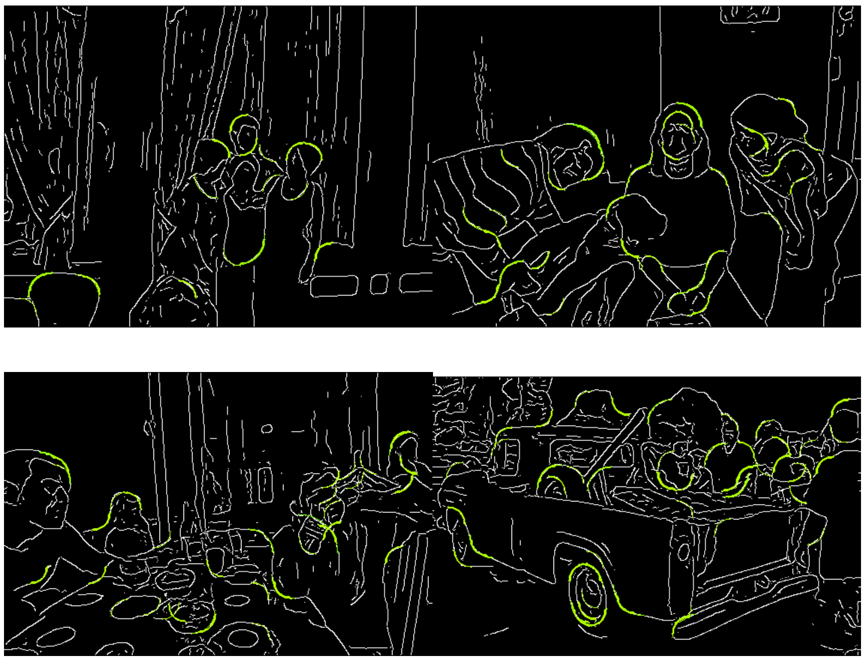

(b) Hausdorff distance at a threshold of 1.6

Figure 3.19: Results obtained using a quarter-circle shape

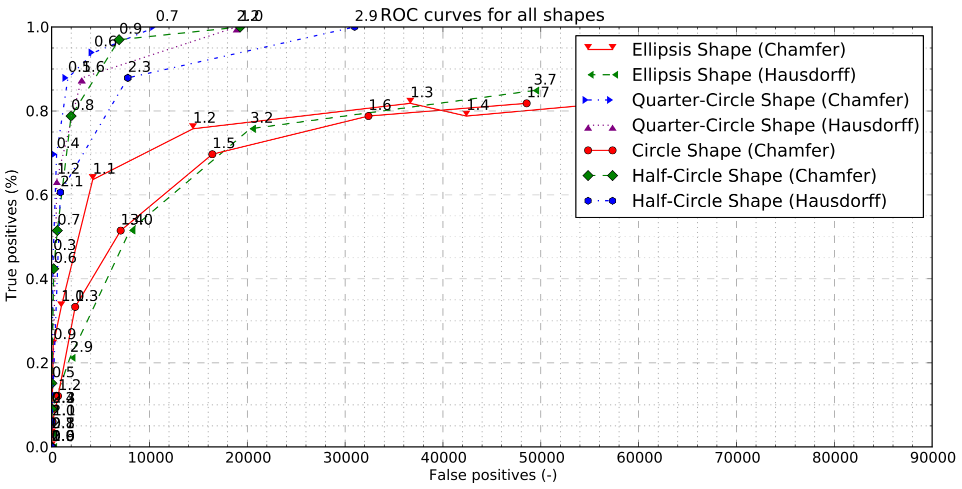


Figure 3.20: Receiver Operating Characteristics curves for all shapes. Each data point is annotated with the threshold that was used

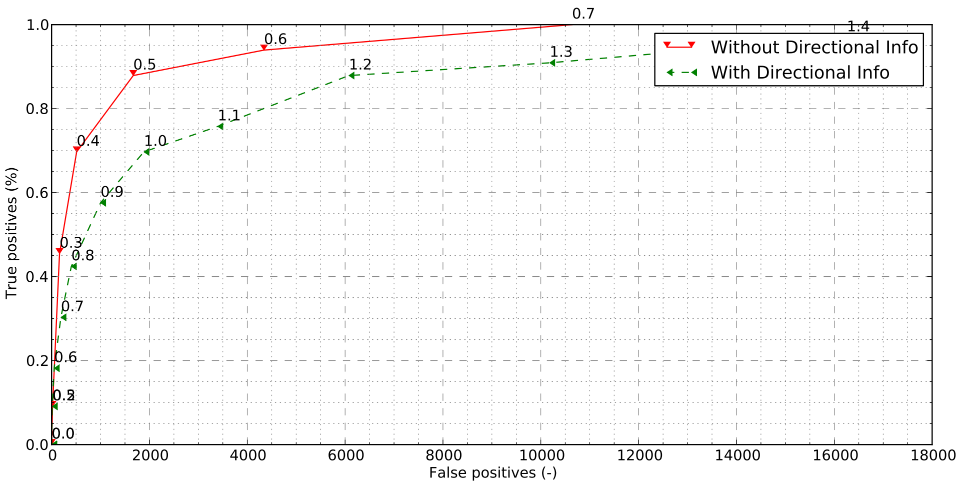


Figure 3.21: Results obtained by including edge orientation when matching shapes in the image. The plot compares the result with those obtained without edge orientation. The λ weight factor was set to 1.

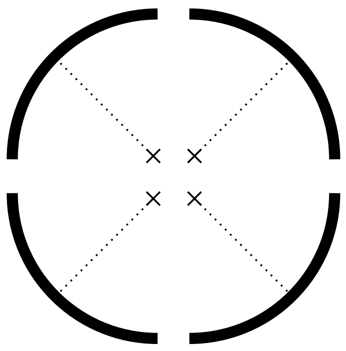
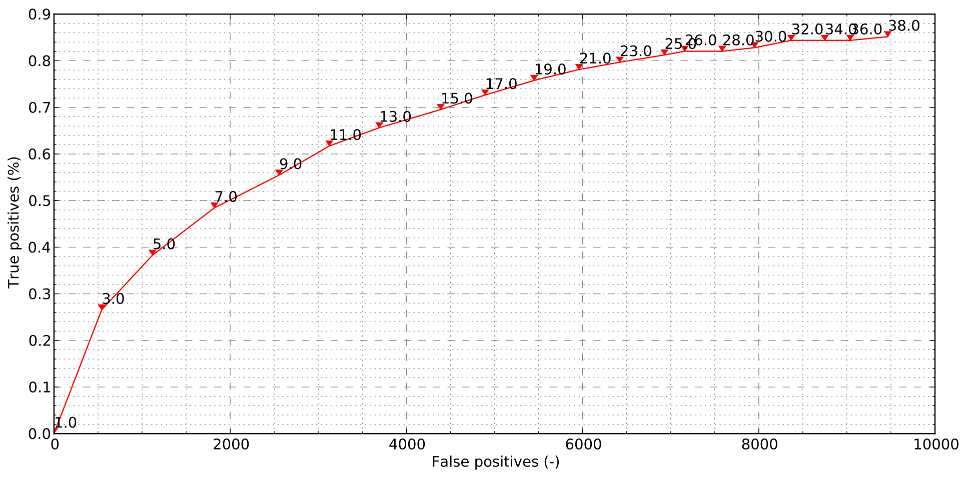
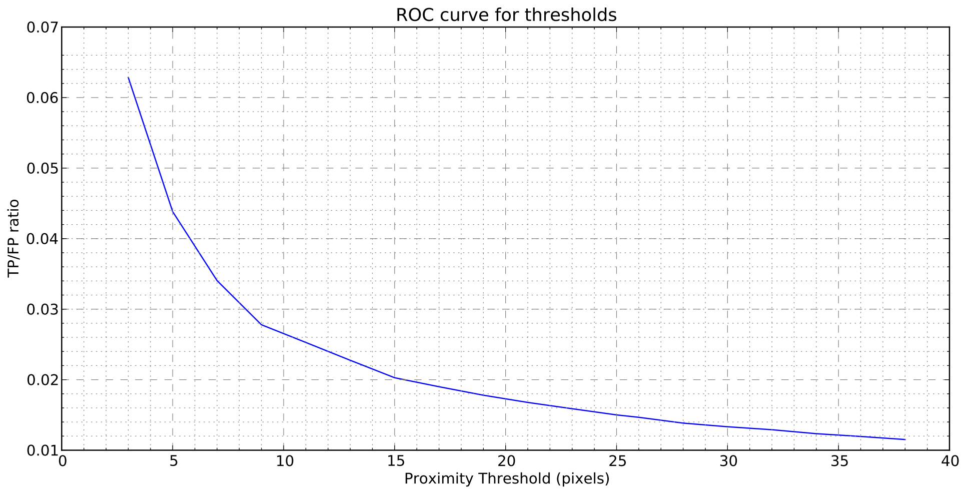


Figure 3.22: Valid configuration of quarter circles. If two or more matches with different orientations are close w.r.t their centers, they are considered as one match.



(a) Receiver Operating Characteristics curve for the quarter-circle shape obtained using different thresholds for proximity between match centers



(b) Curve showing the ratio between true and false positives as a function of the threshold

Figure 3.23

of that region can help confirm whether it is actually a face or head. Furthermore, the shape can give hints about where the rest of the body is. Segmenting the head region comes down to labeling pixels as either “object” or “background”. Two segmentation methods were used to find the contour of the face or head: A method which introduces the notion of fixation [36] and the Grabcut algorithm. The theory behind these methods is described in section A.6 and A.7.

The purpose of this module is to use seed points from the other detection modules and determine whether the region is a head or not. Unfortunately it wasn’t possible to develop this module to the point where the results were truly useful. The following section shows the results that were obtained and provides suggestions for further development.

3.10.1 Tests and Results

The method was tested on a number of images in order to assess its ability to find the correct face or head contour when given a fixation point from the other detectors. 4 of the images that were used for testing are shown in Figure 3.24.

Examples of the results obtained are shown in Figure 3.25 and 3.26 for head and non-head regions. Both figures compare the results obtained using fixation and Grabcut.

In many cases, the fixation and Grabcut algorithms are able to isolate the head from the background. Unfortunately there does not appear to be much consistency in the shape of the segmented region. The head regions do however tend to have a round shape which could be useful for distinguishing them from the non-head regions that have more random shapes. Furthermore, the segmented region sometimes contain a part of the shoulders. It is conceivable that certain features of the contour could be used to classify them as either head or non-head. This would require more testing and a deeper analysis which has not been possible within the time frame of this project. While this detection module has shown some promise it has not been included in the final detection system.

3.11 System Integration

The 3 detection modules were combined into a single detection system by comparing their output and using a voting scheme to decide if a detected area is valid or not. If 2 or more detectors agree on a given region, it is considered valid. The HOG and Viola-Jones detectors both produce a detection window whereas the template matching detector produces a point and a circle radius. In practice, determining the validity of a detected region has involved checking if its center position is inside a detection window from another detector. The decision scheme is illustrated in Figure 3.27.

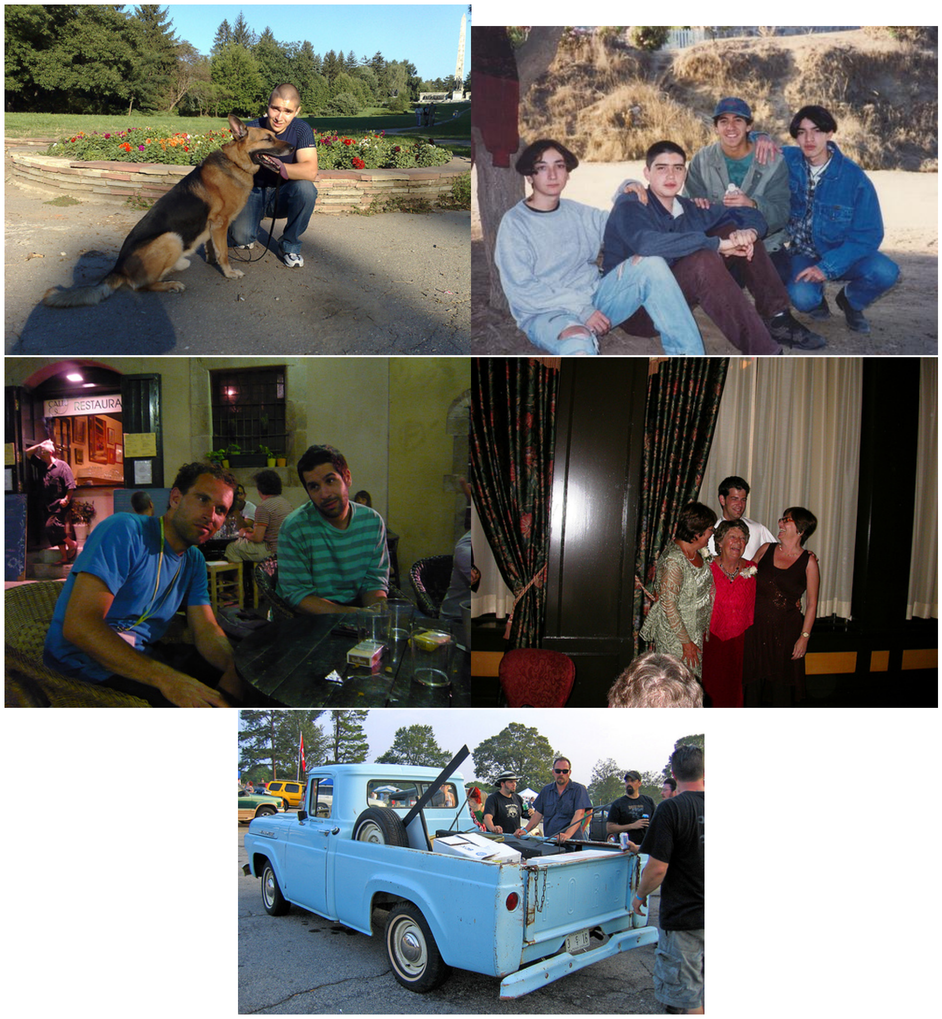
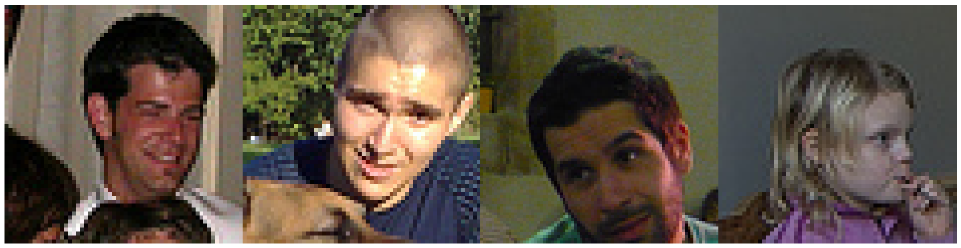
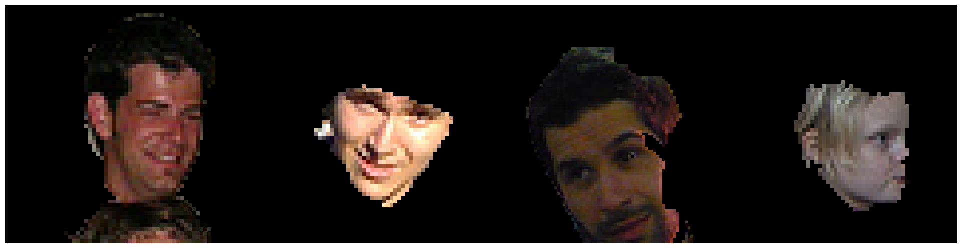


Figure 3.24: Images used to test the face/head segmentation module


(a) Head regions detected by the other modules


(b) Region segmented using the Grabcut algorithm

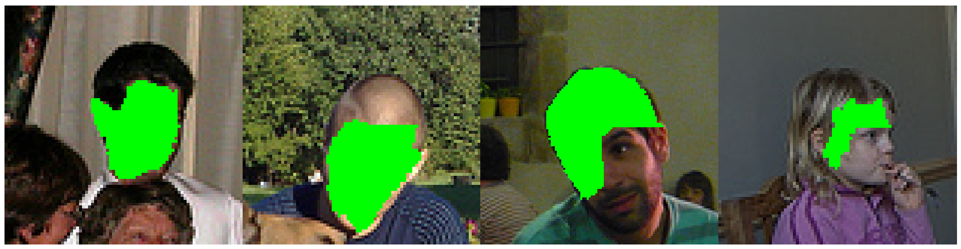
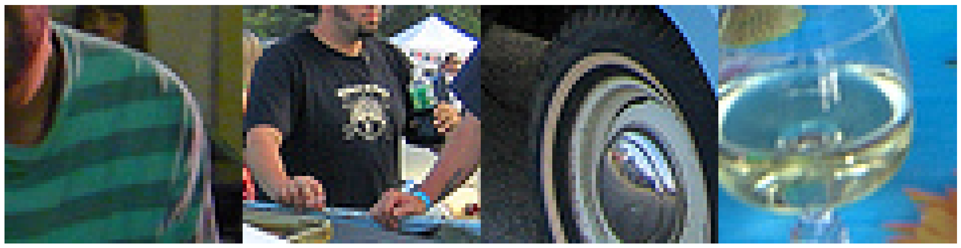
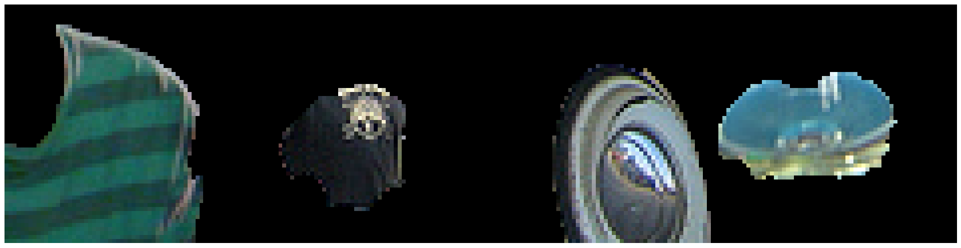

(c) Region segmented using fixation and graphcuts

Figure 3.25: Examples of results obtained from regions containing a human head


(a) Head regions detected by the other modules


(b) Region segmented using the Grabcut algorithm

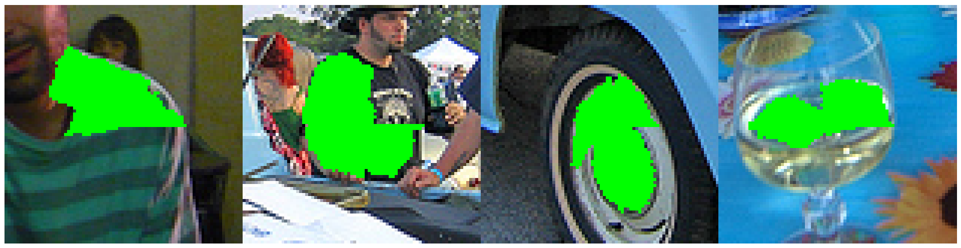

(c) Region segmented using fixation and graphcuts

Figure 3.26: Examples of results obtained from regions that do not contain a human head

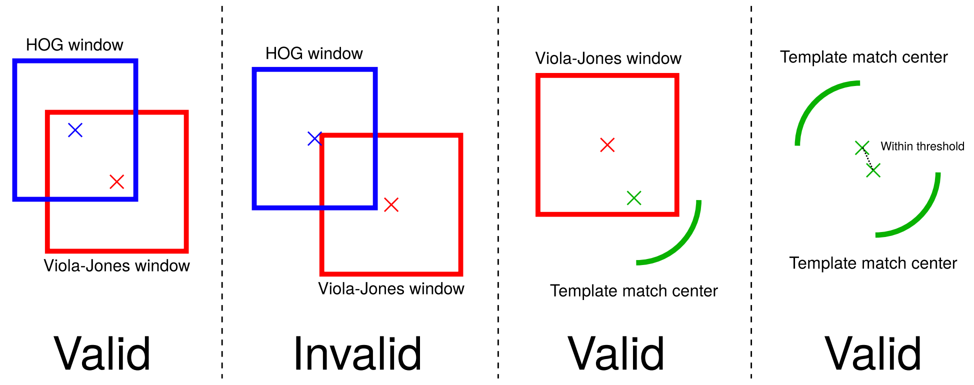


Figure 3.27: Illustration of how output from different detectors is combined into a final decision

The output from the template matching detector is a special case since two or more matches that are within a given distance of each other and have different orientations is also adequate for a region to be considered valid. Since each detector has several parameters that determine detection rate and false positive characteristics, several tests were performed with different parameters used. The results of these tests can be found in chapter 4.

3.12 Computational Performance

In order to assess whether the proposed detection system could be used for real time processing of a video feed, the computation time for each detector has been measured when processing 10 images with resolutions around 500x330. The tests were performed on a laptop PC with a Linux operating system, an Intel Core i7-3610QM at 2.30 GHz, 8 GB of system RAM and a GeForce GT650m graphics card with 1GB of dedicated RAM. The results can be seen in Table 3.7.

It is apparent that the majority of time is spent in the template matching module. The actual template matching takes more than 8 seconds per image on average. This part of the module has been implemented in C++ to run on the CPU, however the implementation has not been optimized and uses a single thread only. There is therefore plenty of room for improving the performance of this part. The Pb contour detector uses more than 5 seconds on average to process the image. Since this detector has been implemented to run on the GPU, it is unlikely that this part can be optimized much further and it is therefore a serious bottleneck in the processing pipeline. Lowering the image resolution could improve performance, but may hurt the detection rate.

function	mean (s)	std dev (s)
HOG detector	**0.12**	**0.0089**
detection	0.12	0.0092
window trimming	0.0023	0.0014
Viola-Jones detector	**0.077**	**0.0064**
Template match detector	**13.85**	**0.69**
Pb contour generation	5.31	0.17
Memory allocation	0.06	0.018
Matching	8.27	0.52
Finding minima	0.19	0.011
Validation	0.024	0.014

Table 3.7: Computation times obtained by running the detectors on 10 test images. The table shows the average over all 10 images and the standard deviation. The total average time per image for each detector are written in bold. Some of the functions are implementation specific and included for completeness only.

The HOG and Viola-Jones detectors use native implementations and are thus very efficient. The tests indicate that these detector can perform image analysis in close to real time. It is very likely that their performance could be improved even further if implemented to run on the GPU instead of the CPU. In conclusion, it is possible that the detection system could be made to run in real time if a different edge detector is used, or if it is possible to optimize the Pb implementation. However, the platform would have to have the processing power of a modern laptop PC.

3.13 Chapter Summary and Conclusion

This chapter has described how a human detection system implemented on a robotic platform fits into the larger picture of a disaster scenario. Since it is unlikely that a system can be created that is better than a human at spotting victims in the camera feed, a human operator still needs to observe the output from the robot. However, with the aid of a human detection system on-board the robot, an operator may be able to observe the progress of more than one robot at the same time.

4 detection modules have been proposed for solving the computer vision task of detecting human victims in an image obtained from the robot camera: A human head detector based on the HOG descriptor and an SVM classifier, a face detector based on the Viola-Jones object detection framework, a head detector based on template matching with quarter circle templates and finally a module to analyze the contour of an object to determine whether it has has the shape of a human head. The 3 detectors have been implemented and tested in software. The contour analyzer could not be made to work within the time frame of the project.

All detectors achieve detection rates above 50 % but the HOG and template match detectors suffer from a large number of false positives. These performance figures are largely dependent on the training and parameter choices of the different detectors.

The next chapter explains how the combined detection system was tested and shows the results that were obtained.

Chapter 4

Tests, Results and Discussion

This chapter explains how the combined detection system has been tested in order to assess its ability to detect humans in a wide variety of images. The HOG, Viola-Jones and shape matching detectors have been combined as described in the previous chapter. Two image data sets were used for testing. The first consists of images from the VOC2012 database [54] and is the same set that was used for testing the individual detectors. A few examples are shown in figure 4.1. This set contains images of humans in different environments and lighting conditions. While this set is useful due to the great variety of the images, it does not have much similarity to a real disaster scenario. The image resolutions varies but is generally around 500x300.

A more realistic image data set has been produced at the UNSW robotics lab and has been used with permission for testing the system described in this thesis. The images were captured in an indoor area that was built to simulate a disaster environment such as a building in the wake of an earthquake. The images feature several persons in various poses that are sometimes covered by objects in the scene. The image set has been extracted from a continuous video sequence and consists of more then 2000 images. A smaller subset has been used that was randomly selected from the larger set. The fact that a video sequence has been used means that many images in the set are similar. The images in this set all have a resolution of 320x240. Examples from the data set can be seen in figure 4.2.

A final set was selected from the VOC2012 database to use solely in the final test. Many of humans in the selected images were considered to be difficult for the system to detect. Examples of these are shown in figure 4.3.

4.1 Tests and Results

Finding optimal parameters and training data for the individual detectors turned out to be difficult, so a number of tests were performed on both data sets using

Figure 4.1: Examples from the VOC2012 image set used for testing the combined detection system

different combinations of parameters and training data. The results are shown in the following sections, first for the VOC2012 test set that has been used throughout the thesis, then for the VOC2012 validation set and finally the test set created in the robotics lab at UNSW. The majority of tests were performed on the VOC2012 test set. The best combination and settings with the best performance was then used on the remaining two datasets.

4.1.1 VOC2012 Test Set

The combinations used for this test set are shown in the following table along with the results

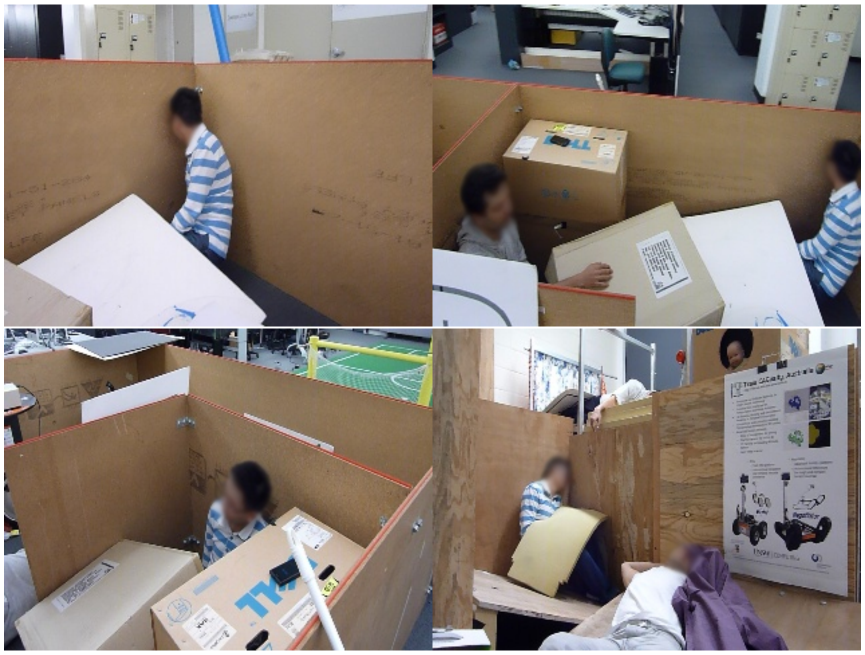


Figure 4.2: Examples from the image set produced at the robotics lab at UNSW. Faces have been blurred by request of the author.

#	HOG	Template Match	Viola Jones	Det. rate	TP/FP
1	G + IW || B + V + U	Thrsh : 0.5	Std	89 %	0.14
2	G + IW || B + V + U	Thrsh : 0.5, proximity	Std	55 %	1.28
3	G + IW || B + V + U + MN	Thrsh : 0.5	Std	55 %	0.93
4	G + IW || B + V + U + MN	Thrsh : 0.5, proximity vote	Std	62 %	0.41

The threshold for the template match detector was constant for all tests. The proximity criteria was employed in two ways: both as a requirement for template matches to be considered valid and also just as an additional vote (meaning 2 votes from the template matching detector instead of 1). It can be seen that there is still a clear trade-off between detection rate and TP/FP ratio. However, the combined detectors generally performed better than any single detector. In terms of detection rate the configuration in test 4 fared best and also achieved a reasonable TP/FP ratio. This configuration was therefore used with the remaining data sets

4.1.2 VOC2012 Validation Set

Figure 4.3: Examples from the image set produced at the robotics lab at UNSW. Faces have been blurred by request of the author.

HOG	Template Match	Viola Jones	Det. rate	TP/FP
G + IW || B + V + U + MN	Thrsh : 0.5, proximity vote	Std	40 %	0.52

With this data set, the detection system did not perform as well as on the test set. This is not too surprising since the humans in the images are believed to be more difficult to detect compared to the test set.

4.1.3 Robotics Lab Set

HOG	Template Match	Viola Jones	Det. rate	TP/FP
G + IW || B + V + U + MN	Thrsh : 0.5, proximity vote	Std	14 %	1.02
G + IW || B + V + U + MN	-	-	27 %	0.37

The detection system did not perform well on the robotics lab test set when all detectors were combined. When used alone, the HOG detector achieved a higher detection rate than the combined system. This large gap in performance between this set and the others may be due to the smaller resolution of the robotics lab set. Unfortunately it wasn't possible to obtain a version of the set with larger resolution images to verify this.

4.2 Discussion

The results obtained from the tests described in this chapter have been mixed. In the test set used throughout the thesis, the combined detection system performed better than any single detector and achieved a detection rate of 62 % with a TP/FP ratio of 0.41. In the validation set the detection rate dropped to 40 % but with a slightly higher TP/FP ratio of 0.52. The detection system did not work well with the robotics lab test set, yielding a detection rate of only 14 % with a TP/FP ratio of 1.02. On this data set the HOG detector performs better in isolation. It is not easy to predict how well the detector would work in an actual disaster scenario, however, the tests have shown that under certain conditions, the system is well capable of detecting humans with only a reasonable number of false negatives.

Chapter 5

Conclusion and Perspectives

This thesis has looked into the historical background and current state of disaster management in various countries. Countries such as the United States and Japan have a high frequency of natural disaster and have therefore developed systems and structures for efficient management and response handling. Technology plays a large role in forecasting natural disaster and communicating this information efficiently. Search and rescue operations where rescue workers attempt to find and save the lives of victims of the disaster is largely carried out by humans with use of low-tech tools. Robots have been employed in some cases such as in the aftermath of the World Trade Center attacks in 2001, however, their usefulness has so far been limited. Among the many problems that have been reported, some could be addressed by making the robots capable of autonomous operation.

Autonomous robots for use in disaster response is a research field that has received some attention. The focus is generally spread out over all aspects of the robotic platform with an incline towards the movement and mapping aspects. Several computer vision methods have been proposed for detecting victims from the camera of a robotic platform and these methods all have roots in the more general field of automatic human detection. This thesis has provided a survey of some of these methods and discussed the applicability in detection humans in a disaster scenario.

A victim disaster system will be a part of a larger system and some thoughts have been given on how the system could fit into a larger context. With present technology it is not realistic that a robot will be able to detect disaster victims with an accuracy that is on par with a human operator. A feasible solution would therefore be to combine the best of both worlds and have the detection system work as a visual aid to the human operator by analyzing the video feed and indicating when a possible human victim is detected.

The human detection system that has been implemented for this thesis consists of 4 modules that try to detect the human head or face. This choice was made since the head and face are some of the most recognizable features of the human body and whose appearances are not affected too much by body pose or articulation. A detector has been implemented that is uses HOG descriptors and an SVM classifier to determine if a region of the input image contains a head. The detection rate depends on which data sets are used for training and there is a trade-off between detection rate and the number of false positives. The performance has been gauged by the detection rate, which is the percentage of all the labeled heads in the image set that were correctly detected and TP/FP ratio which is the number of true positives divided by the number of false positives. A high TP/FP ratio means few false positives compared to true positives. The detection rate of the HOG detector ranges from 34 % with a TP/FP ratio of 0.81 to 81 % with a TP/FP ratio of 0.08.

Another detector has been implemented that uses the Viola-Jones object detection framework to detect human faces. This module uses the standard OpenCV implementation with a pre-trained classifier and achieves 44 % detection rate with a TP/FP ratio of 3.97.

A detector has been implemented that searches for quarter circle shapes in an edge map generated from the input images. Several circular shapes were tested such as half-circles and ellipses and the quarter circle was found to give the best results. Using this shape it was possible to obtain a detection rate up to 90 %. However, this is at the cost of a very large number of false positives. In its present form the template matching detector is therefore not well suited to work on its own.

The last detector was intended to verify if an area in the image detected by one of the other detectors actually contains a human head. Several methods were tested for this purpose but it was not possible to develop this module to a point where it was useful within the time frame of the project.

It is very likely that the detection system could be modified and optimized to process images in real time on a modern platform. The largest obstacle in fulfilling this goal has been the use of the Pb detector for generating edge maps. This method is computationally expensive even when implemented to run on the GPU. By using a different edge detector it is believed that the system could be made to run in, or close to real time.

By combining the 3 detectors it was possible to achieve better results on the test set than by using any single detector. On this set the detection rate was 61 % with a TP/FP ratio of 0.41. On the more difficult validation set, the detection system only achieved a detection rate of 40 % with a TP/FP ratio of 0.52. Finally, on a simulated disaster image set the detector performed poorly with only 14 % detection rate and a TP/FP ratio of 1.02.

The tests show that the detection system does a descent job of detecting humans by their head with a reasonable number of false positives under some circumstances. This is not surprising since humans can appears in endless ways in images. Even with a detection rate of 50 %, the system could be of help to a human and the system has thus shown a lot of promise. Since there are numerous parameters and variation within each detector, it is likely that the results could be improved further with more tuning.

5.1 Perspectives

There are many ways in which the detection system described in this thesis could improved and expanded. First of all, the implementation could be optimized to make it possible to process images in real time. Furthermore, better performance could probably be obtained from each detector by further tuning of parameters and choice of training data. The HOG and Viola-Jones detectors could be modified to search for heads or faces at different orientations.

The detection system could also be expanded to look for other parts of the body than the head region or search for the head and shoulders together. This would probably require the use of more sophisticated methods than those employed in the present system such as part-based models of the human body. The contour analysis module was originally intended to be used to determine the extent of the body and several methods were reviewed that showed some promise.

This project has dealt only with the software aspects of a detection system. Obviously a human detection system is not of much use if it can only run by itself on a PC. However, since the video feed from a disaster robot will be transmitted to an operator anyway, it is not necessarily a requirement that the processing take place on the robotic platform itself. The system could work very well as an overlay on the video feed that is shown to the operator. Of course this would limit the possible use of the information obtained from the analysis since the robot itself would have no knowledge of potential victims (unless two-way communication is established). A compromise would thus have to be made between complexity of the system and flexibility in terms of using the information.

Bibliography

[1] M. Ohkubo. online. [Online]. Available: http://www.flickr.com/photos/mah_japan/4478778647/in/photostream/

[2] A selective history of emergency management. Internet. National Emergency Management Association. [Online]. Available: http://www.norman.noaa.gov/NSWW2008/Ashwood.pdf

[3] C. Fisher. (2004) Rebuilding after the great fire of london. Internet. aalinfo-about.com. [Online]. Available: http://london.allinfo-about.com/features/rebuilding.html

[4] T. Saito. Disaster management of local government in japan. [Online]. Available: http://www.hyogo.uncrd.or.jp/hesi/pdf/peru/saito.pdf

[5] (2011, Feb) Disaster management in japan. Internet. Cabinet Office, Government Of Japan. [Online]. Available: http://www.bousai.go.jp/1info/pdf/saigaipanf_e.pdf

[6] (2012, Nov) Hurricane sandy, one of costliest natural disasters. Internet. Fox 19. [Online]. Available: http://www.fox19.com/story/19979510/hurricane-sandy-one-of-costliest-natural-disasters-in-us-history

[7] (2010) Japan meteorological agency. Internet. Japan Meteorological Agency. [Online]. Available: http://www.jma.go.jp/jma/en/Activities/brochure201003.pdf

[8] (2012) National hurricane center website. National Hurrican Center. [Online]. Available: http://www.nhc.noaa.gov/

[9] *INSARAG Preparedness - Response*, Office For The Coordination Of Humanitarian Affairs, Apr 2012. [Online]. Available: http://www.un.org/en/ecosoc/julyhls/pdf12/has_insarag_side_event_concept_note.pdf

[10] (2006, Jan) White paper on emergency communications. Internet. Space & Advanced Communications Research Institute. [Online]. Available: http://spacejournal.ohio.edu/issue10/PDF/Final_Version_White_Paper.pdf

[11] About sine. Internet. Center for Beredskabskommunikation. [Online]. Available: http://www.sikkerhedsnet.dk/about-sine/who-are-the-users/

[12] J. Casper and R. R. Murphy, "Human-robot interactions during the robot-assisted urban search and rescue response at the world trade center," *IEEE Transactions on Systems Man and Cybernetics*, vol. 33, pp. 367–385, 2003.

[13] Solen, mine clearing vehicle. Internet. Army Guide. [Online]. Available: http://www.army-guide.com/eng/product1796.html

[14] What's irs. Internet. NPO International Rescue System Institute. [Online]. Available: http://www.rescuesystem.org/IRSweb/en/whatsIRS.html

[15] F. Matsuno and S. Tadokoro, "Rescue robots and systems in japan," in *Proceedings of the 2004 IEEE International Conference on Robotics and Biomimetics*. IEEE, Aug 2004, pp. 12–20.

[16] K. Richardson. (2011, Apr) Rescue robots - where were they in the japanese quake relief efforts? Internet. The Institution of Engineering and Technology. [Online]. Available: http://eandt.theiet.org/magazine/2011/04/rescue-robots.cfm

[17] L. Greenemeier. (2011, Mar) Robots arrive at fukushima nuclear site with unclear mission. Internet. Scientific America. [Online]. Available: http://www.scientificamerican.com/article.cfm?id=robots-arrive-fukushima-nuclear

[18] (2011, Apr) Nasa robots probe japan's fukushima nuclear reactor. Internet. Daily Galaxy. [Online]. Available: http://www.dailygalaxy.com/my_weblog/2011/04/-packbot-code-red-duty-nasa-robots-probe-japans-nuclear-reactor.html

[19] Robocup rescue. The Robocup Federation. [Online]. Available: http://www.robocup.org/robocup-rescue/

[20] S. Balakirsky, S. Carpin, A. Kleiner, M. Lewis, A. Visser, J. Wang, and V. A. Ziparo, "Towards heterogeneous robot teams for disaster mitigation: Results and performance metrics from robocup rescue," *Journal of Field Robotics*, vol. 24, no. 11-12, pp. 943–967, November-December 2007.

[21] S. Carpin, M. Lewis, J. Wang, S. Balakirsky, and C. Scrapper, *RoboCup 2006: Robot Soccer World Cup X*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2007, ch. Bridging the Gap Between Simulation and Reality in Urban Search and Rescue, pp. 1–12.

[22] C. Y. Wong, G. Seet, S. K. Sim, and W. C. Pang, "A framework for area coverage and the visual search for victims in usar with a mobile robot," in *Sustainable Utilization and Development in Engineering and Technology (STUDENT), 2010 IEEE Conference on*, nov. 2010, pp. 112 –118.

[23] R. Hahn, D. Lang, M. Haselich, and D. Paulus, “Heat mapping for improved victim detection,” in *Safety, Security, and Rescue Robotics (SSRR), 2011 IEEE International Symposium on*, nov. 2011, pp. 116 –121.

[24] M. Aziz and B. Mertsching, “Survivor search with autonomous ugvs using multimodal overt attention,” in *Safety Security and Rescue Robotics (SSRR), 2010 IEEE International Workshop on*, july 2010, pp. 1 –6.

[25] J. Meyer, P. Schnitzspan, S. Kohlbrecher, K. Petersen, M. Andriluka, O. Schwahn, U. Klingauf, S. Roth, B. Schiele, and O. Stryk, “A semantic world model for urban search and rescue based on heterogeneous sensors,” in *RoboCup 2010: Robot Soccer World Cup XIV*, ser. Lecture Notes in Computer Science, J. Ruiz-del Solar, E. Chown, and P. PlÃ¶ger, Eds. Springer Berlin Heidelberg, 2011, vol. 6556, pp. 180–193.

[26] G. Liu, H. Tong, and R. Zhang, “An intelligent control architecture for search robot based on orthogonal perception information,” in *Fuzzy Systems and Knowledge Discovery (FSKD), 2012 9th International Conference on*, may 2012, pp. 2348 –2352.

[27] M. Andriluka, P. Schnitzspan, J. Meyer, S. Kohlbrecher, K. Petersen, O. von Stryk, S. Roth, and B. Schiele, “Vision based victim detection from unmanned aerial vehicles,” in *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, oct. 2010, pp. 1740 –1747.

[28] B. Soni and A. Sowmya, “Classifier ensemble with incremental learning for disaster victim detection,” in *Robotics and Biomimetics (ROBIO), 2012 IEEE International Conference on*, 2012, pp. 446–451.

[29] M. Gerdzhev, J. Tran, A. Ferworn, K. Barnum, and M. Dolderman, “A scrubbing technique for the automatic detection of victims in urban search and rescue video,” in *Proceedings of the 6th International Wireless Communications and Mobile Computing Conference*, ser. IWCMC ’10. New York, NY, USA: ACM, 2010, pp. 779–783.

[30] R. Shamroukh and F. Awad, “Detection of surviving humans in destructed environments using a simulated autonomous robot,” in *Mechatronics and its Applications, 2009. ISMA ’09. 6th International Symposium on*, march 2009, pp. 1 –6.

[31] S. Bahadori and L. Iocchi, “Human body detection in the robocup rescue scenario,” in *CDROM Proc. Int. RoboCup Symposium*, vol. 3. Citeseer, 2003.

[32] C. Castillo and C. Chang, “A method to detect victims in search and rescue operations using template matching,” in *Safety, Security and Rescue Robotics, Workshop, 2005 IEEE International*, june 2005, pp. 201 –206.

[33] N. Viertl, C. Beleznai, and J. Birchbauer, “A pedestrian detection system combining motion detection, spatial grouping and classification,” in *Image and Signal Processing and Analysis, 2009. ISPA 2009. Proceedings of 6th International Symposium on*, 2009, pp. 194–199.

[34] R. Feraund, O. Bernier, J.-E. Viallet, and M. Collobert, “A fast and accurate face detector based on neural networks,” *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 23, no. 1, pp. 42–53, 2001.

[35] Y. Li, A. Goshtasby, and O. Garcia, “Detecting and tracking human faces in videos,” in *Pattern Recognition, 2000. Proceedings. 15th International Conference on*, vol. 1, 2000, pp. 807–810 vol.1.

[36] A. Mishra, Y. Aloimonos, L. F. Cheong, and A. Kassim, “Active visual segmentation,” *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 34, no. 4, pp. 639 –653, april 2012.

[37] L. Tao and H. bin Wang, “Detecting and locating human eyes in face images based on progressive thresholding,” in *Robotics and Biomimetics, 2007. ROBIO 2007. IEEE International Conference on*, 2007, pp. 445–449.

[38] K. Bhuvaneswari and H. Abdul Rauf, “Edgelet based human detection and tracking by combined segmentation and soft decision,” in *Control, Automation, Communication and Energy Conservation, 2009. INCACEC 2009. 2009 International Conference on*, 2009, pp. 1–6.

[39] S. Jia, S. Wang, L. Wang, and X. Li, “Robust human detecting and tracking using varying scale template matching,” in *Information and Automation (ICIA), 2012 International Conference on*, 2012, pp. 25–30.

[40] F. He, Y. Li, S. Wang, and X. Ding, “A novel hierarchical framework for human head-shoulder detection,” in *Image and Signal Processing (CISP), 2011 4th International Congress on*, vol. 3, 2011, pp. 1485–1489.

[41] Z. Lin, L. Davis, D. Doermann, and D. DeMenthon, “Hierarchical part-template matching for human detection and segmentation,” in *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, 2007, pp. 1–8.

[42] D. T. Nguyen, P. Ogunbona, and W. Li, “Detecting humans under occlusion using variational mean field method,” in *Image Processing (ICIP), 2011 18th IEEE International Conference on*, 2011, pp. 2049–2052.

[43] N. Dalal and B. Triggs, “Histograms of oriented gradients for human detection,” in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 1, june 2005, pp. 886 –893 vol. 1.

[44] C. Li, L. Guo, and Y. Hu, “A new method combining hog and kalman filter for video-based human detection and tracking,” in *Image and Signal Processing (CISP), 2010 3rd International Congress on*, vol. 1, 2010, pp. 290–293.

[45] M. J. J. Paul Viola, “Robust real-time face detection,” *International Journal of Computer Vision*, vol. 57, no. 2, pp. 137–154, May 2004.

[46] C. Huang, H. Ai, Y. Li, and S. Lao, "High-performance rotation invariant multiview face detection," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 29, no. 4, pp. 671–686, 2007.

[47] K. Shi, S. Pang, and F. Yu, "A real-time face detection and recognition system," in *Consumer Electronics, Communications and Networks (CECNet), 2012 2nd International Conference on*, 2012, pp. 3074–3077.

[48] T. H. Le and L. T. Bui, "An approach to combine adaboost and artificial neural network for detecting human faces," in *Systems, Man and Cybernetics, 2008. SMC 2008. IEEE International Conference on*, 2008, pp. 3411–3416.

[49] [Online]. Available: http://pascallin.ecs.soton.ac.uk/challenges/VOC/voc2012/

[50] F.Tarrés and A. Rama. Gtav face database. Internet. [Online]. Available: http://gps-tsc.upc.es/GTAV/ResearchAreas/UPCFaceDatabase/GTAVFaceDatabase.htm

[51] Psychological image collection at stirling (pics). Internet. University of Stirling. [Online]. Available: http://pics.stir.ac.uk

[52] D. Martin, C. Fowlkes, D. Tal, and J. Malik, "A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics," in *Proc. 8th Int'l Conf. Computer Vision*, vol. 2, July 2001, pp. 416–423. [Online]. Available: http://www.eecs.berkeley.edu/Research/Projects/CS/vision/bsds/

[53] Urtho's training data for eye detection using opencv haarcascade. Internet. [Online]. Available: http://face.urtho.net/

[54] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results," http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html.

[55] M.-Y. Liu, O. Tuzel, A. Veeraraghavan, and R. Chellappa, "Fast directional chamfer matching," in *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, 2010, pp. 1696–1703.

[56] T. Fletcher. (2009) Support vector machines explained.

[57] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *Journal of computer and system sciences*, vol. 55, no. 1, pp. 119–139, 1997.

[58] J. Canny, "A computational approach to edge detection," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. PAMI-8, no. 6, pp. 679–698, 1986.

[59] D. Martin, C. Fowlkes, and J. Malik, “Learning to detect natural image boundaries using local brightness, color, and texture cues,” *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 26, no. 5, pp. 530 –549, may 2004.

[60] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik, “Contour detection and hierarchical image segmentation,” *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 33, no. 5, pp. 898 –916, may 2011.

[61] P. Arbelaez, “Boundary extraction in natural images using ultrametric contour maps,” in *Computer Vision and Pattern Recognition Workshop, 2006. CVPRW ’06. Conference on*, june 2006, p. 182.

[62] Y. Boykov and G. Funka-Lea, “Graph cuts and efficient n-d image segmentation,” *International Journal of Computer Vision*, vol. 70, pp. 109–131, 2006. [Online]. Available: http://dx.doi.org/10.1007/s11263-006-7934-5

[63] Y. Boykov and V. Kolmogorov, “An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision,” *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 26, no. 9, pp. 1124 –1137, sept. 2004.

[64] C. Rother, V. Kolmogorov, and A. Blake, “"grabcut": interactive foreground extraction using iterated graph cuts,” *ACM Trans. Graph.*, vol. 23, no. 3, pp. 309–314, Aug. 2004. [Online]. Available: http://doi.acm.org/10.1145/1015706.1015720

Appendix A

Theoretical Background

This chapter describes the theoretical background for the system that has been designed for this thesis. The descriptions given here are general and not necessarily focused on the problem of detecting humans in images. Chapter 3 describes how these methods have been used for human detection.

The first two sections describe the HOG descriptor and Viola-Jones object detection framework. Both methods have been widely used for detecting different objects in images. Next, an overview of edge detection techniques is given. Edge detection plays an important role in the template matching and contour analysis modules in the system. Template matching is described next and finally segmentation via graph cuts and the GrabCut method.

A.1 The HOG Descriptor

Histogram of Oriented Gradients is a widely used feature descriptor that has proven to be very efficient for classification of objects in images. In the original paper [43], the authors used HOG descriptors and a Support Vector Machine classifier to perform fast detection of humans.

The HOG descriptor is computed by splitting the input image into smaller cells, usually of size 8x8 pixels. The process is illustrated in Figure A.1. The gradient orientations in the cell are calculated and placed in bins that cover different ranges of angles, e.g. $0-20°$ for the first bin, $20-40°$ for the second, and so on. Each pixel in the cell has a number of votes that depend on the magnitude of their gradient. The votes can either be placed in one bin or distributed into two using bilinear interpolation.

To make the representation more robust to changes in lighting, cell histograms are normalized w.r.t. other cells and grouped into blocks. Blocks are normally made up of 4 cells and can have overlapping cells between them. The final descriptor

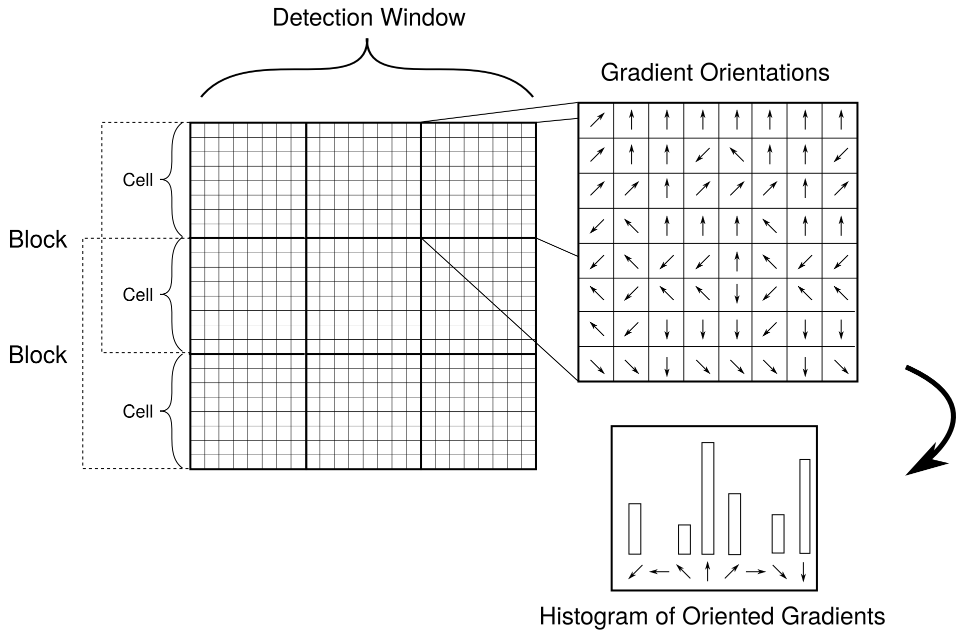


Figure A.1: Illustration of how cells and blocks of oriented gradients are calculated from the image. Each cell is represented as a histogram.

is a vector composed of all blocks computed from the detection window. The total number of features extracted from one detection window can be calculated as follows:

$$\begin{aligned} \text{nFeatures} &= \text{floor}\left(\frac{\text{Win Height}}{\text{Block Stride}} - \left(\frac{\text{Block Size}}{\text{Cell Size}} - 1\right)\right) \\ &\quad \cdot \text{floor}\left(\frac{\text{Win Width}}{\text{Block Stride}} - \left(\frac{\text{Block Size}}{\text{Cell Size}} - 1\right)\right) \\ &\quad \cdot \frac{(\text{Block Size})^2}{(\text{Cell Size})^2} \cdot \text{nBins} \end{aligned} \tag{A.1}$$

where all sizes are specified in pixels. Blocks and cells are assumed to have square dimensions. The block size and stride have to be a multiple of the cell size.

A HOG based object detector is created by training a classifier to recognize HOG descriptors computed from a set of images. The most popular way to classify HOG descriptors is with a Support Vector Machine (SVM). The SVM is described in the following section.

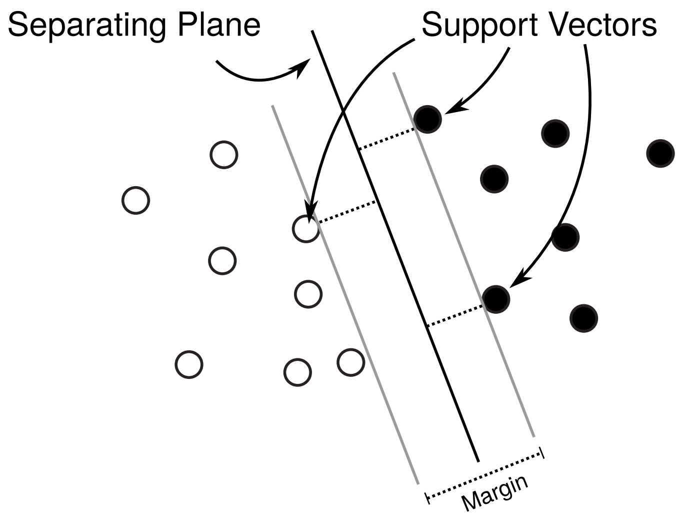


Figure A.2: Illustration of an SVM classifier that separates data points from two classes using a plane. The points that lie closest to the plane are the support vectors. The goal of training is to find an orientation of the plane that maximizes the margin between the plane and the data points.

A.1.1 Support Vector Machine (SVM) Classifier

The Support Vector Machine (SVM) is a two-class classifier that uses a number of important training data points, called support vectors, to determine if a new point belongs to one class or the other. The SVM classifier has been explained in detail [56]. In it's most basic form, the classifier is linear, meaning that it uses a hyperplane to separate the two classes. A plane in an n-dimensional space is defined as follows:

$$\boldsymbol{x}_i \cdot \boldsymbol{w} + b = 0 \tag{A.2}$$

Where

$\boldsymbol{x}_i$	is a point with n coordinates on the plane
$\boldsymbol{w}$	is a vector with n elements that is orthogonal to the plane
b	is a constant

The sign of (A.2) will indicate on which side of hyperplane $\boldsymbol{x}_i$ lies, i.e which class it belongs to. By defining $\mathbf{x} = [\boldsymbol{x}_0, \boldsymbol{x}_1, ..., \boldsymbol{x}_n]$ and $y_i \in \{-1, +1\}$ as the class of the data point $\boldsymbol{x}_i$, the goal is to determine $\boldsymbol{w}$ and b such that:

$$\begin{aligned} x_i \cdot \boldsymbol{w} + b &\geq +1 \text{ , } y_i = +1 \\ x_i \cdot \boldsymbol{w} + b &\leq -1 \text{ , } y_i = -1 \end{aligned}$$

These two expressions can be combined into one:

$$y_i\left(\boldsymbol{x}_i \cdot \boldsymbol{w} + b\right) - 1 \geq 0 \forall_i$$

For the points that lie closest to the hyperplane, the following must therefore hold:

$$\begin{aligned} x_i \cdot \boldsymbol{w} + b &= +1 \;,\; y_i = +1 \\ x_i \cdot \boldsymbol{w} + b &= -1 \;,\; y_i = -1 \end{aligned}$$

The points that satisfy this equation are called support vectors as illustrated in Figure A.2. It can be shown that obtaining the largest margin between the hyperplane and the closest point from either class is achieved by minimizing the expression

$$\min \frac{1}{2}\left\|\boldsymbol{w}\right\|^2 \quad \text{such that} \quad y_i\left(\boldsymbol{x}_i \cdot \boldsymbol{w} + b\right) - 1 \geq 0 \forall_i \tag{A.3}$$

Using Lagrange multipliers the objective function to be minimized can be written as

$$L_P = \frac{1}{2}\left\|\boldsymbol{w}\right\|^2 - \sum_{i=1}^{L} \alpha_i y_i \left(\boldsymbol{x}_i \cdot \boldsymbol{w} + b\right) + \sum_{i=1}^{L} \alpha_i \tag{A.4}$$

Where α_i are the Lagrange multipliers. It can be shown that by finding $\boldsymbol{w}$ and b that minimizes and $\boldsymbol{\alpha}$ that maximizes (A.4), the multipliers where $\alpha_i \geq 0$ will correspond to the support vectors that maximize the margin. Once the support vectors have been determined, the remaining training data can be discarded.

This formulation of SVM requires that the data points are linearly separable. As this is not always the case, it is useful to extend the formulation to allow classification of more complicated data distributions. This can be done by introducing so-called slack variables or by transforming the data points using kernels. For details on these extensions the reader is referred elsewhere [56].

A.2 Viola-Jones Framework for Object Detection

The Viola-Jones object detection framework [45] can be used for efficient and very fast face detection in images. The defining characteristics of the framework is that it uses Haar-like features and a cascade of boosted classifiers to detect objects in images. Detection is done on gray scaled images.

A Haar-like feature of an image is the sum of pixel values within a rectangular area. In the Viola-Jones framework the following features are used:

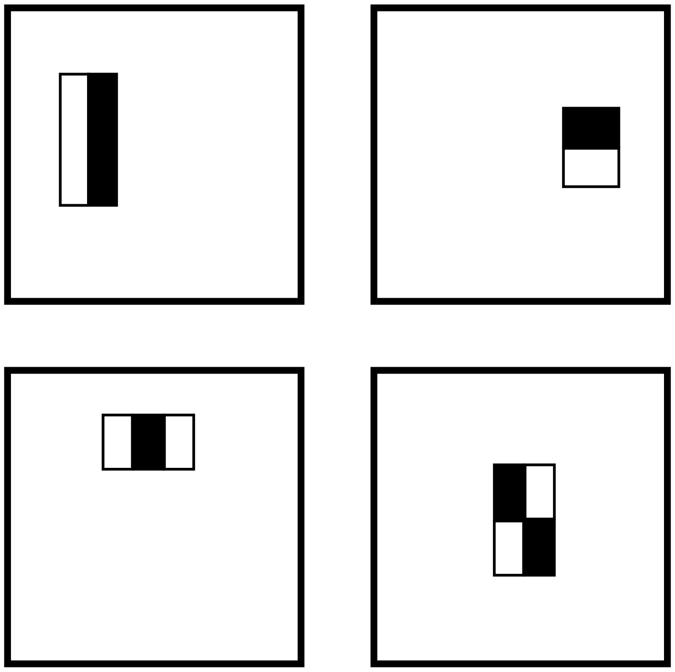


Figure A.3: Examples of Haar-like features in 4 different detection windows [45]

(i) Two-rectangle feature

(ii) Three-rectangle feature

(iii) Four-rectangle feature

Examples of Haar-like features are shown in Figure A.3. A feature is evaluated by calculating the difference between the sums of white and black areas within a detection window. To speed up the sum operation an integral image is generated from the input. It can be defined as follows:

$$ii\,(x,y) = \sum_{i=1}^{y}\sum_{j=1}^{x} I(x,y)$$

Where $I(x,y)$ is the pixel value at coordinates x and y. In practice the integral image is calculated iteratively using the following formula:

$$\begin{aligned} ii(x,y) &= I(x,y) + ii(x-1,y) + ii(x,y-1) \\ ii(x,y) &= 0,\; x \le 0,\; y \le 0 \end{aligned}$$

The entire integral image can therefore be generated in a single pass. With the integral image, calculating the sum of any rectangular area defined by upper left coordinates (x_1, y_1) and lower right (x_2, y_2) is as simple as:

$$s(x_1, y_1, x_2, y_2) = ii\,(x_2, y_2) - ii\,(x_2, y_1) - ii\,(x_1, y_2) + ii\,(x_1, y_1)$$

Evaluating a Haar-like feature is thus extremely fast and the computation time does not depend on the size of the area.

Haar-like features are useful for face detection since the face normally exhibits certain regions that are consistently darker than others such as the area around and below the eyes as well as between them. This is illustrated in Figure A.4

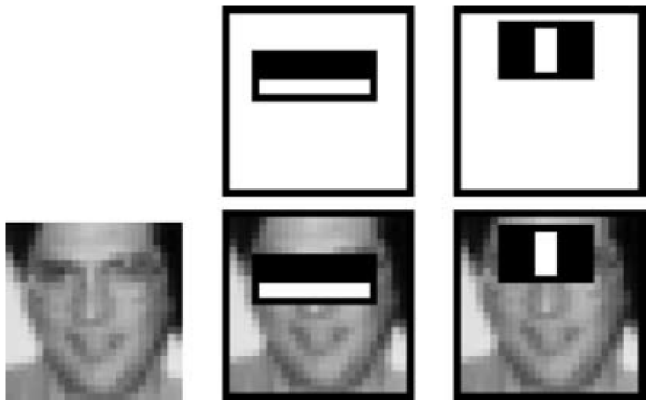


Figure A.4: Illustration of how Haar-like features are useful for detecting certain patterns visible in the human face[45]

A.2.1 Cascade of Boosted Classifiers

The goal of the Viola-Jones detection framework is to be as fast as possible at discarding negative detection windows. This is achieved by setting up a cascade of classifiers which start out simple and grow increasingly complex with each stage. The advantage of a cascaded solution is that each stage is very fast to evaluate, and since negative samples are more likely to be discarded during the first stages, the average processing time per window is low.

The premise is that a small number of features are sufficient for detecting faces and can be used to classify almost all positive training samples correctly. The features in Figure A.4 are good examples. With these features it may be possible to classify 99 % of all positive training samples correctly. However, the features may also classify a high number of negative samples incorrectly e.g. 40 %. This is not a problem since the next stage in the cascade can be trained specifically on the incorrectly classified samples, again reaching a true positives rate (TPR) of 99

% and false positives rate (FPR) of 40 % on this subset of images. At this stage the total FPR is $0.4^2 = 0.16$ while the TPR has only dropped to $0.99^2 = 0.98$. Adding more stages will continue this pattern as shown in Figure A.5. The results is a very efficient classifier with a high TPR and a low FPR.

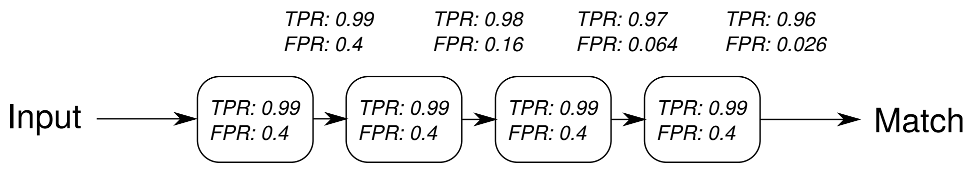


Figure A.5: Example of a cascade of classifiers. With each stage, the false positives rate (FPR) drops substantially while the true positives rate (TPR) is kept high

Each classifier stage is trained using the AdaBoost algorithm [57] and consists of several weak classifiers. A weak classifier is defined by a single feature f, a threshold θ and a polarity p. Formally it is defined as:

$$h(x, f, p, \theta) = \begin{cases} 1 & \text{if } pf(x) < p\theta \\ 0 & \text{otherwise} \end{cases}$$

where x is the detection window. The training set consists of positive and negative images of the same size as the detection window. Every image is assigned a weight w_i that is initially set to $\frac{1}{n}$ where n is the total number of training samples. The classification error is then defined by:

$$\epsilon_t = \sum_i w_i \left| h(x_i, f, p, \theta) - y_i \right| \tag{A.5}$$

where

x	is the set of training samples
y	is the set of training labels, being 0 or 1 for positive and negative samples respectively

Training a weak classifier is therefore a matter of minimizing (A.5). A single classifier will generally not be able to classify all training samples correctly. For every pass the misclassified samples are therefore given higher weights. The weights are updated as follows:

$$\begin{aligned} w_{t+1,i} &= w_{t,i}\beta_t^{1-e_i} \\ \beta_t &= \frac{\epsilon_t}{1-\epsilon_t} \\ e_i &= \begin{cases} 1 & \text{if } x_i \text{is classified correctly} \\ 0 & \text{otherwise} \end{cases} \end{aligned}$$

The update is followed by normalization to ensure a total sum of 1:

$$w_{t,i} = \frac{w_{t,i}}{\sum_{j=1}^{n} w_{t,j}}$$

As training progresses, more weight will be put on the training samples that are difficult to classify. The final decision is determined by:

$$\begin{aligned} C(x) &= \begin{cases} 1 & \sum_{t=1}^{T} \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t \\ 0 & \text{otherwise} \end{cases} \\ \alpha_t &= \log \frac{1}{\beta_t} \end{aligned}$$

Each weak classifier is thus weighted inversely to its error rate.

A.3 Edge Detection Techniques

Classifying objects in an image often involves determining their location and boundary. Since an image is essentially a 3D scene being projected onto a 2D surface it can be difficult to determine if one area of an image is a part of or connected to another area, i.e. if they are both a part of the same object. For a simple 2D image there is no way to determine depth information with absolute certainty. However, in many cases it is possible to use cues from pixel intensities to estimate the boundaries of objects in the image. This process is generally known as edge or boundary detection.

Edge detection is based on the fact that pixels intensities tend to change rapidly around object boundaries as can be seen in Figure A.6. The following sections describes the most basic method of edge detection and some of the popular, more sophisticated methods.

A.3.1 Basic Edge Detection

The simplest form of edge detection involves calculating the difference between each consecutive pixel intensity value along the abscissa and ordinate axes of the image as follows in order to approximate the image gradient:

$$\begin{aligned} \mathrm{d}I_{\text{ord}}(x,y) &= I(x,y) - I(x,y-1) \\ \mathrm{d}I_{\text{abs}}(x,y) &= I(x,y) - I(x-1,y) \end{aligned}$$

The magnitude and orientation of the gradient can then be calculated as follows:

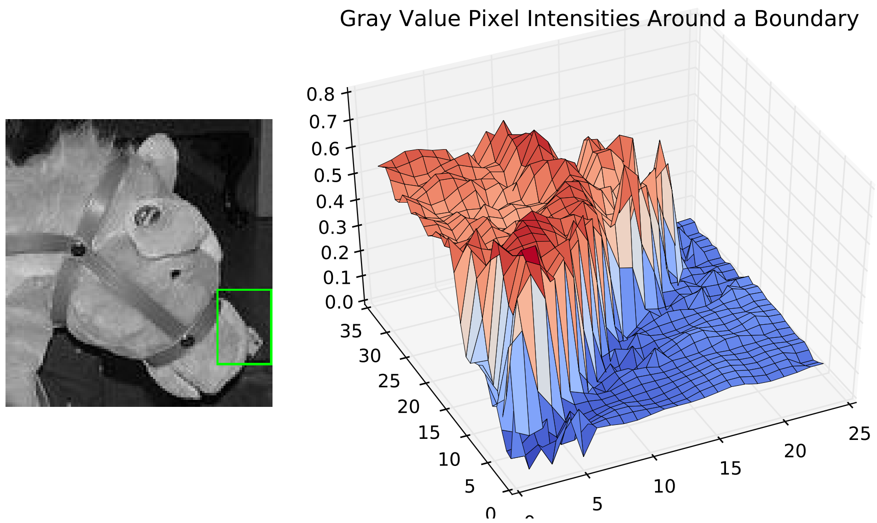


Figure A.6: Example of a large change in pixel intensities around an object boundary

$$\begin{aligned} \mathrm{d}I_{\text{mag}}(x,y) &= \sqrt{\mathrm{d}I_{\text{ord}}(x,y)^2 + \mathrm{d}I_{\text{abs}}(x,y)^2} && \text{(A.6)} \\ \mathrm{d}I_{\theta}(x,y) &= \arctan 2\left(\mathrm{d}I_{\text{ord}}(x,y),\ \mathrm{d}I_{\text{abs}}(x,y)\right) && \text{(A.7)} \end{aligned}$$

The difference operations can be expressed more compactly as convolutions with the two kernels:

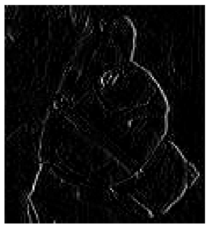
$$\begin{aligned} \mathbf{K}_{\text{ord}} &= \begin{bmatrix} 1 \\ -1 \end{bmatrix} && \text{(A.8)} \\ \mathbf{K}_{\text{abs}} &= \begin{bmatrix} 1 & -1 \end{bmatrix} && \text{(A.9)} \end{aligned}$$

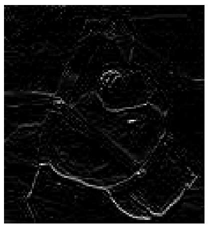
To obtain an edge map, a threshold operation is performed on the gradient magnitude. The results of these steps are shown by example in Figure A.7.

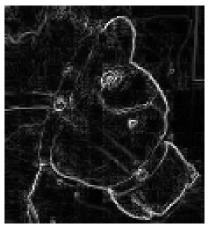
A.3.2 Sobel Operator

A better approximation of the image gradient can be obtained by using Sobel kernels:

(a) Grayscaled image


(b) Difference operation along ordinate axis


(c) Difference operation along abscissa axis


(d) Gradient magnitude

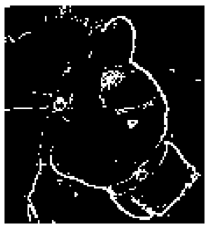

(e) Magnitude thresholded

Figure A.7: Steps involved in simple edge detection using the kernels in (A.8) and (A.9)

$$\mathbf{K}_{\text{Sob,ord}} = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \tag{A.10}$$

$$\mathbf{K}_{\text{Sob,abs}} = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \tag{A.11}$$

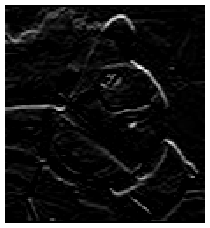
The results of using the Sobel kernels are shown in Figure A.8.

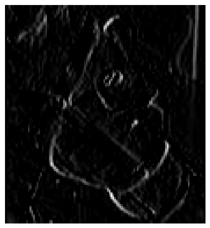
A.3.3 Canny Edge Detector

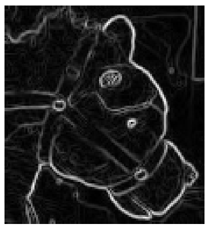
The Canny edge detector was introduced [58] in an attempt to define an optimal edge detector given the following criteria:

- Good detection. Real edges should be detected with high probability and vice versa for non-edges.
- Good localization. Detected edge points should lie at the center of the actual edge in the image.

(a) Grayscaled image


(b) Difference operation along ordinate axis


(c) Difference operation along abscissa axis


(d) Gradient magnitude

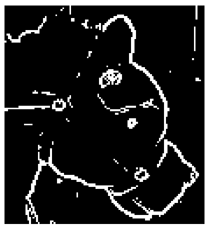

(e) Magnitude thresholded

Figure A.8: Results obtained when using the Sobel operator kernels in (A.10) and (A.11)

- A single response to one edge. The detected edge should have only a width of 1 pixel.

There exists different versions of the Canny edge detector but the one described here is from the original article. The detector is based on the 2 dimensional Gaussian filter:

$$f(x,y) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2+y^2}{2\sigma^2}\right) \tag{A.12}$$

The author finds an optimal detector for a theoretical step edge and shows that this detector can be approximated closely by a difference of Gaussians filter which is shown in Figure A.9.

Directional filters for 2 dimensions are obtained by calculating the partial derivatives of (A.12) relative to x and y:

$$f_{\mathbf{n}} = \mathbf{n} \cdot \begin{bmatrix} \frac{\partial}{\partial x} f \\ \frac{\partial}{\partial y} f \end{bmatrix}$$

Where **n** is the direction. For ordinate and abscissa axes the filters are as follows:

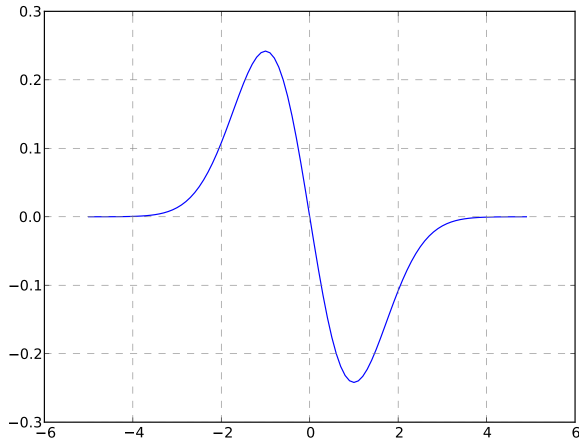


Figure A.9: Gaussian 1st order derivative with $\sigma = 1$

$$
\begin{aligned}
f_x &= -\frac{x}{2\pi\sigma^4}\exp\left(-\frac{x^2+y^2}{2\sigma^2}\right)\\
f_y &= -\frac{y}{2\pi\sigma^4}\exp\left(-\frac{x^2+y^2}{2\sigma^2}\right)
\end{aligned}
$$

In kernel form with standard deviation 1.5 the filters are as follows:

$$
\begin{aligned}
\mathbf{G_x} &= \begin{bmatrix} 1 & 1 & 1 & 0 & -1 & -1 & -1 \\ 1 & 2 & 2 & 0 & -2 & -2 & -1 \\ 1 & 3 & 3 & 0 & -3 & -3 & -1 \\ 1 & 2 & 2 & 0 & -2 & -2 & -1 \\ 1 & 1 & 1 & 0 & -1 & -1 & -1 \end{bmatrix} \frac{1}{107}\\
\mathbf{G_y} &= \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 2 & 1 \\ -1 & 2 & 3 & 2 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ -1 & -2 & -3 & -2 & -1 \\ -1 & -2 & -3 & -2 & -1 \\ -1 & -1 & -1 & -1 & -1 \end{bmatrix} \frac{1}{107}
\end{aligned}
$$

The gradient magnitude and orientations are then calculated as in (A.6) and (A.7). Figure A.10 shows the results of applying these filters. The edges are slightly more blurry than those obtained with the Sobel operator.

The next step involves locating the pixels that have the largest magnitude response within a local region. This is known as nonmaximum suppression. Formally it means satisfying the following equation:

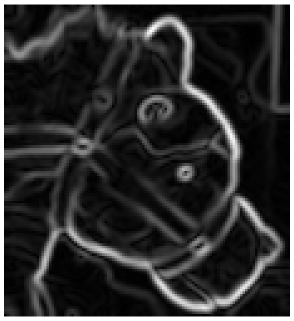


Figure A.10: Edge magnitude response when applying the Gaussian derivative filters to both axes

$$\begin{aligned}\frac{\partial}{\partial \mathbf{n}} f_{\mathbf{n}} \star I &= 0 \\ &\Downarrow \\ \frac{\partial^2}{\partial \mathbf{n}^2} f \star I &= 0\end{aligned}$$

Where **n** is the edge orientation. In words the edge map is searched along the direction of an edge for the largest magnitude response which corresponds to the 2nd order derivative being 0.

The final step involves applying a threshold operation to the edge map in order to remove edges with low magnitude responses. A main concern brought up in the paper is that simple thresholding can lead to edges that have holes in them, also known as streaking. This happens when an edge has a magnitude that lies just above the threshold and occasionally dives below it.

The proposed solution is a threshold with hysteresis. A higher bound is defined as the threshold at which an edge is detected and retained. Once an edge magnitude above this level is encountered, the edge is traced along both directions. During this trace, a lower bound is used as a threshold for when the edge magnitude response is no longer large enough for the edge to be retained.

The final result of using the Canny edge detector is shown in Figure A.11.

A.3.4 Pb Boundary Detector

The Pb Boundary Detector [59] uses a number of different cues in an image to estimate the contours of distinct objects. The cues chosen are oriented energy, intensity gradients, color gradients and texture gradients.

Oriented energy is defined as:

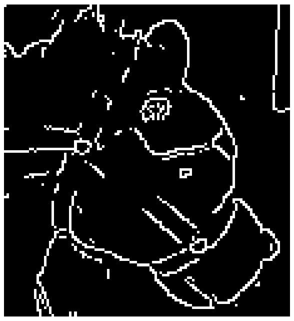


Figure A.11: Output from the Canny edge detector with thresholds 220 and 200

$$\mathrm{OE}_{\theta,\sigma} = \left(I \star f^{e}_{\theta,\sigma}\right)^{2} + \left(I \star f^{o}_{\theta,\sigma}\right)^{2} \tag{A.13}$$

The two filters $f^{e}_{\theta,\sigma}$and $f^{o}_{\theta,\sigma}$ are even and odd symmetric, second order derivative, 2D Gaussian filters. Each is defined for different orientations θ and scales σ. Both are convolved with the image and their squared responses are combined to make up the oriented energy map of the image.

All gradient based cues are calculated using a function called $G(x, y, \theta, r)$. The function selects all pixels within a radius of r from the point (x, y) and creates a histogram of values for each half disc. The two histograms are then compared to produce a response. The orientation of the disc is specified by θ. This approach is illustrated in Figure A.12.

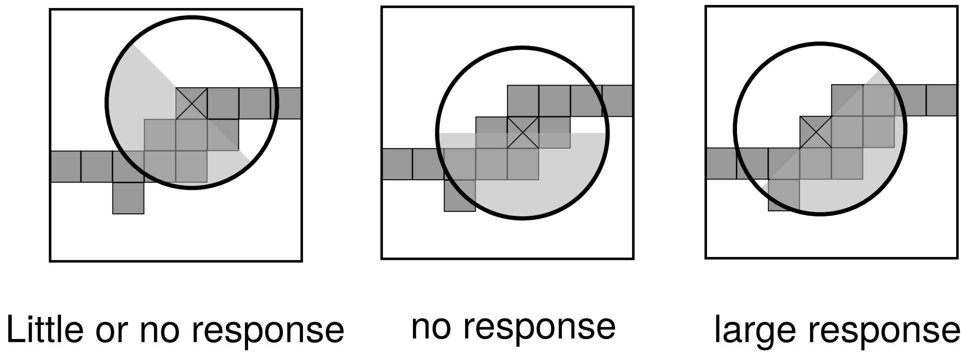


Figure A.12: Diagrams showing how gradients are calculated with the Pb Boundary Detector

To calculate texture gradients the the notion of texture elements, called *textons* is introduced. The image is convolved at every pixel point with a bank of Gaussian filters with different orientations of their axes. The filter responses are then stored in a single vector, resulting in as many vectors as there are pixels in the image.

Vectors are then clustered together using k-means and the center of each cluster is called a texton. All pixels in the image are then labeled by the cluster they belong to.

A number of experiments were performed [59] in order to find the best way to combine all cues into a final probability boundary function $\mathrm{Pb}\,(x, y, \theta)$. First parameter tuning was performed for each cue, comparing the results to ground truth contour images. Finally, a proper set of weights were determined for combining the cues together.

The output of the Pb function has been used to extend the resulting boundaries to form closed contours [60]. This is done using a variant of the watershed transform (OWT), applied to the probability boundary map while taking into account the orientation of edges to avoid certain artifacts. Furthermore the ultra metric contours method [61] is used. Figure A.13 shows the result of applying all these steps to an image.

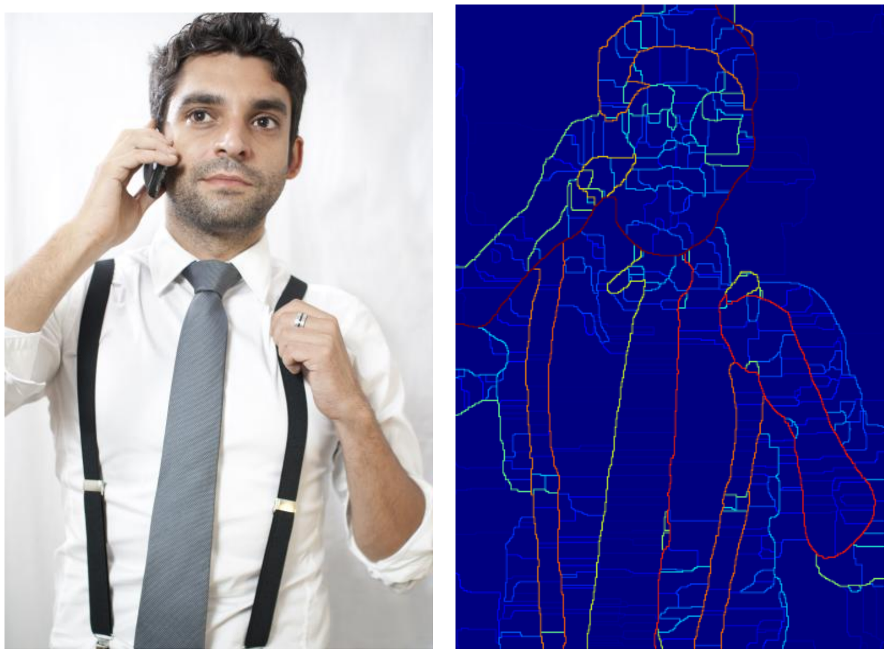


Figure A.13: Result of applying the Pb boundary detector, oriented watershed transform (OWT) and ultra metric mapping (UCM) to an image. The image on the right shows many artifacts in the form of weak contours that run in straight lines between stronger ones. OWT uses orientation information to minimize the strength of these false contours.

The result of applying the Pb detector to our test image is shown in Figure A.14.

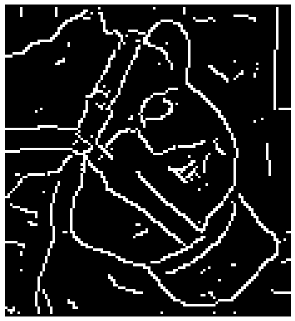


Figure A.14: Edge map obtained by applying the Pb detector

A.3.5 Comparing Edge Detectors

In order to assess the performance of each edge detector and compare them to each other, 3 images were processed. The results can be seen in Figure A.15. From the detectors that were tested and the image used the Pb detector gave the best results.

A.4 Template Matching

Template matching is a general technique where an image is searched exhaustively for occurrences of a predefined smaller image or shape, often by sliding a window across the larger image and comparing all pixels within the window with the predefined template. This is illustrated in Figure A.16. In the case where the image is an edge map the pixel pattern is binary and the template will be sparse. A simple example is:

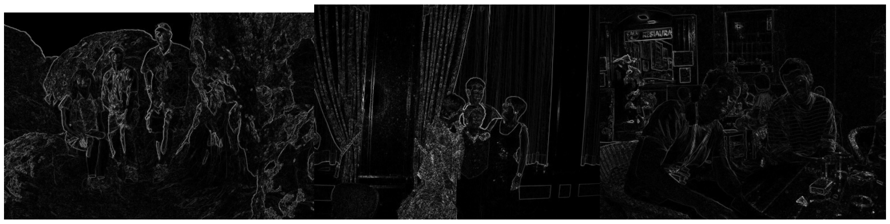
$$\mathrm{T} = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \end{bmatrix} \tag{A.14}$$

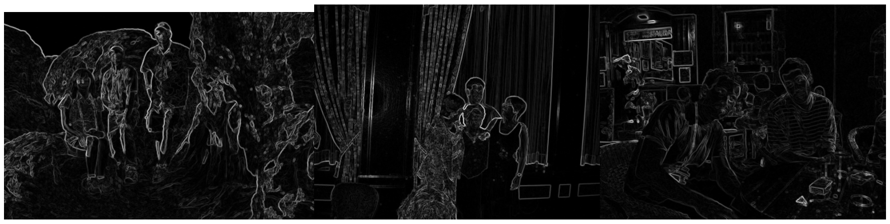
which is a template with a mirrored "S" shape. Only entries with 1 are considered when matching against a region in the image. This structure is more efficiently stored as a list of pixel coordinates, i.e:

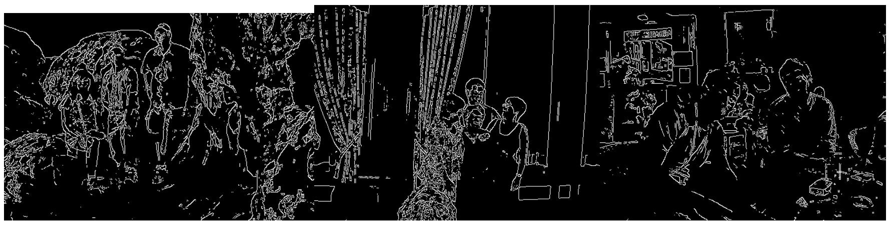
$$\mathrm{T} = \{(p_{0,x}, p_{0,y}), (p_{1,x}, p_{1,y}), ..., (p_{n,x}, p_{n,y})\} \tag{A.15}$$

There are several ways of comparing a template window to the contents of the image. Some of these are described in the following section.

(a) Original images


(b) Simple edge detection kernel


(c) Sobel kernel


(d) Canny edge detector with thresholds 200 and 180

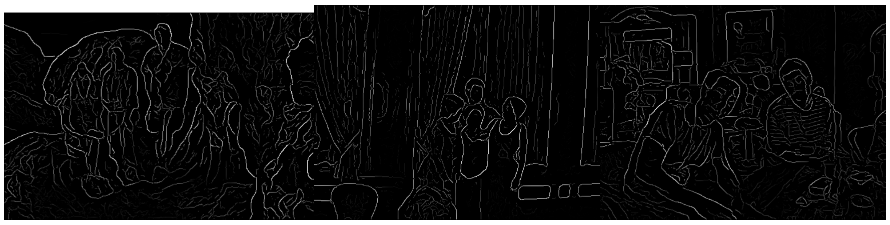

(e) The Pb contour detector

Figure A.15

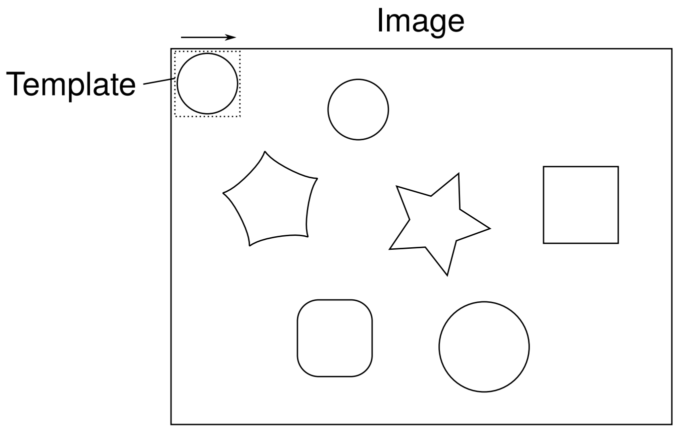


Figure A.16: Illustration of template matching. The small template window slides across the larger image and thus "scans" it for occurrences of a particular pixel pattern.

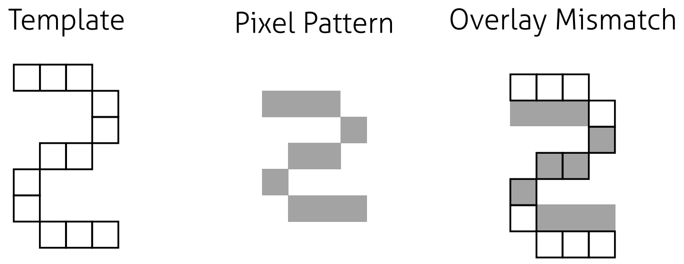


Figure A.17: Example of a large mismatch between two shapes that appear similar. The template only overlaps 4 out of 10 pixels in the pixel pattern and will thus show a low level of similarity.

A.4.1 Similarity Measure

The simplest way to measure similarity between a template and a small region of an image is to count the number of edge pixels that overlap with the template and divide by the total number of entries in the template:

$$S(\mathrm{T}, \mathrm{P}) = \frac{1}{N} \sum_{p \in \mathrm{T}} 1 \tag{A.16}$$

where

T	is the set of pixels in the template
P	is the set of pixels in the detection window
N	is the total number of pixels in the template

A big problem with this measure is that the result will be very sensitive to even small deviations between the template and the actual shape in the detection window, as illustrated in Figure A.17. This is because the algorithm will only register contour pixels that precisely overlap the template. A better approach is to measure the proximity of each element in the shape to the contour pixels in the detection window. One way is to use the Hausdorff distance (HD) which is formally defined as:

$$\mathrm{H}(\mathrm{T},\mathrm{P}) = \max\left(\max_{t\in\mathrm{T}}\left\{\min_{p\in\mathrm{P}}|t-p|\right\}, \max_{p\in\mathrm{P}}\left\{\min_{t\in\mathrm{T}}|t-p|\right\}\right) \tag{A.17}$$

The Hausdorff distance calculation can be explained as follows

1. Calculate the euclidean distance between all pairs of points between T and P. Find the smallest distances for each point in T to any point in P. Call the list $D_{T,P}$
2. Find the largest distance in $D_{T,P}$, call it $\max_{T,P}$
3. Swap T and P and repeat 1 and 2
4. Finally, the Hausdorff distance is the largest of $\max_{T,P}$ and $\max_{P,T}$

Essentially the Hausdorff distance will depend on the point in the two shapes that is furthest away from any point in the other shape. As such, it will penalize shapes that have just a single outlier between them, even if they match each other perfectly otherwise. This form of the Hausdorff distance is not very well suited for template matching, since the detection window can contain any number of contour pixels. Even if the shape that is searched for is present in the detection window, a single outlying contour pixel in the corner will make $\max_{P,T}$ become the longest distance. It is therefore better to simply use $\max_{T,P}$ as the distance between the template and the contour pixels:

$$\mathrm{H}_{\mathrm{mod}}(\mathrm{T},\mathrm{P}) = \max_{t\in\mathrm{T}}\left\{\min_{p\in\mathrm{P}}|t-p|\right\} \tag{A.18}$$

A related measure that is less sensitive to outliers is obtained by calculating the average minimal distance between each point in T and any point in P. This is known as the Chamfer distance (CD) [55] and is defined as:

$$\mathrm{D}_{\mathrm{C}}(\mathrm{T},\mathrm{P}) = \frac{1}{N}\sum_{t\in\mathrm{T}}\min_{p\in\mathrm{P}}|t-p| \tag{A.19}$$

Calculating the modified Hausdorff or Chamfer distance directly is expensive since both require the algorithm to calculate the euclidean distance between every pair of points in the two sets. Fortunately the computations can be sped up by first applying the distance transform to the contour map. In its basic form, the distance transform replaces all pixels in an image with the distance to the nearest edge or

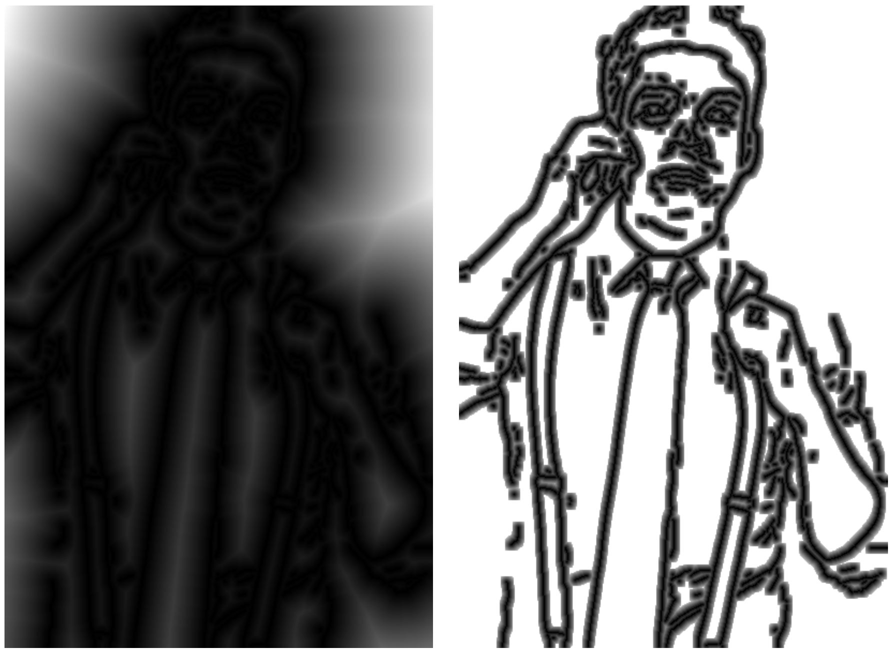

(a) No limit on the distance (b) A distance limit of 5 pixels. Beyond this distance pixels are set to a high value

Figure A.18: The result of applying the distance transform to the image

contour, as shown in Figure A.18a. The implementation can vary depending on the accuracy needed, e.g. the distance measure can be euclidean or Manhattan. To further reduce the computation time, the maximum search radius has been set to 5 pixels, as this turned out to be sufficient for good results. The remaining pixels are set to a high value. This produces the image shown in Figure A.18b.

The following formula is used to calculate the Hausdorff distance from a distance transformed image:

$$\mathrm{H_{mod}}(\mathrm{T}, \mathrm{I_{DT}}) = \max_{t \in \mathrm{T}} \mathrm{I_{DT}}(t) \tag{A.20}$$

where $\mathrm{I_{DT}}$ is the image region within the detection window. Likewise the Chamfer distance is calculated as follows:

$$\mathrm{D_C}(\mathrm{T}, \mathrm{I_{DT}}) = \frac{1}{N} \sum_{t \in \mathrm{T}} \mathrm{I_{DT}}(t) \tag{A.21}$$

A.5 Segmentation Via Graph Cuts

Segmenting an image essentially means dividing it into meaningful regions. This problem can be formulated in terms of a graph where each node corresponds to a pixel, and all nodes have one edge for each neighboring pixel. Segmenting the image is then a matter of cutting edges between nodes until the desired regions have no edges connecting each other. The description in this section is closely related to the fixation based segmentation method described in section A.6.

A general framework for transforming an image into a graph has been described [62]. Formally the image is defined as a graph $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$ where $\mathcal{V}$ is the set of vertices or nodes and $\mathcal{E}$ is the edges between the nodes. Every pixel in the image is a node in the graph. Edges between neighboring pixels are denoted by $e = \{p, q\}$. These edges are henceforth called N-links. In the case where the graph is to be divided into two regions (object and background), each node has two additional edges leading to terminal nodes. These are called T-links and they signify how closely a node is tied to either terminal. All edges have a weight assigned to them and the total cost of any segmentation is given by the sum of weights of the edges being cut:

$$|C| = \sum_{e \in C} w_e \tag{A.22}$$

where C is the set of edges being cut and w_e are their respective weights. If a vector A is defined that contains the binary labels of every node in the graph as:

$$A = \left(A_1, A_2, ..., A_p, ..., A_{|\mathcal{P}|}\right) \tag{A.23}$$

then an energy function that is to be minimized can be defined as follows:

$$E(A) = \lambda \cdot R(A) + B(A) \tag{A.24}$$

where

$$R(A) = \sum_{p \in \mathcal{P}} R_p\left(A_p\right) \tag{A.25}$$

$$B(A) = \sum_{\{p,q\} \in \mathcal{N}} B_{\{p,q\}} \cdot \delta\left(A_p, A_q\right) \tag{A.26}$$

$$\delta\left(A_p, A_q\right) = \begin{cases} 1 & \text{if } A_p \neq A_q \\ 0 & \text{otherwise} \end{cases} \tag{A.27}$$

$R(A)$ is a measure of the regional cost of the labeling defined by A. It is based on how a certain area fits into the model of the object to be segmented.

$B(A)$ is the boundary cost associated with the labeling. It is based on how different p and q are. It penalizes similarity in the case where $A_p \neq A_q$.

The goal is to minimize the cost function $E(A)$ by cutting the edges of the graph so the image is separated into an optimal object and background configuration. In this case, the labels are "inside" and "outside". The framework in [62] is intentionally unspecific about the details of $R(A)$ and $B(A)$ as these are intrinsic to the problem being solved. In [36] the authors define the energy function and costs specifically as follows[1]

$$Q(A) = \sum_{p \in P} U_p(A_p) + \lambda \sum_{(p,q) \in \Omega} V_{p,q}\, \delta(A_p, A_q) \tag{A.28}$$

where:

$$U_p(A_p) = \begin{cases} D & \text{for } A_p = 1 \wedge \forall p \in (r,\theta), r = 0 \\ D & \text{for } A_p = 0 \wedge \forall p \in (r,\theta), r = r_{\max} \\ 0 & \textit{otherwise} \end{cases} \tag{A.29}$$

$$V_{p,q} = \begin{cases} \exp\left(-\eta E^{\text{pol}}_{\text{pb},pq}\right) & \text{if } E^{\text{pol}}_{\text{pb},pq} \neq 0 \\ k & \textit{otherwise} \end{cases} \tag{A.30}$$

$$\delta(A_p, A_q) = \begin{cases} 1 & \text{if } A_p \neq A_q \\ 0 & \textit{otherwise} \end{cases} \tag{A.31}$$

$$E^{\text{pol}}_{\text{pb},pq} = \frac{E^{\text{pol}}_{\text{pb}}(r_p,\theta_p) + E^{\text{pol}}_{\text{pb}}(r_q,\theta_q)}{2} \tag{A.32}$$

$$\lambda = 50$$
$$D = 1000$$
$$\eta = 5$$
$$k = 20$$

$E^{\text{pol}}_{\text{pb}}(r,\theta)$ is the contour strength at a point in the polar map and the regional and boundary costs are now called $U_p(l_p)$ and $V_{p,q}$ respectively. The sole purpose of the regional cost is to make sure that the first and last columns in the polar map remain attached to their initial (opposite) terminals. In a more general case $U_p(l_p)$ could be based on how well a certain pixel fits a given color model. However, since single channel contour maps are used, such a term does not make much sense. Therefore only the first and last column are directly tied to a terminal and their T-link is given the weight D which is large. For all remaining nodes in the graph, the T-links have zero weight. This configuration is illustrated in Figure A.19.

[1]For consistency, some variable names in the original formulae have been changed to match the ones previously used.

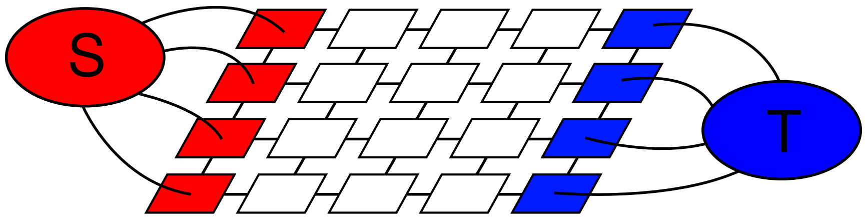


Figure A.19: Diagram showing the initial graph setup [36]. Only the first and last column are directly tied to a terminal. S terminal is colored red and T is colored blue.

The boundary cost $V_{p,q}$ is based on the mean value of two neighboring pixels. Since the image is a contour probability map, two pixels will have a large mean value if they lie on or near a contour and therefore the cost of cutting their edge will be small. For lower mean values the cost increases and in case the value is 0, the cost jumps to k which is large.

Using these rules, the entire graph can be defined for an image. The next step is to minimize $Q(A)$ to obtain an optimal segmentation. This was done using an algorithm called Min-Cut/Max-Flow [36]. The details of this algorithm are described in the following subsection.

A.5.1 Min-Cut/Max-Flow Graph Cut Algorithm

In order to minimize (A.28) and find the cheapest cut through a graph it turns out to be useful to consider all edges as pipelines that can carry a certain amount of flow between nodes. The capacity of an N-link or T-link is then defined by its weight. The goal is to push the maximum amount of flow through the graph from one terminal to the other, saturating all links that are at full capacity. When maximum flow is achieved, the two terminals will be separated by saturated N-links[2]. In this case, the image will be segmented into two regions. This idea is illustrated in Figure A.20.

This idea of flow is the basis for the Min-Cut/Max-Flow algorithm [63]. The algorithm is iterative and involves the following 4 stages:

1. Initialization
2. Growth
3. Augmentation
4. Adoption

[2]Technically by saturated T-links as well. However, these can be ignored in this case as T-links only have non-zero values for the first and last columns.

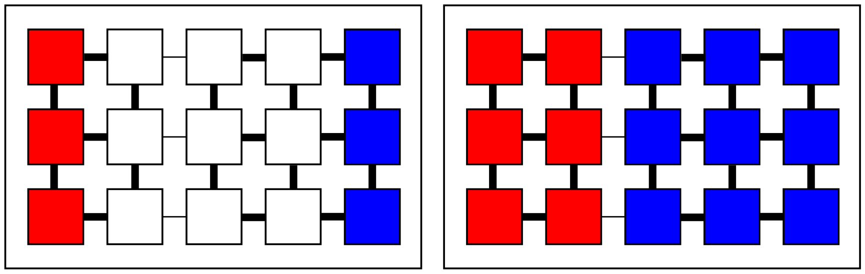


Figure A.20: Illustration of how a graph cut problem can be viewed as pushing maximum flow through a series of pipelines. The width of an edge indicates the capacity. The terminals will be separated by the edges that become bottlenecks when enough flow is passed through. Note that the T-links are not drawn explicitly since they only exist for the first and last column. The remaining nodes are tied to a terminal through neighboring nodes.

Initialization Upon initialization only the first and last column are tied to a terminal and the remaining nodes are free. All N-links are assigned a capacity based on their weight. For both terminals, a search tree is generated that contain the nodes they are currently tied to: the first column for S-terminal and the last column for T.

Growth In the growth stage the search tree is expanded along unsaturated N-links of all nodes in the search tree of each terminal. If an unsaturated N-link points to a free node, that node is added to the tree and is thus tied to the terminal through its parent nodes. The expansion continues until a node from one terminal tree meets a node from the other tree through an unsaturated N-link.

Augmentation Once the two terminals meet each other, a path is traced back through both search trees and flow is then pushed through this path. The amount of flow is determined by the N-link in the path that has the lowest capacity. This means that for every augmentation, at least one N-link will reach full capacity and become saturated.

If an N-link between a node and its parent in a search tree becomes saturated, the node becomes an orphan and is removed from that search tree. The node still maintains a temporary tie to the terminal as do its children.

Adoption In the adoption stage, all orphaned nodes will attempt to find a suitable parent that is tied to the same terminal and to which they have an unsaturated N-link. If such a parent exists, the orphaned node is adopted and added to the search tree again along with all its children. If no parent exists, the node becomes free and loses its children nodes that now become orphans themselves. This stage continues until all orphan nodes have been either adopted or become free.

The growth, augmentation and adoption stages are repeated until no unsaturated N-links remain that connect the two terminals. All pixels are then labeled by their terminal.

A.6 Fixation Based Segmentation

The idea behind fixation is to choose a seed point in the image, that lies within the object one wishes to segment. This is achieved by use of a graph-cut algorithm. The idea of constraining the search by use of a seed point is not a novel one. However, when searching for an optimal closed contour around a point, graph-cut algorithms tend to favor the shortest contour, even if this is not optimal from a visual perspective. The problem is illustrated in Figure A.21.

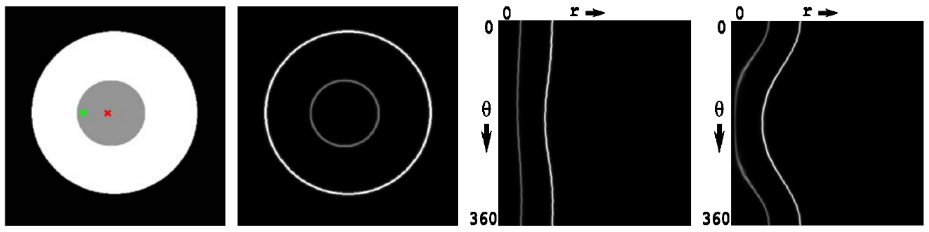


Figure A.21: Example of a segmentation problem where a seed is placed at the center [36]. A graph-cut algorithm will normally find the shortest contour around the point, which in this case lies on the gray circle, even though the strongest edge is found on the boundary of the larger white circle. In a polar coordinate system however, the two contours will have equal length and the algorithm will favor the outer most contour

A solution to this problem has been proposed [36] by transforming the image into a polar coordinate system with origin in the seed point before segmentation takes place. In a polar image representation the two contours have the same length and the algorithm will favor the contour with the strongest edge. To generate the edge map the authors use the Pb detector [59]. Furthermore, it is suggested to use stereo and/or motion cues, if available, to improve contour detection.

Rather than to directly transform the image into polar coordinates, the authors introduce a continuous function $W(x, y)$ that is defined as follows:

$$W(x, y) = \sum_{i \in S} \exp\left(-\frac{(x_i^t)^2}{\sigma_{x_i}^2} - \frac{(y_i^t)^2}{4}\right) \cdot E_{\text{pb}}(x_i, y_i) \tag{A.33}$$

where:

$E_{\text{pb}}(x,y)$	is the edge map with values ranging from 0 (no edge) to 1 (strong edge)
S	is the set of all edge pixels in the image
$\sigma_{x_i}^2$	$= \dfrac{K_1}{\sqrt{(x_i - x_0)^2 + (y_i - y_0)^2}}$
K_1	$= 900$
$\begin{bmatrix} x_i^t \\ y_i^t \end{bmatrix}$	$= \begin{bmatrix} \cos\theta_i & \sin\theta_i \\ -\sin\theta_i & \cos\theta_i \end{bmatrix} \begin{bmatrix} x_i - x \\ y_i - y \end{bmatrix}$
θ_i	is the orientation of edge pixel i in S
$\begin{bmatrix} x_0 \\ y_0 \end{bmatrix}$	is the fixation point or *pole*

$W(x,y)$ is the sum of Gaussian kernels placed on all edge pixels in the image and multiplied by their respective edge intensity. Each kernel is rotated such that its x-axis has the same orientation as the edge and the variance is inversely proportional to the distance between the edge and the fixation point. This means that a point in $W(x,y)$ will be brighter if it has many edge pixels of similar orientation nearby and if it is close to the fixation point. Sampling the $W(x,y)$ function obtained from the edge map in Figure A.21 produces the result shown in Figure A.22. Also shown is the same image transformed into polar space.

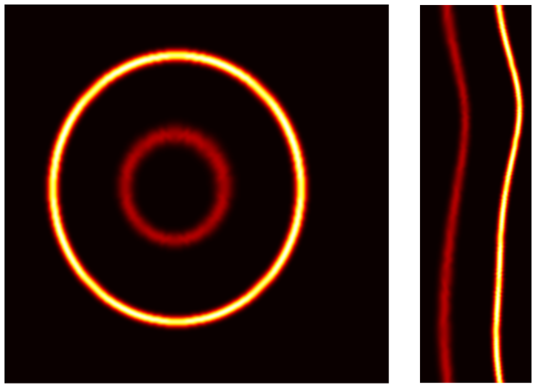


Figure A.22: (Left) Image from Figure A.21 transformed using $W(x,y)$ and a fixation point in the center of the two circles. Pixel intensities are represented by a heat map. (Right) sampled $W(x,y)$ transformed into polar coordinates.

In polar space, all pixels in the leftmost column correspond to the fixation point. The segmentation process will therefore cut the image into two parts, from top to bottom. This is done with a Min-Cut/Max-Flow algorithm [63]. Applying this algorithm on the image from Figure A.22 yields the results seen in Figure A.23

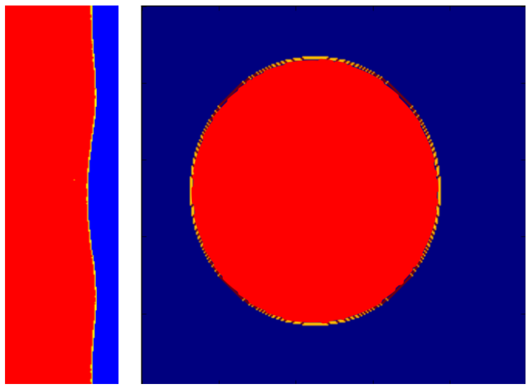


Figure A.23: (Left) The result of segmenting the image in polar space using the Min-Cut/Max-Flow algorithm. (Right) The inner segmented region transformed back into Cartesian coordinates and overlaid on the original image.

A.7 GrabCut

GrabCut [64] is an interactive segmentation algorithm that attempts to separate a user-specified region of the image into foreground and background. The algorithm is based on the the graph cut algorithm described in section A.5. Where the original graph cut algorithm segments the image in a single pass, GrabCut uses an iterative approach. The extension involves defining Gaussian Mixture Models (GMM) for foreground and background pixel RGB values. The energy function that needs to be minimized is then defined by

$$\mathbf{E}\left(\underline{\alpha}, \mathbf{k}, \underline{\theta}, \mathbf{z}\right) = U\left(\underline{\alpha}, \mathbf{k}, \underline{\theta}, \mathbf{z}\right) + V\left(\underline{\alpha}, \mathbf{z}\right)$$

where

$\mathbf{z}$	is a vector of size N containing the RGB values of all pixels in the image
$\underline{\alpha}$	is a vector of size N that contains the labels of every pixel. The label is either 0 or 1 corresponding to foreground or background
$\underline{\theta}$	is a vector that contains the parameters for all components in the GMM corresponding to the label $\underline{\alpha}$. Each mixture component has a mean vector and covariance matrix
$\mathbf{k}$	is a vector that assigns a unique GMM component to each pixel. The label α_n of the pixel decides the GMM
$U(\underline{\alpha}, \mathbf{k}, \underline{\theta}, \mathbf{z})$	$= \sum_n D(\alpha_n, k_n, \underline{\theta}, z_n)$
$D(\alpha_n, k_n, \underline{\theta}, z_n)$	$= -\log p(z_n \mid \alpha_n, k_n, \underline{\theta}) - \log \pi(\alpha_n, k_n)$
$p(z_n \mid \alpha_n, k_n, \underline{\theta})$	is a multivariate Gaussian probability distribution with parameters $\underline{\theta}$ decided by the label α_n and the mixture component k_n of the pixel
$\pi(\alpha_n, k_n)$	are mixture weighting coefficients
$V(\underline{\alpha}, \mathbf{z})$	$= \gamma \sum_{(m,n)\in \mathbf{C}} [\alpha_n \neq \alpha_m] \exp\left(-\beta \lVert z_m - z_n \rVert^2\right)$
γ	is a weight factor
$\mathbf{C}$	is the set of pixels adjacent to the cut
β	is a factor

The energy function $\mathbf{E}(\underline{\alpha}, \mathbf{k}, \underline{\theta}, \mathbf{z})$ is minimized using the min-cut algorithm. For every iteration, the GMM parameters are updated according to the new labels obtained from the graph cut and the process is repeated a specified number of times or until the decrease in energy per iteration goes below a certain threshold. The segmentation can be guided further by the user by explicitly labeling pixels as foreground or background as shown in Figure A.24.

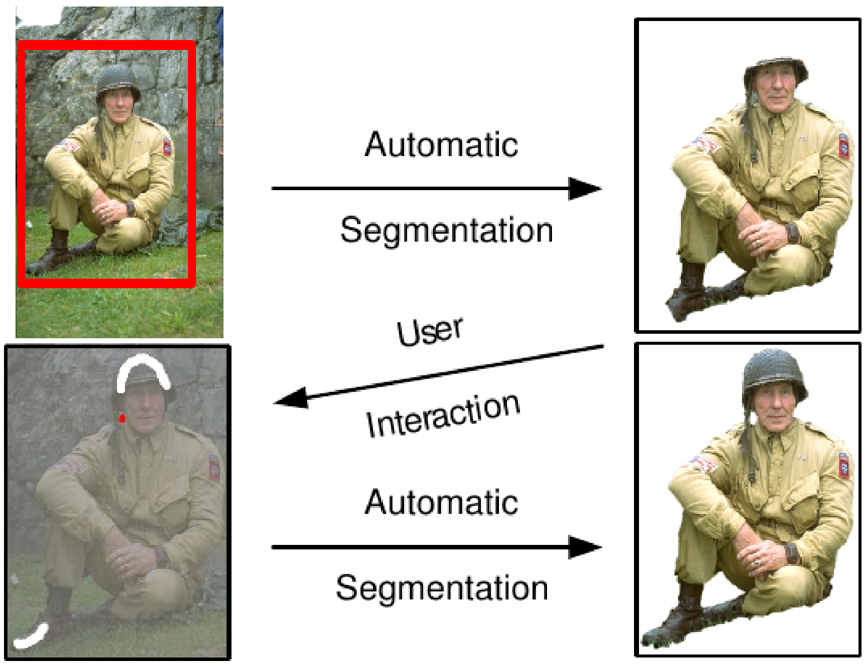


Figure A.24: Example of an image being segmented into foreground and background using the GrabCut algorithm [64]. The user specifies a rectangular area in the image and if the result is not as expected, further areas can be marked as being either part of the foreground or background.

Appendix B

Manual Labeling of Images

This appendix describes how images have been labeled manually in order to measure the performance of the head and face detectors described in this thesis. The script itself can be found on the DVD[1]

To make labeling as fast as possible a Python script has been written that utilizes the powerful Matplotlib library to display images and detect user input. To label a head in an image, the user selects a scale using the numeric keys 1-5 and then clicks on the center of the head to be labeled. The following scales can be selected:

$$[1.0;\ 0.66;\ 0.5;\ 0.33;\ 0.25]$$

Once a new label is added it will appear in the window as a green square. The label added last can be removed again by pressing "R". Figure B.1 shows the labeling interface and an example image.

When run, the script will create a list of all images in the current directory and open the first one. Once the image has been labeled the user can press the right arrow key to move on to the next image. All labels are saved in a text file with the same name as the image.

[1] \code\labelImageObjects.py

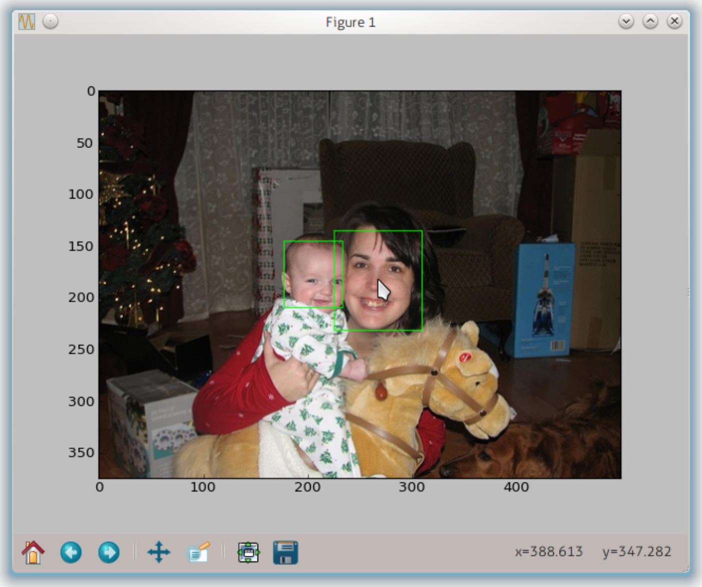


Figure B.1: The graphical interface used for labeling images

Appendix C

Scale-Based Weighting of Template Response

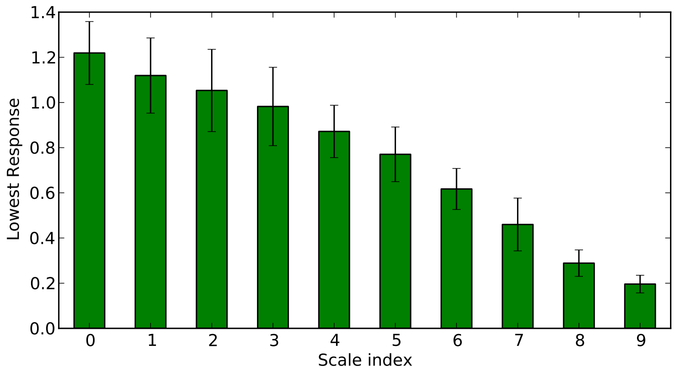
During testing of the template matching algorithm it quickly became apparent that smaller shapes tended to yield lower responses than larger shapes when matched against images. This is also becomes clear when plotting the average minimum response at different scales, as shown in Figure C.1a.

The reason for this is that smaller shapes have fewer pixels in their template and they spread out over a smaller area. This means that small shapes are more likely to match edges in the image more accurately than larger shapes. This dependence on the pixel count is shown more clearly in Figure C.1b. This is not entirely unexpected since smaller circular shapes *are* more likely to occur in images than larger ones. However, in this case the difference in response means that even valid matches of larger shapes are effectively eclipsed by smaller ones. One solution would be to use different thresholds for different scales. Another would be to weight the responses based on their scale, which is what is described in this appendix.

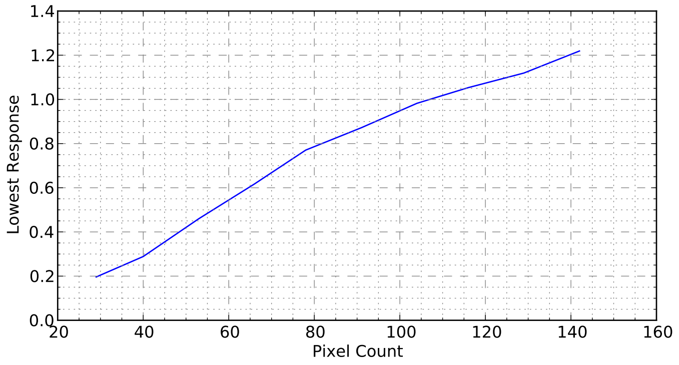
Looking at the shape of Figure C.1b suggests that the relationship between pixel count and minimum response is approximately linear. Using linear regression the curve can be approximated by the following expression:

$$f(x) = 0.00921x - 0.0231$$

We wish equalize this expression by finding $f'(x)$ such that:



(a) The mean minimum response and standard deviation from matching quarter circle shapes at different sizes on 10 images. Index 0 corresponds to a radius of 100 pixels.



(b) Lowest response as a function of the pixel count in the template

Figure C.1: Comparisons between the minimum match response and the scale of the shapes.

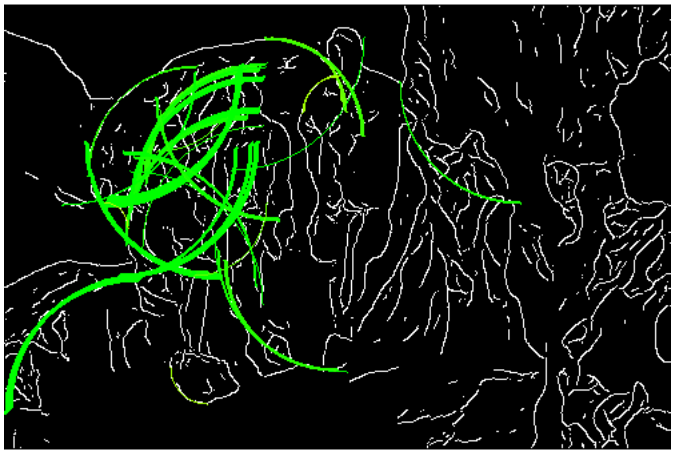


Figure C.2: Results of using equalization to balance the out the responses. It is apparent that larger shapes are now dominating and producing erroneous matches.

$$\begin{aligned} f(x)\, f^{'}(x) &= 1 \\ & \Downarrow \\ f^{'}(x) &= \frac{1}{0.00921x - 0.0231} \end{aligned} \tag{C.1}$$

Equalizing the responses using (C.1) turns out to penalize smaller shapes too much and thus yields invalid matches of very large scale shapes as shown in Figure C.2. It's obvious that a less aggressive weighting scheme is needed. The following weighting function was also tried:

$$f'(x) = \exp\left(\frac{a}{x}\right)$$

Where a is a constant. Several tests were performed using different values of a. The performance of the exponential weights is shown in Figure C.3. The performance Figure shows no substantial improvement although the exponential weights appear to perform slightly better when $a = 1$. For higher values of a the detection rate decreases. The performance when equalizing the responses with a linear expression perform the worst.

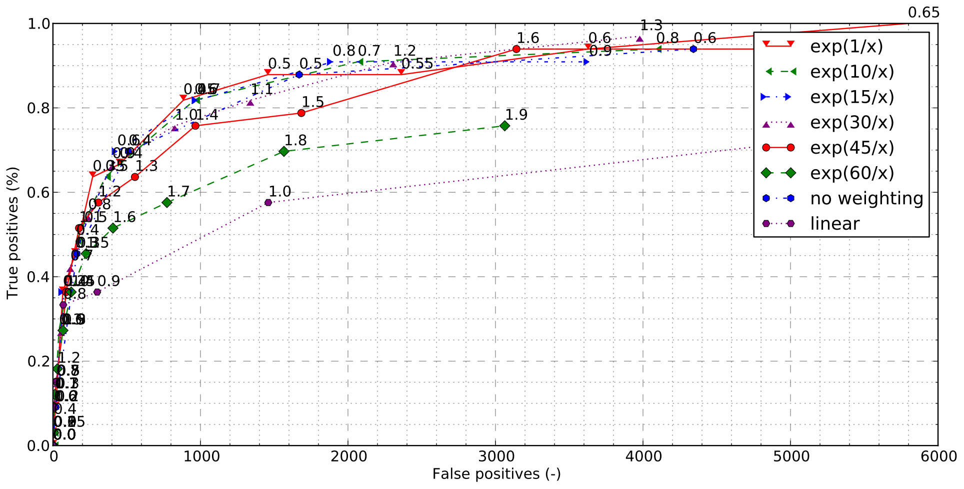


Figure C.3: ROC curves showing the performance of shape matching when applying the different weighting schemes with different constants. Each point is annotated with the threshold value that was used.