Master of Science Thesis

Learning to open new doors

Víctor López

Advisor: Ricardo Téllez
Dr./Drs. on behalf of the Advisor: Javier Béjar

Barcelona, January 2014

Acknowledgments

I am deeply grateful to Daniel Pinyol and Francesco Ferro for all the great opportunities that they have granted me in the past four years and specially for the chance to develop such an interesting project.

I would like to thank Ricardo Téllez and Guilhem Sals for their effort in the initial stages of my work and for their great ideas that helped shape this project.

I am sincerely grateful to Hilario Tomé, Jordi Adell and Jordi Pagès for their assistance and guidance on those subjects that were less known to me.

And to Dr. Javier Béjar for his help through all the stages of development of this thesis.

Finally I would like to dedicate this work to my family and friends who have supported me in many ways during the project.

Abstract

Finding and opening an unknown door autonomously is an unsolved challenge that the most advanced robots in the world have not solved yet.

Door handles have different locations, shapes, operating mechanisms and are made of different materials. Most approaches to door opening require precise information such as its exact location and a 3D model of the door handle that must be opened. This is one of the barriers that prevents robots from being used outside of controlled environments.

In this thesis, we describe an approach to solve the problem of localizing and classifying a door handle with the REEM robot with no human intervention in the process. To do so we use the data obtained from a RGB-Depth sensor to detect the position of the door handle and compute an image of it that is processed by a supervised classifier system to determine the type of door handle. The type of the handle will determine which approach and opening motion is required to open the door.

In this thesis we chose to perform stacked generalization with a feed-forward neural network on the prediction of several binary classifiers. The selection of the neural network model and binary classifiers is based on the experimental results of training and evaluating several combinations of supervised classifiers such as K-NN, SVM, Adaboost and Random Tree Forests with the image feature extraction algorithms Histogram of Oriented Gradients, Covariance Matrices and edge detection.

In the end we obtain a model able to classify handle images with a performance higher than any of the individual binary classifiers trained.

Contents

1 Definition of the problem **1**

2 Definition of the goals **3**

3 State of the art **5**

3.1 Basic concepts and vocabulary 5

3.2 3D image sensing 6

3.2.1 Contact based 6

3.2.2 Non-Contact based 7

3.3 Object recognition 8

3.4 Related work 8

4 Our Approach **11**

4.1 Sensors 11

4.2 Handle detection 12

4.3 Handle classification 12

4.4 Door opening motion 13

5 Handle Detection **15**

5.1 Method Overview 15

5.2 Implementation Details 17

5.2.1 Point cloud acquisition 17

5.2.2 Height filtering and down-sampling 17

5.2.3 Plane detection 18

5.2.4 Non-specular handle detection 18

5.2.5 Specular handle detection 18
5.2.6 Handle Side prediction 20
5.2.7 Image generation 20

6 Handle Classification **25**
6.1 Method Overview 26
6.2 Classification infrastructure 26
6.2.1 Image features 26
6.2.2 Feature selection 27
6.2.3 Over-fitting and Under-fitting 27
6.2.4 Supervised learning algorithms 28
6.2.5 Ensemble Classifier 31

7 Training Setup **33**
7.1 Data gathering 33
7.2 Example preparation 34
7.3 Classifier training 36
7.4 Ensemble Classifier training 38

8 Results **39**
8.1 Considerations 39
8.1.1 Data sets 39
8.1.2 Experiment Configuration 41
8.1.3 Performance Metrics 41
8.1.4 PCA 42
8.2 Classifier Performance 42
8.3 Identifying the best binary classifiers 44
8.3.1 Best Type 0 classifiers 45
8.3.2 Best Type 1 classifiers 45
8.3.3 Best Type 2 classifiers 46
8.3.4 Classifier memory usage 47
8.4 Ensemble classifier 47

9 Evaluation of Results **51**
9.1 Feature and classifier combination analysis 51
9.2 PCA effect analysis 53
9.3 Over-fitting analysis 54
9.4 Ensemble classifier performance 56

10 Discussion and conclusions **57**

A Binary classifier results **61**
A.1 Type 0 cross-validation results 61
A.1.1 Type 1 cross-validation results 65
A.1.2 Type 2 cross-validation results 68

B Tukey's range test results **73**
B.0.3 Type 0 classifier results 73
B.0.4 Type 1 classifier results 74
B.0.5 Type 2 classifier results 74
B.0.6 Ensemble classifier results 75

Bibliography **77**

List of Figures

3.1 Different active scanners used in robots 8

5.1 Handle detection pipeline 16

5.2 Same specular handle displayed, on the left the kinect sensor was able to determine its surface, on the right it was not. 19

5.3 By combining the left and center image using a XOR conjunction we obtain the areas occluded by possible specular handles 19

5.4 We compute the gradient of the coloured points, the ones above an threshold are painted blue, the rest are red. The points above the threshold are considered to be the edge of the door, with the position of the handle relative to the detected edge we can determine the side of the handle with regards to the center of the door. 21

5.5 First approach at door handle image generation 21

5.6 Depth handle image. The higher the grayscale value is, the greater is the distance from the point to the door plane 22

5.7 Hole handle image. Areas without readings are painted white while the rest of the image is black. 23

5.8 From the point cloud of the image on the left, we remove all the points of the handle and binarize it, painting the areas without information as white, to obtain the image on the right. 24

7.1 Asus Xtion Sensor on top of a tripod allowed us to capture images from a point of view similar to the robot's 34

7.2 Color images of the three labels that will be used in the classification. The first row are Type 0 examples, where no handles are present, Type 1 handles in the middle row and Type 2 handles in the bottom low. In the training the shadow images binary images are used, not this color image examples. 35
7.3 The top image is a projection of the point cloud contained in the rosbag, the red boxes are centered on the centroid of the detected handles. The image on the bottom left corresponds to the handle of the door in the image, whereas the image on the right corresponds to the door bolt 36

8.1 Box plot of the cross-validation data of the 10 Type 0 classifiers with the highest $F_{0.5}$ score, the read line indicates the median. 45
8.2 Box plot of the cross-validation data of the 10 Type 1 classifiers with the highest $F_{0.5}$ score, the read line indicates the median. 46
8.3 Box plot of the cross-validation data of the 10 Type 2 classifiers with the highest $F_{0.5}$ score, the read line indicates the median. 46
8.4 Average size in MB for all handle types for the XML representation of each classifier saved in disk. 48

9.1 $F_{0.5}$ score for the most significant feature/classifier combinations. The scale has been adjusted to the 0.5 - 1.0 range to give the image more contrast, $F_{0.5}$ scores lower than 0.5 are displayed as 0.5. 52
9.2 $F_{0.5}$ score average for all handle types in the validation set. 53
9.3 $F_{0.5}$ score for the most significant feature/classifier combinations. Training set results are on the left and validation set on the right. The scale has been adjusted to the 0.5 - 1.0 range to give the image more contrast, $F_{0.5}$ scores lower than 0.5 are displayed as 0.5. 55

List of Tables

6.1 Example output of the classifiers, each row is the output of a set of classifiers. 0 is negative classification and 1 is a positive match 32

8.1 Number of images in the training set for each handle type 40
8.2 Number of images in the train set of the ensemble classifier for each handle type 40
8.3 Number of images in the test set of the ensemble classifier for each handle type . 40
8.4 Number of features generated by each feature extraction algorithm without PCA and having applied different PCA configurations, indicated as PCA-N, where N is the retained variance in percentage 42
8.5 Type 0 classifiers $F_{0.5}$ Measure average results for the cross-validation data set. Highest 10 results in bold. 43
8.6 Type 1 classifiers $F_{0.5}$ Measure average results for the cross-validation data set. Highest 10 results in bold. 44
8.7 Type 2 classifiers $F_{0.5}$ Measure average results for the cross-validation data set. Highest 10 results in bold. 44
8.8 Size in MB of the PCA components XML files for each feature and PCA-variance combination. 47
8.9 Number of images in the validation set for each handle type 48
8.10 $F_{0.5}$ performance of the Ensemble classifiers, the column indicates the number of hidden neural network layers of the ensemble classifier, and the row the id of the set of classifiers used for training. The 10 classifiers with the highest $F_{0.5}$ score are in bold. 49

9.1 Performance of the best binary classifiers and the ensemble classifier on the test data set. 56

B.1 p values returned by Tukey's Range Test, if $p < 0.95$ the hypotheses that the there's no significant difference between the classifiers is considered false. In bold those comparisons where the classifiers are significantly different 73
B.2 p values returned by Tukey's Range Test, if $p < 0.95$ the hypotheses that the there's no significant difference between the classifiers is considered false. In bold those comparisons where the classifiers are significantly different 74
B.3 p values returned by Tukey's Range Test, if $p < 0.95$ the hypotheses that the there's no significant difference between the classifiers is considered false. In bold those comparisons where the classifiers are significantly different 75
B.4 p values returned by Tukey's Range Test, if $p < 0.95$ the hypotheses that the there's no significant difference between the classifiers is considered false. In bold those comparisons where the classifiers are significantly different 75

CHAPTER 1

Definition of the problem

Each passing day brings us closer to a future in which robots are no longer exclusively used in industrial settings. The biggest obstacle in this path is the impossibility to alter the environment in order to adapt it to the robots.

A humanoid robot is designed to reduce the difficulties of operating in a world shaped for humans. Humanoid robots generally have a humanoid-sized head, a set of two arms and two legs. By using their human-like extremities and size the robots are able to reuse the human infrastructures and tools reducing the need to modify the environment.

There have been many developments over the years to make robots able to navigate autonomously in open spaces. Autonomous navigation is the capability of a robot to determine its own position in its current environment and plan a path to a distant point in the space. To do so the robot must previously build a map of the area by navigating around it.

In public spaces such as museums or airports generally this is enough to cover most of the space, but as robots become more widespread, they need be able to navigate autonomously in other areas such as offices, homes or other environments where the different spaces are separated by doors.

Once a robot has identified that it in order to get to a certain point it needs to go through a door, it should be able to identify it, locate the handle, open the door and cross it.

There are many challenges when attempting to detect a door handle, in most situations there is no information about the door handle to be opened, the handle or the door can be replaced or moved around, and this should not affect the ability of the robot to open it.

Many types of different handles are used, they have different shapes, colours and textures and their position is not known beforehand. The door the handles are attached to also have

different sizes, texture and colours.

Even though there's a lot of uncertainty in this setting, all door handles have a similar purpose, they need to be manipulated, usually through a rotation, in order to unlock the door they are attached to. Similar purpose means that all handles have some common characteristics that we can take advantage of in order to detect them. They are on top of the door they are meant to open, they are close to the lateral edge of a door, they are at the same height and have similar size.

CHAPTER 2

Definition of the goals

The goal of this thesis is to give the REEM Robot the capability to detect and open doors. The scope of the project will be limited to detection and localization of a handle when the robot is nearby but not necessarily in front of a door, identifying its type in so that the door opening motions can be executed. The door opening motion is not in the scope of the thesis as it is a complete separate field.

The developed system must be highly precise and emphasize the minimization of the number of false positives, since a false positive might trigger a door opening motion on an incorrect object and cause damage to people or properties.

In order to achieve this we have at our disposal the wide variety of sensors that the REEM robot has, which include tactile sensors, ultrasonic range finders, laser range finders, stereo cameras and rgb-depth cameras.

The doors and door handles will not be modified in any way, nor will the robot sensors or actuators.

A secondary goal for this work is to learn how to use and experiment with different 2D image features and learning algorithms and compare their performance, learning time and other characteristics and choose the combination of feature and learning algorithm that provides the best results.

The 2D features that we will experiment are Histogram of oriented gradients (HOG) features descriptors[DT05], Covariance Matrices[TPM08] and direct pixel values with an edge detector.

For learning algorithms we will experiment with Support Vector Machines[CV95], K-Nearest Neighbours[CH67], Adaboost and Random Trees.

CHAPTER 3

State of the art

The main goal of this master thesis is to give the ability to open doors to a humanoid robot. This task couldn't have been performed 10 years ago, probably not even 5 years ago due to the lack of appropriate actuators and sensors. For this reason, all similar work is quite recent, albeit it is based on fields that have been around for several decades, such as object recognition and machine learning.

This chapter first provides basic concepts and nomenclature used in the related works and that will be used in the rest of the thesis. Afterwards we will explore the state of the art regarding 3D image sensing, object recognition and machine learning classification algorithms. At the end of the chapter we will review works related to opening doors in robots.

3.1 Basic concepts and vocabulary

This section describes the main technical terms used on the document. This is provided to establish a common ground necessary because they can have different interpretations, for this reason it is mandatory to specify their meaning.

TF is a software package that lets the user keep track of multiple coordinate frames over time. It is used to transform the position in one frame of reference to another. For instance to convert a position relative to the camera to a position relative to the robot hand.

TF Frame is a coordinate system in 3D, right-handed, for instance with X forward, Y left and Z up.

TF Transform transformation from one frame to another, can be static such as the transformation between the two eyes of the robot, or they can change in time, such as the transformation between a hand of the robot and its head.

Point Cloud is a data structure used to represent a collection of multi-dimensional points and is commonly used to represent three-dimensional data.

PCL is a large scale, open project[RBC11] for 2D/3D image and point cloud processing. The PCL framework contains numerous state-of-the art algorithms including filtering, feature estimation, surface reconstruction, registration, model fitting and segmentation.

RGB is a color space based con three chromaticities, red, green and blue. It can reproduce most colors combining intensities of one of them.

XYZRGB point a point in a point cloud that contains both the 3D position and RGB color from the point of view of the camera.

Type I error in statistics it is the incorrect rejection of a true null hypothesis. Also known as error or the first kind or false negative.

Type II error in statistics it is the failure to reject a false null hypothesis. Also known as error or the second kind or false positive.

Voxel a volumetric (3D) pixel.

Convex Hull is the smallest convex set that contains a given set of points.

3.2 3D image sensing

In object detection and recognition having the 3D information of the object provides many more tools for performing the task successfully.

With 3D information we can extrapolate the shape of the object in the 3 axis and we can use the depth information to distinguish if an object is on top of another.

There are many technologies available for acquiring the 3D information of an object. One of the most agreed classification[Cur99] establishes that there are two main types of 3D scanners, contact-based and non-contact based.

3.2.1 Contact based

Contact based scanners extract the 3D information of the object by physically touching it. This option was not suitable for this task since for safety reasons, the robot should only try to touch the object when it is certain that it is a door handle.

3.2.2 Non-Contact based

Non-contact based scanners can be active or passive.

Passive Scanners rely on detecting the reflected ambient radiation. The most widely used are stereoscopic systems which use two video cameras and can determine the distance of a pixel from the camera if the distance between the two cameras is known.

The REEM robot has two cameras and uses stereo-vision for many tasks, but this technique lacks precision on texture-less objects. Since doors and handles have uniform textures, we chose not to use this system.

Other passive scanners use a single camera and rely on changing the lightning conditions[CCD+88], moving the object or zooming [LRD93].

Active Scanners Active non-contact base scanners emit light, ultrasound or different radiation types and detect its reflection or radiation absorption.

There a wider array of sensor types, the most used active systems in robotics are time-of-flight cameras and laser range finders and structured light scanners.

Time-of-flight laser 3D scanners are based on a moving laser range finder, which emits pulses of lights and measures the time it takes to get back to the sensor. Since the speed of light is known, the distance to the object is easily obtained. The data obtained from a laser range finder is 2D, due to the low field of view of lasers[GCS+06] therefore the sensor must be tilted up and down and construct a 3D image by composing the captured slices. These devices don't provide the color of the object, just a point cloud with the shape of the object.

These sensors have been used successfully on door handle detection [RMCB09], unfortunately REEM's laser range finders are used for detection of obstacles in the ground and are not able to move or obtain readings at the height of door handles.

Time-of-flight cameras work in a similar way to laser 3D scanners, they consist of a light source that emits a light (typically infrared) for a short period of time, the camera lens gathers the reflected light and estimates the distance to the point that reflected the light.

Structured light scanners are composed of a regular camera and a light pattern projector, the camera obtains the color image of the scene, and from the deformation of the pattern calculates the distance of the pixels from the point of view [SPB04].

These scanners are precise and acquire images at a high rate, and have made a jump in popularity since Microsoft Released the kinect sensor, an affordable structured light scanner that is widely used in robotics nowadays.

(a) PR2 robot tilting laser range finder

(b) ASUS Xtion Pro structured light scanner used in REEM robot

Figure 3.1: Different active scanners used in robots

3.3 Object recognition

Object recognition is the process of detecting and identifying an object in an image or sequence of images. The main challenges for the task are objects obstructed from view, different view points and scales.

The recognition is very general, and can be broken down into different components. Usually the first step is to perform object detection in order to determine where the object more likely is and extract some candidate regions of interest, this will make the process faster and more less error prone. On these candidates we can perform instance recognition, if we know exactly which object are we looking for, or class recognition [Pon06] which involves determining to which class the candidate object belongs to, or if it doesn't belong to any class at all.

The goal of this thesis is to detect and identify handles in novel environments, since we don't have an image or model of the handle beforehand we'll be performing class recognition.

3.4 Related work

There have already been some approaches on teaching a robot to open doors.

If the door and handle are previously known and a model of the object is available the task is quite different and has been successfully implemented in [PSDP07] and [PN07].

More recently there has been and increase in interest and effort to perform the same task without requiring a model of the door handle.

Klingbeil et al. [KSN10] presented a system for opening doors and pressing elevator buttons using a 2D sliding window classifier on images obtained by a pair of stereo cameras. A supervised learning algorithm based on boosting weak decision trees on Haar features [Pap] determines if

the selected window contains at least one handle. Once the handle is detected, using 3D data from the stereo camera they estimate if the handle must be turned to the left or to the right using a logistic classifier based on two features.

Rühr et al. [RSP+12] developed a generalized framework for opening doors and kitchen drawers. They use 3D point clouds clustering to detect candidate handles in the at a certain position regarding the plane of the door. The 3D point clouds are obtained using a Kinect sensor. They also propose a method to handle specular handles, which reflect the sensor's projected infra-red pattern making them invisible to the Kinect sensor, by exploiting the lack of information to infer that there is a specular handle. They then let the robot learn how to open the door by trial-and-error.

CHAPTER 4

Our Approach

4.1 Sensors

In order to open a door, first we need to detect it, and to do so we require one or more devices that provides us with information from the environment.

The REEM robot has 2 laser range finders, 15 sonar range finders, 3 bumpers, a pair of stereo cameras and the option of using a head mounted RGB-Depth sensor.

Given the height of the handles, it is not possible to use REEM's precise laser range finders since their position is fixed and low enough to detect obstacles on the floor, therefore most of the door handles are out of the lasers' field of view.

Ultrasonic range finders are used to detect the presence of obstacles but are not precise enough to determine the exact position or shape of the detected object, and most of them are positioned closer to the ground level. Therefore we cannot use them for this task.

Bumpers are placed around the mobile base and are used to detect physical contact with an obstacle, we cannot use them due to their positioning and that they are meant only to be used in case of collision.

Unlinke Klingbeil et al. we chose to use RGB-Depth sensors over stereo vision as in the work by Rühr et al. because the depth sensors provide a more accurate 3D image of the environment and are being integrated in many more robots every day. Also, since RGB-Depth sensors use a pattern of projected infra red light, they have great performance on texture-less surfaces unlike stereo cameras, we believe this will work in our favour since many doors are texture-less.

4.2 Handle detection

Our approach requires that the robot is in the vicinity of a door and that the door is in it's field of view. Unlike Rühr et al. we don't require any more information with regards to the location of the handle. Since this solution is meant to be used on a real robot, we made our best effort to not add to the user the cumbersome task of detailing the 3D position of each door handle. Instead, when creating a map with REEM robot, the user just needs to indicate when the robot is facing a door that is meant to be opened with the press of a button in the map creation interface. This provides us with a rough position for the door at a very little cost in effort for the user.

With the 3D Point cloud information, we exploit contextual information as in the works by Klingbeil et al. to filter possible handles by height, size or position relative to the plane to determine the candidate handles.

We have also taken into account the specular handles problem described by Rühr et al. but instead of using two different approaches, one for specular handles and one for non-specular, we modify the 3D point clouds so that non-specular handles are perceived as specular handles, this allows us to reduce the complexity of the code of the handle detection and reduces the possible cases in the handle classification.

4.3 Handle classification

Unlike the PR2 robot, REEM robot is not able to perform impedance-based control to determine how to open the handle, to overcome this, we will not attempt to open door cabinets and drawers as in the work of Rühr et al. and we will limit ourselves to doors.

Furthermore, we use a once we have identified the handle, we generate a 2D image and we classifier to determine which type of handle has been detected, if any. The type of handle will provide us with the necessary information on how top open the door.

We started using Support Vector Machines and as in Klingbeil et al. but expanded it to experiment with different feature extraction and supervised classification algorithms and learn how they perform.

We build an ensemble of binary classifiers with the best performing combinations of feature extraction algorithm, application of Principal Component Analysis and binary classifier, and we use the output of this ensemble of classifiers to determine which type of handle, if any, has been detected.

In the related work we haven't seen any mention on how to avoid false positives, or the false positive rate, only a success rate. Since the goal of this project is to be used on a real robot on a regular basis it is highly important that we make sure that no other objects will be detected as door handles, therefore we will be giving more priority to having a low rate of false positives over having a low rate of false negatives.

4.4 Door opening motion

Unlike in the related works, this thesis does not deal with the manipulation problem of opening a door.

Once a handle has been localized and classified we'll use some already implemented solutions such as ROS MoveIt [SC] to open the door.

CHAPTER 5

Handle Detection

This chapter explains the methodology and technical details of the handle detection step and the reasoning behind it. The goal of this step is to process the data obtained from the sensors to detect the objects that are candidates to be classified as handles. This processing includes detecting a door and analysing the objects on top of it, in the end it produces a 2D image of the candidate handle.

Our priority in this step is to have a low number of type I errors (false negative errors) even if this increases the number of type II errors (false positive errors). The reasoning behind this statement, is that the handle classifier should be able to filter most of the false positives and we don't want to be too restrictive in the detection step and miss real handles because of it.

5.1 Method Overview

The whole pipeline of the handle detection is displayed on figure 5.1

The first step in our solution is to pre-process the data obtained from the sensors to detect the objects that are candidates to be classified as handles. The steps are:

1. Obtain point cloud from depth sensor
2. Remove all points outside of the allowed height region
3. Detect the biggest plane in the point cloud and obtain points on top of it
4. Detect non-specular handles
5. Detect specular handles

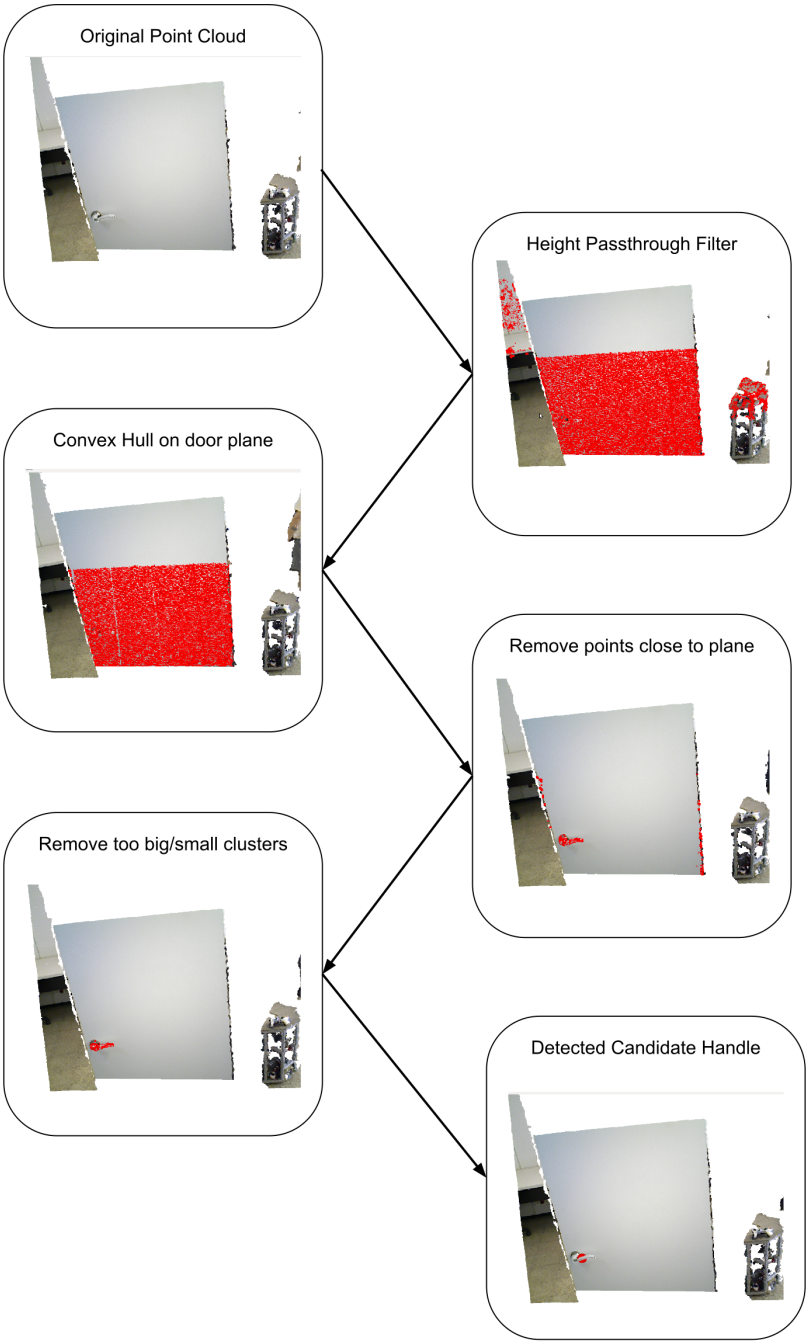


Figure 5.1: Handle detection pipeline

6. Determine side of the handle

5.2 Implementation Details

The handle detector is an application written in C++, it mainly uses the PCL library [RBC11][RC] for point cloud processing and the OpenCV library[Bra00] for 2D image processing. The application is built as a ROS [QCG+09] node, this means that it can be used in any ROS compatible robot.

5.2.1 Point cloud acquisition

The kinect sensor outputs point clouds at 30Hz, even though the robot might not be moving and the image is still, the point clouds can be significantly different due to ambient light changes and noise in the sensor readings. The handle detection can be a very computationally intensive task, since each point cloud contains over 300.000 points with 3D spatial coordinates, a depth value and an RGB value determining the colour of the pixel. Our goal is to process the data in the shortest amount of time possible to decrease the time it takes the robot to detect the handle.

When the point cloud is received, each point position in the cloud is relative to the camera optical frame, so the first step in the detection process is to convert them to the *base_link* frame, which is the root link in the robot, from this link the coordinates are in a frame of reference parallel to the ground and facing in front of the robot.

5.2.2 Height filtering and down-sampling

Once the point cloud is in the *base_link* frame, we apply a height filtering to remove everything over and below our height thresholds.

From research and measuring real doors we discovered that most door handles are placed around 90 centimetres above the ground.

We remove all the points that are above 110 cm and below 70 cm with a pass-through filter, this thresholds can be changed in runtime.

After the filtering is done, two copies are made from the resulting point-cloud, one of the copies will be modified greatly to find the position of the handle, but when the handle is found we will extract the image of the handle from the second copy which is which hasn't been modified and contains more information.

In this step we also down-sample the point cloud to 1 point per centimetre to decrease its point density and to speed up the subsequent steps. The down-sample is obtained by applying a 3D voxel grid over the point cloud, where the voxel are of 1x1x1cm and creating a new point cloud where the points are the centroids of the voxels in the grid.

5.2.3 Plane detection

The next step is to use the PCL library to obtain the predominant plane in the scene, we add some restrictions to this plane and force it to be perpendicular to the ground, since the robot is close to the front of the door, the detected plane must be the door and possibly the wall next to the door.

5.2.4 Non-specular handle detection

We label handles as non-specular when the kinect sensor is able to see most of the handle, because its surface is of a non-reflecting material and it's thick enough to be detected by the sensor.

Isolation of points on top of the door

Once we have the points of the plane, we construct a convex hull that contains all the points of the plane.

We project the convex hull along the normal of the plane towards the robot, this creates a polygon that contains all the points that are in the plane or on top of the plane facing the robot.

Clustering and filtering

From the point cloud that we obtained in the previous step, we subtract the points that form the plane, leaving only the points on top of the plane but not lying in the plane.

To the set of points that are on top of the plane we apply a clustering algorithm that groups neighbour points into sets of points on top of the door.

We analyse the resulting sets, and remove the ones that are too small or too big to be a door handle and the remaining sets are the detected handles of the algorithm.

5.2.5 Specular handle detection

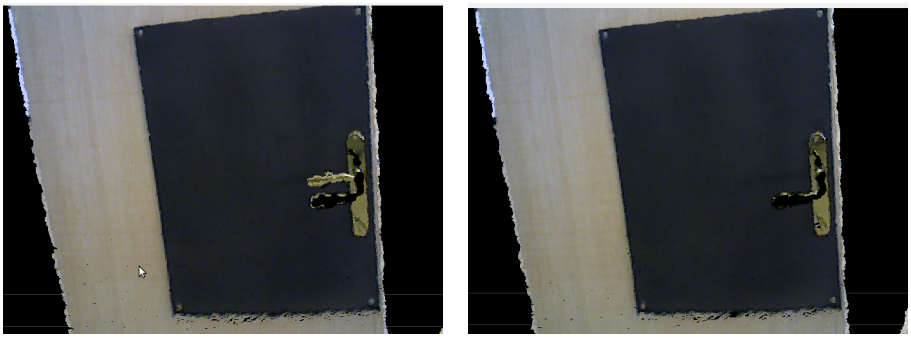
After having developed the handle detection algorithm and the handle classifier, we discovered that our algorithm wasn't able to detect certain door handles.

The surface of some metallic handles, handles with a shiny surface or thin handles reflect the Kinect infra-red pattern in a way that the sensor is unable to determine the surface of those points.

This reflection leaves a hole in the point cloud as can be seen on the right of figure 5.2

The method that we used for detecting the specular handles is based on the works of Rühr et al. [RSP+12] with some modifications.

We take advantage from the lack of information in an area to try to determine if there's a specular handle there.



(a) Specular handle visible (b) Specular handle invisible

Figure 5.2: Same specular handle displayed, on the left the kinect sensor was able to determine its surface, on the right it was not.

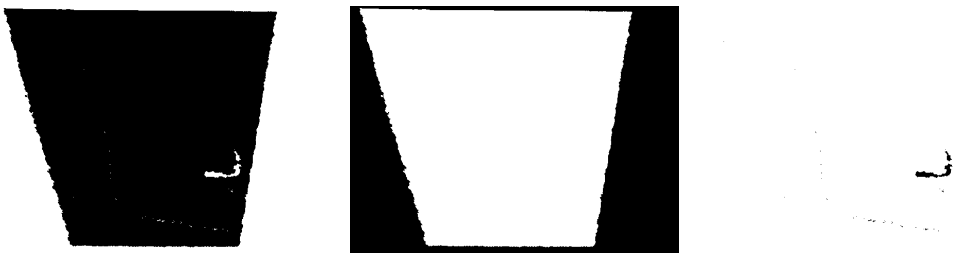
Binary image generation

The point cloud that we receive from the kinect sensor is an array of XYZRGB points, if we get no information from an area due to it being reflective or because the surface in that area is too far away for the sensor we simply don't receive points in that area.

For this reason, in order to detect which areas lack information, we need to project the points received to a 2D binary image like the one in 5.3(a). The white areas are the areas without points.

Then we generate another binary image with the points of the where the projected points of the door plane are painted in white as can be seen in 5.3(b).

To obtain the areas with invalid readings we combine using an XOR projection both binary images, the result can be seen in 5.3(c)



(a) Binary image of invalid measurements, areas without information are painted white (b) Binary image of the door plane, the points of the plane of the door are white. (c) XOR of binary images, the invalid points inside the door plane are black

Figure 5.3: By combining the left and center image using a XOR conjunction we obtain the areas occluded by possible specular handles

Clustering and filtering

As we did with the 3d points of the non-specular handles, we cluster the areas without information in the resulting binary image and remove the areas that are too small or too big.

For the remaining clusters, we compute the convex hull of each cluster and calculate the centroid of the hull to determine the position of the handle.

Handle position projection

After computing the position of the centroid of the candidate hulls, we need to determine in which position the handle is. To do so we project a plane, parallel to the door but 6 centimetres in front of it, this distance was obtained experimentally by measuring the distance from the grasping point of several handles to the door plane.

The grasping point for the detected handle is the point where the projected plane intersects with a vector from the camera optical frame to the centroid of the candidate hull.

5.2.6 Handle Side prediction

Once we have determined the position of the handle we need to determine if the handle is in the left side of the door or in the right, from this information we can deduce the handle rotation direction for opening and the door opening motion.

Since the point cloud obtain does not usually contain the whole frame of the door, we cannot tell what is the position of the handle with regards to the vertical center of the door.

In order to compute it we want to locate the contour of the door. A door handle is most commonly found close to an edge of the door. The idea is to compute the distance between the handle and this edge, if it is below a threshold we know at which side of the door the handle is.

We use a Canny edge detector [Can86] and a Hough Transform [DH72] to the 2D image in order to detect the edges that are part of straight lines and we remove all the lines that are not perpendicular to the ground.

We compute the distance and direction from the detected lines to the handle and determine its side using this information as can be seen in Figure 5.4

5.2.7 Image generation

Once we have detected the position of the candidate handles, we need to extract a 2D image of it for the classifier algorithm.

During the development of this thesis, we tested different ways of generating the images of the candidate handles to determine which would provide the best results.

All the images generated were of the 150x150 pixels and centred on the handle.

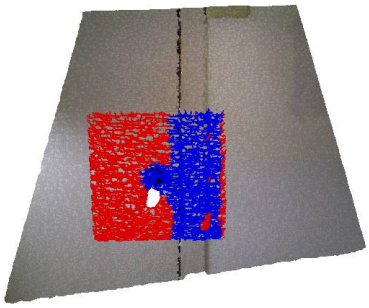
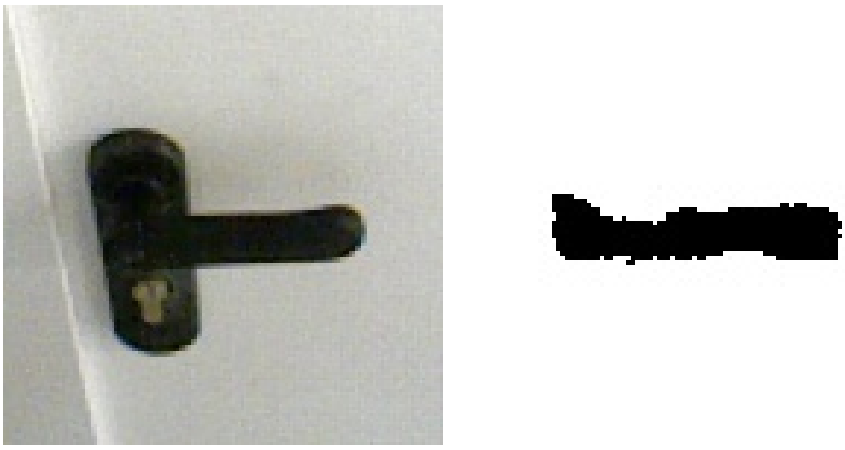


Figure 5.4: We compute the gradient of the coloured points, the ones above an threshold are painted blue, the rest are red. The points above the threshold are considered to be the edge of the door, with the position of the handle relative to the detected edge we can determine the side of the handle with regards to the center of the door.

First approach

The first approach was to generate a colour image as obtained by the sensor containing the handle as in figure 5.5(a) and a black and white image where the points with a distance to the door plane greater than a threshold were painted black as in figure 5.5(b).



(a) Color handle image (b) Binary handle image

Figure 5.5: First approach at door handle image generation

These images where used in the early iterations of the learning algorithm and the black and binary proved to be slightly better than the color image.

Height grayscale images

After the results of our first tests, and based on the binary image approach we implemented a way to generate grayscale height images.

Unlike the depth images provided by the Kinect sensor, which are based on the distance from the sensor to the object, we calculated the minimum distance from each point to the plane of the door, effectively calculating the height of a point with regards to the door.

After removing outliers we defined the range of distances in the image by setting the lowest value to distance 0 (the plane of the door) and white to the highest point in the handle, obtaining a resolution in the image of around 0.4mm, an example of this type of image is displayed in Figure 5.6.

Since the resolution of the sensor is 1.0mm we were able to generate images without losing the depth information.

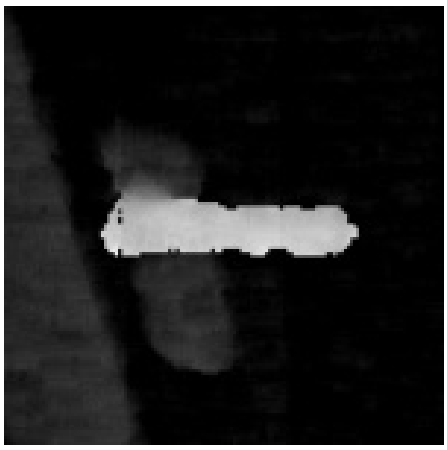


Figure 5.6: Depth handle image. The higher the grayscale value is, the greater is the distance from the point to the door plane

The depth images provided to be 30% better at the classification task than the black and white images described in the previous section, but were discarded because it was impossible to obtain them from specular handles.

Specular handle solutions: Hole image

Most of the work of the thesis was done using the depth images, but once we started facing specular handles we had to look for an alternate solution.

As can be seen in 5.2(b) when in front of a specular handle, sometimes the sensor is not able to detect the surface and doesn't provide any readings of the specular surface.

Our first approach was to try to look for these "holes" in the point cloud, areas inside the door plane that lack information. So we generated a binary image with the invalid points painted white as can be seen in 5.7.

Figure 5.7: Hole handle image. Areas without readings are painted white while the rest of the image is black.

This approach required to build another set of classifiers for the specular handles since the hole images where completely different from the depth images, and the final application would have to decide which one to use according to the type of image provided. This situation left a lot to be desired as we need to double the data acquisition, the training time and the memory and storage usage.

Final solution: Shadow image

In order to remove the need to have separate classifiers for specular and non-specular handles, we implemented a new candidate image type that could be used for both types of handles.

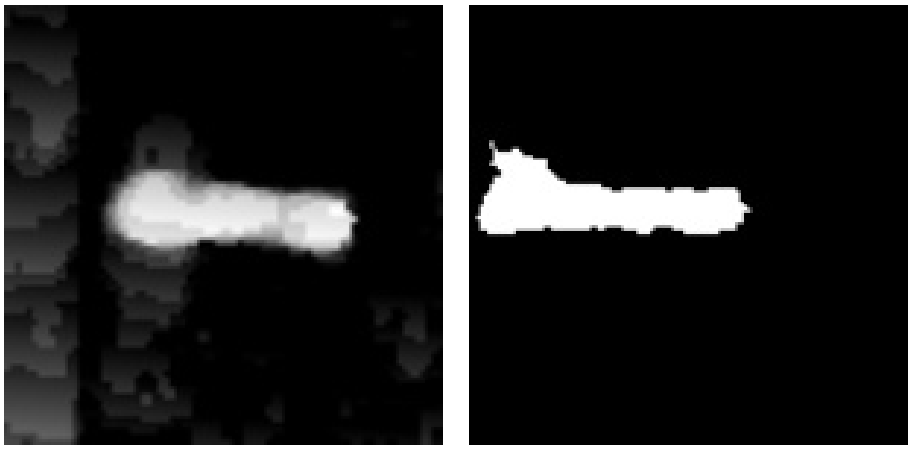
We call these image type "shadow images" and are generated by detecting the plane of the door and removing all the points that on top of the door plane, which we assume are part of the handle. By removing the information of the non-specular handle, we're essentially converting it to a specular-handle, and after this step there is no difference in the processing of specular and non-specular handles.

Another addition in this method, is that instead of generating the 2D image from the RGB-Depth sensor point of view, we estimate the position of the handle by computing it's centroid, and we create a virtual camera that is in front of the handle position a certain distance away in the direction of the normal of the plane.

The 2D images obtained from this virtual camera are viewpoint invariant, it removes the differences caused by obtaining the point cloud from different positions. This step applies some sort of normalization to the 2D images that simplifies greatly the training process.

The resulting images are similar to the binary images that we generated in our first experiments but they are also usable with specular handles.

The shadow image is a compromise between quality of the solution and resource usage, by having really similar images for both specular or non-specular handles, we reduce the complexity



(a) Depth image (b) Shadow image

Figure 5.8: From the point cloud of the image on the left, we remove all the points of the handle and binarize it, painting the areas without information as white, to obtain the image on the right.

of the training, that no longer needs to learn two different image types

Another solution would have been to train twice as many classifiers, a set for recognizing specular handles and a set for non-specular handles, this would require a higher training time, more data gathered and a way of deciding which set of classifiers should be trusted when encountering a novel handle.

CHAPTER 6

Handle Classification

This chapter explains the different machine learning algorithms used in the handle classification, the methodology used to compare them and other design decisions and their justification.

The goal of this step is to determine if what was detected as a handle is really a handle and determine what type of handle is it, as different handle types need different grasping techniques.

Unlike the previous step which a series of well defined and studied manipulations of the input data, this step uses supervised machine learning algorithms to determine whether a handle is present or not and what type of handle it is.

This task is fit to be solved by machine learning algorithms since it is a classification problem, and starting from a small set of examples we want to be able to generalize and be able to identify correctly novel doors and door handles.

In addition to the main goal of this step which is to classify the handles, and due to our lack of expertise with supervised machine learning techniques, we built an infrastructure that would give us the possibility to compare the performance of different combinations of training algorithms, parameters to these algorithms and 2D image features.

Since the output of this step will determine if the robot will attempt to grasp the handle and open a door, our priority in this step is to have a low number of type II errors (false positive errors) even if this increases the number of type II errors (false negative errors). For us it is better to miss the opportunity of opening the door if that increases the safety of the robot.

6.1 Method Overview

Our handle classification solution is composed of two classification steps. The first step will contain N binary classifiers for each type of handle in addition to N classifiers that determine if the given image is not a handle.

The output of these classifiers is an array of $(H + 1) * N$ binary responses, where H is the number of handle types and N is the number of classifier per handle type and for the "not-a-handle" classifier.

Interpreting this binary responses is not a trivial issue, since the classifiers are independent and we can have contradicting answers. To determine whether a handle was detected or not and to determine the type we make use of an Ensemble Classifier [Rok09] that is trained with the output of the first group of classifiers.

6.2 Classification infrastructure

To perform the classifiers training and validation, an infrastructure was developed in order to experiment with different learning algorithms and its parameters, as well as different image features and training configurations.

The goal of this infrastructure was to create an abstraction layer on top of the training algorithms and image features, so they could be interchanged seamlessly without affecting the system in any way.

After some research we decided that we would use the machine learning algorithms implemented in the OpenCV library[Bra00], it was one of the best documented and complete C++ machine learning libraries, and it integrated perfectly with our system since we were using OpenCV already for 2D image processing.

The flexibility of the infrastructure allowed us to perform the training with dozens of combinations of training components (features, algorithms, parameters and feature selection) and compare the results in order to find the most successful combination, and all this was only at the cost of training time.

Each training configuration consisted of a selection of one feature descriptor algorithm, one training algorithm and the possibility to apply a feature selection algorithm before the training algorithm.

The following components were implemented:

6.2.1 Image features

HOG feature descriptor

Histogram of Oriented Gradients [DT05] is a feature descriptor algorithm that is based on counting the gradient orientations present in segments of an image, and uses this information typically for detection applications. It is often used in combination with Support Vector

machines for human and object detection.

Our implementation used 16x16 pixel blocks and initially had two configurable parameters the L2-Hys threshold and the maximum number of detection increases, but we observed that the performance differences were not significant and did not justify the training time cost.

The number of feature descriptors generated by this algorithm was 8100.

Covariance matrices

Usually covariance matrices are not well suited for typical machine learning algorithms, since the feature descriptors do not form a vector space, [TPM08] attempts to solve this by representing the descriptors as a connected Riemannian manifold [Jos08] and this approach is the one we implemented.

Our implementation could be parametrized to change the size of the sub-windows of the algorithm and its configuration. We used 50px and 100px window sizes and three different window configurations that were different compositions of sub-window shapes and sizes.

The number of feature descriptors generated by this algorithm was of 324 for the first sub-window configuration, 1296 for the second sub-window configuration and 3600 for the third.

Pixel Values

Since the images provided by the handle detector are centered around the handle, we created a feature descriptor that would just convert the values of the pixels of the image into a feature vector. There were two available parameters, image size and a boolean to determine if a canny edge detector should be applied.

The number of feature descriptors generated by this algorithm was 2500, 10000 or 22500 depending on the image size (50, 100 or 150 pixels).

6.2.2 Feature selection

Given the high dimensionality of the data that was being extracted by some feature descriptors, we added the possibility to use OpenCV's implementation of Principal Component Analysis[Hot33]. We chose PCA because of its extended use and its simplicity.

We allowed the option to determine the percentage of retained variance, which would determine the number of resulting components and the size of the feature vector.

We experimented with no PCA at all, as well as PCA retaining variance ranging from 75% to 99%.

6.2.3 Over-fitting and Under-fitting

In machine learning a learning algorithm is said to be over-fitted to the training data if it is significantly better at classifying correctly the data in the training set than it is to classify novel data.

Under-fitting occurs when the algorithm is not able to predict correctly the labels, both for the training data and for novel data.

Over-fitting can be solved by reducing the number of features, removing the less significant ones in an effort to try to stop the learning algorithm from learning the noise in the training set. A higher number of examples might help. Also too complex learning algorithms might tend to learn the noise of the training set, so reducing the learning algorithm complexity might help.

Under-fitting can be alleviated by applying the inverse measures that are applied when facing over-fitting.

6.2.4 Supervised learning algorithms

Different supervised learning algorithms were integrated into our infrastructure, some of them where already implemented in OpenCV whereas others had to be implemented from scratch.

As with the feature descriptors, we enabled the possibility to experiment with different values for some parameters of each algorithm with the goal of finding the best combination of feature descriptor, feature selection and learning algorithm and their parameters.

Since we can have as many different handle types, our problem is a multi-class classification problem. Instead of using multi-class classifiers, we chose to build a set of binary classifiers that perform one-versus-all classification, which determine if an image if of the given class or not.

K-Nearest neighbours

K-Nearest neighbours[CH67] is one of the simplest supervised learning classification and regression algorithms, it is a lazy learning algorithm, all the computations are performed at the classification step, the learning process consists only of just storing the data set for later retrieval.

When presented with a sample containing a feature vector that must be classified, the algorithm looks for the K closest examples and chooses the label that is most frequent in those K neighbours. Typically Euclidean distance is used for continuous variables such as the ones we're being used.

If the training set is skewed, and there's more samples from a label than from the rest of labels, the majority voting of K-NN can affect the quality of the results, since the most frequent label can be common among the K neighbours. A way of overcoming this situation can be used by weighting the neighbours with regards to their distance, such that more distant neighbours have less weight that the closer ones.

The K value used depends on the type of data, usually it is small number, greater numbers help reduce the noise on the classification, but the prediction of the samples on the classification boundaries can be affected, since it makes this boundaries less distinct. An odd K number is often used in binary classification to avoid ties.

K is often selected using hyperparameter optimization, selecting the best K for the data being classified. In our experiments we evaluated the performance of different values of K.

Support Vector Machines

Support Vector Machines or SVM are non-probabilistic binary linear classifiers invented by Vladimir N.Vapnik in 1963[VL63].

SVM creates a model by assigning a representation of each sample in the labelled example set to a point in space in a way that maximizes the size of the gap that divides the two different labels. In other words, given the set of training data, the algorithm outputs an optimal hyperplane which categorizes new examples.

Given the training data set $\mathcal{T} = \{(\mathbf{x}_i, y_i) \mid \mathbf{x}_i \in \mathbb{R}^p,\ y_i \in \{-1, 1\}\}_{i=1}^n$, where $\mathbf{x_i}$ are the feature vectors and $\mathbf{y_i}$ the labels of each example, SVM can be seen as an optimization problem trying to maximize the distance $d = \frac{2}{\|\mathbf{w}\|}$ between the two hyperplanes, where $\mathbf{w}$ is the normal vector of the hyperplane described by:

$$y_i(\mathbf{w} \cdot \mathbf{x}_i - b) \geq 1, \quad \text{for all } 1 \leq i \leq n.$$

In many cases the the data sets are not linearly separable in their original finite space, to deal with these situations in 1992 [BGV92] suggested to apply the kernel trick[ABR64] to the original features in order to map them to a high- or infinite-dimensional space.

The Kernel Support Vector Machines use a *kernel function* $K(x, w)$ instead of the dot product in the above formula,

$$y_i(\mathbf{K}(\mathbf{w}, \mathbf{x_i}) - b) \geq 1, \quad \text{for all } 1 \leq i \leq n.$$

Depending on the kernel used, the behaviour of the SVM is altered. With a linear kernel $K(x, w) = x^T w$ the classification is performed in the original feature space, it behaves similarly to the original SVM.

A RBF or Radial Basis Function Kernel function $K(x_i, x_j) = e^{-\gamma||x_i - x_j||^2}, \gamma > 0$ maps the original feature space into an infinite number of dimensions space. It is the preferred Kernel when we don't know much about the data we are trying to model.

OpenCV provides a Support Vector Machines implementation based on LibSVM [cCL01], with five different types of SVM formulations and four different types of SVM kernel functions, from which we will experiment using Linear Kernel and RBF Kernel.

OpenCV provides a method to perform optimization on the different parameters given a training set and a Kernel function, therefore there is no need to select other parameters.

Adaboost

Boosting is a machine learning meta algorithm that combines many weak classifiers, classifiers which are only slightly correlated with the true classifier and computationally inexpensive, in

order to form a strong classifier.

Adaboost[FS95] means adaptive boosting, adaptive because the new classifiers created are targeted to classify correctly the misclassification of the previous classifiers. It is more resistant to over-fitting but sensitive to noise in the data set and outliers.

Adaboost is built in an iterative process, in each iteration a new weak classifier is added, and a weight distribution containing a weight for each example in the training set is updated, the weight of all the miss classified examples is increased and the weight of correctly classified decreases.

Formally, given a training set $\mathcal{S} = \{(\mathbf{x}_i, y_i) \mid \mathbf{x}_i \in \mathbb{R}^p,\, y_i \in \{-1, 1\}\}_{i=1}^m$ and a number of iterations T, firstly the weight distribution D_1 is initialized, with all weights equal to $\frac{1}{m}$.

Then, for each iteration $t = 1, ..., T$ pick the classifier h_t (or train it) that maximizes $h_t = \underset{h_t \in \mathcal{H}}{\operatorname{argmax}} \; |0.5 - \epsilon_t|$ where $\epsilon_t = \sum_{i=1}^m D_t(i) I(y_i \neq h_t(x_i))$ and I is the indicator function.

Then α_t is chosen, usually is $\alpha_t = \frac{1}{2} \ln \frac{1-\epsilon_t}{\epsilon_t}$.

If the error rate ϵ_t is below a previously chosen threshold then the boosting process has finished. Otherwise update the weight distribution D_t such that the examples miss-classified by h_t are weighted more and the properly classified weighted less. In the iteration $t + 1$ the classifier tested against D_{t+1} should perform better against those examples failed by h_t.

$$D_{t+1}(i) = \frac{D_t(i) \exp(\alpha_t (2I(y_i \neq h_t(x_i)) - 1))}{Z_t},$$

where Z_t is a normalization factor that will make D_{t+1} a probability distribution.

After all the iterations, the weak classifiers are combined by assigning a different weight to each classifier to build the strong classifier H:

$$H(x) = \operatorname{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$$

OpenCV's implementation of Adaboost uses decision trees as the weak classifiers, trees where each non-leaf node refers to an input variable and its arcs represent values of the variable, each leaf node contains a value of the classification label.

In our approach we experiment with different depth of the decision trees and different type of boosting algorithm to test their performance.

Random Trees

Random Trees or Random Forest [Bre01] is an ensemble of tree classifiers, each tree is trained with a bootstrap sample of the training set, and it only receives as input a small portion of the features of the training set.

For a training set of size m, with P features, a parameter p that specifies the number of features used in each tree node, and a fixed number of trees to be built, the trees are grown individually as follows:

1. Sample m cases of the training set randomly with repetition, this means that some cases will be repeated and some will be absent.

2. Select p random features from the original feature vector P, and choose the best split from these p features to build additional nodes in the tree.

3. Each tree is grown to the largest extent possible, the resulting trees are not pruned.

Once all the trees are constructed, in order to classify a sample it is pushed down each tree until a node, and it is assigned the label of the training sample in the node for each tree. The output of the random forest is the label that received the majority of votes.

Unlike Adaboost, no weights are assigned to the best performing weak classifiers and no extra effort is done to classify the samples that have been miss-classified before.

Due to it working on subsets of the original features, typically each node receives $p = \sqrt{P}$ features, it can handle huge feature vectors and also gives estimates of which variables ore important in the classification.

OpenCV let's us experiment with the number of decision trees used in the forest, so we experimented with different values to see the how they behave.

6.2.5 Ensemble Classifier

In our classification infrastructure we make use of N sets of $(H+1)$ binary classifiers, where H is the handle types we want to be able to predict. Our output is a a matrix of the same dimensions of binary answers, in an ideal setting with perfect classifiers this matrix would contain a positive answer from the classifiers that detect the handle being examined and a negative answer for the rest of the classifiers, in this situation N could be one because the answers would be always right and simple to interpret.

Unfortunately our classifiers have always a probability of error, in order to improve the accuracy of our problems we combine different sets of classifiers to produce a stronger classifier. In our problem, classification speed is not a limiting factor, since the whole open door procedure can be quite long, we can combine several sets of classifiers to improve the accuracy only at the cost of training time and storage space. Since the training for different learning algorithms will be performed anyway in order to compare their performance, we can pick the N best classifiers or the N sets of classifiers that cover most of the examples and combine their answers in order to build a better classifier.

In Table 6.1 is displayed the typical situation where we have conflicting answers, there are 2 positives answers in the set of classifiers, and two of those answers indicate that there is a Type 1 handle and the others that there's no handle and there's a Type 0 handle.

No-Handle	**Type 0**	**Type 1**
0	1	1

Table 6.1: Example output of the classifiers, each row is the output of a set of classifiers. 0 is negative classification and 1 is a positive match

There are many approaches to this situation. The most simple one would be to add the outputs of the different classification sets and chose the label with the highest count of votes, but this might not be precise enough and we don't have a way to determine what to do when there's more than one label with the highest score.

A different approach that has been explained in the previous section would be boosting which would require to assign a weighting formula for the output of the classifiers.

But we wanted to go a bit further and chose to use stacking [Wol92] to solve this problem. With stacking we apply a supervised machine learning algorithm to the output of the classifiers to determine the label that will be assigned.

There is some work that indicates that stacking performs worse than selecting the best classifier using cross-validation [Zen04] and there are also works that state exactly the opposite [TW97], so we decided to implement it and evaluate for ourselves.

We chose to use the most common kind of feed-forward artificial neural networks, multi-layer perceptrons (MLP) as our ensemble classifier, since it provides out-of-the-box multi-class classification, we had not experimented with it in the previous stage of our solution and it is implemented in OpenCV.

MLP contains one input layer, one or more hidden layers and an output layer. The number of neurons in the input layer must match the number of features, in our case $N * (H + 1)$. The number neurons in the output layer are $H + 1$, so we'll have that many output variables that correspond to the possible labels, each one with a value ranging from -1 to 1 and the one with the highest value will be the selected label.

CHAPTER 7

Training Setup

This chapter will describe the how the process of training our solution is performed, covering from how the images of the door handles are captured to the training process for the handle and the ensemble classifiers, as well as how their performance is measured.

This procedure is divided in four parts, data gathering, example preparation, classifier training and ensemble classifier training. These parts are independent in the sense that the pipeline is modular, on one session the data can be gathered and used several days or weeks later. This allows a greater deal of flexibility as individual components can be modified without having to start over the whole procedure.

There are some constraints that must be respected, if the image generation algorithm is changed, the already trained classifiers are no longer valid since the generated images might be significantly different. And if the handle classifiers are changed, the ensemble classifier must be retrained.

7.1 Data gathering

All supervised machine learning algorithm needs a labelled training set of data, the first step towards having it is capturing the data.

Since we did not know the amount of required examples for correct learning at first, we chose to gather as much data as possible in places easily accessible, and work with it until we had evidence that we required to obtain more data to perform successful classification.

For that reason, we chose to gather images from the PAL Robotics office and from friends and relatives homes. The office images would not be used for training, since they would be

saved for experiments and for evaluating the performance of the algorithms.

The point clouds had to be gathered from a point of view as similar as possible to the one the robot will have when trying to open doors, but taking the robot outside of the office to capture the images requires an investment in time and money by the company, to avoid that we mounted the RGB-Depth on a tripod that could be carried around to gather images from the robot's point of view as is displayed in Figure 7.1.

Figure 7.1: Asus Xtion Sensor on top of a tripod allowed us to capture images from a point of view similar to the robot's

The data was saved in its raw format, as RGBXYZ point clouds, without any processing done using ROS Bags which is a file format in ROS that can store any kind of ROS messages. This allowed us to reuse the data recorded at any time for experimenting with the handle detector algorithm. The only downside is that each point cloud stored is about 2.5MB after compression.

We recorded every door handle from different distances and angles, simulating different approaches made by the robot, also when possible the lighting conditions were changed by turning on and off the lights to train our solution against those different conditions.

The examples were labelled as they were captured, we gathered images from round door knobs, from the most common type of handle, which is a horizontal bar and we gathered negative examples by recording walls and other areas that might look like doors but had no handle. Some examples of them can be seen in Figure 7.2

7.2 Example preparation

Once the examples have been collected, the 2D image of the candidate handles must be generated, to do the rosbag must be processed by the handle detector, which will read each point

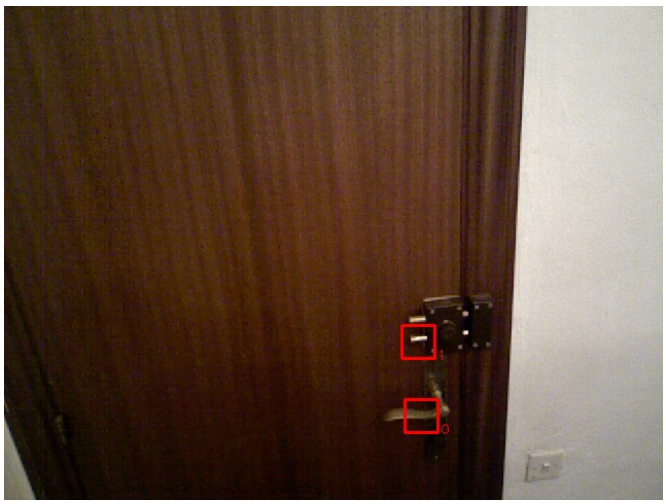
Figure 7.2: Color images of the three labels that will be used in the classification. The first row are Type 0 examples, where no handles are present, Type 1 handles in the middle row and Type 2 handles in the bottom low. In the training the shadow images binary images are used, not this color image examples.

cloud and look for possible handles. handles.

To do so in an environment as close as possible to the real conditions in the robot, we use a simulated version of REEM, and feed the point clouds to the handle detector as if they were generated by the robot's head mounted sensor. Using the simulated robot provides us with the real geometric transformations used by the robot for converting the point cloud from a frame of reference such as the head mount to the base frame of the robot. Having high accuracy in this step is key to the success of the training process, since these geometric transformations will alter the point of view used to capture the images that will be passed to the handle classifier.

The results of executing it for all the point clouds is a set of images of the candidate handles detected in each ROS Bag. Even though the ROS bags were labelled already, we might have some candidate handles that are not real handles. So we need to examine all the images manually and relabel them. The false positives of the handle detector are key to our training, since they will be what the classifier has to differentiate from a real handle. Figure 7.3 depicts the situation where a false positive has been generated by the handle detector.

Once all the newly created images have been generated and labelled, the training can begin.



(a) Original image with handle detections

(b) Positive detected handle

(c) False detected handle

Figure 7.3: The top image is a projection of the point cloud contained in the rosbag, the red boxes are centered on the centroid of the detected handles. The image on the bottom left corresponds to the handle of the door in the image, whereas the image on the right corresponds to the door bolt

7.3 Classifier training

The goal of this step is to train all the combinations of image features, classification algorithms and their parameters to compare their performance and choose the best configurations for our ensemble classifier.

To make this task easier, we wrote a set of tools that created a layer of abstraction on top of the image features and classifications algorithms of OpenCV that allowed us to use them interchangeably without caring about each configuration details. We made an effort to make this layer also an abstraction for our problem, so it can be used in any classification application and might be released as open source in the future.

This application inputs are as many image directories as labels we want to classify, it is assumed that the first directory of the list contains the positive examples for the first label,

and the rest of directories are negative examples for the first label and positive examples for the rest of labels.

The application is configured by giving it a list of PCA retained variances, a list of feature extraction algorithms, and a list of binary classifiers with it's training parameters.

We also prepared the application so after extracting the features and computing and applying the PCA, it performed the training of the binary classifiers in parallel, one thread per binary classifier. This speed up our training time by 2/3.

Algorithm 1 Classifier training

```
for each feature_extractor in feature_extractor_list do
  features=extract_feature(feature_extractor, image_list)
  features=normalize(features)
  for each pca_variance in pca_variance_list do
    reduced_features=apply_pca(features, pca_variance)
    for each classifier in classifier_list do
      for each classifier_parameters in classifier_parameters_list do
        for each label in label_list do
          start_training_thread(label, classifier, classifier_parameters, reduced_features)
        end for
        wait_training_threads()
      end for
    end for
  end for
end for
```

Algorithm 2 Training thread

```
for kfold = 0 to 10 do
  train_set=get_train_set(reduced_features, kfold)
  validation_set=get_validation_set(reduced_features, kfold)
  trained_classifier=train(label, classifier, classifier_parameters, train_set)
  train_evaluation = evaluate_classifier(label, trained_classifier, validation_set)
  save_evaluation(train_evaluation)
  validation_evaluation = evaluate_classifier(label, trained_classifier, validation_set)
  save_evaluation(validation_evaluation)
end for
{Perform a final training and save the trained model}
trained_classifier=train(label, classifier, classifier_parameters, reduced_features)
save_classifier(trained_classifier)
```

Each training is performed using 10-fold cross validation[K$^+$95], where the samples are divided into 10 equal sized partitions, and the training is performed 10 times, using 9 partitions for training and the remaining one for validation of the trained model. This allows us to compare the performance of the different models without favouring those which tend to overfit the training data.

After each training we analyse the performance of the learning algorithms by classifying

all the data in the training set and compare it to the performance of running the validation set. We later use this data to determine which algorithms work better, which are over-fitting or under-fitting and determine which changes need to be performed to the algorithm before training again.

A list of all the misclassified examples is saved for each algorithm, with this information we can detect which are the most frequent problems in the classification, and also we can chose to combine algorithm that complement each other for the ensemble classifier.

7.4 Ensemble Classifier training

In order to build the ensemble classifier, we need to select the set of binary handle classifiers that provides the best performance.

Determining this set is not a trivial matter, the solution that guarantees us the best performance would be to implement an exhaustive way of evaluating the performance of all the possible classifier combinations. The number of combinations is $C(n, r) = n!/r!(n - r)!$ where n is the number of trained classifiers and r the number of classifiers for each handle type used in the ensemble classifier.

The most simple approach would be to pick the r classifiers with the highest performance score, but it might lead to similar classifiers being selected.

A different approach would be to start by selecting the best performing binary classifier and select the rest of classifiers based on how well they complement the initial classifier. This can be done by selecting the ones that classify properly the images that the initial classifier got wrong.

Our idea is to build several ensemble classifiers with the different approaches mentioned in this section and compare their performance.

CHAPTER 8

Results

In this section the results of the training and evaluation of our classifiers are presented and explained.

8.1 Considerations

Due to the large amount of possible combinations of feature extraction algorithms, training algorithms and PCA parameters the training time became a significant problem.

During the initial experiments the training would have taken 18 weeks. After some smaller scale experiments we chose to stick with 1 set of parameters for each feature extraction algorithm, as the the results were much more affected by the changes in the machine learning algorithm parameters and the different applications of Principal Component Analysis. After freezing the set of parameters of the feature extraction algorithm and parallelising our training system, the training was 9 times faster.

8.1.1 Data sets

We used always the same set of data in all the individual training processes, described in Table 8.1.

When training a binary classifier to detect Type 0 images, the Type 0 images were given as positive examples and the Type 1 and 2 as negative examples, when training the Type 1 binary classifier, the Type 1 images where positive and the other types negatives, and the same approach for Type 2 binary classifier.

Image type	**# images**
Type 0 (no handle)	1380
Type 1 (round handle)	575
Type 2 (horizontal handle)	452

Table 8.1: Number of images in the training set for each handle type

During the training we use 10-fold cross-validation to obtain the test performance measures, and use this data to select the best binary classifiers to use in the ensemble classifier.

After the best binary classifiers have been selected, and the ensemble classifier trained with them, we use the test data set described in Table 8.2 to train the ensemble classifier.

And finally we use the data set described in Table 8.3 to evaluate the performance of the ensemble of classifiers.

The data sets used in the ensemble contain images of doors of the Pal Robotics office, which are present in neither the training set nor the validation set of the binary classifiers.

Image type	**# images**
Type 0 (no handle)	190
Type 1 (round handle)	60
Type 2 (horizontal handle)	74

Table 8.2: Number of images in the train set of the ensemble classifier for each handle type

Image type	**# images**
Type 0 (no handle)	144
Type 1 (round handle)	27
Type 2 (horizontal handle)	58

Table 8.3: Number of images in the test set of the ensemble classifier for each handle type

8.1.2 Experiment Configuration

In the rest of this section and in all the tables and graphs that are related to the handle classifier training the following configuration has been used and will be referred by the bold abbreviation.

COV-V Covariance Matrix feature extraction algorithm with 100 pixel window size and sub-windows of 33x33, 33x66, 33x99, 66x33, 66x66, 99x66, 99x33, 99x66 and 99x99 pixels. With with V% of retained variance after PCA.

PIX-V Pixel values of a 100x100 pixel image generated with a canny edge detector. With with V% of retained variance after PCA.

HOG-V Histogram of Oriented Gradients with 128 pixel window size, 16 pixel block size, 8 pixel cell size and a maximum of 32 detection window increases. With with V% of retained variance after PCA.

ADA-DSC-D Discrete AdaBoost with D maximum decision tree depth.

ADA-REA-D Real AdaBoost with D maximum decision tree depth, uses confidence-rated predictions.

SVM-LIN Support Vector Machines with Linear Kernel.

SVM-RBF Support Vector Machines with Radial Basis Function kernel.

KNN-K K-Nearest Neighbours with K neighbours.

RFP-D Random Forest Predictor with D max tree depth.

8.1.3 Performance Metrics

For each combination of PCA application, feature extraction and machine learning algorithm, the following metrics have been obtained both for the training dataset and the validation data set.

TP True positives

FP False positives, or Type I errors

TN True negatives

FN False negatives, or Type II errors

PPV Positive predictive value or precision

$$PPV = TP/(TP + FN)$$

TPR True Positive Rate or recall

$$TPR = TP/(TP + FN)$$

ACC Accuracy, obtained using the formula

$$ACC = (TP + TN)/(TP + TN + FP + FN)$$

F F-Measure or F-Score, the harmonic mean of precision and recall.

F05 $F_{0.5}$ Measure, a variant of F-measure $F_{0.5}$ weights precision higher than recall.

$$F_{0.5} = 1.25 \times \frac{PPV \times TPR}{0.25 \times TPR + PPV}$$

We chose $F_{0.5}$ as our most interesting metric, because it takes into account both precision and recall, but with the slightly increased importance of precision we try to encourage that the selected algorithms are less error prone and do not lead the robot to perform an opening motion where there is no handle.

8.1.4 PCA

We performed the training and experimented with the results of not applying PCA and applying PCA with different values for the retained variance. The table 8.4 displays the number of features for each different feature algorithm.

	COV	**HOG**	**PIX**
PCA-100	1296	8100	10000
PCA-99	51	1731	2133
PCA-85	5	371	1243
PCA-75	3	167	907

Table 8.4: Number of features generated by each feature extraction algorithm without PCA and having applied different PCA configurations, indicated as PCA-N, where N is the retained variance in percentage

8.2 Classifier Performance

The full tables of results of the training and validation sets are displayed in the Appendix B

For each type of handle there are 156 possible combinations of the 3 features (COV, PIX and HOG), 4 PCA Values (75, 85, 99, 100) and 13 machine learning algorithms (KNN-3, KNN-7, KNN-11, KNN-21, SVM-RBF, SVM-LINEAR, ADA-REA-2, ADA-REA-5, ADA-DSC-2, ADA-DSC-5, RFP-2, RFP-6, RFP-18).

It is worth explaining why we used a higher tree depth in the Random Forest Predictors when compared to AdaBoost tree depth. Using a lower tree depth in RFP caused our classifiers to fail to be able to learn, whereas AdaBoost with higher tree depth only caused more over-fitting of the training set.

To make this section more readable, we have reduced the number of combinations to 120 by removing some elements due to their similarity to other elements while preserving at least 2 examples of each examples of each variable we tuned in each machine learning algorithm. The elements that will not be displayed in this section are: KNN-11, KNN-21 and RFP-2.

	COV 075	COV 085	COV 099	COV 100	HOG 075	HOG 085	HOG 099	HOG 100	PIX 075	PIX 085	PIX 099	PIX 100
ADA-DSC-2	.887	.904	.913	.935	.906	.906	.897	.893	.898	.900	.896	.884
ADA-DSC-5	.895	.920	.933	**.952**	.919	.909	.895	.912	.910	.896	.906	.885
ADA-REA-2	.887	.902	.915	.932	.905	.901	.897	.900	.893	.903	.905	.858
ADA-REA-5	.885	.913	.921	.942	.905	.901	.881	.907	.872	.878	.873	.876
KNN-03	.900	.938	**.958**	**.958**	**.955**	**.954**	.910	**.954**	.945	**.952**	**.953**	**.956**
KNN-07	.907	.938	.952	**.953**	.948	.942	.938	.939	.935	.950	.949	.951
KNN-21	.909	.927	.947	.947	.928	.926	.925	.925	.922	.939	.938	.939
RFP-03	.885	.911	.890	.913	.751	.702	.626	.848	.687	.670	.629	.737
RFP-07	.905	.928	.932	.946	.906	.886	.776	.929	.856	.839	.809	.883
RFP-19	.902	.937	.937	.949	.930	.921	.848	.931	.909	.893	.884	.913
SVM-LIN	.887	.886	.925	.924	.926	.935	.929	.934	.922	.922	.922	.923
SVM-RBF	.890	.915	.933	.939	.947	.948	.945	.947	.922	.924	.926	.933

Table 8.5: Type 0 classifiers $F_{0.5}$ Measure average results for the cross-validation data set. Highest 10 results in bold.

	COV 075	COV 085	COV 099	COV 100	HOG 075	HOG 085	HOG 099	HOG 100	PIX 075	PIX 085	PIX 099	PIX 100
ADA-DSC-2	.863	.882	.926	.947	.947	.938	.929	.964	.906	.899	.894	.820
ADA-DSC-5	.850	.923	.956	.964	.966	.966	.962	.972	.936	.940	.927	.876
ADA-REA-2	.846	.889	.934	.947	.940	.943	.933	.968	.929	.918	.920	.843
ADA-REA-5	.834	.920	.959	.957	.965	.952	.953	.972	.921	.918	.928	.834
KNN-03	.848	.910	.927	.942	**.983**	**.983**	.979	.978	.949	.913	.874	.907

KNN-07	.842	.901	.928	.918	**.983**	.979	.969	.971	.947	.915	.845	.880
KNN-21	.843	.897	.908	.909	.956	.953	.952	.955	.919	.894	.797	.857
RFP-03	.870	.900	.719	.908	.159	.0	.0	.956	.065	.065	.0	.775
RFP-07	.875	.921	.943	.958	.926	.808	.017	.976	.829	.723	.577	.849
RFP-19	.869	.931	.957	.962	.966	.951	.748	.976	.922	.902	.901	.855
SVM-LIN	.840	.876	.922	.942	**.981**	.977	**.983**	**.980**	.949	.942	.939	.953
SVM-RBF	.845	.885	.936	.940	**.987**	**.988**	**.985**	**.987**	.946	.958	.947	.953

Table 8.6: Type 1 classifiers $F_{0.5}$ Measure average results for the cross-validation data set. Highest 10 results in bold.

	COV 075	COV 085	COV 099	COV 100	HOG 075	HOG 085	HOG 099	HOG 100	PIX 075	PIX 085	PIX 099	PIX 100
ADA-DSC-2	.544	.677	.763	.822	.776	.743	.731	.762	.705	.686	.675	.580
ADA-DSC-5	.644	.732	.842	**.883**	.811	.810	.804	.807	.680	.677	.671	.625
ADA-REA-2	.587	.714	.765	.834	.772	.743	.748	.772	.678	.687	.666	.588
ADA-REA-5	.564	.744	.783	.834	.803	.790	.706	.757	.635	.602	.642	.539
KNN-03	.620	.752	.841	.846	**.902**	**.889**	**.891**	.873	.865	.828	.836	.827
KNN-07	.643	.749	.802	.810	.875	.860	.865	.856	.859	.827	.817	.821
KNN-21	.663	.719	.763	.755	.838	.831	.836	.83	.806	.780	.783	.780
RFP-03	.254	.700	.0	.754	.0	.0	.0	.461	.0	.0	.0	.032
RFP-07	.649	.758	.824	.861	.706	.394	.0	.829	.052	.021	.0	.502
RFP-19	.662	.795	.860	**.880**	.854	.751	.357	.873	.248	.191	.153	.687
SVM-LIN	.0	.651	.760	.773	.870	.862	**.876**	.872	.765	.762	.764	.766
SVM-RBF	.0	.704	.813	.812	**.910**	**.895**	**.895**	**.897**	.786	.781	.779	.766

Table 8.7: Type 2 classifiers $F_{0.5}$ Measure average results for the cross-validation data set. Highest 10 results in bold.

8.3 Identifying the best binary classifiers

Before building the ensemble classifier we need to determine which of the binary classifiers has the best performance, and this decision cannot be based solely on selecting the one with the highest average $F_{0.5}$.

To do so we'll use the information obtained from the 10 runs of cross validation to examine the variance of the results and we'll use Tukey's range-test also known as Tukey's HSD test

(Honestly Significant Difference) test to determine if the the classifiers in the top-10 group are significantly different from each other. We'll keep only the classifiers that are not significantly different from the best performing one.

The Tukey HSD test uses the Studentized range statistic to determine how large the difference between the means of any two particular groups must be in order to be regarded as significant. It applies simultaneously to the set of all pairwise comparisons $\mu_i - \mu_j$ and identifies any difference between two means that is greater than the expected standard error.

8.3.1 Best Type 0 classifiers

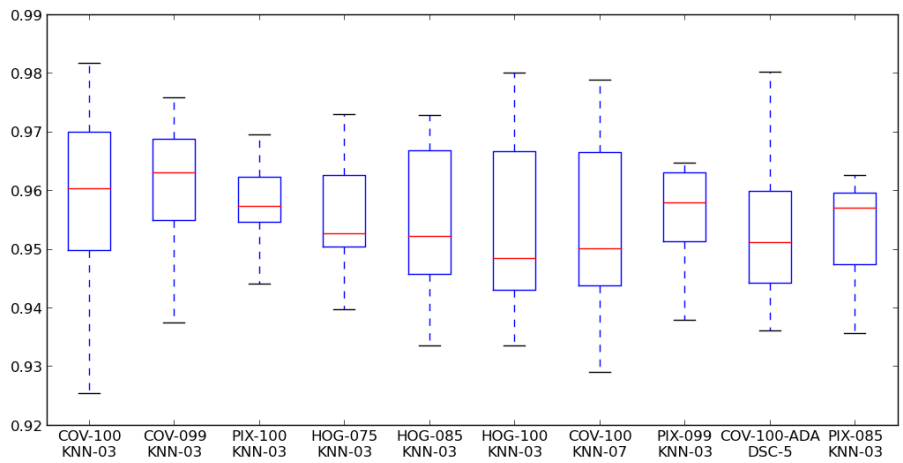


Figure 8.1: Box plot of the cross-validation data of the 10 Type 0 classifiers with the highest $F_{0.5}$ score, the read line indicates the median.

In figure 8.1 the cross-validation data for the top 10 performing classifiers for Type 0 images is displayed.

After performing Tukey's HSD test we determine that the 10 classifiers are not significantly different, therefore we will pick the best classifier from this group with other criteria.

8.3.2 Best Type 1 classifiers

In figure 8.2 the cross-validation data for the top 10 performing classifiers for Type 1 images is displayed.

The Tukey HSD test determined that only the 8 top performing classifiers were equivalent, HOG-075-KNN-07 and HOG-100-SVM-LIN were significantly different from HOG-085-SVM-RBF.

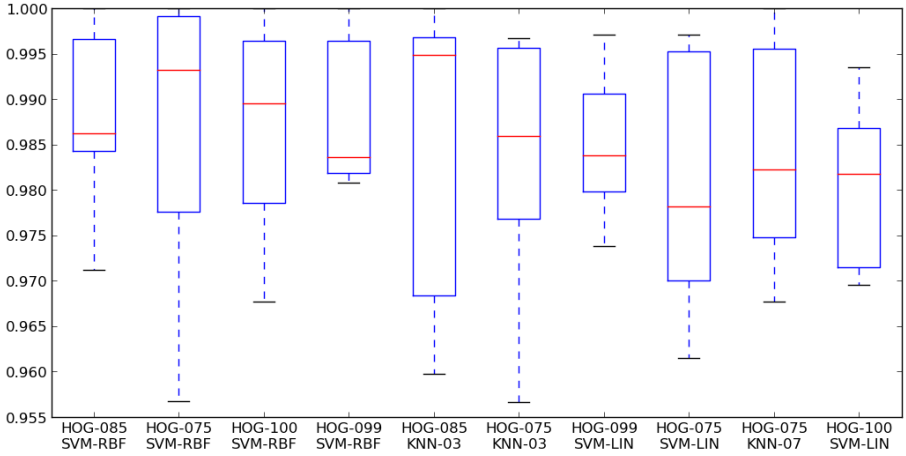


Figure 8.2: Box plot of the cross-validation data of the 10 Type 1 classifiers with the highest $F_{0.5}$ score, the read line indicates the median.

8.3.3 Best Type 2 classifiers

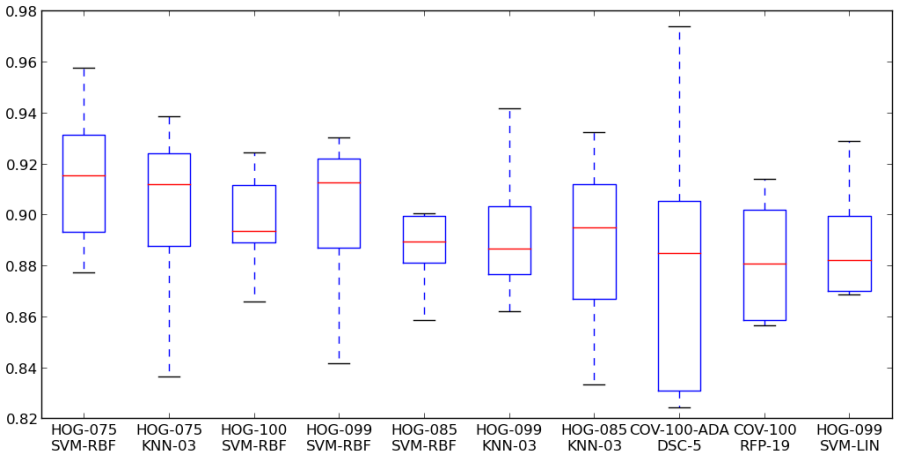


Figure 8.3: Box plot of the cross-validation data of the 10 Type 2 classifiers with the highest $F_{0.5}$ score, the read line indicates the median.

In figure 8.3 the cross-validation data for the top 10 performing classifiers for Type 2 images is displayed.

The Tukey HSD test determined that the top 7 classifiers with the highest $F_{0.5}$ scores were not significantly different and that COV-100-ADA-DSC-5, COV-100-RFP-19 and HOG-099-SVM-LIN were.

8.3.4 Classifier memory usage

When building the ensemble classifier we do not only need to take into account the performance of the binary classifiers, but also their resource usage.

CPU time is not an issue, because the handle detection takes from 5 to 10 seconds to process a point cloud before computing a 2D image of the candidate handle. In all our tests the ensemble classification is performed in less than a few hundred milliseconds, therefore the computational cost of the classification is meaningless.

But to perform the classification with the ensemble of classifiers, we need to load to memory not only the classifiers but also the PCA components to memory.

	COV	**HOG**	**PIX**
PCA-100	0	0	0
PCA-99	1.1	222	337
PCA-85	0.125	48	197
PCA-75	0.084	22	144

Table 8.8: Size in MB of the PCA components XML files for each feature and PCA-variance combination.

In figure 8.4 the size of the trained classifiers is displayed. As expected, the file size increases as there are more features used, this is specially true for K-NN algorithms, since the only way of storing them is to store the examples in the training set as feature vectors.

The size displayed in Table 8.8 and Figure 8.4 is the size of the XML file that contains the models. Once loaded in to memory the RAM memory space used is much less since the data is not saved as characters but as floats. Nevertheless the process of loading these files can be quite slow and the disk size is a good indicator of the memory size that will be used.

The data presented above will be used as a factor when deciding which classifiers to use when there's more than one classifier with equivalent performance. We'll try to reuse PCA to avoid keeping too many PCA components in memory and to avoid computing different features for the same image. Also we'll try to use the classifiers that use the smaller PCA variance values since they are use less memory overall.

8.4 Ensemble classifier

In the previous chapter we explained the two approaches that we would use to pick the best classifiers for the ensemble classifier, pick the best individually performing classifier and the $r-1$ classifiers that best complement it and exhaustively examine all possible combinations of classi-

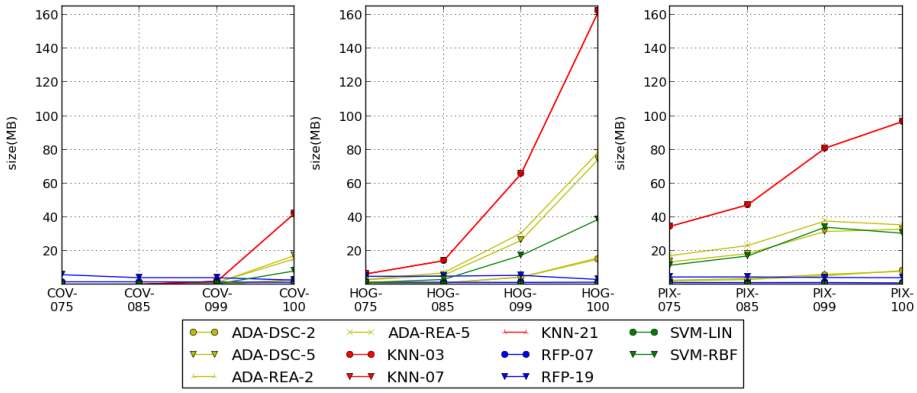


Figure 8.4: Average size in MB for all handle types for the XML representation of each classifier saved in disk.

fiers. Unfortunately the last possibility had to be discarded, since the number of combinations becomes unmanageable as can be seen in Table 8.9.

r	combinations
1	156
2	12090
3	620620
4	23738715

Table 8.9: Number of images in the validation set for each handle type

Therefore to select a set of classifiers to use in the ensemble classifier, we used the other mentioned approach and generated 4 sets of classifiers. We have named them C-r, where C-r is the set of the r classifiers that best complement each other as explained in the previous section, where r goes from 1 to 4.

The best classifiers were determined to be HOG-075-KNN-03 for Type 0 images, HOG-075-SVM-RBF for Type 1 images and HOG-075-SVM-RBF for Type 2 images. They were selected among the best performing classifiers as explained in section 8.3, they all use the same PCA file, reusing the same file removes the need to load multiple PCA files in memory and since it is a low variance PCA file, the PCA components and the classifiers don't use too much space in memory once loaded.

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**
C-1	.799	**.885**	.872	**.888**	.727	.614	.418	.393	.072	.155	.059
C-2	.830	.850	.848	.861	.862	.809	.681	.609	.568	.413	.205
C-3	.871	**.881**	.843	.871	**.885**	**.880**	**.893**	0.781	0.465	.0	.260
C-4	.850	.872	.877	**.882**	.877	**.886**	**.900**	**.889**	.654	.661	.275

Table 8.10: $F_{0.5}$ performance of the Ensemble classifiers, the column indicates the number of hidden neural network layers of the ensemble classifier, and the row the id of the set of classifiers used for training. The 10 classifiers with the highest $F_{0.5}$ score are in bold.

To select one of the best performing classifiers displayed in Table 8.10 we perform again the Tukey HSD test to determine if there's a significant difference between them.

As can be seen in Table B.4 in Appendix B all the ensemble classifiers trained except C4-ANN-8 can be considered not significantly different according to the cross-validation results, therefore we will choose to use the C1-ANN-4 ensemble classifier, since it requires to use less binary classifiers and PCA files than the rest of ensemble classifiers with 3 or 4 classifiers per image type.

CHAPTER 9

Evaluation of Results

In the previous chapter the data we obtained from the training indicated that there is not a unique combination of features and supervised learning algorithm that clearly outperforms the rest at this particular task.

Nevertheless, in this chapter we'll describe those combinations that perform slightly better than others.

9.1 Feature and classifier combination analysis

The heat maps of Figure 9.1 show the $F_{0.5}$ score of the most significant feature/classifier combinations for each handle type.

From it we can extract several conclusions:

- From figure 9.1 we can deduce that the classification of Type 2 handles was significantly more complex than the Type 1 handles, the overall $F_{0.5}$ scores are lower. This is probably due to how the images are processed and the diversity of shapes in the horizontal handles. After the processing done in the handle detector, most of the round handles have the same appearances in the 2D images we use, but the horizontal handles have a much more varied appearance.

- We can also appreciate that for all classifier types KNN and SVM seem to perform better than the rest, specially with HOG features. Also some instances of Adaboost with Covariance Matrices have a remarkable performance for all handle types.

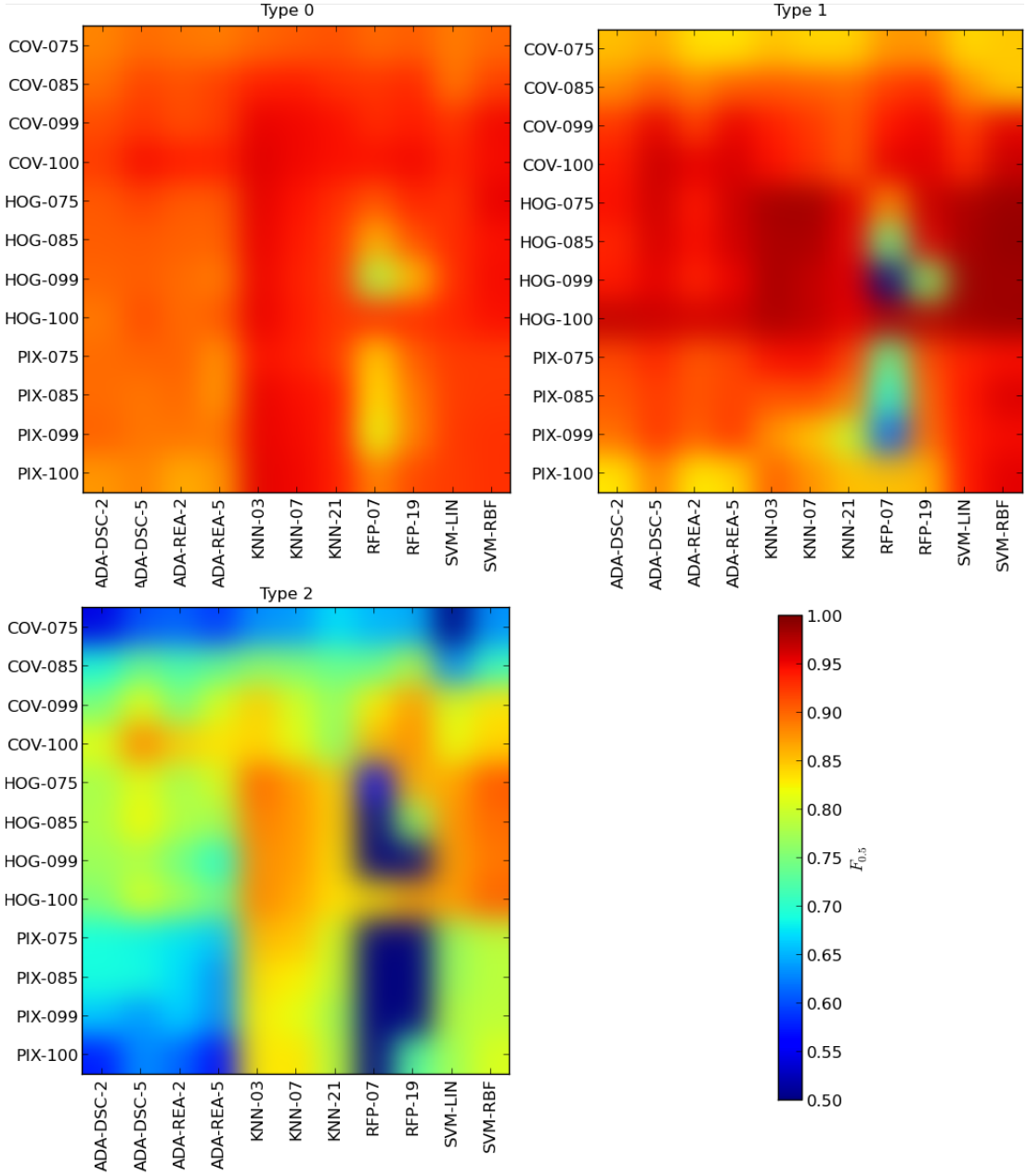


Figure 9.1: $F_{0.5}$ score for the most significant feature/classifier combinations. The scale has been adjusted to the 0.5 - 1.0 range to give the image more contrast, $F_{0.5}$ scores lower than 0.5 are displayed as 0.5.

- COV features with low retained variance and Pixel edge detection features with no PCA are the worst performing features across the board, this will be explained more in depth in the next section.

- The best performance algorithm overall seem to be the Support Vector Machine, which has consistently better performance for all handle types. Specifically it has a significant better performance with Histogram of Oriented Gradients. In the literature HOG is often used with SVM and in these results it can be seen experimentally the great synergy HOG and SVM have.

- We were initially surprised that AdaBoost performed worse than K-Nearest Neighbours in most of the situations, this will be analysed in Section 9.3.

9.2 PCA effect analysis

In figure 9.2 the average of the $F_{0.5}$ score for all handle types in the validation set is displayed.

The bad performance of Random Forest Predictors is due to some instances of training for Type 1 and Type 2 handles where the classifier would predict only positive answers, this situations prevents the $F_{0.5}$ from being calculated and we assigned it a score of 0, this brings the average down.

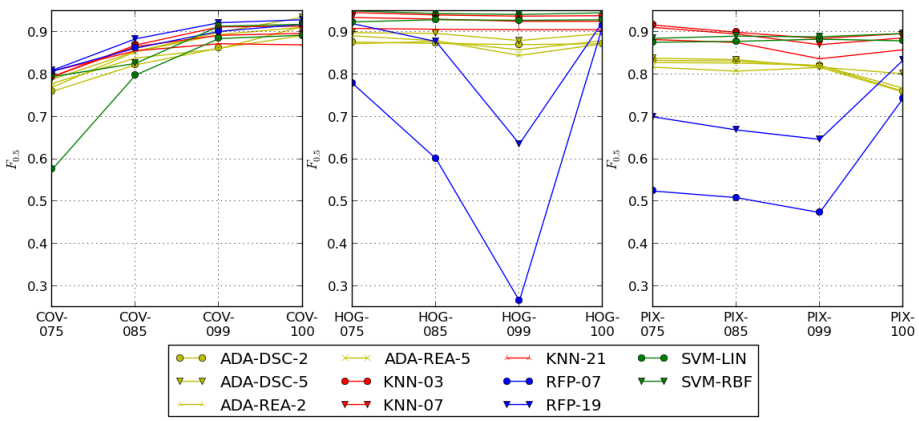


Figure 9.2: $F_{0.5}$ score average for all handle types in the validation set.

- PCA has a much greater impact in Covariance Matrices, where the number of features with 75% variance retained was just 3 and with 99% retained variance of 59. We can deduce that for covariance matrices the PCA was too aggressive and too much significance was lost. Also since the number of features was almost an order of magnitude inferior to the other feature extractor algorithms, PCA with 99% retained variance would have been enough since it retained most of the significance with just 4% of the initial feature vector size.

- With regards to Histogram of Oriented Gradients and Pixel Edge detector, the PCA had a much lower impact. But the results indicate that when the number of features of the PIX

feature extractor increase too much the performance of some classifiers such as AdaBoost gets worse. In this case some features without too much significance were present and the hypotheses space was too large and these machine learning algorithms were over-fitting.

9.3 Over-fitting analysis

Due to the results observed in the previous two sections, we decided to compare the performance of the classifiers in the training set with the performance in the validation set, this comparison is shown in figure 9.3.

- The training results are very interesting, we can see the effect of small variations on the parameters of the machine learning algorithms. Specially for AdaBoost and RFP. For the Type 2 handle we can see that ADA-REA-5 has a much better performance than ADA-REA-2 in the training set, but in the validation set it performs very similarly and in some situations worse. We can deduce that increasing the AdaBoost decision trees depth to 5 caused over-fitting, specially for the Real Adaboost.

- A similar effect can be appreciated with RFP-19 and RFP-07, for Type 0 and Type 1 images, the performance increase in the validation set is significant specially with PIX features, where RFP performs much better with a tree depth of 19. The opposite happens for Type 2 images, where RFP with tree depth of 19 has near perfect performance, but is unable to classify properly the validation set.

- As we suspected the performance of almost all AdaBoost configurations on the training set is far superior than the rest of the classifiers, but on the validation it doesn't perform remarkably well. After some research we've seen using simpler weak classifiers AdaBoost is less prone to over-fitting, if we had used stumps (1-level decision trees) instead of the ones used and less decision trees we might have gotten less over-fitting.

- The KNN and SVM configurations we used seem to not have over-fitted, their higher performance in the training set is expected, but they also perform significantly well in the validation set.

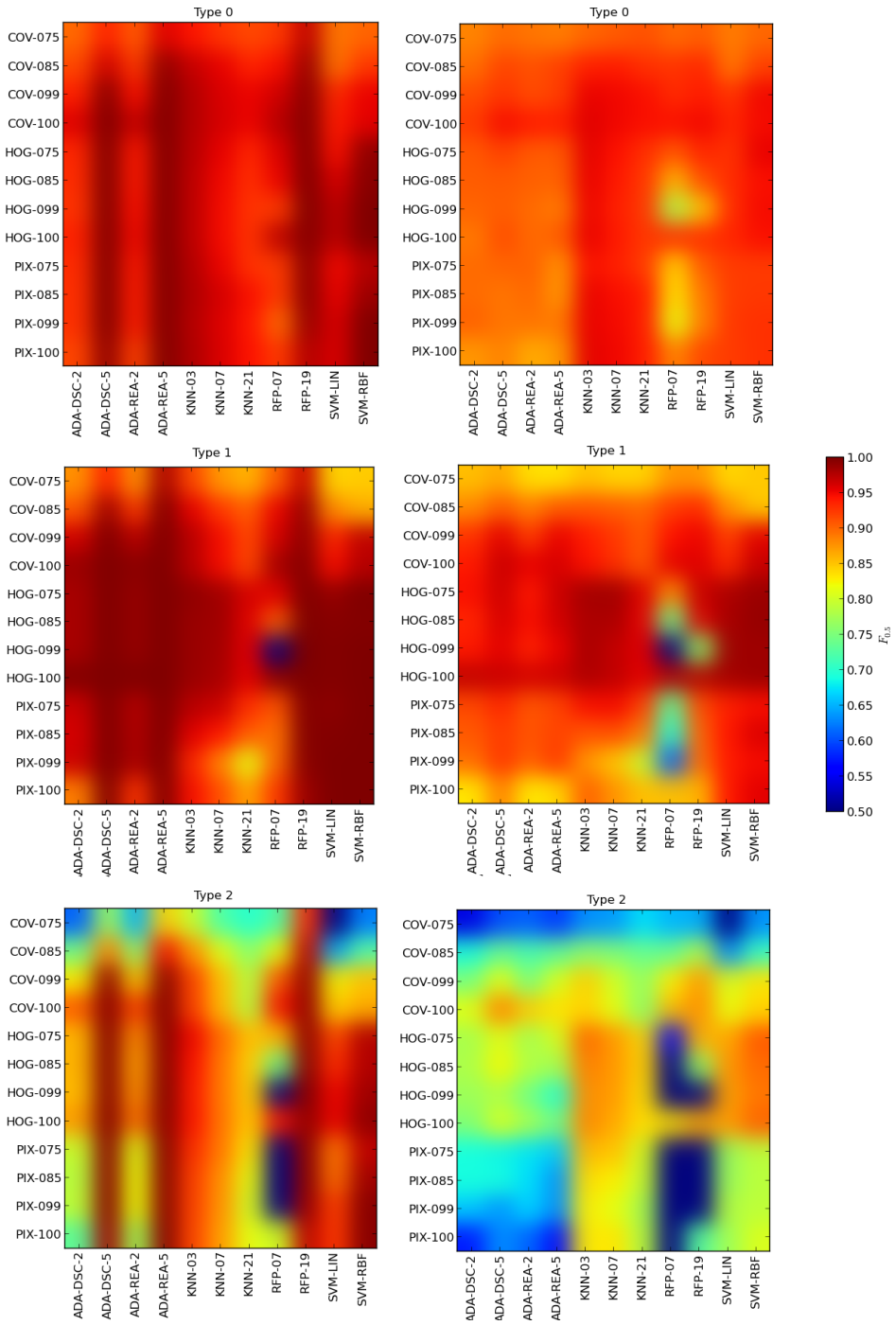


Figure 9.3: $F_{0.5}$ score for the most significant feature/classifier combinations. Training set results are on the left and validation set on the right. The scale has been adjusted to the 0.5 - 1.0 range to give the image more contrast, $F_{0.5}$ scores lower than 0.5 are displayed as 0.5.

9.4 Ensemble classifier performance

During the planning of this thesis we did not want to use a single multi-class classifier, and instead we would use a group of binary classifiers to determine the type of handle. To resolve the situations where the classifiers provided conflicting answers and to try to improve their performance we chose to build an ensemble of classifiers to combine their output.

After selecting the 3 classifiers that would be used in the ensemble classifier, we evaluated the performance of the 3 binary classifiers individually and of the ensemble classifier that uses them against the test set, a group of images not used yet in any step of the process. The results of this the evaluation can be seen in table 9.1

Name	$\mathbf{F}_{0.5}$	**F**	**ACC**	**PPV**	**TPR**
T0-HOG-075-KNN-03	0.835	0.861	0.816	0.818	0.909
T1-HOG-075-SVM-RBF	0.874	0.892	0.973	0.862	0.925
T2-HOG-075-SVM-RBF	0.918	0.923	0.960	0.915	0.931
Ensemble C1-ANN-4	0.920	0.924	0.943	0.918	0.929

Table 9.1: Performance of the best binary classifiers and the ensemble classifier on the test data set.

The Ensemble Classifier created has an $F_{0.5}$ score of 0.920 on the test set, higher than any of the binary classifiers that form it. Not only does the ensemble classifier resolve the situations in which the answers of multiple binary classifier conflict, but in doing so it improves the predictive power by combining the output of the 3 separate binary classifiers.

CHAPTER 10

Discussion and conclusions

In this thesis we have developed a solution to detect door handles using a RGB-Depth sensor and an ensemble of binary supervised classifiers to determine the type of handle. The main goal of the thesis was to develop a robust solution for door handle detection, that would have a high detection rate and a very low amount false detections. A secondary goal was to experiment and combine several binary supervised classification and feature extraction algorithms and evaluate their performance with different applications of Principal Component Analysis.

Our first goal has been successfully achieved. To do it we have built a system that preprocesses the sensor data using a handle detector which provides a 2D image for each possible candidate handle, each candidate is examined by a handle classifier application, which is an ensemble of individually trained binary classifiers, which in the end determine if the candidate is a handle or not, and if it is of which type it is.

The handle detector uses the data provided by the RGB-Depth sensor to look for the common elements of a door handle such as being on top of a flat perpendicular to the ground surface, being at a certain height and of a certain size. To deal with handles that have reflective surfaces, which are invisible to the RGB-Depth sensor, we look for handles in the areas void of point cloud information, that meet the position requirements for containing a handle. To create a unified solution for both specular and non-specular handles, we delete all the points on top of the door plane to treat all handles as if they were specular.

To reduce the variability introduced by capturing the point clouds from different camera positions, we generate a 2D image by projecting the point cloud to a point on view in front of the handle in the direction of the normal of the plane of the door, this 2D image contains the points in the plane of the door that from which we have no 3D information. After filtering

the areas void of information by size and shape, we provide them to the handle classifier as candidate door handles.

In order to build the best handle classifier possible, we combined three feature extraction algorithms (Histogram of Oriented Gradients, Covariance Matrices and Canny Edge Detection), 13 configurations of machine learning algorithms (KNN, SVM, AdaBoost and Random Trees) with different parameters and 4 different applications of Principal Component Analysis for a total of 156 binary classifiers for each handle type and another set to detect whether there was a handle in the image or not.

Each of these classifiers was trained with a set if over 2400 images and its performance evaluated using cross-validation with the training data.

We used the $F_{0.5}$ measure as the performance metric to optimize, since it provides a balance between recall and precision with a higher emphasis on precision to avoid false positives.

After obtaining the performance of each of the 468 classifiers, 312 for two handle types and 156 for the no-handle classifier, we selected the 10 classifiers in each group that had the best performance, and determined if there was a performance difference using Tukey's honest significant difference test on their cross-validation $F_{0.5}$ measures. From the group classifiers that were not significantly different we chose the ones that used the lowest amount of computer resources and marked them as the best binary classifiers.

Once these best binary classifiers had been selected, we evaluated which other binary classifiers had labelled properly the samples miss-classified by the best classifiers, and prepared sets of classifiers with combinations of the best classifier and 1, 2 and 3 complementary classifiers for each image type.

From the created sets of classifiers we trained several ensemble classifiers, using feed-forward neural networks with varying number of hidden layers to determine the best combination of binary classifier set and neural network structure. The training was done on a data set with 324 images not present in the binary classifier training set.

Once again, we used Tukey's HSD test to determine if there was any significant difference among the results and chose the ensemble that required less resources.

The chosen ensemble of classifiers was formed by the 3 selected best binary classifiers using a neural network with 4 hidden layers, there were no need to use the complementary binary classifiers.

We evaluated the performance of the chosen ensemble classifier and it's individual classifiers using a test set with 229 new images and found out that the performance of the ensemble of classifier was higher than the performance of any of the individual binary classifiers used in it.

In the result analysis we determined that SVM and KNN with HOG descriptors provided the best performance and the least over-fitting for the binary classifiers among all the image types. We also determined that we were over-zealous when applying PCA on Covariance Matrices, this caused a lot of significance to be lost and led to the worst performance of all the feature extraction algorithms when PCA didn't preserve enough variance from the original feature vector.

We conclude this final analysis with some directions for future work, once all the classifiers and feature extractor algorithms were integrated, we should have done a training and thorough analysis even if the handle detector was not finished and we did not have the final set of images. Had we done this we would have anticipated most of the problems that appeared at the end of the thesis and had not enough time to correct, for instance we would have realized that the parameters chosen for AdaBoost would lead to over-fitting the training set and having a bad performance with other images, we could have also explored better the parameter space around the most successful classifiers to find even better configurations of PCA, Feature selection and machine learning algorithm parameters.

APPENDIX A

Binary classifier results

A.1 Type 0 cross-validation results

Name	**F05-avg**	$\sigma_{F_{0.5}}$	**ACC**	**TPR**
COV-0100-ADA-DSC-2	0.934	0.025	0.923	0.928
COV-0100-ADA-DSC-5	0.952	0.012	0.946	0.952
COV-0100-ADA-REA-2	0.933	0.013	0.921	0.927
COV-0100-ADA-REA-5	0.942	0.018	0.935	0.945
COV-0100-KNN-03	0.958	0.015	0.942	0.934
COV-0100-KNN-07	0.953	0.015	0.930	0.912
COV-0100-KNN-21	0.947	0.014	0.920	0.897
COV-0100-RFP-03	0.913	0.021	0.901	0.914
COV-0100-RFP-07	0.945	0.014	0.933	0.933
COV-0100-RFP-19	0.949	0.016	0.944	0.955
COV-0100-SVM-LIN	0.924	0.011	0.905	0.902
COV-0100-SVM-RBF	0.939	0.017	0.928	0.934
COV-075-ADA-DSC-2	0.887	0.018	0.876	0.903
COV-075-ADA-DSC-5	0.895	0.018	0.882	0.903
COV-075-ADA-REA-2	0.887	0.030	0.876	0.902
COV-075-ADA-REA-5	0.886	0.017	0.867	0.882
COV-075-KNN-03	0.900	0.017	0.884	0.897
COV-075-KNN-07	0.907	0.012	0.889	0.895
COV-075-KNN-21	0.908	0.020	0.892	0.899

COV-075-RFP-03	0.884	0.022	0.882	0.925
COV-075-RFP-07	0.904	0.025	0.893	0.909
COV-075-RFP-19	0.901	0.025	0.887	0.902
COV-075-SVM-LIN	0.887	0.015	0.870	0.886
COV-075-SVM-RBF	0.890	0.020	0.871	0.880
COV-085-ADA-DSC-2	0.904	0.026	0.893	0.912
COV-085-ADA-DSC-5	0.920	0.022	0.910	0.924
COV-085-ADA-REA-2	0.903	0.013	0.891	0.910
COV-085-ADA-REA-5	0.913	0.023	0.899	0.911
COV-085-KNN-03	0.938	0.014	0.922	0.918
COV-085-KNN-07	0.938	0.014	0.918	0.912
COV-085-KNN-21	0.928	0.017	0.906	0.899
COV-085-RFP-03	0.910	0.015	0.895	0.904
COV-085-RFP-07	0.928	0.012	0.913	0.917
COV-085-RFP-19	0.937	0.017	0.926	0.931
COV-085-SVM-LIN	0.887	0.019	0.870	0.886
COV-085-SVM-RBF	0.915	0.015	0.894	0.892
COV-099-ADA-DSC-2	0.913	0.019	0.900	0.913
COV-099-ADA-DSC-5	0.933	0.016	0.924	0.936
COV-099-ADA-REA-2	0.915	0.014	0.903	0.916
COV-099-ADA-REA-5	0.921	0.013	0.914	0.932
COV-099-KNN-03	0.957	0.016	0.944	0.939
COV-099-KNN-07	0.951	0.014	0.930	0.914
COV-099-KNN-21	0.947	0.019	0.921	0.901
COV-099-RFP-03	0.889	0.019	0.818	0.727
COV-099-RFP-07	0.932	0.027	0.921	0.927
COV-099-RFP-19	0.937	0.017	0.934	0.952
COV-099-SVM-LIN	0.924	0.015	0.905	0.900
COV-099-SVM-RBF	0.933	0.009	0.921	0.926
HOG-0100-ADA-DSC-2	0.893	0.024	0.880	0.900
HOG-0100-ADA-DSC-5	0.911	0.023	0.902	0.920
HOG-0100-ADA-REA-2	0.899	0.022	0.890	0.913
HOG-0100-ADA-REA-5	0.907	0.020	0.895	0.912
HOG-0100-KNN-03	0.953	0.014	0.948	0.955
HOG-0100-KNN-07	0.939	0.018	0.937	0.955
HOG-0100-KNN-21	0.925	0.019	0.923	0.948
HOG-0100-RFP-03	0.848	0.027	0.855	0.947
HOG-0100-RFP-07	0.928	0.017	0.919	0.930
HOG-0100-RFP-19	0.931	0.014	0.927	0.948
HOG-0100-SVM-LIN	0.935	0.025	0.927	0.940

HOG-0100-SVM-RBF	0.947	0.020	0.942	0.952
HOG-075-ADA-DSC-2	0.906	0.016	0.896	0.915
HOG-075-ADA-DSC-5	0.918	0.017	0.913	0.935
HOG-075-ADA-REA-2	0.904	0.024	0.895	0.916
HOG-075-ADA-REA-5	0.905	0.021	0.892	0.908
HOG-075-KNN-03	0.955	0.009	0.953	0.966
HOG-075-KNN-07	0.948	0.020	0.942	0.953
HOG-075-KNN-21	0.928	0.021	0.926	0.949
HOG-075-RFP-03	0.755	0.053	0.757	0.970
HOG-075-RFP-07	0.906	0.014	0.904	0.937
HOG-075-RFP-19	0.930	0.014	0.921	0.933
HOG-075-SVM-LIN	0.925	0.016	0.918	0.933
HOG-075-SVM-RBF	0.947	0.014	0.945	0.962
HOG-085-ADA-DSC-2	0.906	0.016	0.894	0.910
HOG-085-ADA-DSC-5	0.909	0.029	0.901	0.922
HOG-085-ADA-REA-2	0.901	0.013	0.894	0.922
HOG-085-ADA-REA-5	0.900	0.026	0.891	0.915
HOG-085-KNN-03	0.954	0.012	0.950	0.959
HOG-085-KNN-07	0.942	0.019	0.939	0.956
HOG-085-KNN-21	0.926	0.021	0.923	0.949
HOG-085-RFP-03	0.706	0.056	0.693	0.977
HOG-085-RFP-07	0.886	0.030	0.889	0.946
HOG-085-RFP-19	0.921	0.007	0.910	0.923
HOG-085-SVM-LIN	0.935	0.026	0.925	0.935
HOG-085-SVM-RBF	0.947	0.017	0.944	0.958
HOG-099-ADA-DSC-2	0.897	0.018	0.884	0.903
HOG-099-ADA-DSC-5	0.895	0.021	0.888	0.918
HOG-099-ADA-REA-2	0.897	0.010	0.887	0.911
HOG-099-ADA-REA-5	0.880	0.019	0.868	0.894
HOG-099-KNN-03	0.950	0.011	0.947	0.960
HOG-099-KNN-07	0.938	0.019	0.933	0.949
HOG-099-KNN-21	0.925	0.020	0.923	0.950
HOG-099-RFP-03	0.626	0.015	0.573	1.0
HOG-099-RFP-07	0.777	0.027	0.781	0.945
HOG-099-RFP-19	0.848	0.015	0.838	0.891
HOG-099-SVM-LIN	0.929	0.014	0.923	0.938
HOG-099-SVM-RBF	0.946	0.013	0.941	0.956
PIX-0100-ADA-DSC-2	0.884	0.023	0.860	0.864
PIX-0100-ADA-DSC-5	0.885	0.025	0.862	0.866
PIX-0100-ADA-REA-2	0.857	0.023	0.845	0.882

PIX-0100-ADA-REA-5	0.875	0.021	0.856	0.869
PIX-0100-KNN-03	0.956	0.008	0.929	0.906
PIX-0100-KNN-07	0.951	0.013	0.927	0.908
PIX-0100-KNN-21	0.939	0.012	0.910	0.886
PIX-0100-RFP-03	0.737	0.032	0.743	0.990
PIX-0100-RFP-07	0.882	0.030	0.869	0.892
PIX-0100-RFP-19	0.912	0.014	0.884	0.868
PIX-0100-SVM-LIN	0.923	0.010	0.908	0.911
PIX-0100-SVM-RBF	0.929	0.018	0.918	0.928
PIX-075-ADA-DSC-2	0.898	0.016	0.891	0.921
PIX-075-ADA-DSC-5	0.910	0.019	0.896	0.909
PIX-075-ADA-REA-2	0.894	0.032	0.878	0.895
PIX-075-ADA-REA-5	0.872	0.024	0.859	0.888
PIX-075-KNN-03	0.945	0.012	0.937	0.945
PIX-075-KNN-07	0.936	0.014	0.931	0.947
PIX-075-KNN-21	0.921	0.016	0.917	0.941
PIX-075-RFP-03	0.689	0.052	0.673	0.984
PIX-075-RFP-07	0.856	0.024	0.857	0.930
PIX-075-RFP-19	0.909	0.019	0.894	0.906
PIX-075-SVM-LIN	0.922	0.017	0.906	0.911
PIX-075-SVM-RBF	0.922	0.017	0.908	0.915
PIX-085-ADA-DSC-2	0.900	0.019	0.889	0.910
PIX-085-ADA-DSC-5	0.896	0.024	0.883	0.902
PIX-085-ADA-REA-2	0.902	0.023	0.887	0.899
PIX-085-ADA-REA-5	0.877	0.027	0.863	0.885
PIX-085-KNN-03	0.952	0.009	0.928	0.908
PIX-085-KNN-07	0.949	0.013	0.928	0.914
PIX-085-KNN-21	0.938	0.016	0.914	0.895
PIX-085-RFP-03	0.670	0.040	0.646	0.984
PIX-085-RFP-07	0.839	0.034	0.844	0.935
PIX-085-RFP-19	0.893	0.018	0.883	0.908
PIX-085-SVM-LIN	0.921	0.011	0.905	0.908
PIX-085-SVM-RBF	0.924	0.010	0.911	0.918
PIX-099-ADA-DSC-2	0.896	0.019	0.883	0.901
PIX-099-ADA-DSC-5	0.906	0.019	0.894	0.911
PIX-099-ADA-REA-2	0.905	0.012	0.889	0.899
PIX-099-ADA-REA-5	0.872	0.023	0.858	0.885
PIX-099-KNN-03	0.953	0.012	0.924	0.898
PIX-099-KNN-07	0.949	0.017	0.920	0.893
PIX-099-KNN-21	0.937	0.012	0.900	0.863

PIX-099-RFP-03	0.629	0.036	0.578	0.998
PIX-099-RFP-07	0.810	0.026	0.818	0.947
PIX-099-RFP-19	0.885	0.019	0.874	0.904
PIX-099-SVM-LIN	0.921	0.017	0.906	0.911
PIX-099-SVM-RBF	0.926	0.012	0.915	0.924

A.1.1 Type 1 cross-validation results

Name	$F_{0.5}$	$\sigma_{F_{0.5}}$	**ACC**	**TPR**
COV-0100-ADA-DSC-2	0.947	0.020	0.972	0.931
COV-0100-ADA-DSC-5	0.964	0.019	0.980	0.946
COV-0100-ADA-REA-2	0.948	0.018	0.970	0.922
COV-0100-ADA-REA-5	0.958	0.017	0.974	0.929
COV-0100-KNN-03	0.941	0.027	0.973	0.950
COV-0100-KNN-07	0.917	0.027	0.959	0.909
COV-0100-KNN-21	0.909	0.024	0.952	0.877
COV-0100-RFP-03	0.906	0.039	0.943	0.819
COV-0100-RFP-07	0.959	0.020	0.970	0.898
COV-0100-RFP-19	0.964	0.026	0.975	0.922
COV-0100-SVM-LIN	0.943	0.024	0.968	0.917
COV-0100-SVM-RBF	0.940	0.035	0.971	0.938
COV-075-ADA-DSC-2	0.862	0.048	0.925	0.793
COV-075-ADA-DSC-5	0.850	0.028	0.927	0.841
COV-075-ADA-REA-2	0.846	0.035	0.922	0.812
COV-075-ADA-REA-5	0.835	0.030	0.919	0.824
COV-075-KNN-03	0.847	0.041	0.926	0.834
COV-075-KNN-07	0.843	0.031	0.923	0.837
COV-075-KNN-21	0.841	0.022	0.923	0.829
COV-075-RFP-03	0.869	0.033	0.923	0.764
COV-075-RFP-07	0.873	0.028	0.927	0.779
COV-075-RFP-19	0.870	0.029	0.931	0.822
COV-075-SVM-LIN	0.838	0.042	0.921	0.813
COV-075-SVM-RBF	0.844	0.027	0.921	0.805
COV-085-ADA-DSC-2	0.883	0.037	0.942	0.874
COV-085-ADA-DSC-5	0.923	0.031	0.959	0.901
COV-085-ADA-REA-2	0.890	0.033	0.947	0.892
COV-085-ADA-REA-5	0.921	0.022	0.958	0.899
COV-085-KNN-03	0.909	0.036	0.957	0.908

COV-085-KNN-07	0.900	0.027	0.953	0.906
COV-085-KNN-21	0.898	0.024	0.949	0.884
COV-085-RFP-03	0.898	0.040	0.938	0.808
COV-085-RFP-07	0.921	0.031	0.956	0.882
COV-085-RFP-19	0.932	0.034	0.960	0.887
COV-085-SVM-LIN	0.876	0.035	0.939	0.869
COV-085-SVM-RBF	0.886	0.034	0.946	0.895
COV-099-ADA-DSC-2	0.926	0.032	0.962	0.909
COV-099-ADA-DSC-5	0.956	0.024	0.971	0.911
COV-099-ADA-REA-2	0.932	0.028	0.965	0.913
COV-099-ADA-REA-5	0.959	0.020	0.974	0.922
COV-099-KNN-03	0.925	0.023	0.967	0.941
COV-099-KNN-07	0.927	0.024	0.965	0.923
COV-099-KNN-21	0.905	0.030	0.951	0.875
COV-099-RFP-03	0.563	0.370	0.689	0.352
COV-099-RFP-07	0.942	0.013	0.955	0.835
COV-099-RFP-19	0.958	0.019	0.964	0.869
COV-099-SVM-LIN	0.921	0.035	0.963	0.923
COV-099-SVM-RBF	0.937	0.025	0.969	0.934
HOG-0100-ADA-DSC-2	0.963	0.019	0.979	0.944
HOG-0100-ADA-DSC-5	0.973	0.017	0.983	0.956
HOG-0100-ADA-REA-2	0.968	0.018	0.983	0.961
HOG-0100-ADA-REA-5	0.972	0.006	0.981	0.942
HOG-0100-KNN-03	0.978	0.018	0.989	0.975
HOG-0100-KNN-07	0.971	0.016	0.985	0.968
HOG-0100-KNN-21	0.955	0.023	0.977	0.945
HOG-0100-RFP-03	0.955	0.015	0.968	0.888
HOG-0100-RFP-07	0.976	0.012	0.980	0.929
HOG-0100-RFP-19	0.976	0.013	0.978	0.918
HOG-0100-SVM-LIN	0.980	0.008	0.989	0.973
HOG-0100-SVM-RBF	0.987	0.010	0.992	0.979
HOG-075-ADA-DSC-2	0.945	0.019	0.970	0.920
HOG-075-ADA-DSC-5	0.965	0.021	0.980	0.946
HOG-075-ADA-REA-2	0.940	0.016	0.969	0.927
HOG-075-ADA-REA-5	0.964	0.016	0.977	0.929
HOG-075-KNN-03	0.983	0.013	0.990	0.976
HOG-075-KNN-07	0.980	0.018	0.989	0.970
HOG-075-KNN-21	0.956	0.020	0.975	0.934
HOG-075-RFP-03	0.138	0.169	0.395	0.038
HOG-075-RFP-07	0.926	0.020	0.934	0.738

HOG-075-RFP-19	0.965	0.012	0.971	0.892
HOG-075-SVM-LIN	0.980	0.012	0.987	0.957
HOG-075-SVM-RBF	0.987	0.013	0.992	0.978
HOG-085-ADA-DSC-2	0.937	0.021	0.965	0.909
HOG-085-ADA-DSC-5	0.966	0.017	0.979	0.940
HOG-085-ADA-REA-2	0.942	0.014	0.969	0.924
HOG-085-ADA-REA-5	0.953	0.021	0.973	0.932
HOG-085-KNN-03	0.984	0.016	0.991	0.977
HOG-085-KNN-07	0.979	0.011	0.988	0.969
HOG-085-KNN-21	0.953	0.024	0.975	0.937
HOG-085-RFP-03	0.0	0.0	0.0	0.0
HOG-085-RFP-07	0.806	0.032	0.871	0.468
HOG-085-RFP-19	0.952	0.014	0.956	0.828
HOG-085-SVM-LIN	0.977	0.011	0.987	0.966
HOG-085-SVM-RBF	0.988	0.008	0.992	0.979
HOG-099-ADA-DSC-2	0.928	0.032	0.961	0.897
HOG-099-ADA-DSC-5	0.962	0.018	0.975	0.925
HOG-099-ADA-REA-2	0.934	0.024	0.963	0.904
HOG-099-ADA-REA-5	0.951	0.024	0.969	0.902
HOG-099-KNN-03	0.979	0.012	0.989	0.976
HOG-099-KNN-07	0.968	0.020	0.985	0.972
HOG-099-KNN-21	0.951	0.021	0.975	0.943
HOG-099-RFP-03	0.0	0.0	0.0	0.0
HOG-099-RFP-07	0.016	0.032	0.153	0.003
HOG-099-RFP-19	0.739	0.060	0.850	0.373
HOG-099-SVM-LIN	0.982	0.011	0.990	0.972
HOG-099-SVM-RBF	0.986	0.012	0.992	0.980
PIX-0100-ADA-DSC-2	0.820	0.039	0.914	0.824
PIX-0100-ADA-DSC-5	0.869	0.063	0.940	0.867
PIX-0100-ADA-REA-2	0.843	0.028	0.923	0.832
PIX-0100-ADA-REA-5	0.834	0.032	0.922	0.848
PIX-0100-KNN-03	0.907	0.012	0.967	0.986
PIX-0100-KNN-07	0.880	0.030	0.957	0.982
PIX-0100-KNN-21	0.857	0.031	0.947	0.975
PIX-0100-RFP-03	0.775	0.034	0.862	0.451
PIX-0100-RFP-07	0.848	0.060	0.918	0.776
PIX-0100-RFP-19	0.854	0.039	0.928	0.835
PIX-0100-SVM-LIN	0.943	0.016	0.970	0.932
PIX-0100-SVM-RBF	0.951	0.026	0.971	0.918
PIX-075-ADA-DSC-2	0.905	0.026	0.951	0.883

PIX-075-ADA-DSC-5	0.935	0.019	0.963	0.900
PIX-075-ADA-REA-2	0.928	0.028	0.964	0.916
PIX-075-ADA-REA-5	0.919	0.028	0.955	0.880
PIX-075-KNN-03	0.949	0.016	0.978	0.964
PIX-075-KNN-07	0.948	0.019	0.975	0.947
PIX-075-KNN-21	0.918	0.026	0.962	0.924
PIX-075-RFP-03	0.063	0.058	0.461	0.014
PIX-075-RFP-07	0.826	0.049	0.880	0.511
PIX-075-RFP-19	0.922	0.030	0.941	0.784
PIX-075-SVM-LIN	0.950	0.019	0.971	0.924
PIX-075-SVM-RBF	0.944	0.028	0.973	0.940
PIX-085-ADA-DSC-2	0.899	0.027	0.948	0.876
PIX-085-ADA-DSC-5	0.935	0.037	0.964	0.896
PIX-085-ADA-REA-2	0.918	0.035	0.957	0.894
PIX-085-ADA-REA-5	0.917	0.039	0.951	0.855
PIX-085-KNN-03	0.914	0.016	0.968	0.981
PIX-085-KNN-07	0.916	0.020	0.968	0.975
PIX-085-KNN-21	0.894	0.023	0.960	0.966
PIX-085-RFP-03	0.061	0.072	0.383	0.013
PIX-085-RFP-07	0.709	0.093	0.843	0.348
PIX-085-RFP-19	0.901	0.040	0.933	0.764
PIX-085-SVM-LIN	0.942	0.012	0.968	0.921
PIX-085-SVM-RBF	0.956	0.029	0.977	0.941
PIX-099-ADA-DSC-2	0.893	0.025	0.946	0.874
PIX-099-ADA-DSC-5	0.927	0.021	0.959	0.889
PIX-099-ADA-REA-2	0.919	0.032	0.958	0.905
PIX-099-ADA-REA-5	0.927	0.017	0.955	0.862
PIX-099-KNN-03	0.873	0.031	0.955	0.986
PIX-099-KNN-07	0.845	0.023	0.943	0.983
PIX-099-KNN-21	0.794	0.055	0.921	0.978
PIX-099-RFP-03	0.0	0.0	0.0	0.0
PIX-099-RFP-07	0.563	0.104	0.812	0.221
PIX-099-RFP-19	0.898	0.044	0.929	0.744
PIX-099-SVM-LIN	0.939	0.022	0.967	0.915
PIX-099-SVM-RBF	0.948	0.025	0.971	0.925

A.1.2 Type 2 cross-validation results

Name	$F_{0.5}$	$\sigma_{F_0.5}$	**ACC**	**TPR**
COV-0100-ADA-DSC-2	0.822	0.044	0.931	0.807
COV-0100-ADA-DSC-5	0.880	0.050	0.949	0.824
COV-0100-ADA-REA-2	0.835	0.041	0.936	0.821
COV-0100-ADA-REA-5	0.832	0.023	0.934	0.804
COV-0100-KNN-03	0.847	0.032	0.945	0.874
COV-0100-KNN-07	0.810	0.049	0.932	0.852
COV-0100-KNN-21	0.755	0.042	0.910	0.800
COV-0100-RFP-03	0.751	0.049	0.900	0.638
COV-0100-RFP-07	0.858	0.038	0.939	0.775
COV-0100-RFP-19	0.881	0.021	0.943	0.773
COV-0100-SVM-LIN	0.775	0.059	0.907	0.672
COV-0100-SVM-RBF	0.816	0.064	0.925	0.765
COV-075-ADA-DSC-2	0.543	0.065	0.834	0.450
COV-075-ADA-DSC-5	0.640	0.061	0.865	0.598
COV-075-ADA-REA-2	0.588	0.075	0.846	0.523
COV-075-ADA-REA-5	0.563	0.050	0.837	0.528
COV-075-KNN-03	0.621	0.074	0.857	0.589
COV-075-KNN-07	0.643	0.050	0.865	0.578
COV-075-KNN-21	0.660	0.045	0.871	0.558
COV-075-RFP-03	0.211	0.160	0.578	0.074
COV-075-RFP-07	0.643	0.076	0.865	0.476
COV-075-RFP-19	0.661	0.073	0.870	0.527
COV-075-SVM-LIN	0.0	0.0	0.0	0.0
COV-075-SVM-RBF	0.0	0.0	0.0	0.0
COV-085-ADA-DSC-2	0.676	0.056	0.875	0.574
COV-085-ADA-DSC-5	0.727	0.072	0.894	0.648
COV-085-ADA-REA-2	0.714	0.045	0.889	0.648
COV-085-ADA-REA-5	0.745	0.062	0.902	0.723
COV-085-KNN-03	0.751	0.051	0.907	0.758
COV-085-KNN-07	0.748	0.057	0.904	0.719
COV-085-KNN-21	0.720	0.045	0.892	0.684
COV-085-RFP-03	0.699	0.047	0.880	0.536
COV-085-RFP-07	0.755	0.045	0.903	0.660
COV-085-RFP-19	0.794	0.035	0.915	0.702
COV-085-SVM-LIN	0.650	0.045	0.865	0.439
COV-085-SVM-RBF	0.703	0.058	0.883	0.581
COV-099-ADA-DSC-2	0.756	0.055	0.904	0.660
COV-099-ADA-DSC-5	0.842	0.037	0.930	0.730

COV-099-ADA-REA-2	0.763	0.033	0.909	0.722
COV-099-ADA-REA-5	0.782	0.058	0.915	0.740
COV-099-KNN-03	0.840	0.033	0.941	0.852
COV-099-KNN-07	0.801	0.062	0.929	0.848
COV-099-KNN-21	0.760	0.057	0.912	0.784
COV-099-RFP-03	0.0	0.0	0.0	0.0
COV-099-RFP-07	0.824	0.025	0.919	0.670
COV-099-RFP-19	0.858	0.055	0.929	0.689
COV-099-SVM-LIN	0.751	0.095	0.901	0.635
COV-099-SVM-RBF	0.816	0.048	0.920	0.706
HOG-0100-ADA-DSC-2	0.758	0.054	0.904	0.672
HOG-0100-ADA-DSC-5	0.806	0.048	0.919	0.713
HOG-0100-ADA-REA-2	0.769	0.052	0.911	0.718
HOG-0100-ADA-REA-5	0.755	0.044	0.901	0.654
HOG-0100-KNN-03	0.872	0.035	0.949	0.851
HOG-0100-KNN-07	0.854	0.046	0.941	0.804
HOG-0100-KNN-21	0.828	0.036	0.927	0.733
HOG-0100-RFP-03	0.444	0.138	0.839	0.152
HOG-0100-RFP-07	0.825	0.043	0.913	0.585
HOG-0100-RFP-19	0.872	0.016	0.933	0.698
HOG-0100-SVM-LIN	0.870	0.037	0.943	0.792
HOG-0100-SVM-RBF	0.898	0.028	0.953	0.823
HOG-075-ADA-DSC-2	0.775	0.053	0.909	0.688
HOG-075-ADA-DSC-5	0.813	0.055	0.924	0.763
HOG-075-ADA-REA-2	0.767	0.060	0.910	0.713
HOG-075-ADA-REA-5	0.802	0.062	0.918	0.703
HOG-075-KNN-03	0.902	0.030	0.959	0.869
HOG-075-KNN-07	0.876	0.031	0.950	0.849
HOG-075-KNN-21	0.838	0.035	0.931	0.756
HOG-075-RFP-03	0.0	0.0	0.0	0.0
HOG-075-RFP-07	0.701	0.067	0.873	0.336
HOG-075-RFP-19	0.851	0.039	0.918	0.587
HOG-075-SVM-LIN	0.870	0.040	0.941	0.772
HOG-075-SVM-RBF	0.908	0.035	0.958	0.839
HOG-085-ADA-DSC-2	0.738	0.050	0.898	0.649
HOG-085-ADA-DSC-5	0.811	0.038	0.921	0.724
HOG-085-ADA-REA-2	0.741	0.043	0.900	0.695
HOG-085-ADA-REA-5	0.787	0.048	0.912	0.676
HOG-085-KNN-03	0.888	0.031	0.954	0.861
HOG-085-KNN-07	0.860	0.043	0.943	0.814

HOG-085-KNN-21	0.829	0.042	0.928	0.741
HOG-085-RFP-03	0.0	0.0	0.0	0.0
HOG-085-RFP-07	0.349	0.193	0.752	0.117
HOG-085-RFP-19	0.746	0.045	0.884	0.392
HOG-085-SVM-LIN	0.861	0.038	0.938	0.772
HOG-085-SVM-RBF	0.896	0.029	0.954	0.844
HOG-099-ADA-DSC-2	0.725	0.056	0.893	0.619
HOG-099-ADA-DSC-5	0.802	0.053	0.919	0.715
HOG-099-ADA-REA-2	0.742	0.065	0.901	0.677
HOG-099-ADA-REA-5	0.700	0.066	0.884	0.573
HOG-099-KNN-03	0.892	0.023	0.956	0.869
HOG-099-KNN-07	0.866	0.030	0.943	0.802
HOG-099-KNN-21	0.835	0.039	0.929	0.739
HOG-099-RFP-03	0.0	0.0	0.0	0.0
HOG-099-RFP-07	0.0	0.0	0.0	0.0
HOG-099-RFP-19	0.343	0.107	0.830	0.101
HOG-099-SVM-LIN	0.875	0.039	0.944	0.791
HOG-099-SVM-RBF	0.895	0.038	0.953	0.826
PIX-0100-ADA-DSC-2	0.578	0.058	0.847	0.412
PIX-0100-ADA-DSC-5	0.621	0.071	0.859	0.555
PIX-0100-ADA-REA-2	0.588	0.045	0.846	0.535
PIX-0100-ADA-REA-5	0.538	0.068	0.831	0.468
PIX-0100-KNN-03	0.826	0.033	0.936	0.841
PIX-0100-KNN-07	0.819	0.048	0.928	0.767
PIX-0100-KNN-21	0.781	0.048	0.908	0.662
PIX-0100-RFP-03	0.031	0.067	0.167	0.007
PIX-0100-RFP-07	0.500	0.078	0.842	0.187
PIX-0100-RFP-19	0.680	0.052	0.875	0.469
PIX-0100-SVM-LIN	0.769	0.062	0.901	0.607
PIX-0100-SVM-RBF	0.787	0.043	0.914	0.717
PIX-075-ADA-DSC-2	0.705	0.044	0.886	0.638
PIX-075-ADA-DSC-5	0.681	0.047	0.877	0.597
PIX-075-ADA-REA-2	0.674	0.062	0.877	0.617
PIX-075-ADA-REA-5	0.630	0.055	0.862	0.523
PIX-075-KNN-03	0.863	0.037	0.944	0.819
PIX-075-KNN-07	0.859	0.037	0.939	0.792
PIX-075-KNN-21	0.806	0.027	0.916	0.681
PIX-075-RFP-03	0.0	0.0	0.0	0.0
PIX-075-RFP-07	0.054	0.073	0.333	0.012
PIX-075-RFP-19	0.245	0.056	0.822	0.064

PIX-075-SVM-LIN	0.762	0.030	0.902	0.624
PIX-075-SVM-RBF	0.788	0.055	0.912	0.696
PIX-085-ADA-DSC-2	0.685	0.073	0.879	0.617
PIX-085-ADA-DSC-5	0.674	0.075	0.876	0.595
PIX-085-ADA-REA-2	0.687	0.054	0.880	0.619
PIX-085-ADA-REA-5	0.602	0.044	0.852	0.503
PIX-085-KNN-03	0.828	0.037	0.937	0.850
PIX-085-KNN-07	0.823	0.053	0.932	0.797
PIX-085-KNN-21	0.781	0.045	0.912	0.714
PIX-085-RFP-03	0.0	0.0	0.0	0.0
PIX-085-RFP-07	0.019	0.058	0.082	0.004
PIX-085-RFP-19	0.173	0.127	0.734	0.045
PIX-085-SVM-LIN	0.764	0.055	0.899	0.602
PIX-085-SVM-RBF	0.781	0.032	0.912	0.713
PIX-099-ADA-DSC-2	0.673	0.072	0.876	0.605
PIX-099-ADA-DSC-5	0.671	0.047	0.874	0.576
PIX-099-ADA-REA-2	0.665	0.058	0.873	0.604
PIX-099-ADA-REA-5	0.641	0.053	0.864	0.528
PIX-099-KNN-03	0.834	0.037	0.937	0.829
PIX-099-KNN-07	0.816	0.031	0.925	0.753
PIX-099-KNN-21	0.784	0.059	0.906	0.627
PIX-099-RFP-03	0.0	0.0	0.0	0.0
PIX-099-RFP-07	0.0	0.0	0.0	0.0
PIX-099-RFP-19	0.151	0.080	0.738	0.036
PIX-099-SVM-LIN	0.762	0.023	0.902	0.635
PIX-099-SVM-RBF	0.777	0.054	0.911	0.700

APPENDIX B

Tukey's range test results

B.0.3 Type 0 classifier results

	HOG-075-KNN-03	**HOG-075-KNN-07**	**HOG-075-SVM-LIN**	**HOG-075-SVM-RBF**	**HOG-085-KNN-03**	**HOG-085-SVM-RBF**	**HOG-099-SVM-LIN**	**HOG-099-SVM-RBF**	**HOG-100-SVM-LIN**	**HOG-100-SVM-RBF**
HOG-075-KNN-03	1.0	0.999	0.999	0.999	1.000	0.997	1.000	0.999	0.999	0.999
HOG-075-KNN-07	0.999	1.0	1.000	0.980	0.999	0.941	0.999	0.995	1.000	0.985
HOG-075-SVM-LIN	0.999	1.000	1.0	0.986	0.999	0.956	0.999	0.997	1.000	0.990
HOG-075-SVM-RBF	0.999	0.980	0.986	1.0	0.999	1.000	0.998	1.000	0.972	1.000
HOG-085-KNN-03	1.000	0.999	0.999	0.999	1.0	0.998	1.000	0.999	0.999	0.999
HOG-085-SVM-RBF	0.997	0.941	0.956	1.000	0.998	1.0	0.993	0.999	0.924	0.999
HOG-099-SVM-LIN	1.000	0.999	0.999	0.998	1.000	0.993	1.0	0.999	0.999	0.999
HOG-099-SVM-RBF	0.999	0.995	0.997	1.000	0.999	0.999	0.999	1.0	0.993	1.000
HOG-100-SVM-LIN	0.999	1.000	1.000	0.972	0.999	0.924	0.999	0.993	1.0	0.979
HOG-100-SVM-RBF	0.999	0.985	0.990	1.000	0.999	0.999	0.999	1.000	0.979	1.0

Table B.1: p values returned by Tukey's Range Test, if $p < 0.95$ the hypotheses that the there's no significant difference between the classifiers is considered false. In bold those comparisons where the classifiers are significantly different

B.0.4 Type 1 classifier results

	HOG-075-KNN-03	**HOG-075-KNN-07**	**HOG-075-SVM-LIN**	**HOG-075-SVM-RBF**	**HOG-085-KNN-03**	**HOG-085-SVM-RBF**	**HOG-099-SVM-LIN**	**HOG-099-SVM-RBF**	**HOG-100-SVM-LIN**	**HOG-100-SVM-RBF**
HOG-075-KNN-03	1.0	0.999	0.999	0.999	1.000	0.997	1.000	0.999	0.999	0.999
HOG-075-KNN-07	0.999	1.0	1.000	0.980	0.999	0.941	0.999	0.995	1.000	0.985
HOG-075-SVM-LIN	0.999	1.000	1.0	0.986	0.999	0.956	0.999	0.997	1.000	0.990
HOG-075-SVM-RBF	0.999	0.980	0.986	1.0	0.999	1.000	0.998	1.000	0.972	1.000
HOG-085-KNN-03	1.000	0.999	0.999	0.999	1.0	0.998	1.000	0.999	0.999	0.999
HOG-085-SVM-RBF	0.997	0.941	0.956	1.000	0.998	1.0	0.993	0.999	0.924	0.999
HOG-099-SVM-LIN	1.000	0.999	0.999	0.998	1.000	0.993	1.0	0.999	0.999	0.999
HOG-099-SVM-RBF	0.999	0.995	0.997	1.000	0.999	0.999	0.999	1.0	0.993	1.000
HOG-100-SVM-LIN	0.999	1.000	1.000	0.972	0.999	0.924	0.999	0.993	1.0	0.979
HOG-100-SVM-RBF	0.999	0.985	0.990	1.000	0.999	0.999	0.999	1.000	0.979	1.0

Table B.2: p values returned by Tukey's Range Test, if $p < 0.95$ the hypotheses that the there's no significant difference between the classifiers is considered false. In bold those comparisons where the classifiers are significantly different

B.0.5 Type 2 classifier results

	COV-100-ADA-DSC-5	**COV-100-RFP-19**	**HOG-075-KNN-03**	**HOG-075-SVM-RBF**	**HOG-085-KNN-03**	**HOG-085-SVM-RBF**	**HOG-099-KNN-03**	**HOG-099-SVM-LIN**	**HOG-099-SVM-RBF**	**HOG-100-SVM-RBF**
COV-100-ADA-DSC-5	1.0	1.000	0.937	0.768	0.999	0.993	0.999	0.999	0.994	0.983
COV-100-RFP-19	1.000	1.0	0.946	0.789	0.999	0.995	0.999	0.999	0.995	0.986
HOG-075-KNN-03	0.937	0.946	1.0	0.999	0.996	0.999	0.999	0.785	0.999	0.999
HOG-075-SVM-RBF	0.768	0.789	0.999	1.0	0.956	0.998	0.989	0.533	0.998	0.999
HOG-085-KNN-03	0.999	0.999	0.996	0.956	1.0	0.999	0.999	0.998	0.999	0.999
HOG-085-SVM-RBF	0.993	0.995	0.999	0.998	0.999	1.0	0.999	0.945	1.000	1.000
HOG-099-KNN-03	0.999	0.999	0.999	0.989	0.999	0.999	1.0	0.986	1.000	0.999

HOG-099-SVM-LIN	0.999	0.999	0.785	0.533	0.998	0.945	0.986	1.0	0.949	0.905
HOG-099-SVM-RBF	0.994	0.995	0.999	0.998	0.999	1.000	1.000	0.949	1.0	1.000
HOG-100-SVM-RBF	0.983	0.986	0.999	0.999	0.999	1.000	0.999	0.905	1.000	1.0

Table B.3: p values returned by Tukey's Range Test, if $p < 0.95$ the hypotheses that the there's no significant difference between the classifiers is considered false. In bold those comparisons where the classifiers are significantly different

B.0.6 Ensemble classifier results

	C1-ANN-2	**C1-ANN-4**	**C3-ANN-2**	**C3-ANN-5**	**C3-ANN-6**	**C3-ANN-7**	**C4-ANN-4**	**C4-ANN-6**	**C4-ANN-7**	**C4-ANN-8**
C1-ANN-2	-	1.000	1.000	1.000	1.000	1.000	1.000	1.000	0.999	0.957
C1-ANN-4	1.000	-	1.000	1.000	0.999	1.000	1.000	1.000	1.000	**0.943**
C3-ANN-2	1.000	1.000	-	1.000	1.000	1.000	1.000	1.000	0.999	0.981
C3-ANN-5	1.000	1.000	1.000	-	1.000	1.000	1.000	1.000	0.999	0.964
C3-ANN-6	1.000	0.999	1.000	1.000	-	1.000	1.000	1.000	0.999	0.984
C3-ANN-7	1.000	1.000	1.000	1.000	1.000	-	1.000	1.000	1.000	**0.949**
C4-ANN-4	1.000	1.000	1.000	1.000	1.000	1.000	-	1.000	0.999	0.973
C4-ANN-6	1.000	1.000	1.000	1.000	1.000	1.000	1.000	-	0.999	0.958
C4-ANN-7	0.999	1.000	0.999	0.999	0.999	1.000	0.999	0.999	-	**0.876**
C4-ANN-8	0.957	**0.943**	0.981	0.964	0.984	**0.949**	0.973	0.958	**0.876**	-

Table B.4: p values returned by Tukey's Range Test, if $p < 0.95$ the hypotheses that the there's no significant difference between the classifiers is considered false. In bold those comparisons where the classifiers are significantly different

Bibliography

[ABR64] A. Aizerman, E. M. Braverman, and L. I. Rozoner, *Theoretical foundations of the potential function method in pattern recognition learning*, Automation and Remote Control **25** (1964), 821–837.

[BGV92] B.E. Boser, I.M. Guyon, and V.N. Vapnik, *A training algorithm for optimal margin classifiers*, Proceedings of the fifth annual workshop on Computational learning theory, ACM, 1992, pp. 144–152.

[Bra00] G. Bradski, *The OpenCV Library*, Dr. Dobb's Journal of Software Tools (2000).

[Bre01] Leo Breiman, *Random forests*, Machine Learning **45** (2001), no. 1, 5–32 (English).

[Can86] John Canny, *A computational approach to edge detection*, Pattern Analysis and Machine Intelligence, IEEE Transactions on (1986), no. 6, 679–698.

[CCD+88] V. Cantoni, L. Carrioli, M. Diani, M. Savini, and G. Vecchio, *Photometric approach to tracking of moving objects*, Image Analysis and Processing II (V. Cantoni, V. Gesù, and S. Levialdi, eds.), Springer US, 1988, pp. 253–260 (English).

[cCL01] Chih chung Chang and Chih-Jen Lin, *Libsvm: a library for support vector machines*, 2001.

[CH67] T. Cover and P. Hart, *Nearest neighbor pattern classification*, Information Theory, IEEE Transactions on **13** (1967), no. 1, 21–27.

[Cur99] Brian Curless, *From range scans to 3d models*, SIGGRAPH Comput. Graph. **33** (1999), no. 4, 38–41.

[CV95] C Cortes and V Vapnik, *Support-vector networks*, Machine learning **297** (1995), 273–297.

[DH72] Richard O Duda and Peter E Hart, *Use of the hough transformation to detect lines and curves in pictures*, Communications of the ACM **15** (1972), no. 1, 11–15.

[DT05] Navneet Dalal and Bill Triggs, *Histograms of oriented gradients for human detection*, In CVPR, 2005, pp. 886–893.

[FS95] Yoav Freund and Robert E. Schapire, *A decision-theoretic generalization of on-line learning and an application to boosting*, 1995.

[GCS+06] Philip Garcia, Katherine Compton, Michael Schulte, Emily Blem, and Wenyin Fu, *An overview of reconfigurable hardware in embedded systems*, EURASIP Journal on Embedded Systems **2006** (2006).

[Hot33] H. Hotelling, *Analysis of a complex of statistical variables into principal components*, 1933.

[Jos08] Jürgen Jost, *Riemannian geometry and geometric analysis (5. ed.).*, Springer, 2008.

[K+95] Ron Kohavi et al., *A study of cross-validation and bootstrap for accuracy estimation and model selection*, IJCAI, vol. 14, 1995, pp. 1137–1145.

[KSN10] Ellen Klingbeil, Ashutosh Saxena, and Andrew Y Ng, *Learning to open new doors*, Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on, IEEE, 2010, pp. 2751–2757.

[LRD93] J.-M. Lavest, G. Rives, and M. Dhome, *Three-dimensional reconstruction by zooming*, Robotics and Automation, IEEE Transactions on **9** (1993), no. 2, 196–207.

[Pap] CP Papageorgiou, *A general framework for object detection*, Computer Vision, 1998. . . . .

[PN07] Anna Petrovskaya and Andrew Y Ng, *Probabilistic mobile manipulation in dynamic environments, with application to opening doors.*, IJCAI, 2007, pp. 2178–2184.

[Pon06] Jean Ponce, *Toward category-level object recognition*, vol. 4170, Springer, 2006.

[PSDP07] Mario Prats, Pedro J Sanz, and Angel P Del Pobil, *Task planning for intelligent robot manipulation.*, Artificial Intelligence and Applications, 2007, pp. 237–242.

[QCG+09] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully B. Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y. Ng, *ROS: an open-source robot operating system*, ICRA Workshop on Open Source Software, 2009.

[RBC11] Rusu Radu Bogdan and Steve Cousins, *3d is here: Point cloud library (pcl)*, IEEE International Conference on Robotics and Automation (ICRA) (Shanghai, China), May 9-13 2011.

[RC] Radu Bogdan Rusu and Steve Cousins, *Point cloud library website.*

[RMCB09] Radu Bogdan Rusu, Wim Meeussen, Sachin Chitta, and Michael Beetz, *Laser-based perception for door and handle identification*, Advanced Robotics, 2009. ICAR 2009. International Conference on, IEEE, 2009, pp. 1–8.

[Rok09] Lior Rokach, *Ensemble-based classifiers*, Artificial Intelligence Review **33** (2009), no. 1-2, 1–39.

[RSP+12] Thomas Ruhr, Jurgen Sturm, Dejan Pangercic, Michael Beetz, and Daniel Cremers, *A generalized framework for opening doors and drawers in kitchen environments*, 2012 IEEE International Conference on Robotics and Automation (2012), 3852–3858.

[SC] Ioan A. Sucan and Sachin Chitta, *MoveIt! [Online] Available: http://moveit.ros.org.*

[SPB04] Joaquim Salvi, Jordi Pages, and Joan Batlle, *Pattern codification strategies in structured light systems*, Pattern Recognition **37** (2004), no. 4, 827–849.

[TPM08] Oncel Tuzel, Fatih Porikli, and Peter Meer, *Pedestrian detection via classification on Riemannian manifolds.*, IEEE transactions on pattern analysis and machine intelligence **30** (2008), no. 10, 1713–27.

[TW97] Kai Ming Ting and Ian H. Witten, *Stacked generalization: when does it work?*, in Procs. International Joint Conference on Artificial Intelligence, Morgan Kaufmann, 1997, pp. 866–871.

[VL63] V. Vapnik and A. Lerner, *Pattern recognition using generalized portrait method.*

[Wol92] David H. Wolpert, *Stacked generalization*, Neural Networks **5** (1992), 241–259.

[Zen04] Bernard ˇ Zenko, *Is combining classifiers better than selecting the best one*, Machine Learning, Morgan Kaufmann, 2004, pp. 255–273.