TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

APS	Automated Picking Station
CNN	Convolutional Neural Network
CoRo	Control and Robotics laboraty at ETS
ETS	École de Technologie Supérieure
HSV	Hue Saturation Value
IMU	Inertial Measurement Unit
ROS	Robot Operating System
ROI	Region Of Interest
SVM	Support Vetor Machine

INTRODUCTION

In the past six decades, we have come a long way in the world of autonomous machines. Science fiction had placed robots in our homes in the year 2000. Even though we do not all own humanoid robots to help us with our daily tasks, automated systems are necessary for our way of life. Nonetheless, these stories we heard as children have guided some of us to attempt replicating human abilities with robots (Boston Dynamics (2017), Honda (2017), Bebionic (2017), etc.). Amongs all the complex tasks we learn naturally as humans, one of the first ability we wished to replicate with robots was grasping. We have come a long way in the automatic grasping domain but, robots are still far from matching human capibilities when it comes to adapting their grasp when exposed to novel objects. We often see images of industrial robots accomplishing complex tasks in a very robust and efficient way but, with the lean supply chain movement (small volume and high mix productions), robots must now be able to adapt to a variety of different objects. There is a growing demand for adaptive systems in the manufacturing world that can be attributed to the consumer's want for personalized products. This movement has not only affected small productions but also, high volume production lines. Robotic integrators and tooling experts have had to push their imaginations to transform the typical rigid assembly line into a flexible production line. We have seen the introduction of cobots in the manufacturing world and the notion of lean robotics who are ment to help integrators maintain a high flexibility and quick deployment of production lines. This transition towards new manufacturing techniques have even prompted industry giants, such as Google and Amazon, to invest time, money and effort into more flexible and efficient production lines.

For a robotic integrator, it is a common task to teach a robot how to grasp an object. Equipped with the proper end effector and sensors, a robot can be shown how to grasp and also assess the quality of its grasp on an object. But, replace the object and it can no longer grasp the new object properly. This is quickly becoming a problem for the modern flexible production lines. As researches, we wish to develop new kinds of tools and methods to allow easier and faster integration of flexible robotic cells. More specifically, our research is interested in developing new techniques in order for a robot to learn, not how to grasp an object, but how to determine if the grasp is stable. This interest came from analysing the grasp strategy used by humans. We turned our attention to the biomedical research of grasping. We noticed there is a planning phase (Feix *et al.* (2014a) and Feix *et al.* (2014b)), were our brain computes the necessary trajectory and grasping technique, greatly based on vision and our knowledged of the object to grasp. But, there is a whole second phase that starts when we come in contact with the object (Fu *et al.* (2013)), we react and adapt our grasp based on a whole new set of sensors directly located in our hands. We questioned ourselves on how a human evaluates the quality of the grasp as it is happening. In the case of biological intelligence, we know that we use a combination of different sensors, mixed with experience to assess the outcome of our actions. Johansson and Flanagan (2009b) have shown that an essential sense for grasp assessment is touch. Many researchers (Hyttinen *et al.* (2015), Huebner *et al.* (2008), Dang and Allen (2013)) have used tactile sensors for grasp planning and adjustment.

Grasp planning has been highly developed and we already have very valuable tools such as *GraspIt* (Miller and Allen (2004)) to simulate and implement grasp trajectory but, in order to apply a correction to the originally planned grasping by a robot, we first need to be able to estimate the quality of the executed grasp. In this line of thought, we decided to take a look at the problem of determining the quality of a grasp in an efficient, flexible and rapid manner. The research presented in this thesis aims at developing a method to asses a robotic grasp at the moment of contact with the object by using exteroceptive information provided by tactile sensors. We will present the work we have done in attempt to build a grasp stability prediction system using tactile sensors developped in our laboratory (CoRo). First, we will present the robotic cell we have built to gather data by executing a simple grasp planning algorithm. The experimental setup will also be used to test our different systems. Afterwards, we expose the unsupervised learning technique used to extract high level features from the pressure images from our tactile sensors. The machine learning technique we used on pressure images was inspired by image reconstruction techniques. Finally, we propose different simple architecture to classify our data on a grasp stability scale.

CHAPTER 1

LITERATURE REVIEW

In order to better understand what is already being done in the robotic grasping domain, we studied the different sensors and techniques that are used in industry and also in research laboratories. This chapter will give a review of the information we found helpful for our research.

1.1 Overview of Relevant Sensors Used in Robotic Manipulation and Grasping Tasks

According to the Oxford English Dictionary the definition of robot is *A machine capable of carrying out a complex series of actions automatically, especially one programmable by a computer.* These robots, just as humans, are very limited in the tasks they can execute if they do not have the proper *tools*. In order to perform more and more complex tasks, researchers and engineers have developed sensors to give more flexibility to robotic systems.

1.1.1 Computer Vision

In the case of an industrial robot working in an assembly line, repeating the same tasks with the same objects over and over, the sense of *sight* is not necessary. But, if the object is slightly different or presented to the robot in a different location, it will immediately fail. Vision systems have been in constant development and evolution in order to solve some of these problems.

Capturing images long preceeds robots but, it is only in the 1960s that we first started using images in combination with a computer. Indeed, image processing is typically a very demanding task. Nowadays, we have the ability to capture both 2D and 3D data with different sensors.

One could believe that 2D passive cameras are out of date but in fact, they are still very useful in the industry. As Cheng and Denman (2005) have demonstrated, using a 2D vision system can improve accuracy, flexibility and intelligence of a robotic system. Indeed, we have become very proficient at detecting edges, patterns and shapes in 2D imagery. Also, with the evolution of computer processors, these tasks are executed very rapidly, to the point where they can be used in real-time.

On the other hand, we live in a three dimensional world and, to develop more elaborate systems we need the information from the third dimension. The output of 3D vision systems is in the form of a point cloud. There are many different technologies to retrieve this type of information but, they can be seperated into two main categories. First, stereo vision uses two passive cameras, the depth perception is inferred by computing a disparity field at each pixel location. The Bumblebee© camera by FireWire is a popular choice. The second category is the active range which can be seperated into two sub categories: projected light and time of light technologies. Projected light, as its name suggests, projects a pattern (visible or not) and uses triangulation to compute depth. The most common version of this technology is found in the Microsoft Kinect©. Time of light uses our knowledge of the speed of light. Again, light is projected onto the space we want to see but, this time, depth is computed by calculation the time delay between the emission and detection of the light source. The Lidar systems are based on this technology. Each of these cameras have their strong points and their weaknesses.

Here are a few examples of researchers who have used 3D image processing in an automated system. Viet *et al.* (2013) have used a Bumblebee stereo camera in their algorithm to control an electric wheelchair for severely disabled people. The goal of their research was to avoid object during the movement towards a target position in highly clustered areas. Padgett and Browne (2017) have also worked on obstacle avoidance but using the Lidar technology. Furthermore, researchers such as Fan *et al.* (2014) proposed a computer vision system using both a passive 2D camera coupled to a 3D sensor for depth perception. Their system was built to efficiently determine the position of randomly placed objects for robotic manipulation.

1.1.2 Force Sensing

Vision systems give the robots the ability to know their surroundings but do not give the robot direct feedback of its effect in their workspace. The second sensors we wish to review are the force-torque sensors which can be used to many effects. Again, there are many different technologies to obtain what the end effector of a robot *feels*.

A common use of force torque sensors is in applications where a robot must keep a constant pressure on a workpiece. Mills (1989) propose a complete dynamic model to be used for such tasks. Another interesting application for force torque sensors is the handling of a robot by hand. Typically, during the development of a robotic program, the integrator must use a keypad or a joystick to move the robot to desired positions in order to teach them. Loske and Biesenbach (2014) propose a solution to hand-drive an industrial robot using an added force-torque sensor and companies like Universal Robot, who build collaborative robots, have integrated such technology directly into their controllers by using force feedback from the individual joint of their robotic arms.

In the case of our research, we are more interested in monitoring what the robot *feels* in a grasping operation. Some authors, such as Hebert *et al.* (2011), propose a method to fuse data from a vision system, force-torque sensor and gripper finger position to evaluate the position of an object within the robotic hand while other, such as Moreira *et al.* (2016), propose a complex architechture to assess the outcome of a grasping operation. These systems have a common point where the robot must actually pick the object to evaluate the grasp.

Force-torque sensors can give us valuable information on the grasped object. We can image a system that would detect if an object has been dropped by reading the variations of *weight* from the sensor but, it is much harder to detect the object slipping in the gripper.

1.1.3 Tactile Sensors

Force-torque sensors give us a certain feedback on the interactions of the robot with its workspace but, in grasping we need the sense of *touch*. When it comes to manipulation tasks, Johansson and Flanagan (2009a) shows us that tactile information is greatly used by the human brain to asses and correct the grasp in order to maintain stability. Inspiring themselves from nature, researches and engineers have developed tactile sensors to collect contact information, to be used in several different situations.

A wide variety of sensors have been developed to mimic human fingertip sensors. Some are built to acquire pressure data while other collect vibration or shear to detect slippage and weight shifts. Some tactile sensors are based on electrical properties such as resistive (Weiss and Worn (2005)) or piezoelectric effect (Liu *et al.* (2017)) or capacitive energy (Rana and Duchaine (2013)) while others are based on optical (Lepora and Ward-Cherrier (2015)) or fibre optics (Fujiwara *et al.* (2017)) solutions.

These different tactile sensors have been used by many researchers such as Bekiroglu *et al.* (2011b), Bekiroglu *et al.* (2011a), Dang and Allen (2013) and Romano *et al.* (2011). Many more are experiencing with tactile sensors but, we mentioned only these researchers since they are all working on robotic grasping with tactile feedback. To do so, they are analysing pressure images of the gripper's contact with the objects.

1.2 Examples of Existing Intelligent Grasping Robots

Most of the sensors listed above have been in the market for some time. Some of which have made an impact in the robotic assembly lines industry. In this section, we will review the current typical assembly line robots followed by the developments that are being introduced in the modern intelligent automated manipulators.

1.2.1 Classical Examples of Robots Picking and Moving Products on an Assembly Line

When robots were introduced in the manufacturing industry, they were meant to accomplish highly repetitive tasks in a very controlled environment. The objective was to output higher volumes of products while perfectly replicating the process over and over. Typically, these robots have minimal input sensors that are very simple, such as position sensors letting the robot know if the next part is present or not. Once they are properly programmed and debugged, these assembly line robots are highly proficient in their work which is executed blindly.

The introduction of more complex assembly procedures has brought more sensors into the assembly line industry. It is not uncommon to see vision systems to correct any imperfections in the part locations and force-torque sensors to correct or conduct different operations in highly complex manufacturing procedures.

Moreover, we have felt a change of mentality when it comes to product development. We all wish to be different meaning that we want more customization options in the products we buy. This has created a demand for more modulable automated systems in order to keep up with the high volume output we need, coupled with the variety of the same product line.

1.2.2 Modern Intelligent Manipulating Robots

The evolution of the sensors, the effectors, the computation technology coupled with a want of more versatile robotic solutions has lead to big advances in the robotic grasping world. For example, Miller and Allen (2004) have made a versatile simulator for robotic grasping, *GraspIt!*, publicly available for other researches in the field of robotic grasping. Much effort is being placed into making intelligent robots that can *evolve* by learning.

Here, we are far from the industrial robot grasping the same object blindly at a set position. Researches such as Lin and Sun (2014), Kehoe *et al.* (2013), Bekiroglu *et al.* (2011a), Bekiroglu *et al.* (2011b) and Levine *et al.* (2016) have all introduced machine learning algorithms to their grasp strategies. Both grasp planning and grasp quality assessment are being *learned* by the robot.

This new generation of robots are far from being simple manipulators coupled to a few proximity sensors. We are talking about multi-sensor and sometimes multi-manipulator intelligent robots. With this new *intelligence*, the robots are no longer limited to industrial manufacturing but are slowly making their way into our homes (the Herb2 robot by Srinivasa *et al.* (2012) is a great example). Even in the industry, we are seeing collaborative robots come into play. These robots are much more *aware* of their surroundings allowing them to accomplish task alongside of humans instead of a highly securized cell.

1.3 Machine Learning in Robotic Grasping Strategies

In the past, machine learning was slowed down by a lack of computational power. With the advances in micro processors and general computer technology, machine learning is a highly studied field. But, in order to train complex learning machines, we first have a need for data. Researches Goldfeder *et al.* (2009) have built a grasp database and made it public. The advantage of such a public database is that different researches can compare their finding on a similar base. On the other hand, grasping data is very complex and can vary depending on the type of strategy.

Moving back to machine learning, we have found different approaches to the problem. Bekiroglu *et al.* (2011b) compares methods based on AdaBoost, support vector machines(SVMs) and hidden Markov models(HMMs) to assess grasp stability using tactile information as an input.Others, like Levine *et al.* (2016) have used convolutional neural networks (CNNs) to learn hand-eye coordination in robotics grasping. CNNs seem to be very interesting in the computer imagery field. Contrary to multi-layer perceptrons, CNNs have repetitive blocks of neurons which can move across the space of an image.

Considering that we were very interested in tactile presure *images*, we were interested in image restoration techniques. A common method for image restoration is sparse coding (Mairal *et al.* (2009), Mairal *et al.* (2008)), which is an unsupervised machine learning algorithm to extract high level features in unlabeled data. Sparse coding is not limited to image restoration but, we pushed our research in this direction since we can correlate many of the ideas used to our tactile pressure images. In fact, Raina *et al.* (2007) have proposed an approach that uses spase coding with unlabeled data in a supervised classification task.

CHAPTER 2

AUTOMATED PICKING STATION

In order to develop a grasp stability prediction system, we first needed a robotic grasping system. This robotic cell will be useful for collecting data, understanding the different grasping techniques and finally as a test bench to our system. In this chapter we will describe the final version of our automated picking system (APS). The original version of this system was developed to compete in a robotic picking challenge. The objective of the challenge was to automatically pick objects out of a bin with no prior knowledge of said object. The system was then refined over time to become stable and safe without the need of a safety perimeter. The final APS was a very useful tool to accomplish the work that was done in this thesis and was aslo useful for other work done at the CoRo lab.

2.1 Experimental setup

In this section, we will expose the material we chose to use for our APS and the reasons that led to these decisions. Also, physical installation specifications will be indicated as a reference point for future work.

2.1.1 Choosing the material

Our first challenge was to detect and locate the object to be picked in a three dimensional space. We decided to develop a vision system to accoplish this task. We needed a camera that was fairly easy to integrate considering that we had no vision expert in the team at the time. Our search stopped on the first version of the Microsoft Kinect©. The Microsoft Kinect© is widely used in research laboratories. The drivers for the camera and a lot of code was also available in open source which made development much quicker. Finally, the availability and low cost allowed us to obtain the camera rapidly.

With a vision system in place, we had to choose a robotic arm to build our APS. In light of the current tendencies of robot technology, we wanted our system to be safe for human contact. We believe that modern industry will create a demand for robots that collaborate with workers instead of always completing the work in a closed environment. We chose to use a UR10 manipulator from Universal Robots. The UR10 has six independent rotational joints and a maximum payload of 10 kg. Not only does the UR10 respect our collaborative station criteria, it was also very easy to integrate Robotiq's products.

The CoRo works in close collaboration with Robotiq, which facilitated the choice of end effector. We installed Robotic's FT-300 force torque sensor to our UR10. Also, we interfaced the two-finger 85 model gripper from Robotiq. This two-finger gripper is an underactuated adaptive parallel gripper. The underactuation of the fingers allows us to pick a variety of objects that can not be picked in a stable manner with a rigid parallel gripper but, the underactuation can also lead to unpredictable events that will be discussed later. We modified the two-finger gripper by replacing the original finger pads by tactile sensors developed in the CoRo lab.

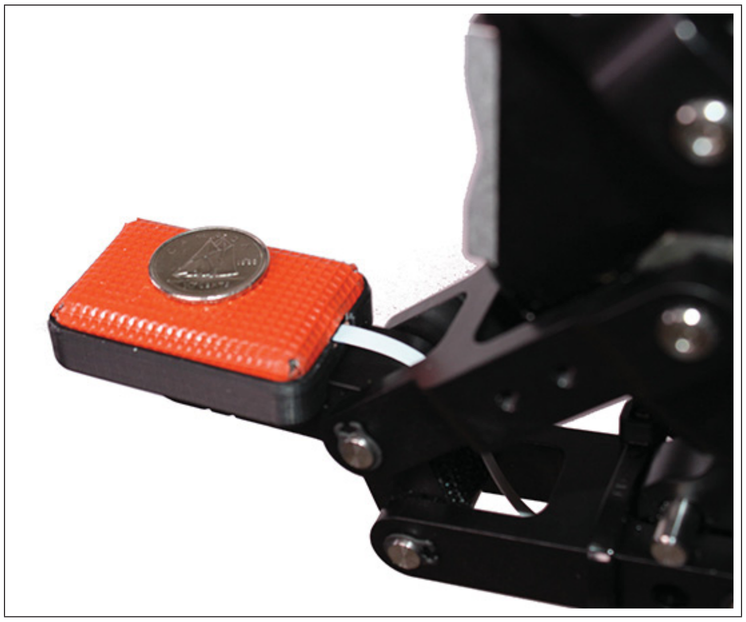


Figure 2.1 Tactile sensor

The tactile sensors we integrated (see figure 2.1) to our system are an evolution of our previous version, commercialized (patented) by Kinova Inc., and Robotiq Inc. The sensor we used was developed in our laboratory, at École de technologie supérieure (ÉTS). The sensor can measure both static (pressure images) and dynamic variations in pressure over time, although only the first type of information is used in this work. The sensor acquires both static and dynamic data at the same time and location since the sensor uses two independent acquisition channels. The device relies on capacitive sensing to acquire both types of data, with pressure data acquired at a rate of 25 Hz with a resolution of 4×7 taxels. The sensor has a wide measurement range and relatively high sensitivity because of its micro-structured polyurethane dielectric. The dielectric was built using a direct laser-etching technique, unlike the moulded dielectric in our laboratory's earlier version of the sensor by Duchaine and Rana (2014). The sensor can withstand up to 400 kPa. In comparison, Dargahi and Najarian (2004) show that a typical human grasp is between 10-100 kPa.

2.1.2 Installation specifications

As mentioned earlier, the original APS was developed for a robotics competition. This competition stipulated that the objects would be picked out of a bin. We first positioned our Microsoft Kinect © over the bin to have a top view of the objects to pick. To determine the distance between the camera and the bin we first reviewed the work by Andersen *et al.* (2015). We concluded that we have to place the camera between 1 and 2 metres from the bin. We put together a simple aluminium frame to hold our camera and attached it to the side of a worktable. Finally, the robot was positioned to respect proper reach to the bin and also to the general area on the table. Using the material we had in the laboratory, we constructed the station that can be seen in figure 2.2. We included the distances in milimetres on our image as a reference.

2.2 Programming environment and architecture

With all our material on hand, we had to figure out the best way to integrate all these different components. We chose to develop our software in a ROS environment. ROS is an open

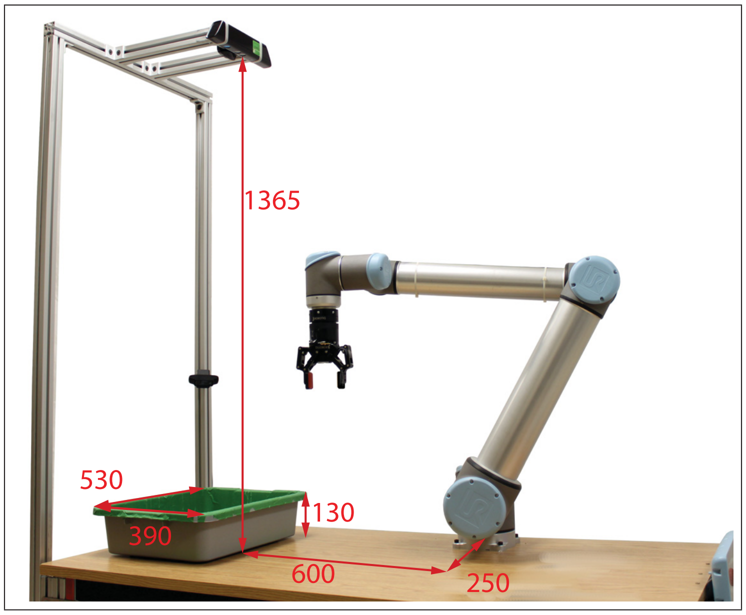


Figure 2.2 Mecanical setup

source flexible framework widely used in research laboratories. ROS manages all that is communication between the different material internally and offers easy solutions for transferring information from one module to an other. There exists two main communication techniques within the ROS environment (see figure 2.3). The first is a unidirectionnal communication protocol named **topics**. The server *publishes* information on a topic and then, clients can *subscribe* to the topic to read the information. The second is a bidirectionnal communication protocol named **services**. The server expects a *request* from the client, computes the data and send a *response* to the client. *Topics* and *services* are user defined which gives great liberty for the programmers to easily transfer information from one module to another. Hence, it is easy to integrate different hardware into one environment. Being an open source system, many packages are already available to use. Our APS uses open source packages for the Kinect© , the Robotiq force-torque sensor and gripper and the UR10 robot.

Two main programming languages are available in the ROS environment : Python and C++. We chose to work with C++ because of our prior knowledge of the programming language and for the slightly improved run time of the compiled language. We designed our software architecture in a star shape (see Figure 2.4). This architecture best uses the ROS environment

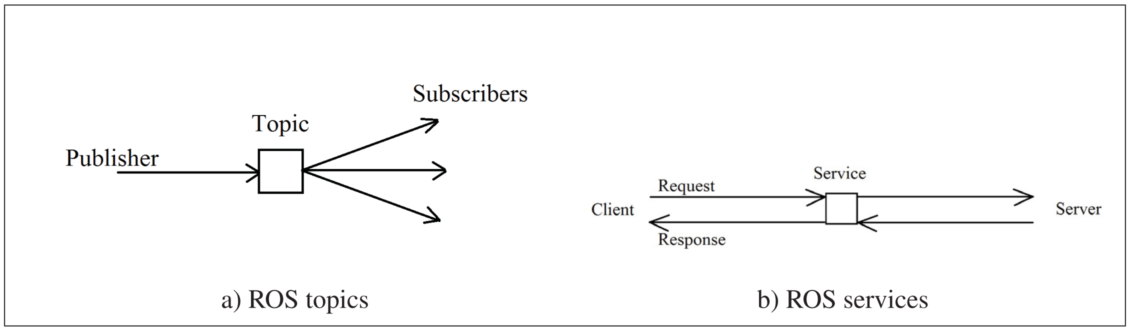


Figure 2.3 ROS basic communication techniques

services and makes the sofware easy to update or modify. The central program is a master client that dictates the flow of the program. For every task that must be accomplished, the master client interrogates the proper service, which correspond to the orbiting programs. The master client is the only program that can send commands to the different hardware. This structure is not only easily modulable, it also makes the system safer by centralising the hardware controllers. Finally, having a sequential central program removes all the synchronising problems of a multy thread program.

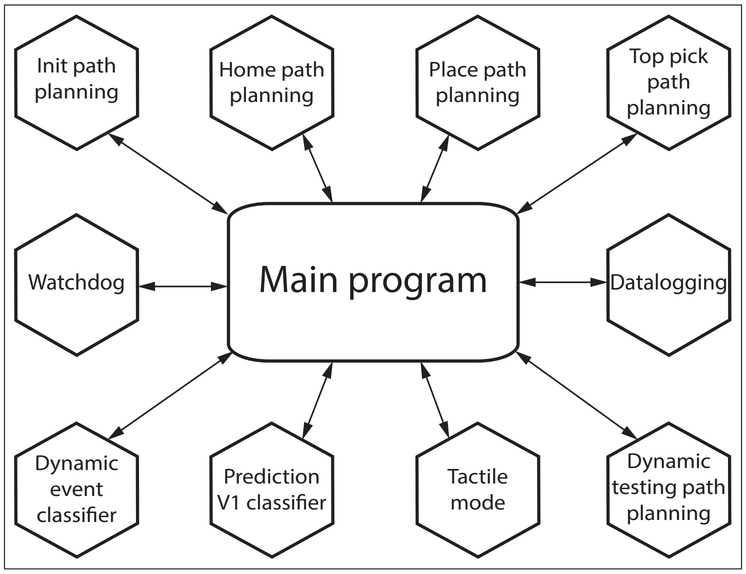


Figure 2.4 Main program architecture

The flow of the main program is quite staightforward and can be viewed in figure 2.5. As we said earlier, this is the final version of the program which has evolved into an automatic picking station requiring minimal supervision. In order to achieve minimal supervision, the test objects are picked 10 times from the bin. After being picked, the robot drops the object back into the same bin. Considering that simply picking and dropping the object does not represent the reality of a robotic task, we wanted to simulate the robot's navigation in the workspace. We developed what we called *dynamic testing*. The objective of the dynamic testing is to ensure the grasp is really stable and not simply holding by a thread. We simulated a robot navigation by creating linear and rotationnal accelerations with the robot.

Parallel to my work, one of my collegues, Jean-Philippe Roberge, developed a slip detection algorithm using dynamic events computed by our tactile sensors. In order to test and collect addition data, we integrated his slip detection system in our main program. The algorithm is launched by our main program during the dynamic testing. Moreover, we wanted to collect all the possible data from our different grasp attemps. Therefore, our program automatically starts and stops a datalogging thread. The datalogging thread saves all the possible information from the different sensors and from the robot. It was integrated to build a rich data set to be used by many reaserchers from our laboratory.

Finally, in the optic of having minimal supervision of the system during testing, we developed an automatic labelling system. Unlike the datalogger, the automatic labelling system is not a separate thread. It was directly integrated into the main program. A detailed description of this system is available further on in this thesis. Please refer to chapter 4.

2.3 Vision system

One of the main modules of our APS is the vision system. Visions systems have come a long way in the robotics world. From simple 1 channel systems to complex multi channel point clouds, researchers have developed highly efficient image recognition systems. Also, some of the best grasping algorithms rely on these high level vision systems. One of the goals of

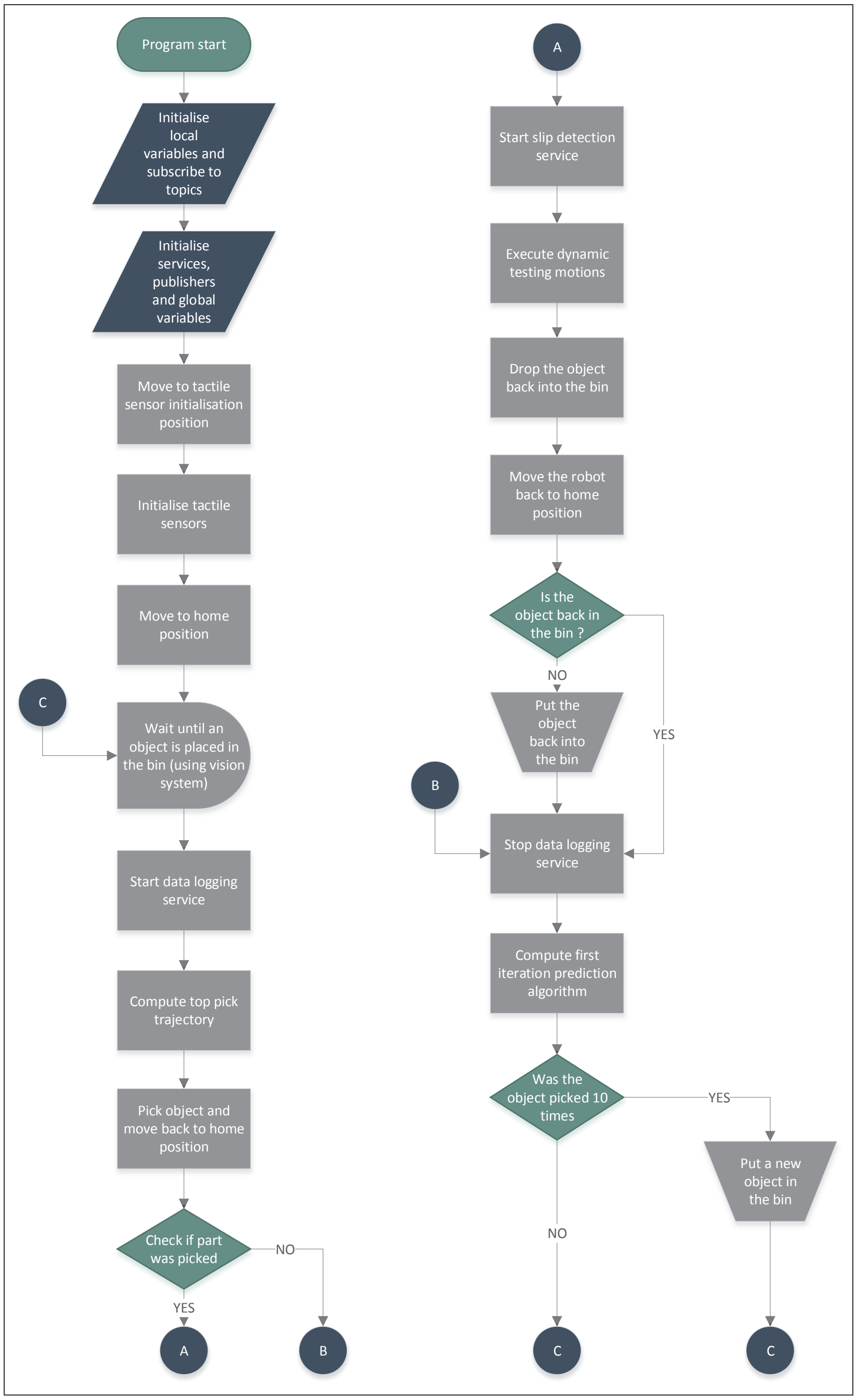

Program start
Initialise local variables and subscribe to topics
Initialise services, publishers and global variables
Move to tactile sensor initialisation position
Initialise tactile sensors
Move to home position
C
Wait until an object is placed in the bin (using vision system)
Start data logging service
Compute top pick trajectory
Pick object and move back to home position
Check if part was picked
NO
YES
A
B
A
Start slip detection service
Execute dynamic testing motions
Drop the object back into the bin
Move the robot back to home position
Is the object back in the bin ?
NO
YES
Put the object back into the bin
B
Stop data logging service
Compute first iteration prediction algorithm
Was the object picked 10 times
YES
Put a new object in the bin
NO
C
C

Figure 2.5 Main program - Flow chart

research is to be less dependant of these high computation vision systems. In this section we will expose the simple yet efficient vision system we developed for our APS.

Considering the fact that we did not wish to orient our research towards a vision system contribution, we set ourselves some simple criteria. First, we only wish to have an approximate position of an unknown object relative to our robot. Therefore, we want our vision system to return the geometrical center and the main axes of the object. This information, coupled with the camera reference frame, will be enough to compute a simple robot trajectory. As mentioned at the beginning of this chapter, the original use of our APS was for a picking challenge. Certain criteria were imposed from this challenge. The objects will have to be placed in a bin. The bin placement would have to be variable but, we limited to one object in the bin at a time.

2.3.1 ROS package and C++ libraries

Before jumping into software development, we first had to extract the raw data from the camera. Lucky for us, researchers had already made ROS drivers available to accomplish this task with the Kinect© . We simply needed to pick the driver that was most suited to our needs. After a few days of research, we found that the **freenect-launch** driver (an evolution of the **OpenNI** driver), not only fit within our requirements but also did a lot of pre processing of the raw data. That data is automaticaly published into ROS topics making the package easy to use.

As mentioned earlier, we had no vision expert in the project at the time. Therefore, we decided to keep our algorithm within this autor's vision knowledge. We decided to do most of the image processing on the two dimensional HSV image. We accomplished this task with a well known image processing library named OpenCV. Many developers use this library, making a lot of code available to accomplish different vision tasks with OpenCV. Also, the OpenCV library has integrated most of the complex vision algorithms that are widely used for image segmentation. And finally, OpenCV has functions to easily work with ROS topics.

Working with the two dimensional image came with a limitation, all of our computation was done in a pixel based system. In order to calculate a picking trajectory, we needed three dimensional space information of the object. Lucky for us, the freenect-launch driver aligned a point cloud with the pixel based image. In order to extract the information from the point cloud, we needed another library. We chose to use PCL to complete this task. PCL is a highly developed point cloud manipulation librairy. It could have been used to do the object segmentation completely but, all we needed was to extract cartisian information from specific locations in the point cloud. Integrating the PCL library in our project widened the expansion development possibilities of our vision system.

2.3.2 Object detection algorithm

As mentioned in subsection 2.3.1, the images produced by our camera are placed in ROS topics for further use. This came with two problems : first, the ROS topic format could not be used directly by our vision libraries and second, considering that we wanted to extract information from two different sources, we needed to synchronise the subscription to two different topics at the same time. Fortunatly for us, both libraries offered functions to translate our ROS topics into manageable formats.

With raw data in hand, we had to detect the object in the bin. Since we had no prior information on the objects, we could not use a database based strategy to locate those objects. We chose to use an elimination process on our original image to extract this information. As we can see in figure 2.6, the first segmentation we applied was to find our region of interest (ROI), which is the bin. In order to do so, we first converted the image to a grayscale format (Figure 2.7a) and we applied the Canny filter (Figure 2.7b). The Canny filter detects the edges in the image. Then, geometrical properties were used to locate the rectangles in the image. We assumed that the largest rectangle in the camera's scope would be the bin. With the bin detected, we could then focus our search in this ROI (Depicted as a red rectangle in figure 2.9) to locate the object.

These next steps were inspired by chroma keying, also know as green screen, from the special effects domain. The idea of chroma keying is to determine what is the dominant color of the backgroud in order to eliminate it or replace it by another background. In order to accomplish

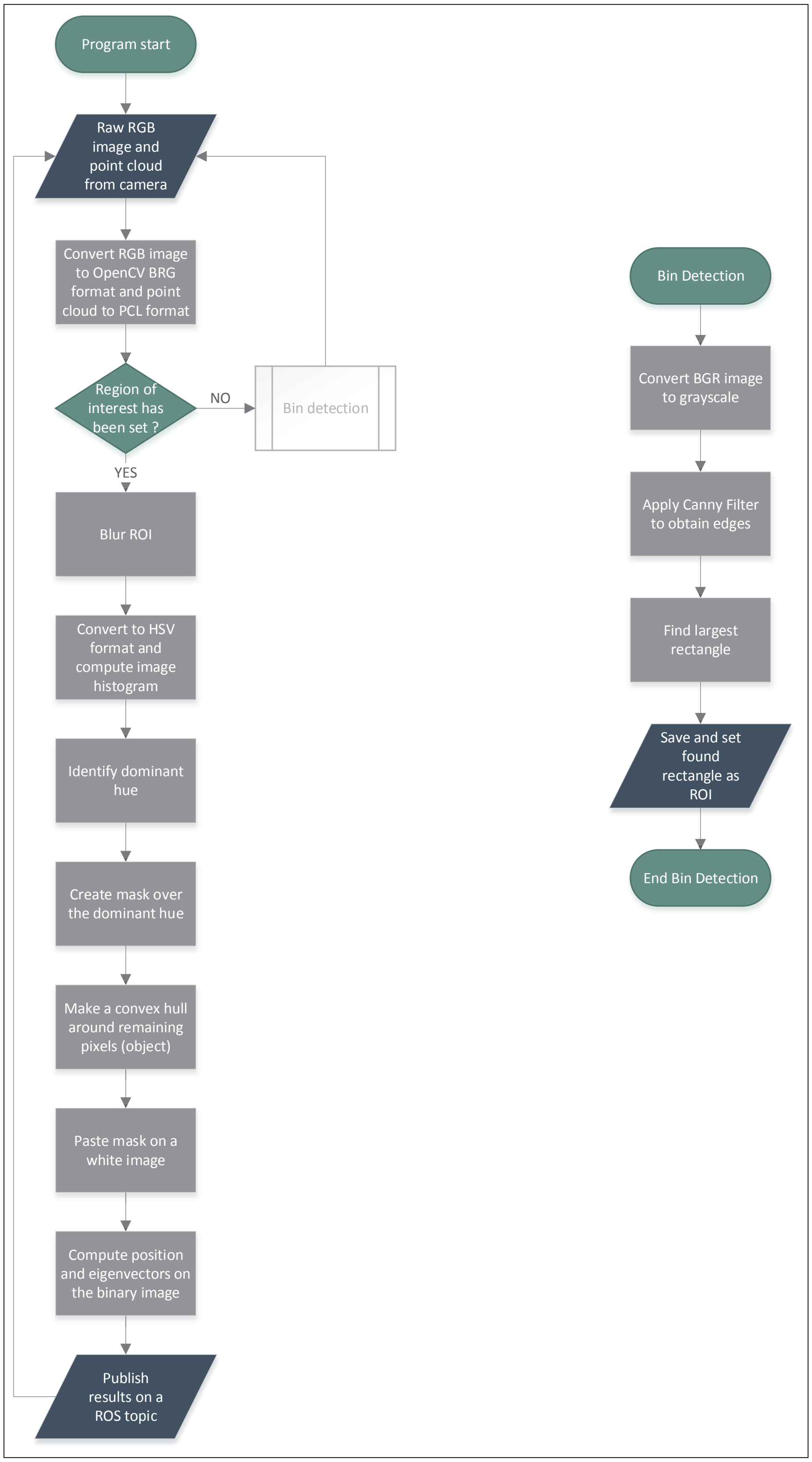

Program start
Raw RGB image and point cloud from camera
Convert RGB image to OpenCV BRG format and point cloud to PCL format
Region of interest has been set ?
NO
Bin detection
YES
Blur ROI
Convert to HSV format and compute image histogram
Identify dominant hue
Create mask over the dominant hue
Make a convex hull around remaining pixels (object)
Paste mask on a white image
Compute position and eigenvectors on the binary image
Publish results on a ROS topic
Bin Detection
Convert BGR image to grayscale
Apply Canny Filter to obtain edges
Find largest rectangle
Save and set found rectangle as ROI
End Bin Detection

Figure 2.6 Vision system - Flow chart

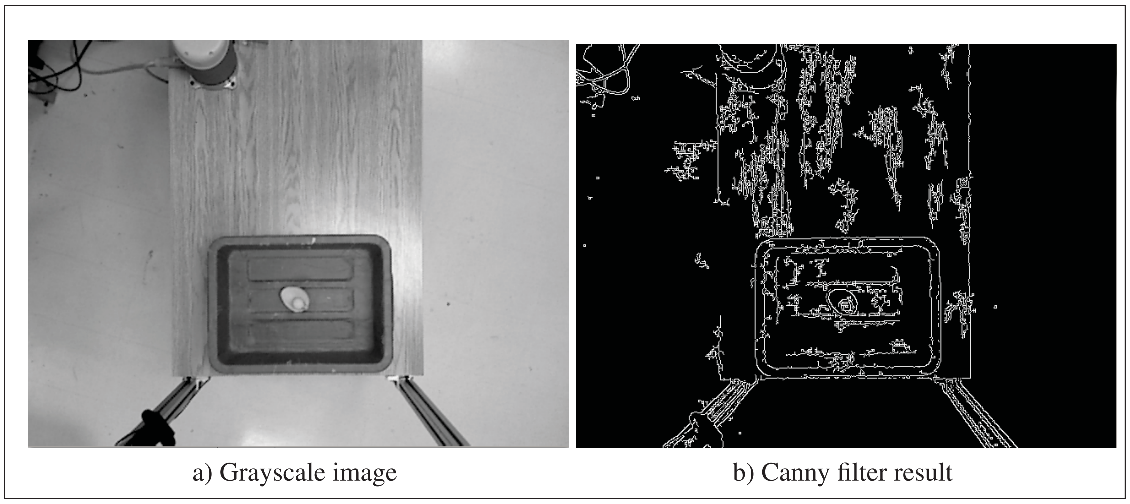


Figure 2.7 Vision system - Locating the bin

this on our ROI, we first blurred the image and converted it to the hue-saturation-value (HSV) format. We then computed the histogram of the image which allowed us to determine the dominant hue of the image. We used this information the create a mask over the backround. Finally by applying this mask to a white image, we isolated the object in a binary image.

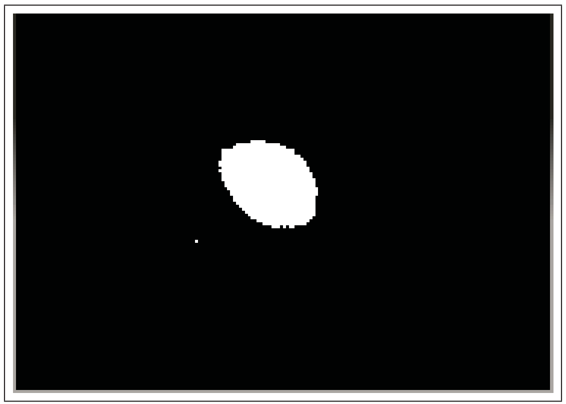


Figure 2.8 Vision system - Binary mask

An example of a binary image is shown in figure 2.8. With this finely defined contour, it is simple to extract the geometrical center and the main axes of the object which can be seen on the final result image (Figure 2.9).

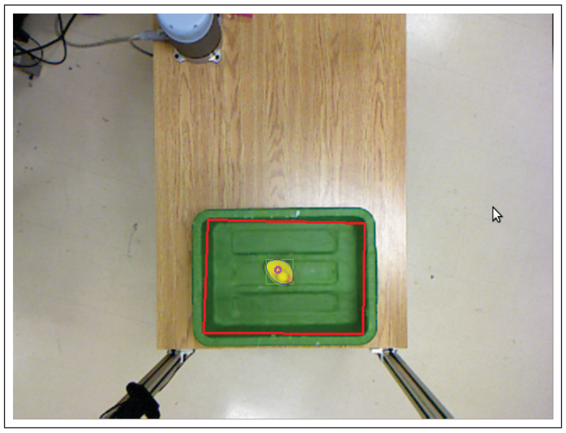


Figure 2.9 Vision system - Final result

Recall that we are working in a two dimensional image which meant that we were still in a pixel based system. Lucky for us, the freenect-lauch driver offers a point cloud which has a mathematical relation between the position of a pixel in the two dimensional image and the real position of the object in relation to the camera frame. In other words, the package already pre-processes the data and creates an overlay between the two dimensional pixel based image and the three dimensionnal point cloud. The relation between the two vectors is a simple mathematical solution, given in equation 2.1. We simply applied this simple equation before publishing the information on a ROS topic.

$$Pixel_X + 640 \times Pixel_Y = point(x,y,z) \tag{2.1}$$

2.4 Grasp strategy

Our APS has evolved to become a very efficient tool. Throughout its development, many grasping strategies have been developed and tested. For instance, Francois Levesque from Université Laval worked on a scooping technique to pick up thin items or object that did not fit in our Robotiq gripper. Other algorithms were developed using the shape of the object and its position in the bin to compute the best picking position. Finally, we also developed what we consider the most typical robotic pick, the simple top pick. The top pick is a simple trajectory where the robot approaches the center of the object from above to attempt the grasping.

At ÉTS, we chose to force the simple top pick at all times. Considering that we wanted to use the system mostly for tactile data collection, we found it important that the robot motion be similar from one pick to another. Computing the top pick trajectory begins by the final pick position. This position is computed from the vision system information. We placed the target at the geometrical center of the object and aligned the orientation of the gripper with the eigenvectors. From this target, we computed an approach position straight above the target at a secure distance to clear the bin. From the approach position, the robot can navigate safely within the cell. Again, to keep a certain similarity between the picks, the parameters of the gripper were set to fixed values.

Our top pick algorithm was not designed to be perfect. In some cases, it is bound to failure from the beginning but, we did not want a perfect system, we wanted to collect valuable data. By forcing the same picking strategy to collect data, we eliminated some variables allowing us to concentrate on studying the performace of our classifiers. Finally, using this technique, we obtained an automatic picking station that generates successful and failed grasping attemps.

CHAPTER 3

GRASP STABILITY PREDICTION SYSTEM USING TACTILE SENSORS

In this chapter, we will present the first version of our grasp stability prediction system. The general idea is to determine the outcome of a grasp before moving away with the object. At the moment of the robotic grasp, tactile information is used as an input to our system in order to predict the outcome of the grasp.

3.1 Proposed approach

The goal of this work is to improve robotic grasping by enabling a robot to distinguish between stable and unstable grasps for a variety of objects. To achieve this, we are proposing an approach that lets our system find the features of tactile images that are most relevant for the task of predicting whether a grasp attempt will succeed or fail. Our grasp analysis method is based on pressure images captured by a tactile sensor. The original aspect of our work comes from the fact that we used an unsupervised feature learning algorithm to achieve our goal, rather than hand-crafting the features.

In the past, several researchers have improved their robots' abilities to grasp a variety of objects by proposing different techniques. For instance, Bekiroglu *et al.* (2011a,b) have used hand-crafted features from pressure image moments and Romano *et al.* (2011) got their inspiration from human tactile sensing. However, the success of hand-crafted feature techniques is entirely reliant upon the researchers' abilities to determine the most relevant features.

In contrast, in our approach the auto-encoder (specifically a sparse coding algorithm) itself determines the most relevant high-level features of the unlabeled pressure image data. These high-level features are then used to classify the pressure image data with a support vector machine (SVM). The SVM classifier chooses the most relevant features (from among the high-level features) for distinguishing between the two groups. Mairal *et al.* (2014) show us that this type of classifier is well-suited to accomplish this task. By encoding the data and finding most relevant high-level features, we are hypothesizing that this will lead to knowledge of the combination of high-level features that most strongly correlates with the group of successful grasps (and likewise for the group of failed grasps). Thus the algorithm and SVM are working together to find the common denominators behind all successful (and all failed) grasps.

3.1.1 Data collection

At this point in our research, the APS (described in chapter 2) had not reached its final form. Nonetheless, the APS was able to automatically collect information by attempting robotic grasping on different objects.

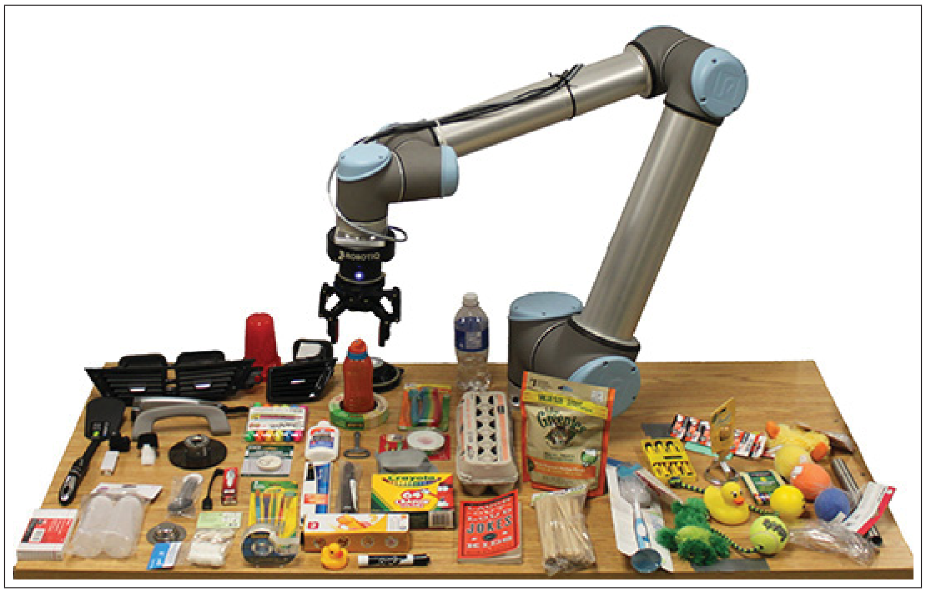


Figure 3.1 Objects used for first dataset collection

The dataset used in this chapter was composed of 540 different grasps. These grasps were done on 54 everyday objects which we attempted to pick 10 times each. The 54 objects can be viewed in figure 3.1. Pressure images were collected from the two tactile sensors at the moment when the gripper was in full contact with the object.

The data was manually labelled as being successful or failed. A successful grasp was defined as a grasp that allowed the robot to pick the object, navigate to the output bin and properly drop the object. Wheter the object was never picked or dropped while moving to the output bin, the data was labelled as a failed grasp.

3.1.2 Data Auto-Encoding

In this section, we describe the techniques we used to encode the raw data for automatic feature extraction. Much like the work of Bekiroglu *et al.* (2011a,b), we consider our static tactile pressure data to be an image. Here, we use *tactile image* to refer to the two pressure images from the sensors that were recorded at the moment of the grasp and placed side-by-side to make one composite pressure image (an example of a filtered sensor image can be seen in Fig. 3.11).

To give a theoretical overview of sparse coding, it works by creating a dictionary of *basis vectors*. Each basis vector is a high-level feature of the input data, and they are used to reconstruct the original image. In other words, the dictionary is used to represent our original tactile image patches as a linear combination of the dictionary's basis vectors. Our sparse coding approach uses image patches that follow the format of our tactile sensors. Since we combined our two pressure images, two patches are needed to reconstruct the pressure image. Our resulting dictionary is composed of basis vectors of dimension 28 (4×7).

In order to automatically generate our dictionary, here is the mathematical theory of sparse coding that we used. Let $\boldsymbol{x}^{(1)},...,\boldsymbol{x}^{(m)} \in \mathbb{R}^k$ be the m patches of a certain tactile image $\boldsymbol{X} \in \mathbb{R}^{k\times m}$, such that each patch has k taxel intensity values. The idea is to find a sparse vector $\boldsymbol{\alpha}^{(i)} \in \mathbb{R}^n$ for each $\boldsymbol{x}^{(i)}$ by using some *a priori* learned basis from a dictionary $\boldsymbol{D} := [\boldsymbol{d}_1,...,\boldsymbol{d}_n] \in \mathbb{R}^{k\times n}$, such that:

$$\boldsymbol{x}^{(i)} \approx \sum_{j=1}^{n} \boldsymbol{d}_j \alpha_j^{(i)} \qquad\qquad i = 1,...,m. \tag{3.1}$$

To obtain the sparse vectors $\boldsymbol{\alpha}^{(1)}, ..., \boldsymbol{\alpha}^{(m)}$ that capture high-level features of $\boldsymbol{X}$ in the dictionary, the dictionary of basis $\boldsymbol{D}$ must first be learned. This is done by minimizing the following objective function:

$$\min_{\boldsymbol{D},\boldsymbol{\alpha}} \sum_{i=1}^{m} \left(\left\| \boldsymbol{x}^{(i)} - \sum_{j=1}^{n} \boldsymbol{d}_j \alpha_j^{(i)} \right\|^2 + \beta \sum_{j=1}^{n} ((\alpha_j^{(i)})^2 + \varepsilon)^{1/2} \right). \tag{3.2}$$

The first term of eq. 3.2 inside the summation is the squared representation error, thus penalizing the objective function for poor representation of the input vectors. Regarding the second term, β is an arbitrarily-set scalar that will define the importance of *sparsity*. Sparsity is the ratio between the quantity of active basis vectors to the total number of basis vectors in the dictionary. A high sparsity corresponds to a low amount of active basis vectors. This second term penalizes the objective function when non-sparsity is high, and thus is responsible for making each $\boldsymbol{\alpha}^{(i)}$ sparse.

The double minimization problem stated in eq. 3.2, is a complex one that is known to be computationally expensive. However, Lee *et al.* (2007) have shown that it can be split into two convex optimization problems, which can then be solved iteratively. Moreover, dictionary learning can be performed offline so that it does not affect live operations.

When the dictionary $\boldsymbol{D}$ is complete it can be used to represent our data. In order to determine which basis must be used and with which intensity, the following equation must be minimized:

$$\min_{\boldsymbol{\alpha}} \left\| \boldsymbol{x} - \sum_{j=1}^{n} \boldsymbol{d}_j \alpha_j \right\|^2 + \gamma \sum_{j=1}^{n} |\alpha_j|. \tag{3.3}$$

To solve these mathematical problems, we used the MATLAB code made available by Lee *et al.* (2007). Fig. 3.2 illustrates how the dictionary is used to reconstruct a patch of our sensor image. The sparse vector α is obtained using eq. 3.3.

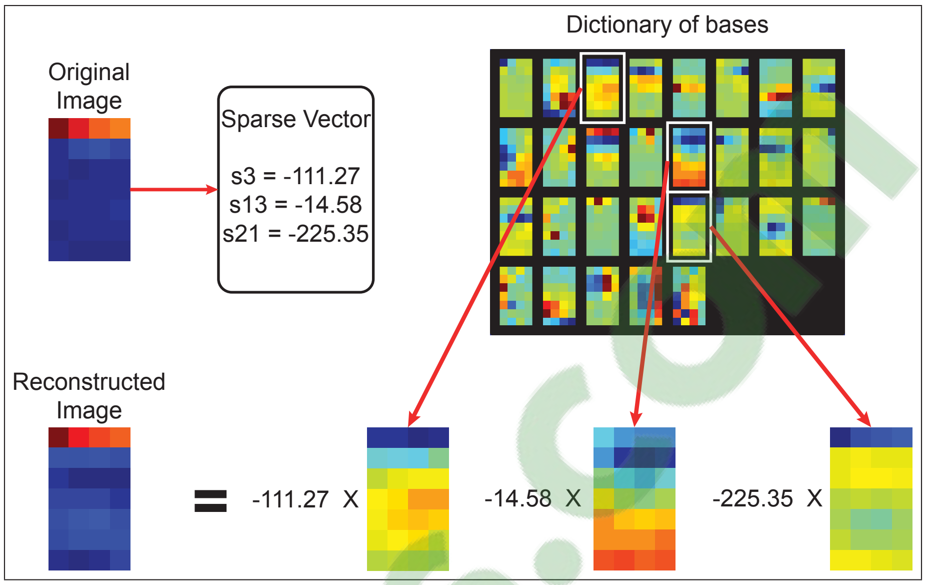


Figure 3.2 Reconstruction of a tactile pressure image using a dictionary of basis

The result of sparse coding is a set of sparse vectors composed of the coefficients for each basis vector needed to reconstruct the input vector. This can also be interpreted as a decomposition of our input vector into high level features.

3.1.3 Optimisation process

Sparse coding is a double optimization problem that has the objective of reconstructing an image as best it can under the constraint of using a limited amount of elements in a dictionary. Since the "optimal" dictionary is the one that yields the best classification results under the constraints of its sparse coding parameters, there are an infinite amount of optimal dictionaries because there are infinite variations of sparse coding parameters. Each of these "optimal" dictionaries will lead to different classification results. Our goal is to find the sparse coding parameters that will result in, as closely as possible, the best classification of our grasp data into success and fail categories.

There are many parameters in the sparse coding algorithm, as shown in the previous subsection. We chose to focus on only three of these parameters: the size of the dictionary (num_bases),

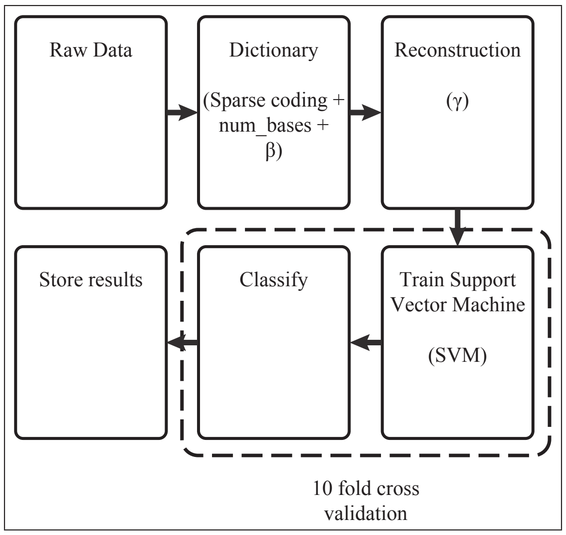


Figure 3.3 Optimisation process

the sparsity penalty factor of the dictionary construction (β), and the sparsity penalty factor used during the reconstruction phase (γ). For computational reasons, the dictionary learning and reconstruction phases use different sparsity penalties. The *epsilonL1* penalty function is used during dictionary learning (eq. 3.2), whereas the *L1* penalty function is used during reconstruction (eq. 3.3). Allowing differences between the two penalty factors (β and γ) enables the optimization process to have different sparsity levels during the two phases.

Since there is no straightforward closed-form solution that determines the optimal dictionary parameters (num_bases and β) and the optimal sparsity during the reconstruction phase (γ), we used a brute force approach: the grid search method. With every iteration of the grid search, we modified one parameter and computed the results. The optimization algorithm steps are the following: first, generate the optimal dictionary of basis using the efficient sparse coding algorithm; second, encode all the raw data using the reconstruction algorithm explained in section 3.1.2; finally, use a ten-fold cross validation to train linear SVMs to compute a weighted success rate (see Fig. 3.3). For every iteration, we saved all the data generated by the process.

The following is the weighted success rate used in the third step of the optimization process:

$$WeightedSuccessRate = \left(\frac{CCS}{TS} + \frac{CCF}{TF}\right) * \frac{100\%}{2}. \qquad (3.4)$$

This weighted success rate compensates for the fact that in our labeled data, the grasp successes outnumber grasp failures. It works by computing the ratios between correctly classified data (CCS and CCF) and the total data (TS and TF) for both labels. Often it is better to have a false negative (to incorrectly classify a success as a fail) than to have a false positive (to incorrectly classify a fail as a success). By applying this simple equation to our data, we give equal importance to correctly classified successes and correctly classified fails.

The results of this automated testing algorithm were then used to determine the optimal parameters for our system by extracting the best weighted success rates. The span and step for every variable were determined by manual testing of different combinations of parameters prior to launching the automated testing algorithm.

3.2 Experimentation

As mentioned earlier, MATLAB was used to accomplish our experimentations. Since we were using a grid search method, we chose to execute our experimentations in finite batches of tests. This allowed us to parallelize the experimentations, optimizing the usage of the multicore CPUs of our computer. Even by executing the optimisation process in many threads, the highly expensive computation of the results took many days. Once we had finished the entire grid, we assembeled the results into a big matrix to easily analyse them.

3.2.1 Experimental results

To validate our approach, we ran the optimization algorithm described in section 3.1.3 with the collected data. To compare each dictionary size (variation on the number of basis), we extracted the best result for each size by varying the other two parameters. The resulting weighted success rates for the best parameter combination at each basis size are plotted in Fig. 3.4. One can observe that the weighted success rates increase sharply until around an 11 basis dictionary. From this point, only small variations are computed. The overall best result is obtained with a dictionary of 29 basis.

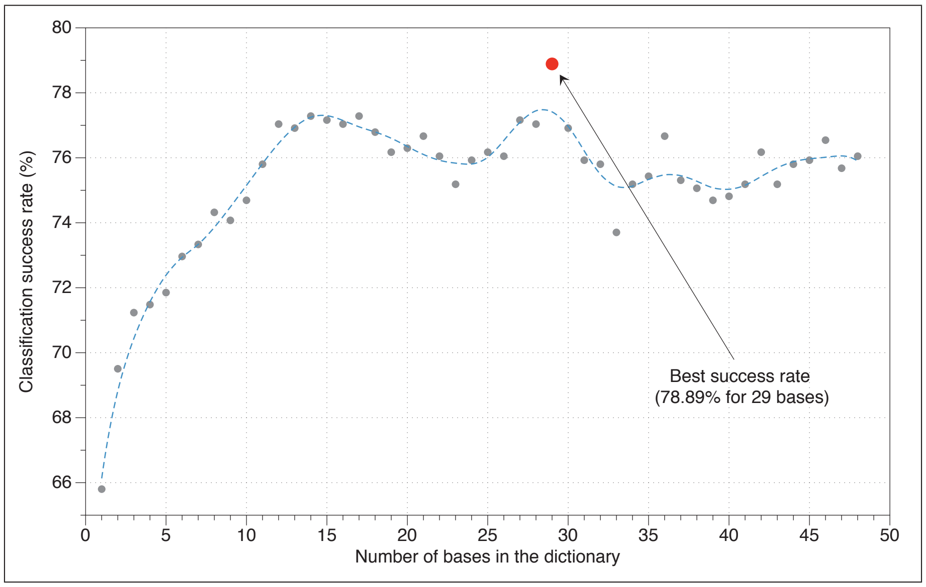


Figure 3.4 Top classification results per number of basis in the dictionary - First system

This dictionary can be visualized in its raw format in Fig. 3.5a. We also added a filtered version of the dictionary in Fig. 3.5b to help us understand the features represented by each basis. Observing the figures, we can recognize some attributes of the high-level features created by the sparse coding algorithm. For example, we can see in the third (3) basis of the dictionary a feature that may be described as an edge contact made at the tip of the sensor. Fig. 3.2 depicts the reconstruction phase of the sparse coding process. We note that our optimal dictionary is composed of 29 basis, whereas our original tactile image is composed of 28 taxels, so our optimal dictionary corresponds to the first iteration of an overcomplete dictionary.

When we used this dictionary to create sparse vectors for all of our raw data, we obtained an average sparsity of ***86.31%***. We used a 10-fold cross validation on our 540 sparse vectors to

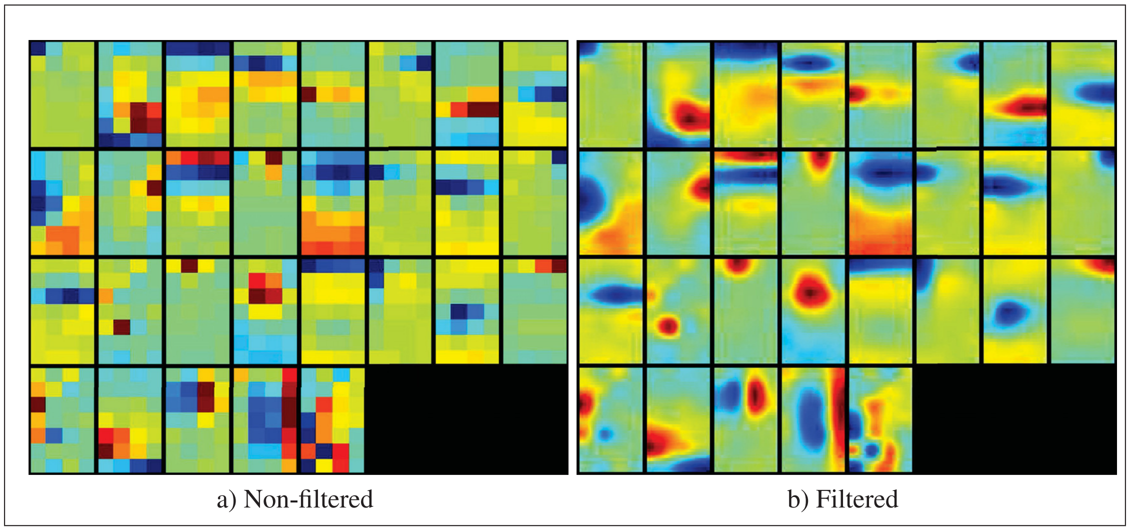


Figure 3.5 Optimal dictionary of basis - First system

verify the classification efficiency of the simple linear SVMs. Considering that we do not have an equal amount of successful and failed picks in our data, we used a weighted success rate to evaluate the efficiency of our classifier. We obtained a ***78.89%*** weighted success rate with the following parameters: 29 basis in the dictionary, a penalty on the dictionary optimization process of $\beta = 950$ and a sparsity penalty on the reconstruction process of $\gamma = 1000$. In comparison, Hyttinen *et al.* (2015) achieved a better success rate of 89% but, by using a higher number of inputs. In fact, our system uses only two tactile sensors as an input while their system relies on tactile information, hand configuration (joint angles) and 3d shape data derived from a vision system input. Therefore, we are confident that we could probably get better results by adding additional inputs to our system.

To further understand the performances of our classifiers, Fig. 3.6 shows the confusion map of our results. The success rate for classifying failed grasps is ***83.70%***. We would like to point out that we consider the success rate for failed grasps to be one of the most important indicators of the system's performance. We included the correctly classified successes in our weighted success rate because it was necessary for having the robot attempt the grasps. Otherwise, since we wanted to prioritize not dropping the object, but we did not place the same importance on not aborting a potentially successful grasp, the robot would logically decide to attempt fewer grasps, which would make for a frustratingly inefficient robot.

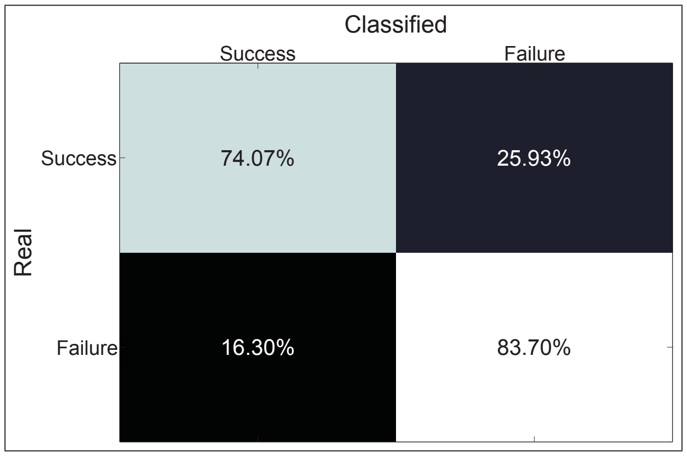


Figure 3.6 Confusion matrix - First system

3.2.2 Sparse coding analysis

The results presented in the previous section are promising for future work with tactile intelligence. We will now explore the effects of using sparse coding within our classification method to better understand what needs to be done to augment the performances of our classication system.

The sparse coding algorithm needs input data that fully represents our population. One may ask how much data is needed for a statistically sound representation of our population. In our problem, the population is very hard to represent considering we want a system that can grasp any object that can fit in our gripper. As mentioned in section 3.1.1, we collected data from 540 picks. By pure coincidence, our labels were separated perfectly into 75% successful and 25% failed picks: we had 405 examples of successful picks and 135 examples of failed picks. Since our priority is to capture failed grasps, we most likely could improve our results by adding more data from failed picks.

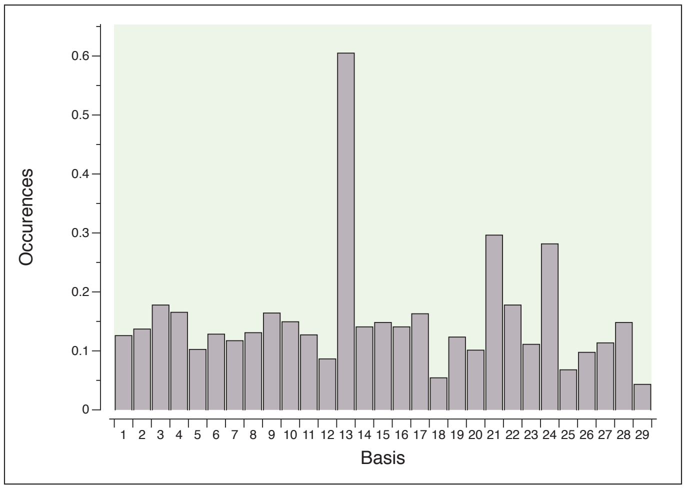


Figure 3.7 Successful picks basis usage

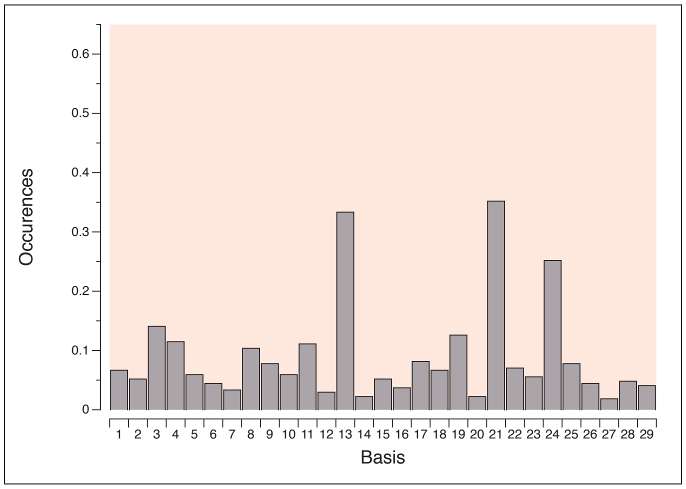


Figure 3.8 Failed picks basis usage

Now, we turn our attention to the sparse representation of our input vectors. We were interested in understanding how our optimal dictionary was used to represent our input vectors. As shown in eq. 3.3 we included a *sparsity penalty* which translates into using as few basis from the dictionary as possible (given the other parameters) to represent each patch of our image. We computed the average quantity of basis used to represent successes and fails. On average, 4.4259 basis are used to represent a successful pick patch and 2.5926 basis are used to represent a failed pick patch. By observing the tactile images, we saw that most successful pick images had larger active areas (compared to failed pick images), indicating that more of the sensor was in contact with the object during successful picks. We can infer from this that successful pick images require more basis for their reconstruction than failed pick images.

This led us to wonder whether certain basis of the dictionary are used mainly to represent one class of picks. In Fig. 3.7 and Fig. 3.8 we computed the weighted occurrences for every basis of the dictionary by label. Each bar is a normalized occurrence of the basis by label. For example in Fig. 3.7, basis 13 is at approximately 60% occurrence, meaning that this basis was involved in the representations of 60% of successful picks. Unexpectedly, the same three basis are used more then 25% of the time for both successful and failed picks. These basis are extracted in Fig. 3.8. If we analyze the basis in this figure, we notice that they are mostly fingertip edge contact features, but we still do not know how these features help the classifier.

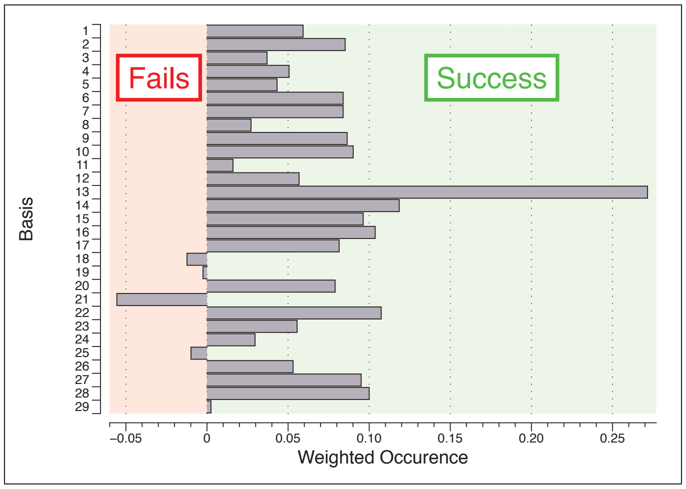


Figure 3.9 Basis usage differential

We decided to push this analysis further. We wished to know whether some of the basis are used mainly for one label or the other. To find out, we computed the differential of the two graphs in Fig. 3.7 and Fig. 3.8, with the results shown in Fig. 3.9. If the bars are in the left section of the graph, the basis are more present in failed picks, and vice versa. In this case, we see a new set of basis that are most often used for failed picks. Only basis #21 is in both sets (used most often for failed picks in both the original graph and the differential graph). This basis is used to construct a pressure image of a pick on the very edge of the fingertip, whereas the other basis of this analysis seems to represent very weak contact points. If one were to try to hand-craft features, we speculate that #21 would probably be one of the basis to consider using.

Here we investigate the role of coefficients. In Fig. 3.9 we can see the differential of the absolute coefficient averages per basis. This time, if the bar is towards the left of the graph, we can say that the basis has a larger coefficient value when used to represent a failed grasp. We notice that failed grasps are more likely to have a strong coefficient. If we couple this information with the fact that failed picks are usually represented using fewer basis, we could hypothesize that a strong activation of a few specific basis would allow us to classify the pick as a fail.

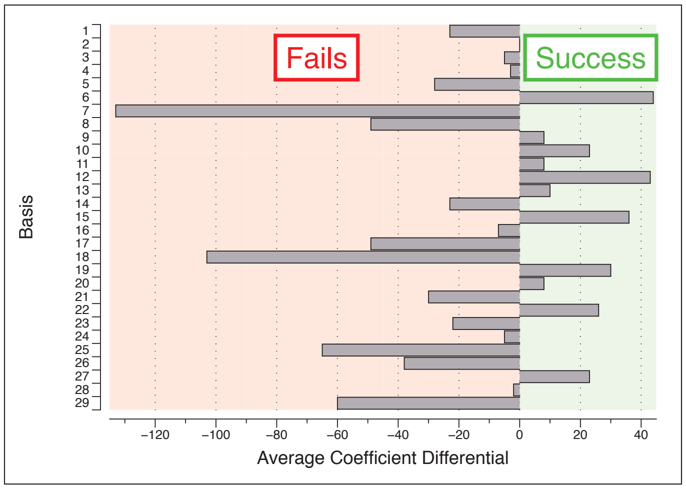


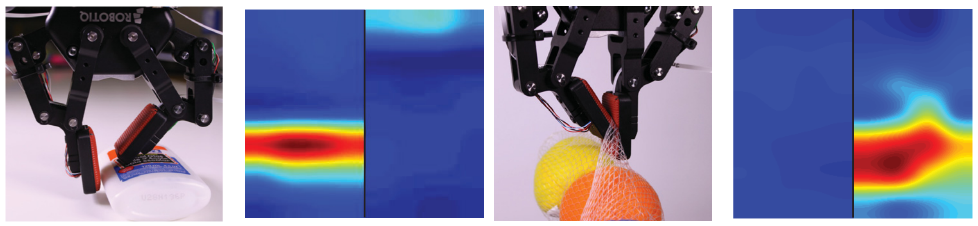
Figure 3.10 Extracted basis of second analysis of sparse vectors

3.2.3 The classifier's performance analysis

In addition to our main goal, we also hoped to find the common denominators behind successful and failed grasps for all objects. The fact that a simple linear classifier like an SVM can separate successful and failed grasps for 54 objects (with 78.89% weighted success rate) is encouraging because it indicates that other objects could possibly be classified similarly. However, the way the k-fold was performed (the data were shuffled randomly) means we cannot be sure that the test data sets were from objects not used for the data in the training sets.

To address this concern, we performed the same experiment using new data collected for the experiments of chapter 5. A description of this new dataset is available in chapter 5. Suffice to say here that this experiment was conducted with data from 50 new objects, never before seen by the system. These 50 new objects are similar to those presented in Fig. 3.1, such as the ones presented by Calli *et al.* (2015). The results of this new experiment rendered a weighted success rate of 71.36%. Although we did not reach the same level of performance, the results are encouraging. They are especially promising when we consider that we had to replace the sensors after the first experiment (because they broke), and the sensors are still in the development phase so they are not as reliable as industrial ones.

In further work, we will consider whether to continue with SVMs or switch to other popular classification methods such as convolutional neural networks (CNNs). Either way, we will gather more data. Considering that we wish to properly classify grasp outcomes for unknown objects, we need to be able to represent as many pressure images as possible. Moreover, with more data, we could add a testing set instead of using only K-Fold.



a) Low quality grasp - failure b) Low quality grasp - success

Figure 3.11 Low quality grasp examples

Lastly, as mentioned earlier, we noticed that the underactuation of the gripper can create some confusion in the data (one example can be seen in Figs. 3.11a and 3.11b). In both cases, if we only concentrate on the pressure images, they can seem very similar. Our algorithm only uses the static pressure images, so in future work we might include more information to potentially get better classification results. We will attempt to integrate some gripper information into the algorithm, such as by using the integrated inertial measurement units (IMUs) to compute the finger positions in 3-dimensional space. Therefore, we will study the different data fusion techniques to hopefully correctly classify this confusing data.

CHAPTER 4

AUTOMATED LABELLING SYSTEM

A common problem with machine learning is lack of data. As we hypothesised in the previous chapter, we believe that with more data, we would be able to observe better results. Data collection can be time consuming. Also, it is critical that the criteria to label our data be robust. In light of these remarks, we decided to develop an automated labelling system to add to our APS.

4.1 Defining the labels

In the spirit of developing tools for our laboratory, the automatic labelling system we put in place was not only for the research presented in this thesis. Therefore, before starting the automatic labelling system, we first decided to define the different labels for the grasping attemps.

In chapter 3, we limited our labels to *success* and *failures*. But, as mentioned in chapter 2, the APS and the data it generates is not only used for the work presented in this thesis. In order to satisfy the different researchers of our laboratory, we had a meeting to discuss the needs of all. We all agreed that the *success* label would not be changed. A *successful* grasp is still defined as such: a grasp where the robot is able to pick the object, submit it to a dynamic test and return the object to the original bin.

We then turned our attention to the different failures possible. At the time, there were two main research branches in the laboratory using tactile data. First, the analysis of static tactile data (such as the work presented in this thesis) and second, the study of dynamic events using tactile sensors. This lead us to define two distinct failure labels: *static failures* and *dynamic failures*.

We defined a *static failure* as a grasp attempt where the object is never picked by the robot and we defined a *dynamic failure* as a grasp attempt where the object slips out of the gripper fingers during the robotic motion.

Our definition of a *static failure* implies two distinct phenomena : either the object is never bound to the gripper or the grasp is of such low quality that it is dropped at the first sign of movement by the robot. This second phenomenon was highly debated within the researches because it could also be labeled as a *dynamic failure*. Considering that the *dynamic failure* label was going to be used by researches to detect slipping events with the tactile sensors, we all agreed that a grasp attempt bound to fail from the start (and not because of slipping) would be categorized as a *static failure*.

4.2 Labelling algorithm

With our newly defined labels in hand, we now focused our attention on developing the labelling algorithm. We decided to base our automatic labelling algorithm around the vision system of the APS. This was the simplest and quickest way to implement the system. We can actually visualize the labelling system in figure 2.5 of chapter 2. The algorithm was directly inserted into the flow of the main program.

Lets take a few steps back and remember the main lines of the APS functionality. First, the robot computes a picking trajectory for the object placed in the bin and executes it. Then, the robot moves away from the bin and executes a dynamic test. Finally, the robot returns the object to the bin for the next grasping attempt. Keeping this sequence in mind we can *link* our labels to the presence of the object in the bin at different times during the execution of the main program.

At the beginning, we know there is an object in the bin because the execution of the program needs vision input to compute the grasping trajectory. Afterwards, the robot attemps the grasp and moves out of the bin. At this point, we check if the object is still present in the bin. If we can see the object in the bin, we know the robot failed at grasping said object therefore, we can already label this grasping attempt as a *static failure*. On the other hand, if we do not see the object in the bin, we move along to the dynamic tests. As mentioned earlier, the dynamic tests were put in place to confirm the stability of the grasp. Once the tests are complete, the robot returns the object to the bin and returns home. At this point, we use the vision system to once again check if the part is in the bin. By cause and effect, if the object is not seen in the bin, we can conclude that it slipped out of the robot's gripper, making this grasping attempt a *dynamic failure*. Finally, if the object did make the journey back to the bin, we label the attempt as a *successful* grasp.

4.3 Evaluating our automated labelling system

As we mentioned in chapter 3, we needed to gather more data to continue our research. This was a perfect opportunity to deploy and test our automatic labelling system. This new dataset was performed using our APS (described in chapter 2) on 100 different every day objects. Most of the objects from our first dataset were used for this new iteration of data collection. More on this dataset will be explained in chapter 5. At this point, all we need to know is that out of the 1000 data points, 778 were *successful*, 193 were *static failure* and 29 were *dynamic failure* gasping attempts.

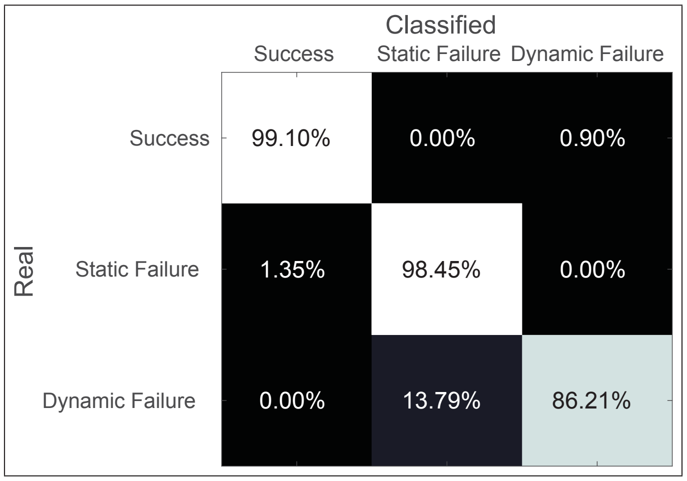


Figure 4.1 Confusion matrix - Automatic labelling with 3 labels

When we collected the data for our second dataset, both human input labels and automatic labels were saved. When we compare the labels from the automatic labelling system to the labels entered by a human, we get a 98.6% success rate for the automatic system. We computed the detailed results into a confusion matrix in figure 4.1. When we analyse the results, we first notice that the automatic labelling system is very effective at labelling successful grasps. The confusion happens more often between *static failures* and *dynamic failures*. Overall, we are pleased with the results we obtained with the automatic labelling system. One could argue that this slight imperfection in the system adds just the right amount of confusion in the data to make it realistic. But, our goal was to obtain a perfect automatic labelling system. We will discuss ways of improving the system in the next section of this chapter.

In section 4.1 we explained the reasons behind the new label (*dynamic failure*) but, the goal of this thesis is not to make a distinction between *static* and *dynamic* failures. Keeping that in mind, we decided to evaluate our system in our context (only *success* and *failure* labels). To do so, we combined both labels into one unique *failure* label. We computed the results into a data matrix, figure 4.2. If we analyse this new confusion matrix, we notice that the labelling errors are very small thus, making this system viable in a two label context.

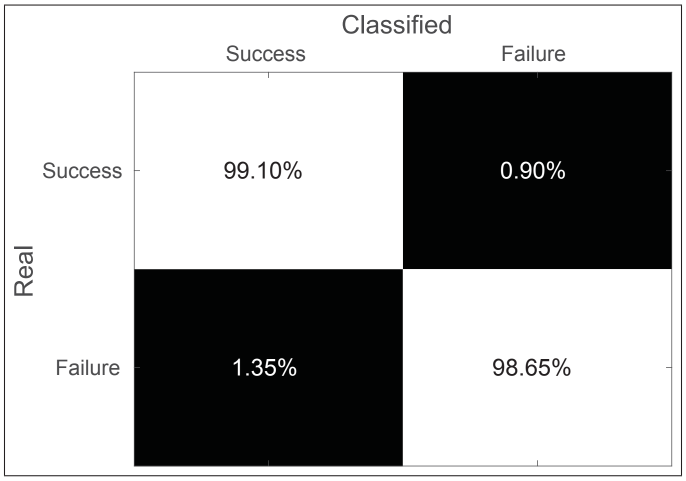


Figure 4.2 Confusion matrix - Automatic labelling with 2 labels

4.4 Possible improvements on the automatic labelling system

In order to suggest possible improvements to our system, we first had to understand the source of the errors. If we look at figure 4.1, we can see that the automatic labelling system had three types of errors: dynamic failures categorized as static failures, static failures categorized as successes and successes categorized as dynamic failures. Lets analyse each one of these errors individually.

The highest source of errors are dynamic failures categorized as static failures. This error is also the easiest to explain. Keeping in mind that the automatic labelling system is based only on a vision system, we knew that some events would not be properly caught. This error occured when the robot grasped the object but, it slipped and fell right back into the bin. Since static failures were defined by a before and after grasping attempt criteria, these slippages (or dynamic failures) were categorized as static failures.

The lowest source of errors are successes categorized as dynamic failures. From an outside point of view, these errors could be very hard to explain. Since we had a first row seat during the data collection, we quickly realized that these errors were associated to very specific types of objects. Most of these errors happened with glass objects and clear wrappers. Depending on the way these objects fell back into the bin, certain times the vision system would simply not detect them. Therefore, these objects were properly grasped, underwent the dynamic tests successfully but were not seen when returned to the bin. The other type of object that caused this error simply bounced out of the bin when dropped. Again, the object underwent the grasping and dynamic test successfully but could not be seen by the vision system in the end.

The final source of errors was static failures categorized as dynamic failures. These errors are easy to explain but are very subjective to the human in charge of inputting the manual label. These events are grasping attemps that pulled the object out of the bin but were bound to fail from the start. Let us explain, in some cases, the objects got caught on the gripper not because of a proper grasp but more of a *lucky* mecanical event. This would cause the part to be pulled out of the bin but never properly grasped by the robot. In these cases, the object would always fall out of the bin durnig dynamic testing.

Our system could easily be improved by using the data from the force-torque sensor. This data would allow us to catch some of the two first types of errors. In the case of dynamic failures categorized as static failures, we could motnitor the variation in weight to detect the slippage above the bin. Using the same approach, we could monitor the weight before dopping the object back into the bin, reducing the successes categorized as dynamic failures due to the vision system failing to see the object in the bin. But, when it comes to detecting the last type of errors, the weight from the force-torque sensor would probably not be enough. These type of subjective decisions are the motivation for our work. We could probably imagine highly complex algorithms using mutiple input sensors to detect these events but, we believe it would simply be easier if we could predetermine the outcome of a grasp before executing it.

CHAPTER 5

EVOLUTION OF OUR GRASP STABILITY PREDICTION SYSTEM USING INTEGRATED IMUS

At the end of chapter 3 we first concluded that we needed more data to continue our research. This was addressed by collecting new data for 1000 grasping attempts. This time, we used a newer version of our tactile sensors which had integrated IMUs. To collect this data, we used our APS with 50 objects taken from our original lot of objects and introduced 50 new objects to augment the variety of grasps. Our second conclusion was that some confusion was added by the gripper's underactuation possibility. To address this hypothesis, we will propose different techniques to integrate the IMU data into our algorithm.

Prior to executing any new tests on the dataset, we first verified its content. From the 1000 grasping attempts, 778 resulted in successful grasps and 222 were failed grasps. In this dataset we do not have a perfect 75%-25% distribution between successes and failures but, since we are still using our weighted success rate (equation 3.4) we will be able to compare our new results with the first iteration.

In the first part of this chapter, we will validate our original approach by testing our first systems parameters on our new dataset. Then, we will explain the parameters of our new tests to finally expose our final results in this research.

5.1 Testing our old system with the new data

In this section we will validate the approach we use in our original system. Our first test will be to feed our new data to the original system which has been fully trained using the old dataset. Afterwards, we will train new systems using our new data but, we will keep the metaparameters of our original optimized system.

5.1.1 Validation of our tactile classifier

The easiest test we conducted was done during the data collection. We integrated our prediction system directly in the APS. We collected the results at the same time as the new data. It was interesting to follow the results of the predictions system in real time. Also, it allows us to measure the impact on computation time of our prediction system.

Using the original prediction system on our new 1000 grasping attemps, we obtained a weighted success rate of 69.16% and a success rate of 73.30%. At a first glance, these results are disappointing when compared to the 78.89% and 76.48% results we obtained with the original system. The first observation we made is that our weighted success rate is lower then the success rate. This can be translated to: we are less effective at predicting failures. Again, this observation left us wondering why the system had performed this way.

To better understand what is happening and to compare with our original results, we present the results in a confusion matrix.

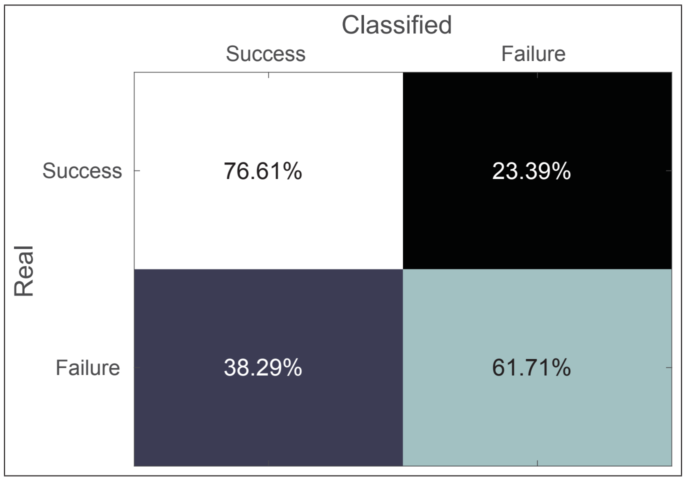


Figure 5.1 Confusion matrix - Old system with new data

When we compare figure 5.1 with figure 3.6 we can quickly see that our system has mostly failed on predicting failed grasps. We thought there might be two possible explanations. First, changing the tactile sensors could have had a small effect on the results. But this can not justify getting lower results because our system has to be able to acheive similar performances regardless on the tactile sensor used. Second, we thought that this further justifies our original hypothesis that we did not have enough data to train our original system. If the original dataset did not have enough failed grasp examples to train both the dictionary and the classifier, it is probable that giving new types of failed grasps to the system would lower its performances. On the other hand, we got better performances at classifying successful grasps then our original tests.

5.1.2 Training new classifiers with old metaparameters

In the previous section, we saw that our optimal system from our first dataset was not very good at classifying new failed grasps. Our hypothesis is that we did not have enough data to properly train the system. Consider that our optimization was done in two distinct steps: first, we contructed a dictionary to get a sparse representation of our data, second, we trained a linear SVM to classify the sparse data.

In this section, we will try to demonstrate that our hypothesis was valid. Also, we want to understand whether the sparse coding or the SVM caused the poor results or again, if it was the combination of the two.

5.1.2.1 New SVM with old dictionary

We first decided to evaluate the SVM. To do so, we first got a sparse representation of our data using the original optimal dictionary (see figure 3.5) and then, we trained a new SVM using the same techniques as described in chapter 3.

We computed the results and placed them in a confusion matrix (see figure 5.2). These results correspond to a weighted success rate of 79.52% and a general success rate of 79.90%. If we compare these results to our first iteration, we notice similar performances (in this case, a bit better overall performances). Also, it is important to note that this new SVM renders a higher score in the general success rate then the weighted success rate. This tells us that our new SVM is more performant at classifying successful grasps over failed grasps.

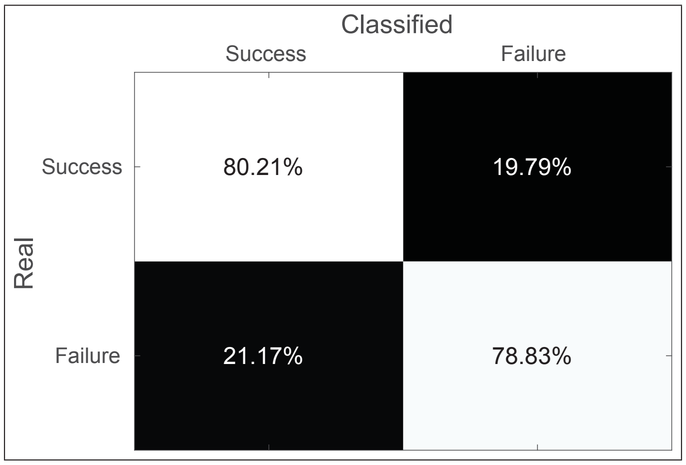


Figure 5.2 Confusion matrix - Old dictionary, new SVM and new data

The results we obtain is a good indication that our original dictionary was rich, meaning that it is performant at representing new data. On the other hand, we learned that the classifier (linear SVM) did not transport its performances with new data. At this point, we started to wonder if we should continue working with the linear SVM as a classifier.

5.1.2.2 New dictionary and SVM

As researchers, we wished to fully understand the effect of the different metaparameter we had found in our first optimization process. In this section, we trained a new dictionary of basis using the optimal metaparameters found in chapter 3.

We rendered our new dictionary in both its raw and filtered version, just as in figure 3.5, in figure 5.3. If we compare these two dictionaries, we notice many similarities in the basis. This was to be expected considering we were building a dictionary of basis to represent the same type of data. The beauty of sparse coding is to break down our data into *basic features* to represent our data. On the other hand, we do notice some new basis that we had not seen before. This further confirms our hypothesis that more data would render better results (even thougt these are only *partial* results).

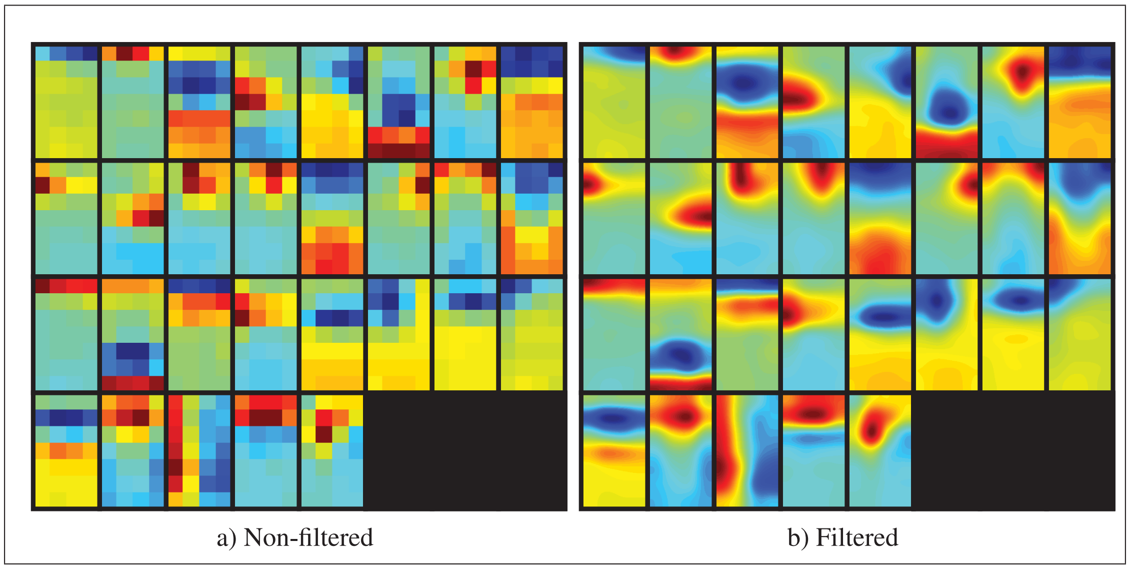


Figure 5.3 Optimal dictionary of basis - Old metaparameters with the new data

Using the new dictionary of basis represented in figure 5.3, and using the same techniques described in chapter 3, we trained a new classifier. Once again, we computed our results in a confusion matrix that can be viewed in figure 5.4. These results correspond to a weighted success rate of 78.43% and a general success rate of 76.20%. These results are disappointing since they do not improve on any of our previous systems.

What we can conclude at this point is that greater data does not give us a better system. It may have helped us generate a better dictionary of basis but, it has not given us an overall better classifier. Perhaps, if we ran the entire optimisation process again to find new metaparameters we could obtain better results.

5.2 Running the optimization process with the new data

We decided to run the optimization process once again using our new data. The process is exactly the same as we described in section 3.1.3. Once again, we plotted all the best results

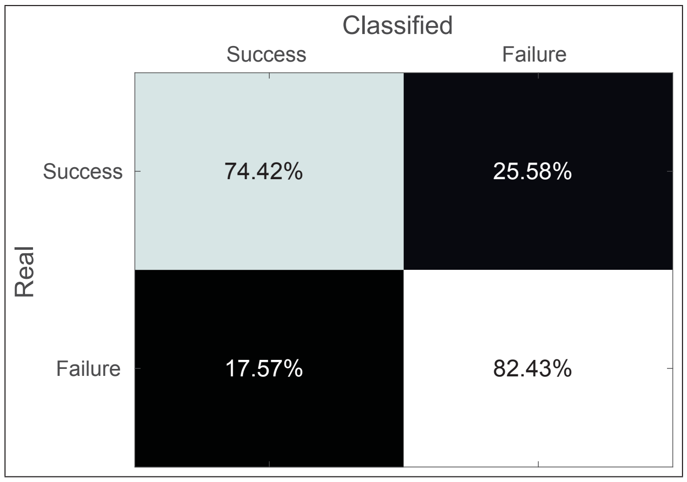


Figure 5.4   Confusion matrix - New dictionary, new SVM and new data

per number of basis in the dictionary. In figure 5.5, we can see that 29 is no longer our optimal number of basis. We obtain a better result using a dictionary containing 41 basis. Nonetheless, we also notice that the results remain pretty stagnant after 29 basis therefore, we consider that all the systems could be used to classify the data. In order to remain true to ourselves, all the subsequents test in this thesis will used the optimal dictionary with 41 basis.

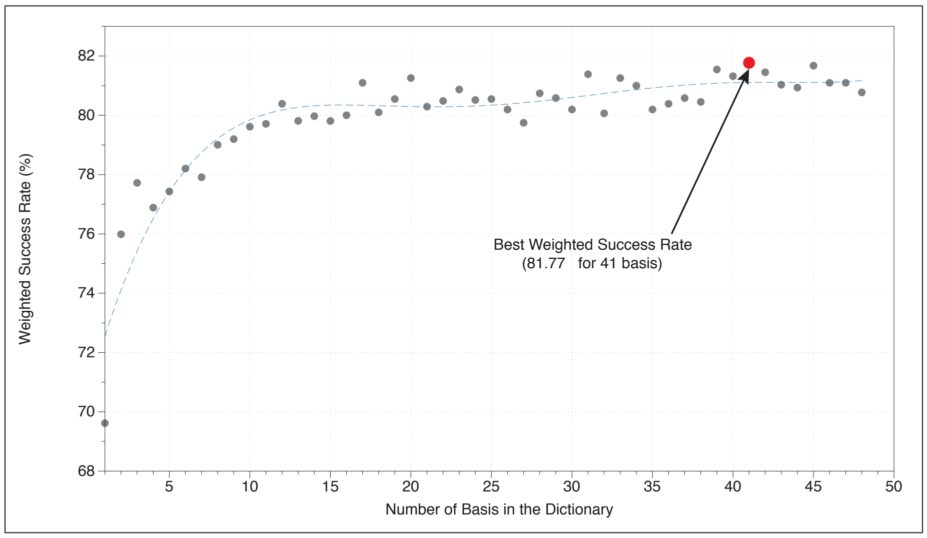


Figure 5.5   Top classification results per number of basis in the dictionary - second system

In figure 5.6 we can see the high level features created by the sparse coding algorithm. If we compare these basis with the ones found in figure 3.5, we notice that there are fewer recognizable features. We can still see some typical features like the tip of the gripper finger (basis #40) but most of the others seem to be randomly generated spots. The way we analyse this new dictionary is that the basis have now evolved to an even higher form of feature, allowing the classifier to cut up the tactile pressure images into smaller portions, rendering better classification results.

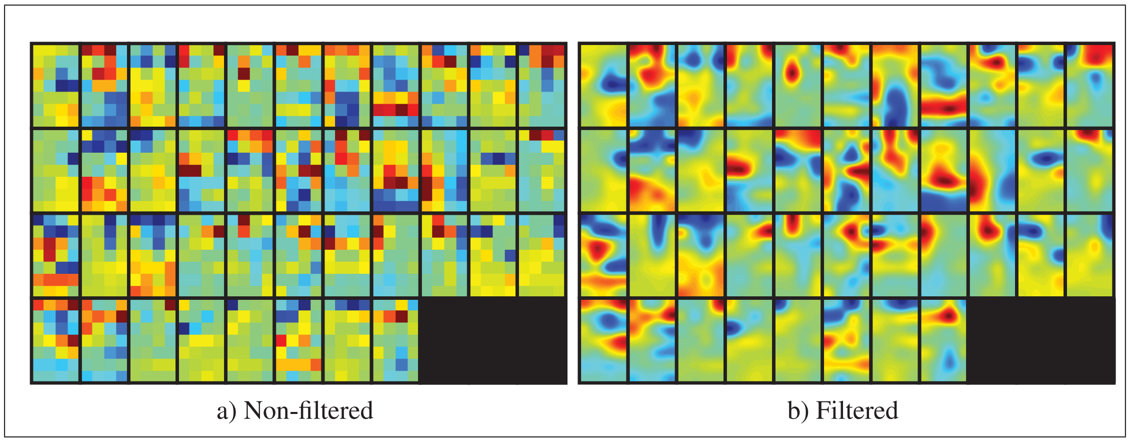


Figure 5.6 Optimal dictionary of basis with the new data

Using this new dictionary of basis, we trained a new classifier. This new classifier gave us a weighted succes rate of 81.77%. In order to better compare it with our previous systems, we computed the results in a confusion matrix (see Figure 5.7)

5.3 Integration techniques of the new data in our system

In the following sections, we will pursue our second hypothesis to augment the success rate of our classifier. We will propose different approaches to fuse a new type of data input to our system. We will attempt to develop a new classifier that will use the gripper finger's positions along side to the tactile images or the grasping attemps.

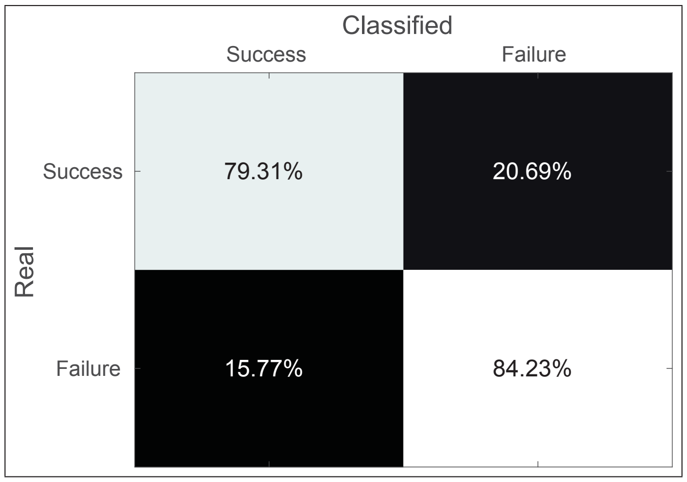


Figure 5.7 Confusion Matrix - Full optimization with the new data

Before exposing the different fusion techniques, we will explain how we defined the relative position of the gripper fingers using the integrated IMUs in our tactile sensors.

5.3.1 Defining the IMU data

Earlier in this thesis, we made the hypothesis that we needed to add different data as an input to our system to get better results. Other than the tactile pressure images, we also have the integrated IMU data and the gripper position data we wish to use.

When it comes to the gripper position data, it is expressed as a percentage from the Robotiq two finger gripper. We opted to use this value directly as an input to our systems. When it comes to the IMU, retrieving the data was not as direct. Recall that we want to use the integrated IMUs to represent the movement of the underactuation of the Robotiq two finger gripper.

Considering that the Robotiq two finger gripper is a parallel gripper, we know that the underactuation creates a movement of the gripper pads on the X-Y plane of the robot's tool center point. We therefore decided to simplify the IMU raw data into two angles, one for each gripper pad, on the X-Y plane. We defined the reference position of the angles to be the not underactuated position of the respective gripper pad. Finally, we chose to keep the angles positive, making

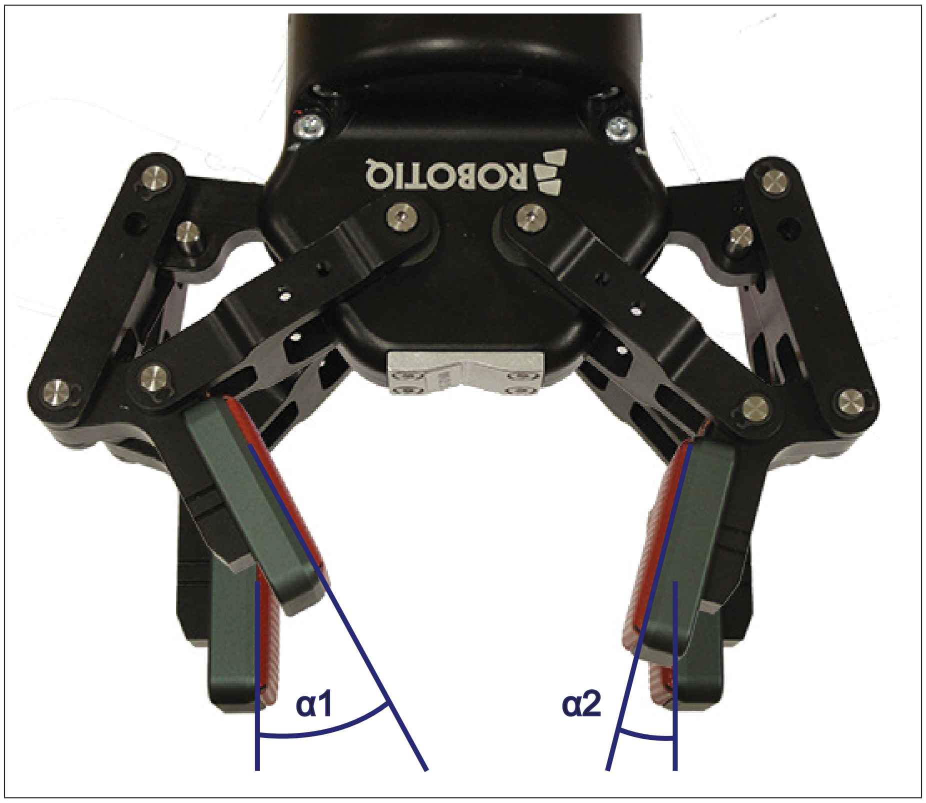


Figure 5.8 Defining the angles obtained from the IMUs

each gripper pad's angle defined as a different rotation direction. Figure 5.8 demonstrates this definition of the angle. In this figure, both $\alpha1$ and $\alpha2$ are positive.

To summarize, in this second dataset, we have collected three new entries to our data: the gripper opening position (a percentage), and two gripper pad angles (positive angle relative to the pad's resting position). From this point on in this thesis, we will refer to this new data as the IMU data.

5.3.2 Testing our systems on the same base

In the next sections, we will be proposing different techniques to build better robotic grasping prediction classifiers. In order to compare the results of the different systems, we chose to define the building procedure in training and testing.

Having a larger dataset made it possible to divide the data into training and testing sets. We randomly separated our 1000 grasping attemps into an 80-20 distribution. This separation was done only once therefore, all the different systems we will present have been trained with the same dataset. Similarly, all the system will be tested using the same test dataset.

Moreover, the results will also be presented in a normalized manner. Our satisfaction criteria is still based on the weighted success rate (see equation 3.4) but, to better understand the properties of our systems, we will present the results in confusion matrices. This way, we can analyse the systems not only by an overall criteria but also, by its performance in classifying *failed* grasping attemps.

5.3.3 Using tactile and IMU data to build classifiers

This section will expose the different classifiers we built using both tactile pressure images and IMU data from robotic grasping attemps. Prior the combining the data, we first trained and tested two separate SVMs ; one using the tactile data and the other, using the IMU data. We collected the resulting levels of confidence of the two SVMs.

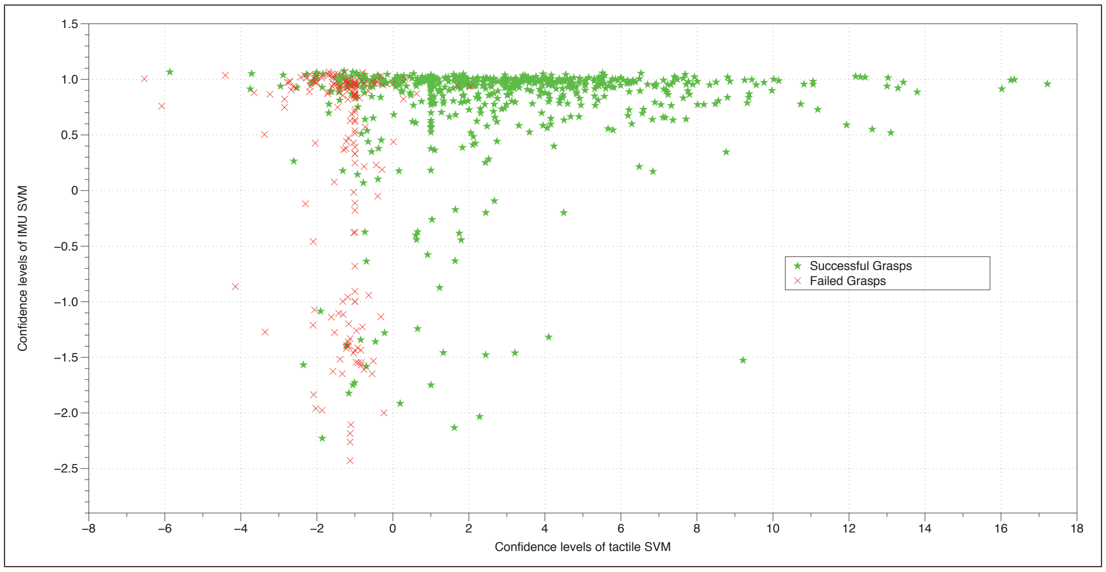


Figure 5.9 Levels of confidence of the tactile and IMU SVMs

The level of confidence of an SVM can been imagined as the distance of the datapoint from the separating hyperplane of the classifier. In our case, a positive value expresses a successful grasping attempt. With a properly trained SVM, the higher the absolute value of the level of confidence, the higher probability that the classification is correct. In order to compare and also determine corrolations between our two classifiers, we plotted the levels of confidence of each previously mentioned classifier into one figure. Figure 5.9 shows us the position of each datapoint. The X axis corresponds to the tactile SVM level of confidence where the Y axis corresponds to the IMU SVM level of confidence. The color and shape of the point indicates the real label of the datapoint. Green stars are *successful* grasping attemps and red crosses the *failed* ones.

5.3.3.1 Blending the data using a handmade classifier

When we analye figure 5.9 the first thing we notice is that the IMU SVM seems performant on classifying the *successful* grasping attemps and the tactile SVM seems more performant on classifying *failed* grasping attemps. Since there is no clear separation of the grasping attemps, we first decided to try and build a simple classifier based on the analysis of figure 5.9.

Recall that our primary objective is to detect robotic grasping failures. Keeping this in mind, we built the simple classifier illustrated in figure 5.10. Our simple classifier was built following an elimination process. First, if both the tactile and IMU SVMs agree, the output is set to this value. Then we prioritize the tactile SVM for its classification of ***failed*** grasping attemps. Finally, the remaining attemps are separated by choosing the strongest of the two levels of confidence.

Using the classifier depicted in figure 5.10, we obtained a weighted success rate of 82.15%. At first glance, we beat our best results to date which were of a weighted success rate of 81.77% from the full optimization process with the new data. This result is a first step to confirming our hypothesis that using IMU data would be beneficial in the classification of robotic grasping attemps. On the other hand, when we computed the results in a confusion

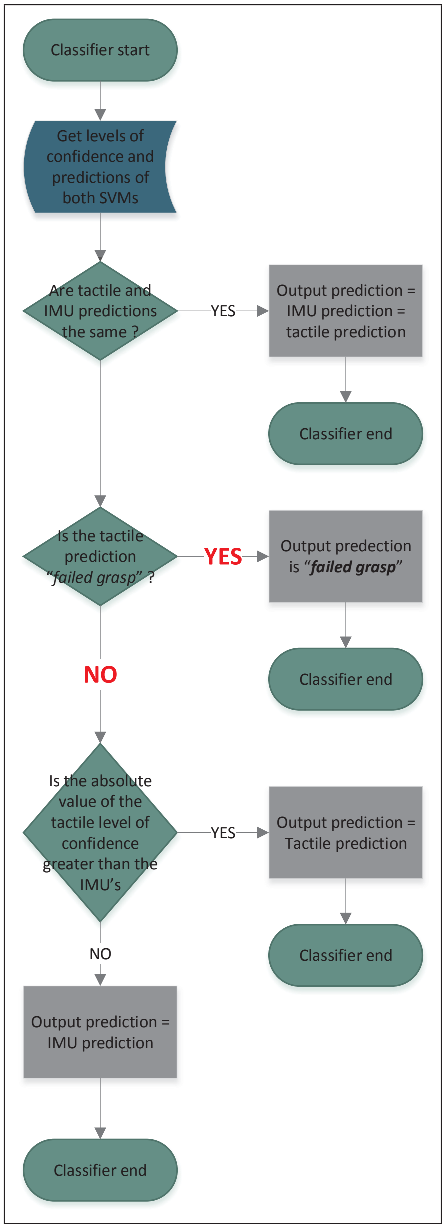

Classifier start
Get levels of confidence and predictions of both SVMs
Are tactile and IMU predictions the same ?
YES
Output prediction = IMU prediction = tactile prediction
Classifier end
Is the tactile prediction "failed grasp" ?
YES
Output predection is "failed grasp"
Classifier end
NO
Is the absolute value of the tactile level of confidence greater than the IMU's
YES
Output prediction = Tactile prediction
Classifier end
NO
Output prediction = IMU prediction
Classifier end

Figure 5.10 Simple handmade classifier - Flow chart

matrix (see figure 5.11), we notice that our increse in performance is due to better results in classifying *successful* grasps. Nonetheless, we decided to build another classifier using learning algorithms in hope of obtaining yet better results.

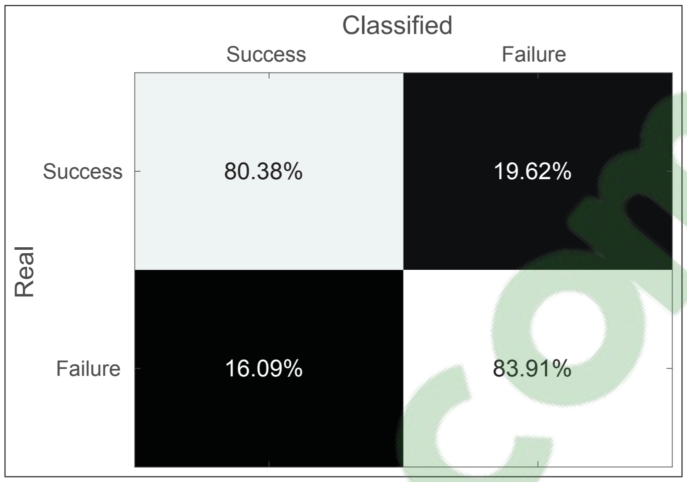


Figure 5.11 Confusion Matrix - Simple handmade classifier

5.3.3.2 Constructing a multilayer SVM system

In this section, we propose to use a third SVM to do the final classification. Instead of using a handmade classifier, we used the levels of confidence of the tactile and IMU classifiers to train a final SVM. In figure 5.12 we can see our architechture, which was inspired by the work of Waske and Benediktsson (2007). First, we independently train two SVMs, one using only the tactile training data and the other, only the IMU training data. We then collect the levels of confidence of both classifiers and use these values as an input for the training of a final SVM.

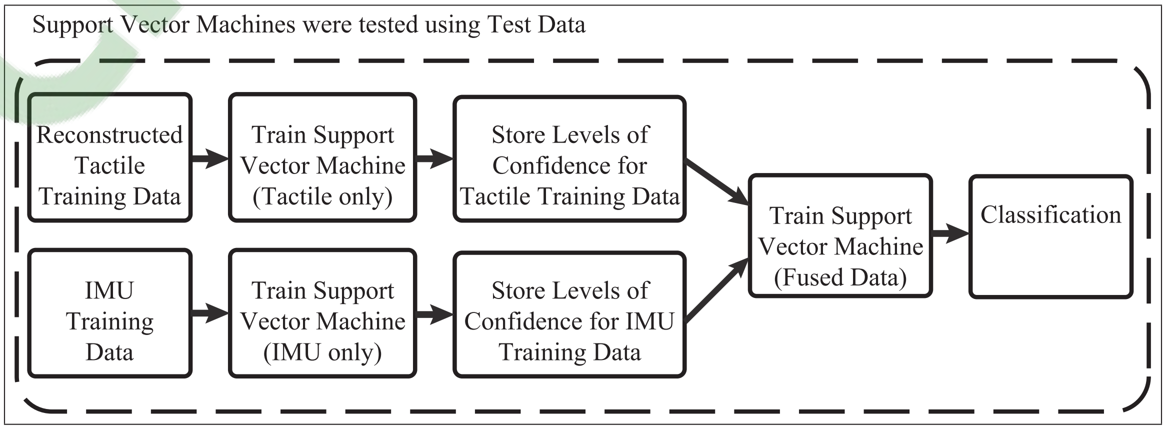

Support Vector Machines were tested using Test Data
Reconstructed Tactile Training Data
Train Support Vector Machine (Tactile only)
Store Levels of Confidence for Tactile Training Data
IMU Training Data
Train Support Vector Machine (IMU only)
Store Levels of Confidence for IMU Training Data
Train Support Vector Machine (Fused Data)
Classification

Figure 5.12 Double layer SVM architecture

After training all three SVMs using the training data, we ran all the test data in our system. We must admit being disappointed with the results. We obtained a weighted success rate of 81.82% wich is lower than the result obtained with our handmade classifier. Nonetheless, we computed the results into a confusion matrix (see figure 5.13).

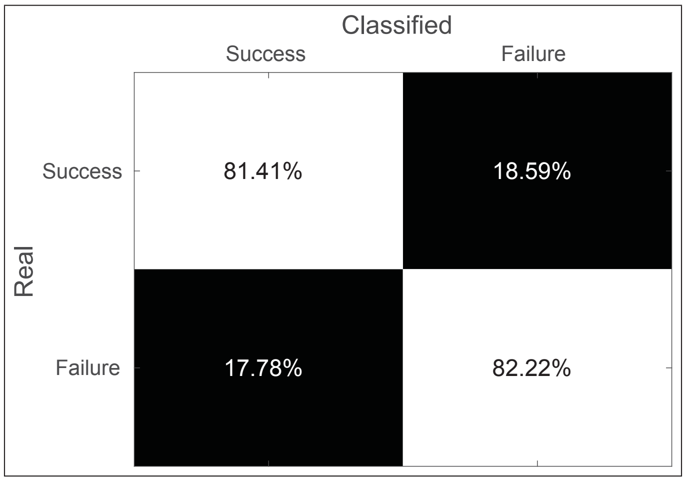


Figure 5.13 Confusion Matrix - Double Layer SVM

These results are even more disappointing since we actually lost performance on classifying *failed* grasping attemps. For further analysis, we decided to submit our final SVM to a 10 fold cross validation test. For this test, we merged the levels of confidence training and test data into a single structure. Using this data, we ran a 10 fold cross validation of the final SVM.

Figure 5.14 illustrates the results we obtained from the 10 fold cross validation on our final SVM. These values correspond to a weighted success rate of 87.98%, with an impressive 94.59% prediction of *failed* grasping attemps. These results can lead us to believe that it is possible to build a performant system using our architechture but, it will not transfer easily to new grasping attempts.

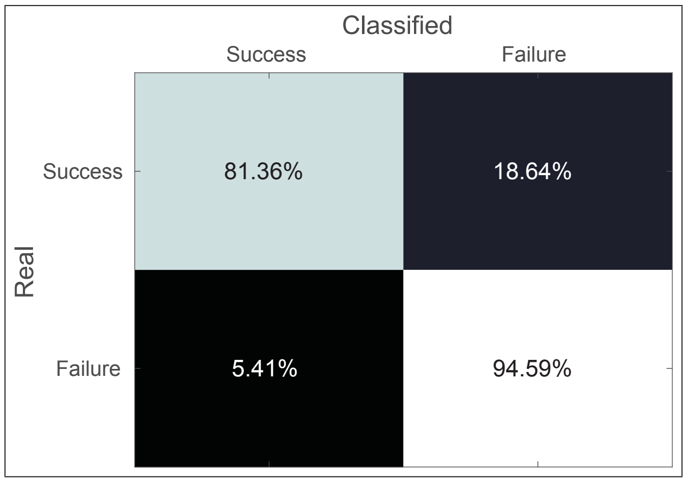


Figure 5.14 Confusion Matrix - Double layer SVM - 10 Fold cross validation

5.4 Future work on data fusion

In this chapter, we have proposed a multi layer SVM system to classify robotic grasping attemps using both tactile pressure images and IMU data. Our results show that using both types of data seems promising but, our architechture does not render the desired results. Let us remember that we are using very simple linear SVMs with traditional kernels. The linear SVM might give good results when using sparce data as an input but, we do not believe it is the proper classifier when it comes to using multiple types of data as an input.

On the other hand, our analysis of sparse coding to extract high level features and the addition of gripper position data did render good results. In future work, we believe that different SVMs should be explored while keeping similar architectures as we proposed. In figure 5.9 we can see that the data is not seperable linearly but, perhaps if we were using non linear SVMs or again clustering classifiers in the different levels, we could obtain better classification results. Furthermore, we believe that other methods should be explored. While we were running the final optimization processes, we started exploring convolutionnal neural networks (CNNs). With the release of open source libraries, such as tensorflow, building a CNN has become much simpler. Contrary to SVMs, CNNs can be built into very specific architechtures wich could possibly solve our multiple data type entries.

Finally, we believe that future researchers should experiements with some of the variables that we kept fixed. As an example, we collected all our data with a fixed top pick approach. This was done to concentrate our efforts on studying our new approach but, this does not represent all robotic grasping possibilities. We would recommend starting by integrating side pick data to see if our techniques could transfer to this new type of data.

CONCLUSION AND RECOMMENDATIONS

The premise of our work was to propose new tools to evaluate the quality of a grasp in order to predict the outcome of said grasp. Using tactile sensors, we hoped to give a good prediction of the outcome of a grasping attempt at the moment of contact with the object. We have shown that it is possible to build a rich dictionary of high level features to represent tactile pressure images using sparse coding. In want of keeping the systems as simple as possible and also to evaluate the performance of using self-taught, unsupervised high level feature extraction, we limited ourselves to simple linear SVM classifiers. As we can see in the following table, the results are very promising for future work.

Table 5.1 Summary of results - Weighted success rate

Optimal system version 1	**New data Optimal V1**	**Optimal system version 2**	**Handmade classifier**	**Multilayer SVM**
78.89%	69.16%	81.77%	82.15%	81.82%

On the other hand, we do notice that we seem to have reached the limits of our approach. We believe that the limitation does not come from the type or manner in wich we represent the data but more from the simplistic SVMs we chose to use. As we have said earlier, grasp assessment is a complex and multi sensory problem and therefore, it is improbable to find a representation of the data that can be separated linearly when we start using mutiple types of data as an entry. We recommend continuing to use unsupervised learning to extract high level features of tactile pressure images and merging this data with information from other sensors but testing with different types of classifiers.

BIBLIOGRAPHY

Andersen, M.R., T. Jensen, P. Lisouski, A.K. Mortensen, M.K. Hansen, Torben Gregersen and Peter Ahrendt. 2015. "Kinect Depth Sensor Evaluation for Computer Vision Applications". *Technical Report Electronics and Computer Engineering*, vol. 1, n° 6.

Bebionic, Ottobock. 2017. "bebionic.com | Home". <http://bebionic.com/>. [Online; accessed 18-November-2017].

Bekiroglu, Y., R. Detry and D. Kragic. Sept 2011a. "Learning tactile characterizations of object- and pose-specific grasps". In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*. p. 1554-1560.

Bekiroglu, Y., J. Laaksonen, J. A. Jorgensen, V. Kyrki and D. Kragic. June 2011b. "Assessing Grasp Stability Based on Learning and Haptic Data". *IEEE Transactions on Robotics*, vol. 27, n° 3, p. 616-629.

Boston Dynamics, Boston Dynamics. 2017. "bostondynamics.com | Home". <https://www.bostondynamics.com/>. [Online; accessed 18-November-2017].

Calli, Berk, Arjun Singh, Aaron Walsman, Siddhartha Srinivasa, Pieter Abbeel and Aaron M Dollar. 2015. "The YCB object and model set: Towards common benchmarks for manipulation research". In *Advanced Robotics (ICAR), 2015 International Conference on*. p. 510–517. IEEE.

Cheng, F. S. and A. Denman. 2005. "A study of using 2D vision system for enhanced industrial robot intelligence". In *IEEE International Conference Mechatronics and Automation, 2005*. p. 1185-1189 Vol. 3.

Cutkosky, Mark R. and Robert D. Howe. 1990. "Dextrous Robot Hands". chapter Human Grasp Choice and Robotic Grasp Analysis, p. 5–31. New York, NY, USA: Springer-Verlag New York, Inc. ISBN 0-387-97190-4. <http://dl.acm.org/citation.cfm?id=88109.88110>.

Dang, H. and P. K. Allen. Nov 2013. "Grasp adjustment on novel objects using tactile experience from similar local geometry". In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. p. 4007-4012.

Dang, H., J. Weisz and P. K. Allen. May 2011a. "Blind grasping: Stable robotic grasping using tactile feedback and hand kinematics". In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. p. 5917-5922.

Dang, Hao and Peter K. Allen. 2014. "Stable grasping under pose uncertainty using tactile feedback". *Auton. Robots*, vol. 36, n° 4, p. 309–330.

Dang, Hao, Jonathan Weisz and Peter K Allen. 2011b. "Blind grasping: Stable robotic grasping using tactile feedback and hand kinematics". In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. p. 5917–5922. IEEE.

Dargahi, J and S Najarian. 2004. "Human tactile perception as a standard for artificial tactile sensing—a review". *The International Journal of Medical Robotics and Computer Assisted Surgery*, vol. 1, n° 1, p. 23–35.

De Boissieu, Florian, Christelle Godin, Bernard Guilhamat, Dominique David, Christine Serviere and Daniel Baudois. 2009. "Tactile texture recognition with a 3-axial force MEMS integrated artificial finger.". In *Robotics: Science and Systems*. p. 49–56. Seattle, WA.

Duchaine, V. and A. Rana. 24 2014. "Dielectric geometry for capacitive-based tactile sensor". <http://www.google.se/patents/WO2014110683A1?cl=sv>. WO Patent App. PCT/CA2014/050,040.

Engel, Jonathan, Jack Chen and Chang Liu. 2003. "Development of polyimide flexible tactile sensor skin". *Journal of Micromechanics and Microengineering*, vol. 13, n° 3, p. 359.

Fan, X., X. Wang and Y. Xiao. Aug 2014. "A combined 2D-3D vision system for automatic robot picking". In *Proceedings of the 2014 International Conference on Advanced Mechatronic Systems*. p. 513-516.

Feix, T., I. M. Bullock and A. M. Dollar. July 2014a. "Analysis of Human Grasping Behavior: Object Characteristics and Grasp Type". *IEEE Transactions on Haptics*, vol. 7, n° 3, p. 311-323.

Feix, T., I. M. Bullock and A. M. Dollar. Oct 2014b. "Analysis of Human Grasping Behavior: Correlating Tasks, Objects and Grasps". *IEEE Transactions on Haptics*, vol. 7, n° 4, p. 430-441.

Fu, Q., A. Ushani, L. Jentoft, R. D. Howe and M. Santella. July 2013. "Human reach-to-grasp compensation with object pose uncertainty". In *2013 35th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*. p. 6893-6896.

Fujiwara, E., F. D. Paula, Y. T. Wu, M. F. M. Santos, E. A. Schenkel and C. K. Suzuki. April 2017. "Optical fiber tactile sensor based on fiber specklegram analysis". In *2017 25th Optical Fiber Sensors Conference (OFS)*. p. 1-4.

Gehring, Clement and Simon Lemay. 2012. "Sparse Coding Applied to Digit Recognition". *sibi*, vol. 1, p. 1.

Goldfeder, C., M. Ciocarlie, Hao Dang and P. K. Allen. May 2009. "The Columbia grasp database". In *2009 IEEE International Conference on Robotics and Automation*. p. 1710-1716.

Hebert, P., N. Hudson, J. Ma and J. Burdick. May 2011. "Fusion of stereo vision, force-torque, and joint sensors for estimation of in-hand object location". In *2011 IEEE International Conference on Robotics and Automation*. p. 5935-5941.

Heyneman, B. and Mark R. Cutkosky. Dec 2012. "Biologically inspired tactile classification of object-hand and object-world interactions". In *Robotics and Biomimetics (ROBIO), 2012 IEEE International Conference on*. p. 167-173.

Hinojosa, Trisha, Ching-Fan Sheu and George F Michel. 2003. "Infant hand-use preferences for grasping objects contributes to the development of a hand-use preference for manipulating objects". *Developmental Psychobiology*, vol. 43, n° 4, p. 328–334.

Honda, Asimo. 2017. "asimo.honda.com | Home". <http://asimo.honda.com/>. [Online; accessed 18-November-2017].

Huebner, K., S. Ruthotto and D. Kragic. May 2008. "Minimum volume bounding box decomposition for shape approximation in robot grasping". In *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*. p. 1628-1633.

Hyttinen, E., D. Kragic and R. Detry. May 2015. "Learning the tactile signatures of prototypical object parts for robust part-based grasping of novel objects". In *2015 IEEE International Conference on Robotics and Automation (ICRA)*. p. 4927-4932.

Johansson, Roland S and J Randall Flanagan. 2009a. "Coding and use of tactile signals from the fingertips in object manipulation tasks". *Nature Reviews Neuroscience*, vol. 10, n° 5, p. 345–359.

Johansson, Roland S. and J. Randall Flanagan. Apr 2009b. "Coding and use of tactile signals from the fingertips in object manipulation tasks". *Nature Reviews Neuroscience*, vol. 10, p. 345 EP -.

Kehoe, Ben, Akihiro Matsukawa, Sal Candido, James Kuffner and Ken Goldberg. 2013. "Cloud-based robot grasping with the google object recognition engine". In *IEEE Int'l Conf. on Robotics and Automation*. p. 8.

Kragten, Gert A and Just L Herder. 2010. "The ability of underactuated hands to grasp and hold objects". *Mechanism and Machine Theory*, vol. 45, n° 3, p. 408–425.

Lederman, SJ and RL Klatzky. 2009. "Haptic perception: A tutorial". *Attention, Perception, & Psychophysics*, vol. 71, n° 7, p. 1439–1459.

Lee, Honglak. 2010. "Unsupervised Feature Learning via Sparse Hierarchical Representations". PhD thesis, Stanford University.

Lee, Honglak, Alexis Battle, Rajat Raina and Andrew Y. Ng. 2007. "Efficient sparse coding algorithms". In *Advances in Neural Information Processing Systems 19*, Schölkopf, B., J.C. Platt and T. Hoffman (Eds.), p. 801–808. MIT Press. <http://papers.nips.cc/paper/2979-efficient-sparse-coding-algorithms.pdf>.

Lepora, N. F. and B. Ward-Cherrier. Sept 2015. "Superresolution with an optical tactile sensor". In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. p. 2686-2691.

Levine, Sergey, Peter Pastor, Alex Krizhevsky and Deirdre Quillen. 2016. "Learning Hand-Eye Coordination for Robotic Grasping with Deep Learning and Large-Scale Data Collection". *CoRR*, vol. abs/1603.02199.

Li, L., W. Wang, Y. Su and Z. Du. Aug 2015. "A data-driven grasp planning method based on Gaussian Process Classifier". In *2015 IEEE International Conference on Mechatronics and Automation (ICMA)*. p. 2626-2631.

Lin, Y. and Y. Sun. Sept 2014. "Grasp planning based on strategy extracted from demonstration". In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. p. 4458-4463.

Liu, W., P. Yu, C. Gu, X. Cheng and X. Fu. 2017. "Fingertip Piezoelectric Tactile Sensor Array for Roughness Encoding under Varying Scanning Velocity". *IEEE Sensors Journal*, vol. PP, n° 99, p. 1-1.

Loske, J. and R. Biesenbach. Sept 2014. "Force-torque sensor integration in industrial robot control". In *15th International Workshop on Research and Education in Mechatronics (REM)*. p. 1-5.

Mairal, J., M. Elad and G. Sapiro. Jan 2008. "Sparse Representation for Color Image Restoration". *IEEE Transactions on Image Processing*, vol. 17, n° 1, p. 53-69.

Mairal, J., F. Bach, J. Ponce, G. Sapiro and A. Zisserman. Sept 2009. "Non-local sparse models for image restoration". In *2009 IEEE 12th International Conference on Computer Vision*. p. 2272-2279.

Mairal, Julien, Francis R. Bach and Jean Ponce. 2014. "Sparse Modeling for Image and Vision Processing". *CoRR*, vol. abs/1411.3230.

Mannsfeld, Stefan CB, Benjamin CK Tee, Randall M Stoltenberg, Christopher V HH Chen, Soumendra Barman, Beinn VO Muir, Anatoliy N Sokolov, Colin Reese and Zhenan Bao. 2010. "Highly sensitive flexible pressure sensors with microstructured rubber dielectric layers". *Nature materials*, vol. 9, n° 10, p. 859–864.

Miller, A. T. and P. K. Allen. Dec 2004. "Graspit! A versatile simulator for robotic grasping". *IEEE Robotics Automation Magazine*, vol. 11, n° 4, p. 110-122.

Miller, Andrew T., Steffen Knoop, Henrik I. Christensen and Peter K. Allen. 2003. "Automatic grasp planning using shape primitives". In *ICRA*.

Mills, J. K. May 1989. "Dynamic modelling for robotic manipulators with a force-torque sensor during compliant motion". In *Proceedings, 1989 International Conference on Robotics and Automation*. p. 1672-1677 vol.3.

Moreira, E., L. F. Rocha, A. M. Pinto, A. P. Moreira and G. Veiga. July 2016. "Assessment of Robotic Picking Operations Using a 6 Axis Force/Torque Sensor". *IEEE Robotics and Automation Letters*, vol. 1, n° 2, p. 768-775.

Oberlin, John and Stefanie Tellex. 2015. "Autonomously acquiring instance-based object models from experience". *Int. S. Robotics Research (ISRR)*.

Padgett, S. T. and A. F. Browne. March 2017. "Vector-based robot obstacle avoidance using LIDAR and mecanum drive". In *SoutheastCon 2017*. p. 1-5.

Raina, Rajat, Alexis Battle, Honglak Lee, Benjamin Packer and Andrew Y. Ng. 2007. "Self-taught Learning: Transfer Learning from Unlabeled Data". In *ICML '07: Proceedings of the 24th international conference on Machine learning*.

Rana, Axay. and Vincent Duchaine. May 2013. "Improved Soft Dielectric for Highly Sensitive Capacitive Tactile Sensor". In *In 2013 IEEE International Conference on Robotics and Automation : Workshop on Research Frontiers in Electronic Skin Technology (Karlsruhe, Germany*. IEEE.

Rana, AxayKumar, Jean-Philippe Roberge and Vincent Duchaine. 2016. "An Improved Soft Dielectric for a Highly Sensitive Capacitive Tactile Sensor". *IEEE Sensors Journal*, vol. 20.

Roberge, Jean-Philippe, Samuel Rispal, Tony Wong and Vincent Duchaine. 2016. "Unsupervised feature learning for classifying dynamic tactile events using sparse coding". In *2016 IEEE International Conference on Robotics and Automation (ICRA)*. p. 2675–2681. IEEE.

Rodriguez, Fernando and Guillermo Sapiro. 2008. *Sparse representations for image classification: Learning discriminative and reconstructive non-parametric dictionaries*. Technical report.

Romano, J.M., K. Hsiao, G. Niemeyer, S. Chitta and K.J. Kuchenbecker. Dec 2011. "Human-Inspired Robotic Grasp Control With Tactile Sensing". *Robotics, IEEE Transactions on*, vol. 27, n° 6, p. 1067-1079.

Schmitz, Alexander, Marco Maggiali, Lorenzo Natale, Bruno Bonino and Giorgio Metta. 2010. "A tactile sensor for the fingertips of the humanoid robot icub". In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*. p. 2212–2217. IEEE.

Smith, Evan and Michael S. Lewicki. 2005. "Efficient Coding of Time-Relative Structure Using Spikes". *Neural Comput.*, vol. 17, n° 1, p. 19–45.

Srinivasa, S. S., D. Berenson, M. Cakmak, A. Collet, M. R. Dogar, A. D. Dragan, R. A. Knepper, T. Niemueller, K. Strabala, M. Vande Weghe and J. Ziegler. Aug 2012. "Herb 2.0: Lessons Learned From Developing a Mobile Manipulator for the Home". *Proceedings of the IEEE*, vol. 100, n° 8, p. 2410-2428.

Tremblay, Marc R and Mark R Cutkosky. 1993. "Estimating friction using incipient slip sensing during a manipulation task". In *Robotics and Automation, 1993. Proceedings., 1993 IEEE International Conference on*. p. 429–434. IEEE.

Vallbo, Å B, RS Johansson et al. 1984. "Properties of cutaneous mechanoreceptors in the human hand related to touch sensation". *Hum Neurobiol*, vol. 3, n° 1, p. 3–14.

Viet, N. B., N. T. Hai and N. V. Hung. Oct 2013. "Tracking landmarks for control of an electric wheelchair using a stereoscopic camera system". In *2013 International Conference on Advanced Technologies for Communications (ATC 2013)*. p. 339-344.

Waske, B. and J. A. Benediktsson. Dec 2007. "Fusion of Support Vector Machines for Classification of Multisensor Data". *IEEE Transactions on Geoscience and Remote Sensing*, vol. 45, n° 12, p. 3858-3866.

Weiss, K. and H. Worn. July 2005. "The working principle of resistive tactile sensor cells". In *IEEE International Conference Mechatronics and Automation, 2005*. p. 471-476 Vol. 1.

Wettels, N, JA Fishel, Z Su, CH Lin and GE Loeb. 2009. "Multi-modal synergistic tactile sensing". In *Tactile sensing in humanoids—Tactile sensors and beyond workshop, 9th IEEE-RAS international conference on humanoid robots*.

Xue, Z., S. Xia and R. Dillmann. Nov 2012. "An efficient grasp planning algorithm based on decomposition of grasp regions". In *2012 12th IEEE-RAS International Conference on Humanoid Robots (Humanoids 2012)*. p. 686-691.

Xue, Zhixing, J. M. Zoellner and R. Dillmann. July 2008. "Automatic optimal grasp planning based on found contact points". In *2008 IEEE/ASME International Conference on Advanced Intelligent Mechatronics*. p. 1053-1058.

Zecca, M., G. Cappiello, F. Sebastiani, S. Roccella, F. Vecchi, M.C. Carrozza and P. Dario. 2004. "15 Experimental Analysis of the Proprioceptive and Exteroceptive Sensors of an Underactuated Prosthetic Hand". In *Advances in Rehabilitation Robotics*, Bien, Z.Zenn and Dimitar Stefanov (Eds.), volume 306 of *Lecture Notes in Control and Information Science*, p. 233-242. Springer Berlin Heidelberg. ISBN 978-3-540-21986-6. doi: 10.1007/10946978_15.

Zhang, Kaibing and Jun Lu. Dec 2010. "Handwritten character recognition via sparse representation and multiple classifiers combination". In *Information Theory and Information Security (ICITIS), 2010 IEEE International Conference on*. p. 1139-1142.

Zheng, Wenbin and Yuntao Qian. Nov 2012. "Non-negative Sparse Semantic Coding for text categorization". In *Pattern Recognition (ICPR), 2012 21st International Conference on*. p. 409-412.