3D MODEL DRIVEN DISTANT ASSEMBLY
Final report

Bachelor Degree Project in Automation
Spring term 2012

Carlos Gil Camacho
Juan Cana Quijada

Supervisor: Abdullah Mohammed
Examiner: Lihui Wang
Executive Summary

Today, almost all factories use robots to a certain extent. In particular, automated assembly operations are common in modern manufacturing firms. However, current industrial applications require a pre-knowledge of assembly parts, their locations as well as the paths that they should follow during the assembly process.

This research introduces a remote assembly control system, which gives the operator access to the robot at any time and any location as long as there is an Internet access. In addition, the remote assembly control uses a user-friendly interface which is easy and quick to use. An operator does not need great knowledge before using the web portal for the remote assembly control. On the contrary, using robot programming software would require the operator spending more time to learn specific skills. The remote assembly is also useful in risky situations in which the operator must keep a distance from the work place of the robot.

The main objective of this project is to design a web-based system that can provide the ability for remote operators to assemble objects in a robotic cell using 3D models of those objects in a virtual environment. The system consists of an industrial robot and a network camera connected to an application server. Using the user interface and the camera, a remote operator can take some snapshots of the real objects in the robotic cell and the system will construct the 3D models of these objects. Consequently, the operator can assemble the real objects using the equivalent 3D models via an existing Wise-ShopFloor virtual environment.

The scope of the project includes:

- Capturing the real objects with a camera mounted beside the gripper of the robot. The position of the camera gives the system more flexibility for taking the snapshots from different angles.

- Analysing the snapshots using different filters in order to identify the silhouettes of the objects. These silhouettes will be labelled to help distinguishing the different objects.

- Constructing 3D models of the objects. This will be done by constructing pillars from the labelled silhouettes of the top view of the objects. These pillars will be trimmed with the silhouettes obtained from snapshots in different angles, until the pillars represent the objects with the needed accuracy.
• Integrating the models into the Wise-ShopFloor virtual environment. This environment can be executed in a web browser.

• Updating the objects in the virtual environment according to the assembly operations.

Since assembly only needs picking and placing objects, the image processing and the 3D modelling do not need as high quality as it would need for other tasks, such as welding. In addition, the operator may be anywhere as long as an Internet access is provided. The Internet connection does not need to be in broadband because the system consumes a limited bandwidth for receiving snapshots from the camera and sending assembly instructions to the robot.

The results of this project will allow the operator to work away from dangerous environments, therefore helping to improve safety. This research can also help improving the adaptability of industry in ad-hoc assembly operations. Besides, the use of the web for the remote assembly can save cost and time in operator training and special-purpose software purchasing. Remote assembly is expected to become an important research area with unlimited application potentials in the future in distributed manufacturing.
Acknowledgements

We want to send our gratitude to the University of Skövde for welcoming and teaching us and especially to the Virtual Systems Research Centre for giving us the opportunity of working on this project.

We want to give our best thanks to Abdullah Mohammed for his advice, patience and support during the development of this project by providing useful resources and always being available.

We also want to thank the support received from the rest of the Wise-ShopFloor research group: Bernard Schmidt and Mohammad Givehchi.

Finally we want to thank our families for helping us to be in Skövde and to participate in this project.
Table of Contents

Executive Summary .................................................................................................................. 1
Acknowledgements .................................................................................................................. 3
Table of Figures ....................................................................................................................... 7
1 Introduction ............................................................................................................................ 9
  1.1 Motivation ......................................................................................................................... 9
  1.2 Objectives ......................................................................................................................... 9
  1.3 System Overview .............................................................................................................. 11
    1.3.1 System Architecture ................................................................................................. 11
    1.3.2 Java Programming Language .................................................................................. 12
2 Literature Review .................................................................................................................. 14
  2.1 Previous Works ............................................................................................................... 14
  2.2 Research Objective ......................................................................................................... 15
  2.3 Novelties ......................................................................................................................... 15
3 Methodology ........................................................................................................................ 17
  3.1 Stages of the project ........................................................................................................ 17
  3.2 Image Acquisition .......................................................................................................... 17
  3.3 Image Processing Stages .................................................................................................. 17
    3.3.1 Conversion to Gray Scale ....................................................................................... 18
    3.3.2 Adjustment of Brightness and Contrast ................................................................. 18
    3.3.3 Gaussian Filtering ................................................................................................. 18
    3.3.4 Image Thresholding ............................................................................................... 18
    3.3.5 Silhouette Labelling .............................................................................................. 19
  3.4 3D Modelling .................................................................................................................. 19
    3.4.1 Camera Calibration ................................................................................................. 19
    3.4.2 Camera Coordinates Frame .................................................................................... 19
    3.4.3 Image Plane Coordinates Frame ............................................................................ 20
    3.4.4 Building the 3D Models ......................................................................................... 21
Appendix ................................................................. 58

7.1 Definitions and Acronyms ........................................ 58

7.2 UML Diagrams ..................................................... 60

7.3 Initial project planning ............................................. 62

8 References .................................................................... 63
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>project main concept.</td>
<td>10</td>
</tr>
<tr>
<td>2</td>
<td>System architecture.</td>
<td>11</td>
</tr>
<tr>
<td>3</td>
<td>Stages during image processing.</td>
<td>18</td>
</tr>
<tr>
<td>4</td>
<td>Position of the camera coordinates frame.</td>
<td>20</td>
</tr>
<tr>
<td>5</td>
<td>image plane frame and camera frame.</td>
<td>21</td>
</tr>
<tr>
<td>6</td>
<td>example of how a cylinder can be represented by pillars and each pillar by two points.</td>
<td>22</td>
</tr>
<tr>
<td>7</td>
<td>example of how the pillars are initially built.</td>
<td>23</td>
</tr>
<tr>
<td>8</td>
<td>pillars which must be trimmed or eliminated.</td>
<td>23</td>
</tr>
<tr>
<td>9</td>
<td>process of trimming and eliminating pillars.</td>
<td>24</td>
</tr>
<tr>
<td>10</td>
<td>Different connectivity modes.</td>
<td>29</td>
</tr>
<tr>
<td>11</td>
<td>Performance of the 2-pass connected components algorithm.</td>
<td>30</td>
</tr>
<tr>
<td>12</td>
<td>Focal length of a lens.</td>
<td>31</td>
</tr>
<tr>
<td>13</td>
<td>lengths of the links of the ABB IRB140 robot. (ABB Robotics ©)</td>
<td>32</td>
</tr>
<tr>
<td>14</td>
<td>Construction of the pillars.</td>
<td>33</td>
</tr>
<tr>
<td>15</td>
<td>data structure of the 3D models of the objects.</td>
<td>34</td>
</tr>
<tr>
<td>16</td>
<td>trimming of one of the pillars.</td>
<td>36</td>
</tr>
<tr>
<td>17</td>
<td>triangulation of the upper or lower surface.</td>
<td>37</td>
</tr>
<tr>
<td>18</td>
<td>triangulation of the vertical sides.</td>
<td>38</td>
</tr>
<tr>
<td>19</td>
<td>appearance of the user interface in the initial stage.</td>
<td>39</td>
</tr>
<tr>
<td>20</td>
<td>appearance of the Reorienting tab.</td>
<td>40</td>
</tr>
<tr>
<td>21</td>
<td>the Modelling tab</td>
<td>41</td>
</tr>
<tr>
<td>22</td>
<td>snapshot positions</td>
<td>42</td>
</tr>
<tr>
<td>23</td>
<td>Scene graph modelling</td>
<td>44</td>
</tr>
<tr>
<td>24</td>
<td>3D model of the gripper</td>
<td>45</td>
</tr>
<tr>
<td>25</td>
<td>conversion of the image from colour to gray scale.</td>
<td>47</td>
</tr>
<tr>
<td>26</td>
<td>Adjusting the brightness and the contrast of the gray scale image.</td>
<td>47</td>
</tr>
<tr>
<td>27</td>
<td>effect of applying the threshold filter.</td>
<td>48</td>
</tr>
<tr>
<td>28</td>
<td>object labelling.</td>
<td>48</td>
</tr>
<tr>
<td>29</td>
<td>construction of the initial 3D models.</td>
<td>49</td>
</tr>
<tr>
<td>30</td>
<td>results of the first trimming.</td>
<td>49</td>
</tr>
<tr>
<td>31</td>
<td>results of the final trimming.</td>
<td>50</td>
</tr>
<tr>
<td>32</td>
<td>final results of the modelling process.</td>
<td>50</td>
</tr>
<tr>
<td>33</td>
<td>The average percentage of time taken by each process in both tested computers</td>
<td>52</td>
</tr>
</tbody>
</table>
Figure 34: Average time taken by each of the image processing stages ........................................ 53
Figure 35: neighbour pixels ............................................................................................................. 58
Figure 36: UML diagram of the image processing classes ................................................................. 60
Figure 37: UML diagram of the 3D modelling classes ...................................................................... 61
1 Introduction

1.1 Motivation

Automation has been growing rapidly during the recent history of industry. Automated applications have been appearing in many areas of the industry and have become very important part of it. Nowadays, almost all the modern factories use industrial robot manipulators to a certain extent. Many industrial operations which used to be manual are now automated therefore they have become faster and more accurate. However, some of these automated tasks still have many limitations when it comes to the current needs of the industry. Automation must improve the flexibility and specially the adaptability.

Current automated industrial applications require a pre-knowledge of the assembly process and the needed parts. The location of these parts and the paths that they should follow must be known in advance unless a human operator is helping or controlling the robot manipulator. Many of the industrial processes are dangerous for human beings therefore the operator must keep a certain distance from the working area. Remote control systems have appeared in order to allow human operators to work in a proper and more comfortable environment. These systems can carry out tasks that used to be impossible for human beings and also help to reduce times and costs of production in tasks in which the operators had to follow certain safety rules.

Many of the remote control systems use the Internet as a way of communication. The main problem with the Internet is that the connection speed or available bandwidth can vary significantly in short periods of time. An unreliable communication may result in undesired effects on remotely controlled operations such as delays. Some solutions have appeared to solve this problem, and one of them is working with three dimensional models in virtual environments instead of streaming images continuously from the working area to the operator.

1.2 Objectives

The main objective of this research is to design a web-based system that can provide operators the possibility of remote assembly. Therefore an operator will be able to work at any place and time as long as he has access to the Internet. This system, like many other remote controlled systems, can be useful to manipulate dangerous objects or in other risky situations. It can also be useful if the operator that has the required knowledge for a certain assembly or disassembly process is not present at the needed time. The operator can use a regular computer with a web browser and work from wherever he is. The remote control system has a user-friendly interface which is intuitive and
easy to use. An operator does not need great knowledge before using the system and he will learn very fast how to use it. This is because the system does not require any knowledge in robot programming.

The system consists of an industrial robot and a network camera connected to an application server. Instead of streaming a video of the working scenario, the camera will send some snapshots to the operator’s computer. The computer will use these snapshots for modelling in three dimensions the situation of the parts to be assembled. The operator will be able to assemble those virtual models and his computer will send the necessary instructions to the robot manipulator. This way, the manipulator will do the same assembling work (Error! Reference source not found.). Therefore the communication delays of the Internet will not affect the system. The adaptability needed in the industry for many assembly operations can be fulfilled because the system will always be controlled by a human operator.

Figure 1: project main concept.
1.3 System Overview

This section first explains the system architecture, or in other words, the hardware parts of the system. Then, it will explain some basic ideas about Java, which is the programming language used for developing the software of the system.

1.3.1 System Architecture

The system consists of an industrial robot and a network camera connected to an application server. The operator will use a standard PC which must have access to the Internet. The operator can access the system’s interface through a web browser and he or she can control the system to take some snapshots of the objects in the robotic assembly cell from different angles. These snapshots will be sent to the server which will build the 3D models of the objects. With these models the operator can remotely control the robot and assemble the objects through the web browser. The structure of this system is described in Figure 2.

[Diagram: System architecture]

The system architecture can be divided into the following sections:

1- **Client**: The client is a standard PC which the operator can use to access the interface anywhere in the world. The operator only needs to introduce an IP address in a web browser and to have a recent version of Java Flash installed in the computer in order to access the application server.

2- **Network Camera**: The camera used in the project is a SNC-CH110 Sony Corporation IP camera [1]. This camera can capture videos and take snapshots in two resolutions, 1280x960 and 640x480. The camera is fixed to the gripper of the robot because in that position it can take advantage of the six degrees of freedom that the robot manipulator can provide. Therefore the camera will achieve any position and orientation as long as the robot can reach it. The camera is connected to the server to allow using the snapshots taken as input of the system.
The HTTP API (Application Programming Interface) of the camera allows editing some options like the brightness, contrast, white balance or saturation.

3- **Application server:** The server consists of a standard PC that uses TCP/IP Ethernet connection to communicate the operator and the robot. The application server contains all the programs needed for constructing the 3D models of the objects and integrating them in the interface. The robot can only be controlled by one operator, if another operator tries to control the system, the server will show an error message. In addition, the user needs a password for obtaining the control right of the robot. This is to avoid unauthorized users who may interfere in the assembly process.

4- **Industrial Robot:** The robot used in the project is an ABB (Asea Brown Bovery) robot, in particular the IRB 140 model [2]. The robot is connected to an IRC5 controller [3] which can be accessed through a standard TCP/IP Ethernet protocol. A gripper has been attached to the robot on the tool mounting plate for grasping the objects during the pick-and-place process.

**1.3.2 Java Programming Language**

Java was developed by Sun Microsystems in 1995. It is the programming language that will be used for the development of this project. It has a very similar syntax to C and C++ because it appeared to replace them. The main difference with respect to C and C++ is that the concept of pointer does not exist in Java; therefore the dynamic memory is automatically controlled by the language instead of the programmer. The programs written in Java are normally compiled in bytecode, in addition to the possibility of compiling it in machine code. The Java Virtual Machine interprets the bytecode programs during the execution, which allows running them on any platform. This makes the programs a bit slower compared to other languages that do not use interpretation. Java gives the opportunity of developing applets, which are small applications that can be inserted in a web page. Web browsers can then download and execute them like any other Java program and with a good security for avoiding malicious programs.

Object oriented programming languages are designed for using structures called objects. These objects can be constructed using a type of structure called class which can be seen as a template that contains data fields and methods. The methods are used for operating with the data fields of the class and with the local variables of the methods. The methods and data fields can be declared as private so that they can only be accessed by methods that belong to the same class. When an object is constructed, the necessary space in memory for the data fields is reserved. In addition, object oriented programming provides the possibility of reusing the classes that are implemented in other
projects easily, or even extend them. When a class is extended, the same class is created and more methods or data fields are added, without editing the original class [4].

Java 3D will be used for modelling the assembly parts in 3D due to the fact that it provides the possibility of creating interactive three dimensional graphics within Java environment. Java 3D programs, like Java programs, have the advantage of being portable to different platforms. The graphics can be displayed on any device and the programmer does not need to worry about conversions from one device to another, as long as the Java 3D API is installed in the device.
2 Literature Review

Previous projects related to this one will be analysed in this chapter in order to provide a better understanding of the importance of the performance of this project. Firstly, some previous works related to this will be analysed, and then the gaps and the coincidences between the works will be discussed. Finally, the novelties of this project with respect to the previous works will be summarised.

2.1 Previous Works

During the history of automation, a wide range of researches have focused on studying the methods for achieving the remote assembly. One of these researches developed a remote control system which can carry out some default assembly tasks [5]. The user, by means of an interface, can edit these tasks and the system builds automatically the 3D models and the path planning for performing the selected process of assembly. Long before this, some investigators of Purdue University also developed a remote control assembly which used a portable structured light unit that scanned the objects on the table [6].

The construction of 3D models is an important objective when developing a remote assembly system. For this reason 3D modelling is the main part of this project. This 3D modelling can be achieved in multiples ways, some more accurate than others. One of the first researches about 3D modelling of objects implemented a method which constructs the 3D model by the intersection of the projections of the silhouettes from different angles [7]. This method has been the base of several later researches, and this project makes use of different concepts explained in this research. Another research group (Rivers, Durand and Igarashi) implemented a method for constructing 3D models from the silhouette of the front, the top and the lateral view of the object using constructive solid geometry (CSG). Through simplified equations of CSG the 3D model is constructed with the unions of the different views in 2D. In another research, a method for building a 3D model using only one image was developed [8]. The model is constructed by inflation of the image but only works with certain shapes. A different approach to 3D modelling is to use a library with predefined models and compare them to the real scanned object [9].

There are previous approaches which have solved the problem of the delays in the communication with a robot manipulator via the Internet [10]. As in this research, the main idea for solving the problem is to create a virtual environment containing 3D models of the manipulator and the assembly parts. Another similarity with this project is that a general purpose PC has been used, the programming has been done in Java and the resulting program apparently is not difficult to use.
The main difference with respect to this project is the methodology followed for the three-dimensional modelling of the objects. The modelling is done using certain parameters of the objects, such as lengths or angles, obtained from the image snapshots. This type of modelling requires a previous knowledge of the shape of the objects; therefore the operator must recognize the objects in the snapshots and select a similar model. Consequently, it will not be possible to model very irregular objects.

The control of the IRB140 robot via the Internet has already been developed by the Wise-ShopFloor research group in previous projects [11]. The same thing happens with many of the filters used for the image processing and enhancement. Therefore this project inherits the remote control of the robot as well as some Java classes that implement image filters from other researches.

2.2 Research Objective

The main objective of this research is to develop an efficient system for remote assembly driven by 3D models of the assembly parts. For obtaining the 3D models of objects with a camera, the images taken must be processed first for obtaining the silhouettes of the objects. The part of image processing has been extended based on a previous project of the Wise-ShopFloor research group [12]. In that project a snapshot of a face is taken by the network camera. A set of filters is then applied to the snapshot and finally the robot is in charge of sketching the face. Many of these filters are used in the current project because the first stages of image processing are very similar.

The development of the 3D modelling algorithm of this project is based on an algorithm which obtains 3D models of complex objects in a fast way [13]. That work is based on the projection of silhouettes from different points of view, introduced in [7]. The modelling of our project is based on both of these works.

2.3 Novelties

Although this project is based on previous researches, it also introduces some novelties which might be interesting for the industry. First of all, the presented system only needs one camera and no turning tables or similar devices. This is because the camera is fixed on the industrial robot manipulator that will be used for picking and placing the assembly parts. The movements of the robot allow placing the camera in different positions. Another novelty is that there is no need to use a database with predefined models of objects. The initial shape of the models is created from the first snapshot; therefore it will be quite similar to the real objects. This allows using fewer snapshots because the models will need less trimming. Another difference with respect to previous researches is that the paths followed by the manipulator during the assembly process are not predefined. The
paths are done by the operator, who will be controlling the manipulator online through an Internet browser. Another novelty of this approach is that it can process the modelling of several objects at the same time. This can shorten the time taken for the modelling stage because previous works model the objects one by one.
3 Methodology

3.1 Stages of the project

This project has been divided into three main stages: The first one is focused on the programming of the image processing part, which includes the acquisition of images and filtering them until the silhouettes of the objects are obtained with the needed quality. Much of the work needed for this part can be taken from the previous year’s project [12]. The second main stage is programming the 3D modelling part. This part of the implementation will use the silhouettes from several snapshots of the previous part to build the 3D models of the objects that the robot is going to handle. The final stage of the project will be integrating the new implementations into the Wise-ShopFloor platform. Therefore the program must be compatible with the Wise-ShopFloor environment, which is executed in a web-based applet. These stages were initially planned over time. Table 5, in the appendix, shows the planned dates for starting and finishing the main tasks of this project.

3.2 Image Acquisition

In this process the system takes several snapshots from different angles that allow the establishment of a good silhouette of the objects in the cell. The system has a network camera which is mounted beside the gripper of the robot. This position gives the camera more degrees of freedom and will help taking the snapshots from different positions with more flexibility.

In order to improve the efficiency of image acquisition, a plane colour is used on the background. The first idea was to use a white background, but this resulted in two main problems. The first one was that the shadows of the objects and the robot were recognized as part of the objects. The second problem was that the yellow objects were difficult to separate from the background. This is because after the gray scale conversion the yellow and the white are very similar. To avoid this second problem, a java class was made so that the program converted the yellow colours into black, before the gray scale conversion. The problem of the shadows could not be easily avoided, so the idea of a white background was discarded and finally a black background was chosen. The problem with this new background is that dark colours of the objects can be confused with the background, but this can be avoided by taking care with the brightness, contrast and threshold parameters.

3.3 Image Processing Stages

During these stages, the images captured in the previous step are now enhanced by a set of filters applied in a certain order. This process is represented in Figure 3. The aim of this process is that the
snapshots allow us to recognize the shape of the real objects for the construction of the three dimensional models.

### 3.3.1 Conversion to Gray Scale

This is done by scanning the image pixel by pixel, and in each one of them the RGB values are replaced by the average of their original values [12]. The conversion to gray scale is necessary because image processing will be simpler without the colour information, as this information is not necessary for the current project.

### 3.3.2 Adjustment of Brightness and Contrast

The aim of this step is to highlight the objects with respect to the background. In other words, the gray tones of the objects are brightened and the background is darkened. This will help to separate the objects from the background for their detection. Two parameters are needed for this step, one for adjusting the brightness and the other one for the contrast. The optimal values of these parameters depend on the environment, especially on the illumination of the working area [12].

### 3.3.3 Gaussian Filtering

The Gaussian filter is a smoothing filter. Smoothing filters are used for blurring images and removing small details in them [14]. This way the background noise of the image is removed and therefore it will be possible to make the threshold of the image in the following step.

### 3.3.4 Image Thresholding

The thresholding process means that the pixels of the image that have a value of the intensity of gray below the threshold parameter are transformed into black, and the pixels that have the value of the intensity of gray above the threshold parameter are converted into white [14]. In some
implementations this works the other way around, so the pixels with darker tones of gray are turned into white and the ones with lighter tones of gray are converted into black. The threshold parameter must be given and its optimal value depends on the environmental light, just like with the brightness and contrast parameters.

3.3.5 Silhouette Labelling

The aim of this phase is to label each silhouette of the objects with a different number for their identification in the following steps. This means that the image is converted into a two dimensional array of numbers. Each number in the array corresponds to a pixel in the image. The value of the number will be a zero if it belongs to the background or a higher number if it belongs to an object. These numbers are the labels of the objects, which will be used for their identification [15].

3.4 3D Modelling

3.4.1 Camera Calibration

The calibration of the camera is needed to obtain certain parameters related to the camera. These parameters are necessary for some calculations during the construction of the 3D models. The parameters obtained from the calibration are the focal lengths, the situation of the optical centre, the radial distortion coefficients and the tangential distortion coefficients. The transformation matrix from the TCP of the robot to the camera frame is also obtained from the calibration [16].

3.4.2 Camera Coordinates Frame

A coordinate frame located on the camera will be needed for the construction of the three dimensional models. As seen in Figure 4, this frame is placed on the focal centre of the camera with the Z axis perpendicular to the camera lens and pointing out of the camera. The X axis of the camera frame points upwards when the robot is in the home position. The Y axis points to the left, when watching the robot from the front.
The transformation matrix from the base to the camera frame is obtained by multiplying the transformation matrix from the base to the TCP and the transformation matrix from the TCP to the camera. The first one of these can be obtained by solving a direct kinematics problem, while the second one is obtained by means of the camera calibration.

### 3.4.3 Image Plane Coordinates Frame

Another coordinate frame must be defined in the image plane. This one is a two dimensional frame placed on a corner of the image plane and it is needed for addressing the pixels in the image plane. The axis of this frame are U and V, see Figure 5.
The coordinates of a point in space projected into the image plane are relatively easy to obtain. It can be done with equations that contain the parameters calculated in the camera calibration. The inverse problem, which is obtaining the coordinates of a 3D point in space from the coordinates of its projection in the image plane, is more difficult. It is not possible to carry it out without certain assumptions. This is because we will only have two equations and we need to calculate the three unknown coordinates in 3D space [16].

3.4.4 Building the 3D Models

The main idea of the three dimensional models is that they will be composed by lines or pillars contained in the shape of the real object. Each of these pillars will be represented by two points in space situated at the ends of the pillar. The example in Figure 6 shows how the pillars can represent an object. In this case it is a cylinder. At the same time, each of these pillars is represented just by two points: one for each end of the pillar. This example has a very small number of pillars since it is just an illustration to explain the idea, but the real models will have a large number of pillars.
In order to build the pillars, the first snapshot taken by the system will always be a top view of the objects. The silhouettes obtained from this snapshot are then filled by vertical pillars with an estimated length. The length should be greater than the height of the objects. The lower ends of the pillars are situated on the table but the higher ends are the ones that must have an estimated height.

Therefore, the result after this step is that the points that represent the higher ends of the lines have an acceptable accuracy in the X and Y coordinates (in world coordinate system), but the Z coordinates of the points on the higher end of the pillars are estimated. The points that represent the lower end of the pillars will have the three coordinates accurately defined. Also there will be some pillars that should not be built at all due to the view of the upper bond of the object from the point of view of the camera. It can be seen in Figure 7.
After building all these pillars, some of them must be trimmed and others completely eliminated. Then they will represent the real objects with an acceptable accuracy. Following the previous example, Figure 8 shows which parts of the pillars should remain and which should disappear, in order to obtain an ideal representation of the objects. This perfect representation of the objects is impossible to achieve, but a very similar one can be obtained using a high number of snapshots in the following steps of the process.
The following step is to use the snapshots from other angles to obtain the silhouettes of the objects from different points of view. These silhouettes will be compared with the projection of the pillars to the same point of view. The parts of the pillars which do not overlap with the compared silhouette will be removed. In some cases an entire pillar will have to be removed. After repeating this process several times with different snapshots the pillars will represent the object with an acceptable accuracy. The accuracy will improve if more snapshots from different points of view are used. Figure 9 shows part of the process of trimming and eliminating pillars.

Figure 9: process of trimming and eliminating pillars.
3.5 Wise-ShopFloor integration

The final phase of the project is integrating all the previously developed programming with the Wise-ShopFloor virtual environment. This phase includes creating a user interface that implements all the needed functionalities for the remote assembly using 3D models. This interface should be as easy and intuitive to use as possible. The easiness of the interface will be tested and then possibly improved. Another objective of this final phase, but with a lower priority, is to add another type of movement to the robot which is currently missing in Wise-ShopFloor. This type of movement is the rotation of the robot around the TCP of the tool. It will be very helpful for a remote operator.

3.5.1 Analyses of 3D objects in Wise-ShopFloor

Before adapting this project to Wise-ShopFloor it is necessary to know how this virtual environment works. The Wise-ShopFloor already contains some 3D models in a specific format. This format consists of triangulated surfaces, or in other words composed by plane triangles. The triangles must be small enough and with many of them it is possible to represent any surface. If the surface is closed then it can represent the whole volume of an object.

3.5.2 Adapting our 3D models to Wise-ShopFloor

The 3D modelling developed until this point will need some modifications in order to integrate it with the virtual environment of Wise-ShopFloor. The most important modification is to create the models in the same format as in the virtual environment. This means that the models must be represented with triangulated surfaces. Another necessary change is that the models created must have the same scale as the model of the IRB140 robot that already exists in Wise-ShopFloor.

3.5.3 Model of the robot cell

Another thing to integrate in the Wise-ShopFloor is the 3D model of the cell where the robot works. This will be useful for an operator in order to avoid collisions of the robot with the cell. The problem is that it can also disturb an operator by obstructing the view of the objects during the assembly. This problem can be solved by implementing semitransparent models of the robot cell, or by implementing an option that gives an operator the ability to alter the transparency of the cell or even completely remove it.

3.5.4 Design of the interface

A user interface is necessary for the implementation of the program in Wise-ShopFloor. The main objective of the interface is that it should be user friendly. This interface should have an appearance in harmony with the interfaces of previous projects already implemented in Wise-ShopFloor. As the
other interfaces, it can be accessed from the drop-down menu in Wise-ShopFloor and have a characteristic and short name such as Remote Assembly. The lower part can contain the same tabs as the interface that already exists in the IRB140 robot. If possible this project will add a new tab for the control of the rotational movements around the TCP of the robot.

3.5.5 Testing the interface

The interface will need many technical tests to check if everything works as expected, just like the previous stages of the programming. It will also need a different test for evaluating the usability of the interface. This can be done with the help of other people who are not involved in this project, such as other automation students. The time that those people take to learn how to use the interface will be valuable information to know whether the interface is user friendly or not. Those people might also provide feedback or suggestions in order to improve the interface.
4 Implementation

This research can be divided into three different sections, which are the main phases in the development of the project. In the first part, the images are processed by applying a set of filters which helps the user to better recognize the silhouette of the pieces. In the second part the 3D models of the objects in the cell are generated from the silhouette of the objects in the different snapshots. In the last section, these 3D models are integrated in the Wise-ShopFloor virtual environment.

4.1 Image Processing Operations

In this phase a Java program was made for processing the images and therefore, results were observed. The libraries made last year for the previous project of Wise-ShopFloor have been implemented in the application. The UML class diagram shown in Figure 36, in the appendix, provides a better understanding the classes used in the development of the Java application. Last year’s project gives more information about this step [12].

4.1.1 Image acquisition

The system has a network camera which can work with two different resolutions. The high resolution is of 1280x960 pixels while the low one is of 640x480 pixels. The low resolution has been used during the first testing stages. However, during the normal operation, the camera will be working in the high resolution mode. The acquisition of images is done by accessing the camera with an IP address followed by a slash and a keyword which indicates the desired resolution. In this case, to obtain a high resolution image the program must access to the following address: “http://192.168.1.211/oneshotimage1”. If a smaller resolution is needed, the previous address must be replaced by: “http://192.168.1.211/oneshotimage2”.

4.1.2 Covering parts of the image

The objective of this stage is to enhance the image for a better performance in the image processing. First, the method “CoverGripper” handles the covering of great part of the gripper with a black layer. This covering avoids that some noise, produced by the bright parts of the tool, can appear in the image.

On the other hand, the method “BlackBackground” has the function of converting most of the background to pure black. In order to achieve this, the method takes each pixel and checks its colour. Then, if the colour is in a range between certain values, it is converted to black. This conversion eases
the recognition of the dark objects. Both processes are made before converting the image to gray scale.

4.1.3 Conversion from colour to gray scale

For this purpose, the application implements the class “ColoredToGreyScale” which was developed last year by the Wise-ShopFloor research group. In this class, a method is used to convert the input image into a gray scale image by using a class defined in Java. In each pixel, the values of red, green and blue are taken and then their average is calculated. The result of the average is used to replace previous values of red, green and blue. Thus, the pixel will turn into gray; because the three RGB values are the same.

4.1.4 Setting the brightness and the contrast

For this step, a class has been created with the parameters of brightness and contrast as inputs. This class reuses the code created last year by the Wise research group. In the first version of the program, two sliders were created to adjust the values of brightness and contrast, and with their help try to optimize the values. The optimal values depend on a wide range of factors such as the lights and the shadows in the working environment.

4.1.5 Gaussian filter application

In this phase, a Gaussian filter is necessary before thresholding the image. This function is used to eliminate the background noise in the image. To implement it, the method of the library from previous years was used. The Gaussian filter applied was the weak, which utilizes a 3x3 Gaussian mask.

4.1.6 Threshold Filtering

This is the final filter applied to the image in order to obtain the shape of the objects. The threshold filter segments the image into two colours depending on the intensity values of each pixel. Thus, in the program the pixels with a high value of intensity are converted to black and the pixels with a small value of intensity are transformed to white. The intensity of the pixels is compared with the threshold parameter, which is an input of the method. This is why it is important to highlight the objects in the cell from the background. During the development of the project, coloured shapes were used; because of that, the utilization of a black background was very helpful. The main advantage of using a black background is that the shadows of the objects do not interfere during the processing of the images.
A class created for the previous project was used for implementing this filter. In this class there are various methods implemented for doing the thresholding. In this project the manual thresholding was used because it is the one that gives the best results for obtaining the silhouettes. The threshold parameter must be given as an input, and its optimal value depends on the environmental lighting, just like with the brightness and contrast parameters. Therefore, during the testing phase, the optimal value of this parameter has been obtained for the situation where the system is going to be working. Nevertheless, the program has the possibility to adjust this parameter for the occasions in which the system is going to be working in a different environment.

4.1.7 Labelling the Silhouettes

This is a very important step in the image processing because with this class the identification of the different objects is performed. This stage labels all the objects, which allows constructing later their 3D models separately. The class was developed with the help of a class from the previous project but with additional changes. These changes were necessary because the class of the previous project was made for labelling edges but in this project the labelling is needed for the entire silhouette.

There are two types of algorithms that can be used for the labelling depending of the connectivity used for analysing the pixels, 4-connectivity or 8-connectivity. Although the 4-connectivity algorithm is faster, the 8-connectivity has been used in the code. This is because it is more accurate for this project, in which entire silhouettes must be labelled. The 4-connectivity is more appropriate for labelling lines or edges. For better observing the difference between the 4-connectivity and 8-connectivity, Figure 10 can help.

![Figure 10: Different connectivity modes.](image)

As it can be observed in the figure, in 4-conectivity only two neighbours of the analysed pixel are taken into account by the algorithm. The neighbour pixels that are checked are the one on top and the one on the left. In other words, if the algorithm is currently analysing a pixel that with the
coordinates \((i, j)\) the neighbours that are checked are the ones with the following coordinates: \((i-1, j)\) and \((i, j-1)\).

The algorithm used is the two pass connected components, which is named like that because the image is scanned twice. First, the threshold image is converted into a two dimensional array of zeros and ones, with one number for each pixel of the image. The zeros mean that the pixel belongs to the background and the ones mean that the pixel belongs to an object. This array is the input of the algorithm. In the first pass the array is scanned. Each pixel is assigned a temporary label, which is the lowest label of its neighbours. All the labels of the neighbours are stored in a list that keeps count of which labels are connected. If the neighbours do not have labels, a new one is created and assigned to the current pixel.

After the first pass, the label list contains the information of the labels that are connected. These groups of labels are reorganised so that each group is represented by the smallest possible integer. During the second pass, the connected labels are replaced by the integer that represents their group, and the output array is updated with these low values of the labels. The result is that the zeros of the array remain as zeros, but the ones are replaced by the label of the object which they belong to. This will allow the identification of the different objects in the first snapshot for their three dimensional modelling. Figure 11 illustrates the performance of this algorithm.

![Figure 11: Performance of the 2-pass connected components algorithm.](image)

During the tests of this part of the program, a method has been used to convert the final output array back into an image for visualising the performance of this process. This image was created pixel by pixel and the colour of each pixel was assigned depending on its label in the array. The result of this when the algorithm was working fine was that the silhouette of each object had a different colour and the background was black. In this step a sample of the colours of each object is taken. The sample is achieved by catching the average of the colour of random pixels in a figure. Afterwards, each colour is stored in a list and they will be used in the construction of the models.
4.2 3D Modelling Operations

The first approach of the 3D models of the objects will be built from the silhouettes obtained from the previous process applied on the top snapshot. The accuracy of these models will be refined with more silhouettes obtained from snapshots taken at different angles. The UML diagram of the actual implemented program can be seen in Figure 37, in the appendix.

4.2.1 Camera Calibration

Before beginning with the 3D modelling, the position of the objects needs to be known with respect to a fixed coordinate frame, like the base coordinate frame. The camera calibration is required to know where the focal centre of the camera is and placing on this point the camera coordinate frame. The focal centre is the point where the light rays that belong to a focused object converge. See Figure 12. This point is in front of the central point of the lens, at a distance called focal length.

![Focal length of a lens](image)

Figure 12: Focal length of a lens.

With the camera calibration process, a homogenous transformation matrix and some parameters such as the focal lengths, the principal point and the distortion parameters are obtained [16]. Figure 5 illustrates this concept. The obtained transformation matrix represents the orientation and the translations from the TCP to the camera focal centre. Using this transformation matrix, the coordinates from the base to the camera and vice versa can be calculated solving an ordinary direct
Kinematics problem [17]. With this information, the coordinates of each object will be known when the snapshots are taken. The camera has been calibrated for the high resolution mode (1280x960).

In order to implementing this in the code, a homogenous transformation matrix from one joint to the next has been calculated using the lengths of the links of the robot and the angles of the joints (Figure 13). These homogenous matrices have been multiplied beginning from one that transforms from the base to joint 1 and finishing with the transformation matrix obtained from the camera calibration. By calculating the inverse of the resulting matrix, the transformation matrix from the camera to the base can be obtained.

![Figure 13: lengths of the links of the ABB IRB140 robot. (ABB Robotics ©)](image)

### 4.2.2 Construction of the pillars

In this phase of the implementation, the construction of the pillars from the top image is explained. This method of construction of pillars is based on the method developed by Niem [13] and basically consists of the creation of pillars that will represent the volume of the silhouette of the object projected into the 3D space. Each pillar is represented by a set of two points situated at the ends of the pillar. Figure 14 shows the 3D model built from the silhouette of a square. Each point of the silhouette will produce two points in space. One of the points is at the same height as the surface of the table and the other point is placed in the same location and by adding the height of the largest
object to the value of the Z coordinates. The vertical pillars are constructed by joining each pair of points and this way a prism with the shape of the silhouette is built in 3D.

The data structure used for storing the 3D models of the different objects in the cell has been divided into three levels. Figure 15 helps analysing the development of the Java class better. The first level is for the set of all objects that are in the cell. This level contains the second level which is formed by each object in the cell individually. Finally, each object contains a third level which consists of the initial and final points of each pillar. For representing this in Java an array list of 2 points was created, each array list represents a pillar, an array list of pillars represents each object and an array list of objects saves the set of objects.
Figure 15: data structure of the 3D models of the objects.

The height of the tallest object in the cell and the height of the table have to be measured. The height of the table needs to be calculated with respect to the robot base and it is difficult to calculate directly because the exact position of the origin of the base coordinate system is unknown. For this reason it was necessary to define a new TCP in a corner of one of the fingers of the gripper. The method used for the definition of the new TCP was the 4 points method. When the new TCP was defined, it was jogged to touch the table on several positions and the Z value of the TCP was read on the teach pendant. The average of the different values is considered as the height of the table with respect to the base coordinate frame. This value allows to calculate the position of the objects with respect to the camera subtracting it from the position of the camera with respect to the base. This way, a Java class is developed which takes each point of the silhouette in the 2D image and transform them to the 3D space adding for each point two values of z, one with the distance between the objects and the camera and the other this same distance plus the height of the tallest object.
4.2.3 Trimming of the pillars

The trimming of the pillars is the step that follows after constructing them and for this reason it is realized from the second to the last snapshots. For carrying out this process, first, the silhouettes of the objects are obtained for each position of the snapshots planning. This is made by processing each image until the thresholding. In this way, the image of the silhouettes and the position of the joints of the robots must be known for trimming the pillars. Once this is made, the objects constructed in previous steps with the pillars are called for the method. With these objects, the two points of each pillar in each object are projected from the 3D world coordinate system to the image plane with respect to the camera using the joints values of the robot. After this, a line of pixels is constructed between the projections of these two points, which represent the pillar in the 2D plane. The lines are constructed using the Bresenham algorithm [18] which allows representing a line in a more efficient way than other algorithm. The efficiency of this algorithm is due to this code only uses integers and reduces the operations to the minimal. In addition, the algorithm can work in the eight octants representing the full line.

Now, the constructed lines are compared with the silhouette of the objects for checking which parts of the pillars do not belong to the object. This is tested by watching if there is intersection between the silhouette and the lines. If the points of the lines belong to the silhouette of the object, these remain intact; otherwise, the points are erased from the line. For carrying this out, first it is necessary to check if the projected lines are inside the image. If part of the line is outside of the image, it cannot be compared with anything, and this part is either maintained or deleted from the line if the silhouette is touching the image border or not respectively. Therefore, if the lines belong to an object, these lines are stored in a list. After this, the trimmed lines are again transformed to the 3D coordinate system. Thereby, the pillars are trimmed in each snapshot and the final model is obtained. Figure 16 explains the process in a graphical way, and it is based on an article consulted for developing this part of the project [13].
4.2.4 Triangulation of the objects’ surfaces

Until now, the models of the objects have been constructed using pillars, but these were not solid models. Through the triangulation process, a solid model of each object is created building a large number of small triangles with all the end points of the pillars. Each piece will be modelled by several parts which are the bottom, the top, the sides and the covers, that are necessary for joining the sides. This algorithm has been constructed using the “TriangleStripArray”, that is an object in Java 3D. Reviewing the webpage [19] helped to understand the construction of the objects’ triangles.

The first thing to do is dividing the group of pillars that represent one object into sections or slices. The subsequent process for the creation of the bottom and top surfaces is the same. It consists of the construction of strips of triangles joining the points of the top or the bottom of all the pillars of two sections. In this case three pillars are necessary for the construction of one triangle; two of them belong to the same section of the object. Figure 17 can help to exemplify the process. The construction of the first triangle is created by taking the first point of the first section, the first point of the second section and the second point of the first section. The last two of these three points will be used with the next one for building the next triangle. Then points are added one by one alternating the section to obtain one strip of triangles. This strip is built for each pair of sections to complete a whole surface.
The surfaces at the vertical sides are constructed in two parts which are assembled together later. These sides are built with pairs of vertical triangles that have the same length as the pillars. Thus, for each pair of vertical triangles four points are taken, the two points at the ends of the first pillar in a section and the same points for the first pillar in the next section. With these four points two triangles are obtained for each pair of sections. This can be easily understood observing Figure 18. The figure represents four sections and with the four points selected, the first pair of triangles has been constructed. Repeating this process for each pair of sections results in a vertical wall. With this method, two walls can be obtained: one from the first pillars of each section and another one from the last pillars of each section.
After this process the first and last sections need to be covered by a surface to complete the entire object. These surfaces are called the covers and cannot be created with the same method as the vertical sides because that one works with pairs of sections. These covers are constructed by taking the lower point of a pillar in one section and the upper point of the next pillar in the same section. When this is done, all the parts of the object are linked and they form a closed 3D shape which represents the object.

4.3 Integration in Wise-ShopFloor

The final part of this project is integrating the work done up to this point in Wise-ShopFloor. This will allow jogging the robot remotely via the Internet and obtaining information from the robot controller such as the position and the joint values. Therefore this section explains the integration process, like the design of the user interface and the addition of certain functionalities that will be useful for this project and possibly for future ones.

4.3.1 User interface

The user interface of this project has been designed following the same style as previous projects integrated in the Wise-ShopFloor. On the top-right corner there is a menu which allows the user to choose one of the selections in Wise-ShopFloor. The option named “Remote Assembly” gives access
to this project. The interface is divided into four main areas: top, right, bottom and central (see Figure 19). The biggest area of the interface is the central part, which is used to display the 3D scene. The scene always includes a model of the IRB140 robot, its controller and the cell in which they are contained. The models of the objects will also appear but only after the modelling process, or part of it, has taken place. The toolbar at the top is exactly the same for all the sections in Wise-ShopFloor. It includes the necessary options for navigating in the displayed 3D scene, such as shifting, rotating and zooming.

The right part of the interface is also very similar to the rest of the previous projects. On the top part, some information of the point of view of the 3D scene is displayed. Below this, there is a device status table, which shows the values of the joint angles of the robot. These values are constantly updated as long as the robot is on-line. Below this table there are two tabs for cameras. One of the tabs is for the camera mounted beside the gripper of the robot, and gives the option of taking a snapshot or getting a live video stream from it. The other tab is used to control an auxiliary camera, which has two electric motors that allow the rotation of the camera around two axes. The tab includes the necessary buttons for taking snapshots, obtaining a live video stream and for controlling the motors.
The lower part of the interface is the one that gives the user the control of the modelling and the robot motion. In order to have access to the control, the user must firstly select the On-line mode (at the left hand side) and insert the required password. This part of the interface has four tabs with the following names: Jogging components, Reorienting, Linear jogging and Modelling. The Jogging components and the Linear Jogging tabs were developed during previous projects. They have been included in this project because their functionalities can be very useful for a remote operator.

The Jogging Components tab is automatically generated by the Wise-ShopFloor system, depending on the machine that has to be controlled. This tab includes two buttons for each joint of the robot. With these buttons, an operator can control each joint by increasing or decreasing the joint angle values. Therefore this tab allows jogging the robot joint by joint. The Linear Jogging tab gives the user the option of moving the tool along a straight line. The line must be parallel to one of the axes of the robot base frame. Therefore the tab includes a total of six buttons that can be used to move in both directions along each of the axes.

The Reorienting tab includes six buttons for reorienting the gripper around the three axes of its frame. The reorienting functionality is explained in another section of this report. As seen in Figure 20, this tab includes two more sections which are not directly related to the reorienting functionality. One of the sections of the tab is for the gripper, and includes two buttons for opening and closing it. The section further to the right has only one button that displays the label Go to object. If this button is clicked, a dialog box will appear with a list of the modelled objects. The user can select one of the objects and click on the Accept button. This will send to the robot controller the coordinates of the centre of the selected object. The robot will then place the TCP of the gripper so that its X and Y coordinates are the same as the ones of the central point, but the Z coordinate of the TCP will not be modified. In other words, the TCP of the gripper will be placed above the central point of the object, ready for the user to reorient the gripper, lower it and close it.

The Modelling tab consists of several parts, as Figure 21 shows. The “Next position” button will make the robot go to one of the predefined snapshot positions when it is clicked. These positions follow a certain order to make sure that the path is free of collisions. Therefore the robot will go to one
position or another, depending on the times that the button has been clicked. A click on this button will also enable the “Image processing” button. When the “Image processing” button is used, the camera will take a snapshot and the system will process it with the parameters that have been saved for the current position. Then, a dialog box will appear showing the results of this image processing and giving the possibility of redoing it manually by changing the parameters. When the processing has finished and the user clicks on the “Accept” button of the dialog box, the “Apply to model” button will be enabled. This button will process the pillars that represent the real objects, based on the results of the image processing. Then the models of the objects will appear on the 3D scene, or will be updated if they already existed. The process with these three buttons can be done up to eight times to obtain the three dimensional models of the objects that are situated in the cell. Finally, the “Start” button in the Automatic Modelling section will make all the modelling process automatically and the final models will appear in the 3D scene when they are finished.

4.3.2 Path planning for the Snapshots

The planning of the path followed for taking the snapshots is an important step for the correct construction of the models. The quantity of snapshots and the position where they are taken will determine the quality of the final models. In this project a fixed number of eight snapshots were taken for the creation of the models. The choice of eight snapshots was taken when searching the balance between the quality of the model and the quickness of the construction. Figure 22 shows the position of the camera for the photos. As it can be observed the path follows a symmetric way. A problem in the path is that there is no snapshot behind the objects. This is due to the limitations of the robot which cannot reach these positions.
The parameters applied in the processing of each snapshot strongly depend on the position of the camera therefore they are another significant factor for the path planning. All positions require their own parameters for the values of brightness, contrast and threshold. These parameters depend on a great variety of factors such as the level of light in the room or the highlights generated by the pieces. Some default parameters for certain conditions have been obtained and included in the automatic modelling. Applying the default parameters sometimes gives poor results therefore it is necessary to have an option of manually changing these values if needed for the correct elaboration of the modelling.

4.3.3 Live 3D scene

Most of the area of the user interface is occupied by a 3D scene implemented in Java 3D. As Figure 23 shows, this scene is added to the Wise-ShopFloor virtual universe through a main Branch Group. The Wise-ShopFloor virtual universe adds the lights to the scene and is also responsible for the mouse behaviour and the navigation in the scene. There is more information about this virtual universe in [20]. This scene is always showing a model of the IRB140 robot, a CS controller and the
cell where they are placed. The model of the cell is composed by different parts, such as the wagon, the cage and the right and left modules. These models and the one of the controller have been obtained from the resources of Computer Aided Robotics course of the University of Skövde. The models were converted to VRML files, which is a file format compatible with Java3D. These models are added to the main Branch Group via another Branch Group and a Transform Group.

The robot model has been added via a Branch Group, using some Java classes previously implemented in Wise-ShopFloor. This model consists of a dynamic component for each link of the robot. The Dynamic components load the necessary VRML files and also represent the movements of the robot by reading its joint values. Further information of this model can be obtained from [20]. The model of the base of the gripper is then attached to the model of the robot using a static component. Then, two more static components (used for the models of the camera and the TCP frame) and two dynamic components (used for the left and right fingers of the gripper) are attached to the base of the gripper. There is an invisible static component attached to the gripper, which is used for detecting the collisions with the objects and simulating the grasping of the gripper.

The modelled objects are added to the scene using a static component for each one. Each of these static components is the parent of a transform group, which at the same time is the parent of a shape 3D that contains the modelled object. All the static components that contain a modelled object are added as children of a root static component, which is necessary for simulating the gripper behaviour. The root static component is a child of a Branch Group which can be added and removed to the live scene each time it has to be updated. It is only possible to add Branch Groups to a live scene, but there must be another Branch Group which is always child of the main one. This adding and removing is represented in Figure 23 by the dashed line between two Branch Groups.
4.3.4 Gripper Implementation

The pick-and-place operations require a tool which allows performing them in a fast way. The robot uses a gripper which works with compressed air to carry out these operations. The gripper consists of two lateral parts and a central one. Each lateral part has two fingers, a small one and a large one to pick objects of different shapes and sizes. The integration of the gripper can be divided into three
different phases: the model of the gripper, the implementation of the performance of the physical gripper and the implementation of the behaviour of the virtual gripper.

The 3D models of the parts of the gripper were taken from the course of Computer Aided Robotics in the University of Skövde (Sweden). The models were converted from the format “obj” to “vrml” which is the format used by Java and imported in the 3D scene. Once this was done, applying a transformation to the gripper was necessary to adjust its position with respect to the robot. The 3D model of the gripper can be viewed in Figure 24.

Secondly, the functions of the grippers have to be activated both in the interface and in the physical gripper. To develop this, a method has been implemented, which carries out the creation of a hash map. This map is created with a command name to identify the process and with an integer that determines if the gripper has to be opened or closed. Later, this map is converted to a packet where a header, which is used to identify the process in the RAPID code in Robotstudio, is added. The performance takes place as follows: when the close or open button is pressed a socket with the previous information is sent to the robot controller. If the value of the integer is 0 the opening signal is activated and the gripper is opened, in the other hand if the gripper is 1 the closure signal is set and the gripper is closed. The movement of the gripper in the interface works in the same way, under similar conditions the fingers of the gripper are displaced to simulate the opening or the closing of the gripper.

Finally, the behaviour of the gripper has to be set to achieve the assembly operations. This has been accomplished with the realization of a basic collision system which detects if an object is in contact with the finger of the gripper to attach them together. In order to detect the collision, the application checks if the points of the upper surface of the object are intersecting with the bounds of a transparent box in the middle of the gripper. Therefore, if the intersection exists the object is ready.
to be taken and then is attached to the gripper when the closing signal is sent. This way, the object will move with the gripper until the gripper is opened and the intersection disappeared.

4.3.5 Reorienting Functionality

The Wise-ShopFloor virtual environment has different functionalities for jogging the IRB140 robot, implemented during previous projects. These are the linear jogging and the jogging by joints. Nevertheless, a useful capability that was not included was the reorienting of the tool. The reorienting of the gripper is very useful for online operations of pick and place of objects. Therefore, this reorienting feature has been developed during this project, based on the linear jogging. The reorienting class developed in Java sends the server a packet that contains the necessary information. This information is the header, the axis of reorientation and the direction. The server passes this information to the robot controller through a socket. The socket is unpacked and the controller executes the desired RAPID instructions. When the socket is unpacked, the first thing that is read is the header which, in this case, will indicate that the desired operation is a reorientation of the tool. The next information that is read is the axis and direction of reorientation. Depending on these parameters, the controller will calculate the next target, which will be on the same position but with a different orientation. Then, if it is possible to achieve the target, the robot will go to it. This class allows reorienting the gripper around the three different axes of the tool frame in any direction, as long as the robot is able to achieve it.
5 RESULTS AND DISCUSSIONS

Figure 25 shows the process of converting the image from colour to gray scale. A black fabric was used to cover the cell to ease the distinguishing between the objects and the background. In this step a covering of the gripper with a black shape is also applied. This avoids some parts of the gripper to appear as shapes which can be recognised later by the system as assembly parts. In the greyscale image a similar technique is applied which deals with converting the background in pure black to ease the work in futures steps.

![Figure 25: conversion of the image from colour to gray scale.](image)

Figure 26 shows the change of the image in gray scale when an adjustment of the brightness and contrast is made. The shininess and the shadows have been greatly reduced thanks to the black fabric used as background. However, some parts of the pieces can disappear due to shadows in the objects. In additions, the noise around the silhouettes can influence in the quality of the model. For these reasons a good adjustment of the brightness and contrast is necessary.

![Figure 26: Adjusting the brightness and the contrast of the gray scale image.](image)
Then a threshold filter is applied. A correct adjustment of the threshold is important to obtain good silhouettes of the objects. Between these two processes a Gaussian filter is used to apply the thresholding filter in the snapshot. Figure 27 shows the effect of applying the threshold filter in the image.

![Brightness and contrast adjustments](image1)

![Threshold Image](image2)

**Figure 27: effect of applying the threshold filter.**

The labelled image can be seen in Figure 28. A label is assigned to each object to distinguish them. In order to visualise the result, each label is represented by a different colour on the image which is the same colour obtained by the camera.

![Threshold Image](image3)

![Labeled Image](image4)

**Figure 28: object labelling.**

Figure 29 shows the primitive 3D models formed by pillars. These pillars are lines projected in the 3D space from the silhouettes, and have been coloured with the sample colours of the objects to distinguish the models of the different objects. These models are basic and have been taken directly from the silhouettes of the top view, so they still have to be trimmed with silhouettes from other points of view to obtain accurate models.
Figure 29: construction of the initial 3D models.

Figure 30 shows the result of the first trimming. With this snapshot, the system has to process the highest number of operations because it carries out the elimination of a lot of spare parts.

Figure 30: results of the first trimming.

In the following image, Figure 31, the result of the last trimming can be observed. This is the final form of the pillars after all the snapshots and their respective trimming processes have taken place. The final 3D models of the objects will be constructed with these pillars.
Finally, the construction of 3D shapes from the pillars can be observed in Figure 32. These shapes are ready to be used by an operator in the remote assembly process.

The camera has been used in the highest possible resolution, which is 1280x960. This resolution can provide a good accuracy for the construction of the 3D models, but slows down all the process since a large number of pixels must be analysed. A lower resolution would make the process faster and probably would keep the accuracy in an acceptable level. However, the used resolution is the high level because it was the one used during the camera calibration and some parameters of the camera depend on the resolution.

The time that each process takes has been measured ten times and then the averages have been calculated. This has been done in two different computers and different results have been obtained. One of the computers has an Intel Core i5 750 2.80GHz central processor, with 4GB of RAM and runs
The results obtained with this computer are shown in the following table, where the times are shown in milliseconds.

<table>
<thead>
<tr>
<th>Testing times (ms)</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>Mean</th>
</tr>
</thead>
<tbody>
<tr>
<td>Loading image</td>
<td>1056</td>
<td>1300</td>
<td>1446</td>
<td>1253</td>
<td>1202</td>
<td>1195</td>
<td>1158</td>
<td>1232</td>
<td>1139</td>
<td>767</td>
<td>1174.8</td>
</tr>
<tr>
<td>Gray scale</td>
<td>180</td>
<td>173</td>
<td>215</td>
<td>205</td>
<td>187</td>
<td>239</td>
<td>317</td>
<td>219</td>
<td>203</td>
<td>409</td>
<td>234.7</td>
</tr>
<tr>
<td>Brightness/contrast</td>
<td>192</td>
<td>206</td>
<td>202</td>
<td>334</td>
<td>202</td>
<td>206</td>
<td>204</td>
<td>187</td>
<td>421</td>
<td>196</td>
<td>235</td>
</tr>
<tr>
<td>Gaussian filtering</td>
<td>113</td>
<td>125</td>
<td>94</td>
<td>114</td>
<td>125</td>
<td>161</td>
<td>145</td>
<td>124</td>
<td>124</td>
<td>125</td>
<td>125</td>
</tr>
<tr>
<td>Thresholding</td>
<td>276</td>
<td>324</td>
<td>263</td>
<td>259</td>
<td>250</td>
<td>326</td>
<td>258</td>
<td>234</td>
<td>250</td>
<td>282</td>
<td>272.2</td>
</tr>
<tr>
<td>Labelling</td>
<td>10507</td>
<td>13918</td>
<td>13234</td>
<td>11320</td>
<td>10015</td>
<td>12531</td>
<td>12211</td>
<td>9266</td>
<td>12200</td>
<td>11294</td>
<td>11649.6</td>
</tr>
<tr>
<td>Pillars</td>
<td>2256</td>
<td>2652</td>
<td>1715</td>
<td>2243</td>
<td>2169</td>
<td>2541</td>
<td>1566</td>
<td>1934</td>
<td>1826</td>
<td>1841</td>
<td>2074.3</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td>14580</td>
<td>18698</td>
<td>17169</td>
<td>15728</td>
<td>14150</td>
<td>17199</td>
<td>15859</td>
<td>13196</td>
<td>16163</td>
<td>14914</td>
<td>15765.6</td>
</tr>
</tbody>
</table>

Table 1: results obtained from the timing tests with the Intel Core i5 computer.

Comparing both tables, it can be seen that the program runs faster on the Intel Core i5 than in the Intel Xeon. This difference in average is smaller than two seconds and therefore is not very significant. The graph in Figure 33 shows the percentage of average time that takes for accomplishing each process in the different hardware. These percentages are very similar in both computers. In the graph it can be seen that if an improvement in the speed is needed it must be done in the labelling process. A first approach for improving the labelling can be to reduce the number of neighbour pixels checked, but this can reduce the performance of the algorithm.
Some of the parts implemented in this project do not have a graphical output and for checking their performance it was necessary to carry out some small experiments. One of these experiments was for testing the transformation matrix from base to the TCP. The joint values and the position of the TCP were given as input and the expected output would be (0, 0, 0). The values obtained were not zeroes but they were small enough to consider them as zeroes and the transformation matrix can be considered as correct. Another experiment tested the transformation matrix from the camera frame to the base. This consisted of jogging the robot to the home position and giving as input the joint values and the origin point. Then joint six was rotated 180 degrees and the inputs were given again. The results obtained were compared and it could be seen that the X coordinate was the same for both, the Y coordinate had almost the same absolute values but with different in signs and the Z values were very similar but not exactly the same. The difference of the Z values is due to the position of the camera with respect to the revolution axis of the sixth joint. The results obtained in the experiment confirmed that the transformation matrices are correct.

The silhouette labelling method has been optimized in order to decrease the time that this process takes. Some minor changes in the code have given a great improvement in the execution time with respect to the old version. The main change done is in the part of the code that searches for the smallest label of the neighbour pixels. The times taken by the image processing stages have been reduced.
measured again in order to compare them with the old ones. The next table shows the results of the new experiment, which has been done with the Intel Core i5 computer.

<table>
<thead>
<tr>
<th>Test #</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>Mean</th>
</tr>
</thead>
<tbody>
<tr>
<td>Loading image</td>
<td>1690</td>
<td>822</td>
<td>1489</td>
<td>834</td>
<td>1205</td>
<td>1038</td>
<td>1051</td>
<td>1548</td>
<td>985</td>
<td>942</td>
<td>1160.4</td>
</tr>
<tr>
<td>Gray scale</td>
<td>788</td>
<td>820</td>
<td>1036</td>
<td>520</td>
<td>1063</td>
<td>1095</td>
<td>540</td>
<td>840</td>
<td>732</td>
<td>1113</td>
<td>854.7</td>
</tr>
<tr>
<td>Brightness/Contrast</td>
<td>178</td>
<td>180</td>
<td>179</td>
<td>181</td>
<td>178</td>
<td>185</td>
<td>182</td>
<td>190</td>
<td>196</td>
<td>188</td>
<td>183.7</td>
</tr>
<tr>
<td>Gaussian filtering</td>
<td>174</td>
<td>125</td>
<td>125</td>
<td>127</td>
<td>141</td>
<td>127</td>
<td>151</td>
<td>146</td>
<td>129</td>
<td>137</td>
<td></td>
</tr>
<tr>
<td>Thresholding</td>
<td>234</td>
<td>269</td>
<td>322</td>
<td>293</td>
<td>287</td>
<td>236</td>
<td>326</td>
<td>341</td>
<td>325</td>
<td>221</td>
<td>285.4</td>
</tr>
<tr>
<td>Labelling</td>
<td>1732</td>
<td>2937</td>
<td>1522</td>
<td>1714</td>
<td>1767</td>
<td>1515</td>
<td>1399</td>
<td>1395</td>
<td>1487</td>
<td>2106</td>
<td>1757.4</td>
</tr>
<tr>
<td>Pillars</td>
<td>481</td>
<td>370</td>
<td>481</td>
<td>475</td>
<td>467</td>
<td>510</td>
<td>466</td>
<td>496</td>
<td>444</td>
<td>383</td>
<td>457.3</td>
</tr>
<tr>
<td>Total</td>
<td>5277</td>
<td>5523</td>
<td>5154</td>
<td>4142</td>
<td>5094</td>
<td>4720</td>
<td>4091</td>
<td>4961</td>
<td>4315</td>
<td>5082</td>
<td>4835.9</td>
</tr>
</tbody>
</table>

Table 3: results of the experiment with the new labelling class, measured in milliseconds.

As the table shows, the average time that the image processing takes is lower than five seconds. This is very low compared to the old result, which was of 15.7 seconds with the same computer. This is because the labelling process has been lowered from an average time of 11.6 seconds to 1.75 seconds. Figure 34 shows a graph of the percentage of time taken by each of the image processing stages. Comparing this graph with the one obtained with the old code, the labelling has passed from taking around 75% of the image processing time to a 36%.

![Percentage of average times taken by processes](image-url)

Figure 34: Average time taken by each of the image processing stages
Some tests have been done in order to measure the time that the automatic modelling function takes. These tests were done for modelling from one piece up to five pieces, and for each number of pieces the test was repeated three times. The results of the experiment are shown in Table 4. These results depend on many other variables apart from the number of modelled objects, such as the lighting of the environment and the positions of the objects. As the results show, the average time needed for modelling grows with the number of objects modelled. The growth of the time with respect to the number of objects is not very significant because most of the calculations needed for modelling the objects are done while the robot is moving to the next snapshot position. Nevertheless, when these calculations take too long (especially for the first snapshots) the robot waits at the next position before taking the next snapshot.

<table>
<thead>
<tr>
<th></th>
<th>1 Piece</th>
<th>2 Pieces</th>
<th>3 Pieces</th>
<th>4 Pieces</th>
<th>5 pieces</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test 1</td>
<td>69</td>
<td>70</td>
<td>73</td>
<td>82</td>
<td>80</td>
</tr>
<tr>
<td>Test 2</td>
<td>76</td>
<td>74</td>
<td>75</td>
<td>73</td>
<td>83</td>
</tr>
<tr>
<td>Test 3</td>
<td>69</td>
<td>76</td>
<td>71</td>
<td>82</td>
<td>92</td>
</tr>
<tr>
<td><strong>Average</strong></td>
<td><strong>71.33</strong></td>
<td><strong>73.33</strong></td>
<td><strong>73</strong></td>
<td><strong>79</strong></td>
<td><strong>85</strong></td>
</tr>
</tbody>
</table>

Table 4: time measured in seconds taken by the automatic modelling in function of the number of modelled objects.

The time that the robot takes for completing the path of the automatic modelling, but without taking any snapshot or doing any processing is 61 seconds. This means that the average time that the robot is waiting for the calculations to finish goes from about 10 seconds for one object up to 24 seconds for five pieces. Taking in mind that the time needed for processing the snapshots is higher for the first ones; the automatic modelling time can be improved by using two parallel processes. One process would be in charge of taking the snapshots in each position, storing them and passing them to the other process when it is needed. The other process would be in charge of using the snapshots to obtain the three dimensional models. This way, the large time that is needed to process the first snapshots could be distributed along the time that the robot is moving and there are no calculations taking place, since the last snapshots can be processed very quickly. Another improvement for the automatic modelling time could be designing a new path for taking the snapshots. The current path takes too long because some of the targets have a long distance between themselves and joint number 5, which is quite slow, requires a big reorientation. Nevertheless, the long distances are necessary for having a big enough modelling area and having snapshots from different angles.
6 CONCLUSIONS AND FUTURE WORK

The performance of the image processing is good enough for the needed accuracy, and is completed in a reasonably short time. The speed could be improved by simplifying some of the developed Java classes, like it was done with the class in charge of labelling the different objects. The final results of the modelling process clearly could be improved. This could be achieved by optimizing the performance of the pillar trimming code. For example, the modelling of pieces with dark colours is very inaccurate because these colours are confused with the background by the code. Another issue is that in some cases there are pillars that remain where they should not be or their lengths are not trimmed enough, thus the pillars end up being too long.

Since the application of this project is for assembly, there is no need of achieving a perfect accuracy because the modelling speed might be more important for the industry. A possible improvement of the 3D modelling is the detection and modelling of concavities of the objects. However the gripper can still grab the objects without considering the concavities although for assembling pieces that fit one inside the other this could be a problem. For this reason, a future work can include the modelling of the concavities and the holes of the pieces.

The integration of this project in Wise-ShopFloor can also be improved. For example, the reorienting tab contains a button for ordering the robot to approach towards a certain object. This button could be better located in a specific tab for assembly operations, but this new tab has not been considered in this project since it would only contain this button. Another issue is the gripper opening and closing. An operator would probably find comfortable to have these two buttons in all the tabs related to the robot jogging. They have not been added in all the tabs because some of them belong to the previous projects in Wise-ShopFloor and this could interfere with them. Another possible improvement for this project is the reorient jogging. This improvement could be limiting the joint values. It would prevent the incorrect movements that the robot makes when it is in a forced position and it receives orders for reorienting.

The function for approaching the gripper to the objects can be enhanced from several points of view. One of these points of view is calculating the centre of the objects. The calculation that has been implemented helps approaching the gripper to the object but the result is not the real centre. This is because the Bounding Sphere class has been used in order to obtain this point. It can also be due to the accuracy of the models. Better results can be achieved by calculating the geometric centre of the shapes of the object surfaces, although this will require some more computation. One more improvement for this function is making the interface friendlier for the operators. One possibility is to include some diagram of the scene with labels in the menu for choosing the object. Another
An interesting future work could be calculating the orientation of the gripper needed for picking each object. Then the robot could pick up the selected object automatically.

An interesting future development of this project can be preparing it for the industry. This can include making the system faster either by simplifying the programming or by reducing the number of snapshots needed. Some fixed cameras in the cell could support the main camera and make the process faster by reducing the paths that the robot has to do for taking the snapshots. An interesting possibility could be using just one camera and a rotating table, similar to the one described in [13]. This would make the system more flexible allowing taking snapshots of the objects from any point of view, giving as result more robust models. It would also make the process faster because there would be no need of moving the robot along large distances.

For the industry it would also be interesting to include some automatic assembly functions. The operator could choose between manual or automatic and this would save some time during the assembly process. The automatic mode could detect the pieces and depending on them choose the necessary assembly process. For the future, it could also be interesting to improve the camera calibration by adding several calibrations for different resolutions. An operator can choose the resolution to adapt it to the available resources and to the needs at that moment. The calibration could even be automated for the cases in which the camera accidentally gets moved from its position. Another interesting further work could be having an off-line assembly where the operator can assemble the parts off-line while the system is recording the process. This will provide the ability for the operator to undo or redo some steps of the assembly, and when the operator is satisfied with the process he can upload it to the server and the robot will perform the task. This could be categorized as a remote robot off-line programming.

An interesting possible addition for the project is the capability of the system to auto-adapt to the ambient light conditions. The system would need one or more light sensors. One of the sensors could be placed on the camera and would give relevant information of the light depending on the orientation of the camera. The image processing parameters would be adjusted depending on the information received from the sensors.

One more improvement for the project is to make the system search the location of the objects and decide the best position of the camera for the first snapshot. This would give more flexibility because the actual modelling zone is restricted to the area of the table that appears in the first snapshot. This would allow placing the object anywhere in the cell as long as the robot reaches the necessary
position for the snapshot. After the first snapshot, the robot could adapt the other snapshot positions according to the situation.

Finally, the gripper used during the development of this research has certain limitations regarding the sizes, shapes and weights of the objects that it can pick. These limitations can be reduced or eliminated by replacing the pressure gripper by an adaptive electric gripper. These modern grippers have the possibility of handling of a great variety of shapes and sizes of objects. The main problem of these grippers is their expensive price. A cheaper alternative is a vacuum gripper, which can handle almost any shape of the objects, as long as there is a flat surface.
7 Appendix

7.1 Definitions and Acronyms

Image: this term is referred to a two-dimensional function \( f(x, y) \) which represents the intensity of light. The variables \( x \) and \( y \) are the spatial coordinates and the value of \( f \) is proportional to the luminosity of the image in the point \( (x, y) \).

RGB: this term is referred to a model in which any colour can be represented by a mixture of the colours of the three primary lights (red, green and blue).

Digital Image: a digital image can be considered like a matrix whose index of rows and columns identify a point of the image and contain the value that identifies the intensity level in that point.

Pixel: is the smallest unit of picture that can be represented in a digital image. It is the intersection between a row and a column.

Neighbours: each pixel of the image has 4 horizontal and vertical neighbours and another 4 diagonal neighbours. The first neighbours are denoted by \( N_4 (p) \) and the second ones are denoted by \( N_8 (p) \). See Figure 35.

![Figure 35: neighbour pixels](image)

TCP (Tool Central Point): it is a point relative to the tool that the robot takes as reference.

IP (Internet Protocol): is a numeric address used for communications over the Internet.

UML (Unified Modelling language): is a diagram used for explaining the structure of object-oriented software.

Shape 3D: a Java3D Node class that contains one or more 3D geometries with a common appearance [20].
**Transform Group**: a Java 3D Group Node class which allows executing spatial transformations (translations and rotations) in order to place the children contained in the group at a desired position [21].

**Branch Group**: a Java 3D Group Node class which can be attached to and detached from live scenes at runtime, as long as the relevant capabilities have been set [22].
7.2 UML Diagrams

Figure 36: UML diagram of the image processing classes
Figure 37: UML diagram of the 3D modelling classes
### 7.3 Initial project planning

<table>
<thead>
<tr>
<th>Task name</th>
<th>Duration (days)</th>
<th>Start</th>
<th>Finish</th>
</tr>
</thead>
<tbody>
<tr>
<td>Image processing</td>
<td>10</td>
<td>3/2/2012</td>
<td>16/2/2012</td>
</tr>
<tr>
<td>Designing</td>
<td>4</td>
<td>3/2/2012</td>
<td>8/2/2012</td>
</tr>
<tr>
<td>Implementation</td>
<td>6</td>
<td>9/2/2012</td>
<td>16/2/2012</td>
</tr>
<tr>
<td>Analysis and testing</td>
<td>3</td>
<td>14/2/2012</td>
<td>16/2/2012</td>
</tr>
<tr>
<td><strong>3D modelling program</strong></td>
<td><strong>37</strong></td>
<td><strong>9/2/2012</strong></td>
<td><strong>30/3/2012</strong></td>
</tr>
<tr>
<td>Documentation of 3D modelling algorithm</td>
<td>37</td>
<td>9/2/2012</td>
<td>30/3/2012</td>
</tr>
<tr>
<td>Designing</td>
<td>3</td>
<td>10/2/2012</td>
<td>14/2/2012</td>
</tr>
<tr>
<td>Implementation of 3D modelling</td>
<td>31</td>
<td>15/2/2012</td>
<td>28/3/2012</td>
</tr>
<tr>
<td>Testing 3D modelling</td>
<td>9</td>
<td>20/3/2012</td>
<td>30/3/2012</td>
</tr>
<tr>
<td><strong>Report writing</strong></td>
<td><strong>94</strong></td>
<td><strong>10/2/2012</strong></td>
<td><strong>20/6/2012</strong></td>
</tr>
<tr>
<td>Introducing program into Wise-ShopFloor</td>
<td>34</td>
<td><strong>16/4/2012</strong></td>
<td><strong>31/5/2012</strong></td>
</tr>
<tr>
<td>Adapting program to Wise-ShopFloor</td>
<td>14</td>
<td>16/4/2012</td>
<td>3/5/2012</td>
</tr>
<tr>
<td>Integrating program into Wise-ShopFloor</td>
<td>14</td>
<td>4/5/2012</td>
<td>23/5/2012</td>
</tr>
<tr>
<td>Analyze and testing</td>
<td>6</td>
<td>24/5/2012</td>
<td>31/5/2012</td>
</tr>
</tbody>
</table>

Table 5: Initial plan of the main tasks of this project
8 References


2. ABB. ABB IRB 140- Robots. [Online] [Cited: 4 April 2012.]

3. —. ABB IRC5 - Controllers. [Online] [Cited: 4 April 2012.]


