|  |
| --- |
| Inholland Composites |
| Technical Report |
|  |

|  |
| --- |
| Joey Meijer  31-1-2021 |

# Document revisions

|  |  |  |
| --- | --- | --- |
| **Date (DD/MM/YYYY)** | **Version** | **Changes** |
| 19/01/2021 | 0.1 | Initial design |
| 20/01/2021 | 0.2 | Finished introduction |
| 21/01/2021 | 0.3 | Chapter 3 finished |
| 22/01/2021 | 0.4 | Chapter 4 and 5 finished |
| 26/01/2021 | 0.5 | Chapter 6 finished |
| 31/01/2021 | 1.0 | Document completed |

Table of contents

[Document revisions 1](#_Toc63035045)

[Summary 2](#_Toc63035046)

[1. Introduction 4](#_Toc63035047)

[2. Document structure 4](#_Toc63035048)

[3. Gaining knowledge 5](#_Toc63035049)

[3.1 Outline current situation 5](#_Toc63035050)

[3.2 Composites samples 6](#_Toc63035051)

[3.3 Determining requirements 6](#_Toc63035052)

[3.4 Basic system design 8](#_Toc63035053)

[3.5 Module choice 8](#_Toc63035054)

[4. Design for primary stakeholder 9](#_Toc63035055)

[4.1 Use case diagram 9](#_Toc63035056)

[4.2 Use case description’s 10](#_Toc63035057)

[4.3 UI Mockup 11](#_Toc63035058)

[5. Determining vision steps 12](#_Toc63035059)

[5.1 Acquisition 12](#_Toc63035060)

[5.2 Enhancement 13](#_Toc63035061)

[5.3 Segmentation 14](#_Toc63035062)

[5.4 Feature extraction 14](#_Toc63035063)

[5.5 Classification 15](#_Toc63035064)

[6. Requirements designs 16](#_Toc63035065)

[6.1 Requirement 1 16](#_Toc63035066)

[6.1.1 Sub-requirement 1 17](#_Toc63035067)

[6.1.2 Sub-requirement 2 18](#_Toc63035068)

[6.2 Requirement 2 19](#_Toc63035069)

[6.3 Requirement 3 & 6 20](#_Toc63035070)

[6.4 Requirement 5 21](#_Toc63035071)

[7. implemented results 22](#_Toc63035072)

[8. recommendations 22](#_Toc63035073)

[8.1 Better lighting and camera equipment. 22](#_Toc63035074)

[8.2 A better path algorithm 22](#_Toc63035075)

# 1. Introduction

Lightweight composites are becoming “the materials of choice for newly created airplanes for their high performance-to-weight ratio.” Since these materials are becoming more widely used, how does the aviation sector keep up when it comes to the repairs of these composites? Inholland Composites tries to answer this question with FIXAR, a 2-year research project that explores ways how these materials can be repaired in an economically viable way using automated technologies.

To answer this question, four work packages have been created for the envisioned improvements for composites repairs:

* Workpackage 1: benefits in automated repair solutions for (offshore) windfarms.
* Workpackage 2: repairability using thermoplastics.
* **Workpackage 3: sustainable composites repair for aerospace.**
* Workpackage 4: Mixed Reality for inspection and validation of repaired composites.

This assignment specifically focuses around Workpackage 3. The goal of this assignment is to create an automated system that is able to remove paint from composite materials. The shape of the material where paint needs to be removed is a radome shape. For now, however this assignment will only focuses on flat surfaces. In the future more work/research needs to be done to make this system work for a radome surface.

The aim of this assignment is to test if an automated system that can remove paint, is a viable solution to reduce labor cost.

# 2. Document structure

The structure of this document is based on the approach that is described in the plan of action. In here the project has been divided in multiple phases, with each phase having multiple results attached to them. These results are described here in there corresponding sub-heading with therein information on how this result was achieved, why these results are important, etc. Most of the scrum related elements have been left out of this document unless they are detrimental to the system design and implementation.

# 3. Gaining knowledge

## 3.1 Outline current situation

The system consists largely of three parts: the vision controller where the actual processing of images is done, the robot controller to control the robot movements and the UI for the human to machine interaction. The actual process flow looks like this:

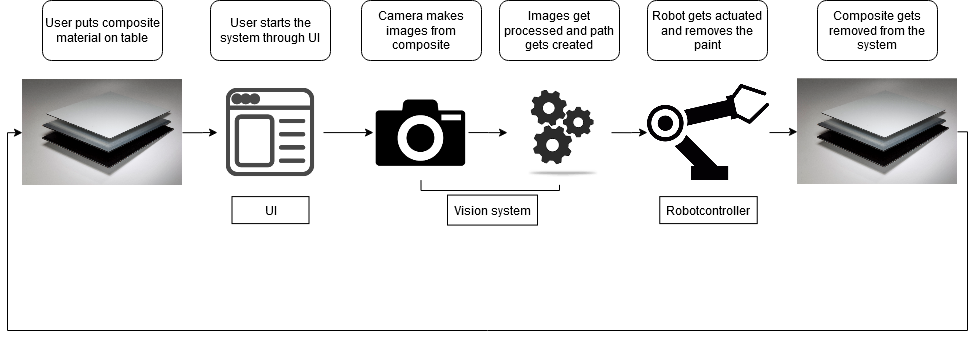


Figure 1: basic flow of system

The user needs to able to put a piece of composite onto the table. The user will then need to press the start button to start the system. When the system starts it will first check the object for paint and will determine where the paint is on the material. With this information the system will try to create a path that can be used by the robot controller to guide the arm along the created path and remove the paint from the object.

To make this system a reality the three parts have to be designed and created from scratch:

* A UI needs to be created to control the system.
* A vision algorithm in combination with the path algorithm to determine where the paint is located on the material and to create a path based on the detection.
* The robot controls to actuate the robot based on the path.

## 3.2 Composites samples

Afbeelding met tekst

Automatisch gegenereerde beschrijvingAfbeelding met tekst

Automatisch gegenereerde beschrijvingThe samples where needed to get a basic understanding what the composite looks like, what type of paint and color is on these materials and how these materials behave with different types of light. Knowing these characteristics, we can determine what the different vision steps will look like for this system. These vision steps are described in paragraph 5.

Figure 2: Object half covered.

Figure 3: Object not completely covered.

## Afbeelding met tekst, kist Automatisch gegenereerde beschrijvingAfbeelding met tekst, kist Automatisch gegenereerde beschrijving3.3 Determining requirements

Figure 4: Object completely covered.

Figure 5: Object not completely covered.

The following requirements were the result of a planned meeting with the client to get a better understanding of what the system should be able to do. Each requirement also has a few acceptance criteria that determine when a requirement is considered finished. They also have a priority attached to them based on the MoSCoW method.

Requirements are ordered in Must, Should, could, and would. Must requirements have to be completed to make the system function, Should requirements also need to be finished but are not required to make the system function, Could requirements are requirements that are only done when there is enough time left and would requirements are requirements that are explicitly refused by the developer(‘s) of the system.

1. As client I want the robot to be able to remove paint, so that the removal process can be automated. (Must)
   1. Software needs to be able to detect paint when it is there.
   2. The arm must be able to remove the paint. (not allowed to swerve more than 50mm)
2. As client I want to be able to manually confirm if the paint detection is accurate, so that possible mistakes can be prevented. (Must)
   1. Must be able to show the detection on screen.
   2. User must be able to accept or decline the detection with the user interface.
3. As client I want a start button, so that I can start the system. (Must)
   1. UI needs to have a start button.
   2. Start button can start the system.
4. As client I want to be able to see the status of the to be removed paint, so that I can determine how long the removal process is going to take. (Should)
   1. User interface must be able to show the status.
   2. Be able to show it in some kind of scale (either percental or time)
5. As client I want that the robot is able to continue where it left of after it was stopped, so that unnecessary time loss can be prevented. (Should)
   1. Be able to somewhere store the current status and location of the robot.
   2. Give the user the ability to continue the arm movement or to stop it entirely.
   3. Be able to show the status and location.
6. As client I want a stop button, so that I can stop the system when it is running. (Should)
   1. UI needs to have a stop button.
   2. Stop button can stop the system.
7. As client I want real-time information about the accuracy of the paint detection, so that I can clearly see if the vision part of the system performs adequately. (Should)
   1. Show accuracy on the UI.
   2. Use the result from requirement 3.
8. As client I want to control the robot manually, so that I can move the arm myself. (Could)
   1. Add interaction to UI.
   2. Show real-time position in UI.
9. As client I want a simulation for the 3d movement of the robot, so that I can determine more precisely what the robot is doing related to safety. (Could)
   1. Be able to see the 3d movement in the UI.

## 3.4 Basic system design

A basic system design was created in the form of state machine diagram. This diagram shows the different conditions the system can be in when its powered on. It also describes the different transitions between the different states. Since the system is made of different sub-systems that operate sequentially from each other with each their own logic, it is from a design perspective more coherent to create different states for each subsystem plus extra states for when the system is doing nothing and when it is paused. The following design was created from the requirements:

Diagram

Description automatically generatedThis design consists of five states: Detecting, Standing by, Removing, Idle and Paused. When the system is on but has not been started yet by the user it is in the Idle state. When the system has been started it will transition to the Detecting state and the system will check if there is paint on the object. When paint has been detected it will transition to the Standing by state otherwise it will return to Idle. Standing by acts as an in between state transition for the Detecting state and Removing state. In the Removing state the robot will remove the detected paint from the material based on the created path.

Figure 6: State machine

Both the Detecting, Standing by and Removing state are part of the composite state Running, Because these three states can exit at any point if pressed on the stop button. The system will then transition over to the paused state where the exited state is stored. When the user presses the resume button it will read the history to return to the original state it had exited from.

## 3.5 Module choice

The software is entirely written in python to make transitioning of the written software easier for people who do not have a lot of experience when it comes to programming/software development. The modules/libraries that are used in this project are there for also written in python.

* RoboDK for controlling the robot.
* PySide2 a python wrapper for the Qt library for the UI.
* OpenCV for the Vision algorithm.

# 4. Design for primary stakeholder

## 4.1 Use case diagram

The use case diagram shows the overall interaction between actor(‘s) and the system. In the case of this system there is only one actor that interacts with the system which is the user. The user can physically interact with the system through the UI either with touch or by looking at results that are displayed. These interactions are shown as ellipses called uses cases. In this design there are a few <<Extend>> and <<Include>> statements in this diagram. An <<extend>> stereotype indicates that the use case at the base of the arrow (the child) may include behavior specified by the use case at the tip of the arrow (the parent). The <<include>> stereotype indicates that the use case at the tip of the arrow (the child) will have its functionality included by the use case at the base of the arrow (the parent).

Diagram

Description automatically generated

Figure 7: Use case diagram.

Because the start of the system includes the removal of paint which include both the detection of paint and the actual sanding procedure, they are written here as an <<include>>. U can think of <<include>> as being a way of concretely describing a more abstract use case and the <<exclude>> as a way of describing a use case that happens after another use case has occurred.

## 4.2 Use case description’s

A use case description is an extension of the already specified use cases in the previous paragraph. Each use case has a description that describes the flow of steps each use case needs to undertake to complete its “task”. Each description consists of an Id, a Title, Description, the primary actor who uses the use case, preconditions, a trigger, The main flow, Postconditions and alternative flow. The basic structure and explanation of every field is described in Table 1.

Table 1: Use case description

|  |  |
| --- | --- |
| ID | A number that starts counting from 1 |
| Title | Title of use case |
| Description | A description of the use case |
| Actor | who or what uses this use case |
| Preconditions | The state the system must be in before the use case can start |
| Trigger | what this use case starts |
| Main | What the normal operation looks like for this use case |
| Postconditions | The state the system ends up in after the use case main flow has finished |
| Alternative flow | An alternative flow of this use case that differs from the main flow. These are usually considered errors or just alternative paths. |

The actual use case descriptions are attached to this document.

## 4.3 UI Mockup

Graphical user interface

Description automatically generatedA picture containing text

Description automatically generatedFigure x and x show the basic design of the UI that is used in this project. The design is created using the Qt designer and then built with python using Pyside2. The design consists of three main tabs, the home tab, simulation tab, and a manual control tab. The home tab consists of an image viewer, a few buttons to start and stop the system, camera controls to debug the camera and a few status bars to who the state of the system and the progress. The Simulation tab shows the simulation. The manual control tab shows the controls for the robot (there was no design created for this since it was at time unknown whether the robot had a pendant).

Figure 8: Simulation tab

Figure 9 Main tab

# 5. Determining vision steps

In the field of computer vision, more specifically machine vision, there are five main steps of processing a single image. These steps consist of Acquisition, Enhancement, Segmentation, Feature extraction and Classification. Acquisition is the act of acquisitioning an image and making sure that all image related variables like lighting and camera settings are set correctly. Enhancement is the act of enhancing images the images for example are particularly blurry. Selecting interesting parts in an image is called Segmentation. Extracting features like width and height from these segmentations or blobs is called feature extraction. Classifying these blobs with these features is classification.

## 5.1 Acquisition

The camera that is used for this project is a Logitech C920 Pro Webcam. This camera does not have a lot of changeable settings. Only the white balance and focus could be changed and even these options were only changeable with sliders not with direct values. Below are some test images that were taken with direct light applied to the object.A picture containing text, case

Description automatically generated

Figure 10: Camera image

Shape, arrow

Description automatically generatedThis type of material, especially with the paint, is very reflective. To make the material reflect as little light as possible, a different lighting setup was necessary. The type of lighting setup that is needed can be determined by the different aspects of the material like the previously mentioned reflectiveness but also the shape of the material. Knowing these aspects, a diffused lighting setup was chosen. This type of lighting diffuses the outgoing rays of light and distributes them evenly on the surface which causes the material to reflect less light. Since shape of the material used by this project is flat, a flat diffuse would have been sufficient.

Figure 11: Flat diffuse. (z.d.).

## 5.2 Enhancement

No actual Enhancement technique was needed for these images. The reason for this is that only paint is being detected. Since segmentation is going to be done primarily on color and not much on smaller objects on the material, no actual filtering is necessary although a smoothing filter is still applied to reduce the number of rough edges in the image. However, the color space is changed from RGB to HSV. HSV, which stands for Hue, Saturation and Value, makes it easier to pick colors from an image.

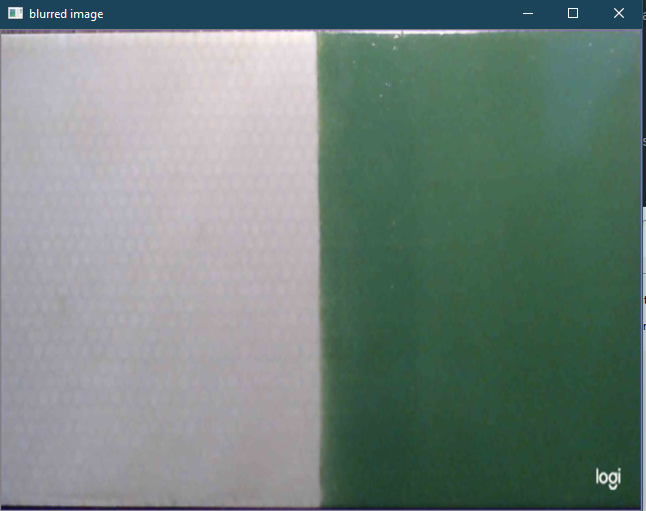


Figure 12: Blurred image.

## 5.3 Segmentation

The segmentation step is simple. In this case the system will look for what is and what is not paint. Since the paint color itself can change, the color of the material itself is used to separate the paint from the not paint. This is done by a predefined HSV value that has been tested with slider bars and has been hard coded into software. The result can be seen in figure 13 where the paint is colored black and the not paint white. An erosion is also used on top of the color extraction to reduce the number of smaller pixels from the image.

Shape

Description automatically generated with low confidence

Figure 13: Segmented image.

## 5.4 Feature extraction

Shape

Description automatically generated with low confidenceSince where only looking for whether something is paint or not, the segmentation step would have been sufficient. However, there are still some smaller blobs inside of the image. To make sure we only grab the much larger paint blobs, the system will also check every blob for its area. The area is the number of pixels the blobs consist of. We declare every blob that has an equal or larger area then the specified value, it is considered paint.

Figure 14: feature extracted image.

## 5.5 Classification

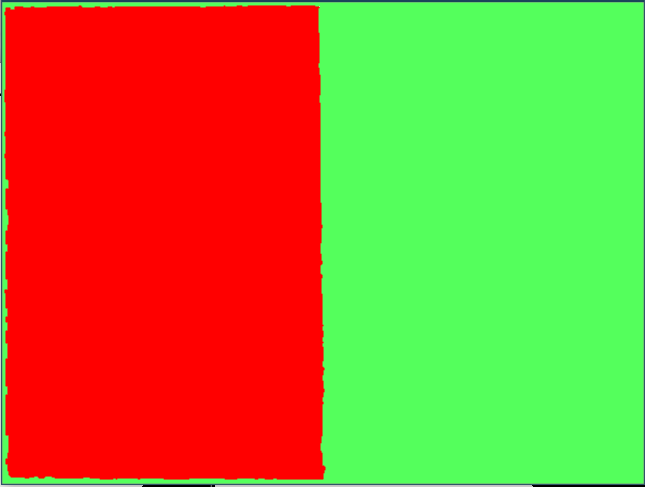
There is not much to classify. It Is either paint or not. Classification has technically already been done in the feature extraction step.

Figure 15: Classified image. Red is material green is paint.

# 6. Requirements designs

This chapter has all the different software designs that have been made during the development process. Just like chapter 4, these designs consist of diagrams specified in UML. In this case these are Class diagrams and sequence diagrams. Not every requirement has a finished design. Some requirements were either not completed because the were deemed unnecessary during the actual development phase or did not have enough time to be completed. These requirements include 4, 7, 8 and 9.

## 6.1 Requirement 1

As client I want the robot to be able to remove paint, so that the removal process can be automated.

Since this requirement is too large to fit into one diagram, it has been split up into two smaller ones.

1. As client I want that the system is able to detect paint, so that the system can remove it during die sanding procedure.
2. As client I want that the robot is able to remove paint, so that this can be automated.

These two sub-requirements both have their own class and sequence diagram. They are also functionally different so splitting them up makes it clearer what each sub-requirement does.

### 6.1.1 Sub-requirement 1

Figure 16 shows this sub-requirement’s class diagram and figure 17 the sequence diagram. Here the MainWindow has an RobotVisionModule object that it calls when it has successfully acquired an image. PySide2 handles the acquirement of images through its event system. When an image has been made, it will be stored in an Image object. The Image class acts as an intermediary between the different image types of PySide2 and OpenCV. The image converter can convert the Image object to either an PySide2 image or OpenCV image.

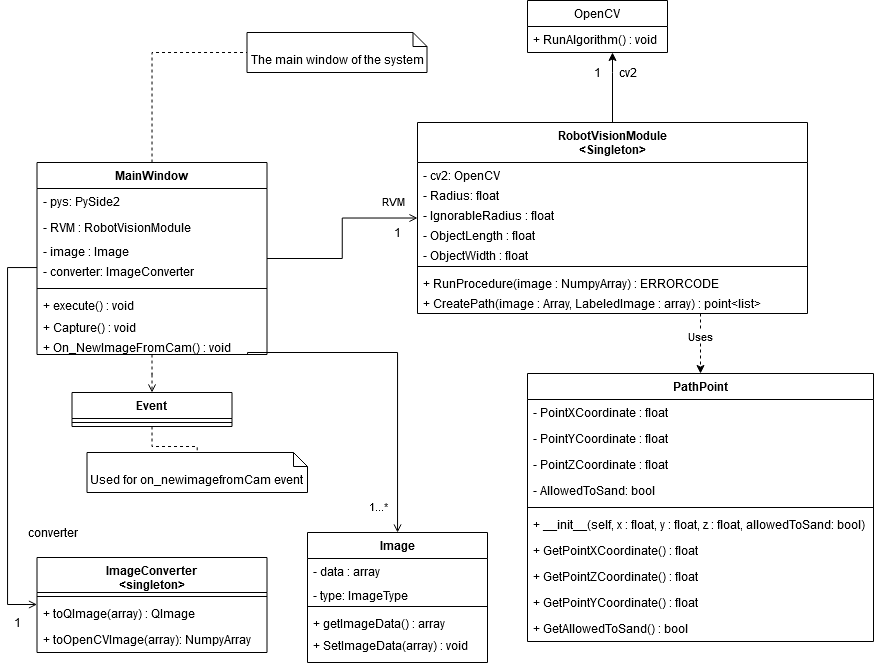
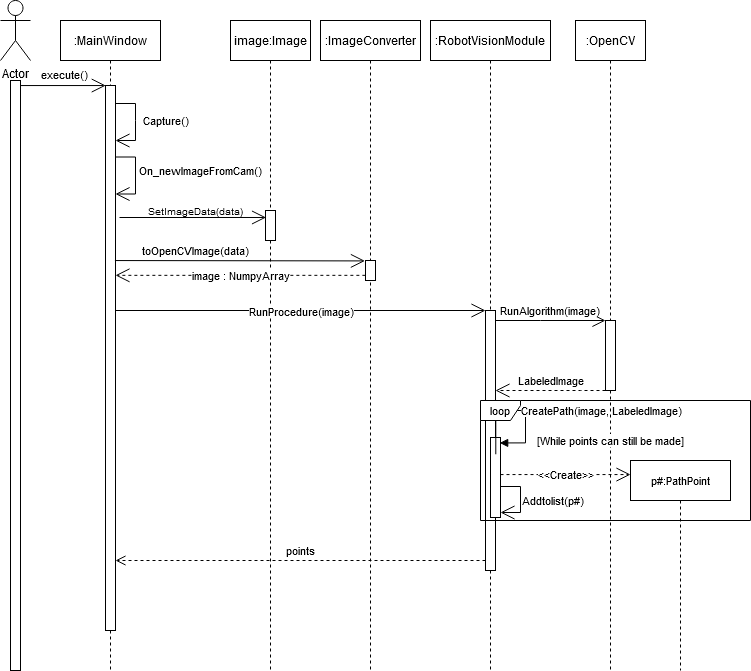
When the image has been converted, the MainWindow will give the image to the RobotVisionModule object and runs its procedure. In the procedure the RVM will first run the vision algorithm and then create a list of points with the PathPoint class and returns this list to the MainWindow.

Figure 16: Sequence diagram

Figure 17: Class diagram

### 6.1.2 Sub-requirement 2

Figure 18 shows the class diagram and figure 19 the sequence diagram of this sub-requirement. In the sequence diagram it shows a :Vision object. This object is only there to show that this sub-requirement follows the previous one. Here the MainWindow calls the RunProcedure Method on the RobotController object with the points list. The robot controller will iterate through this list and move the robot to the desired points.

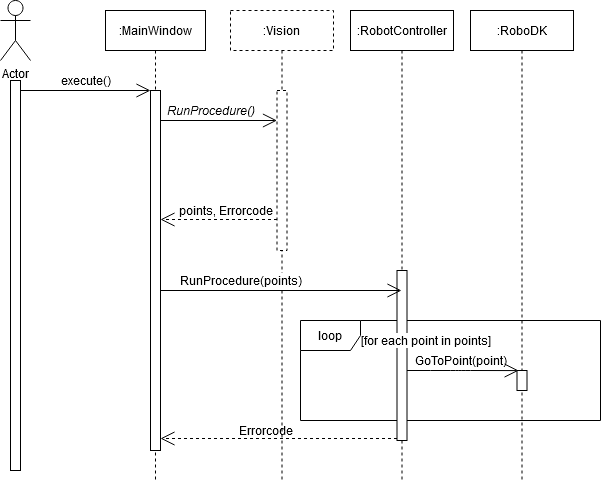
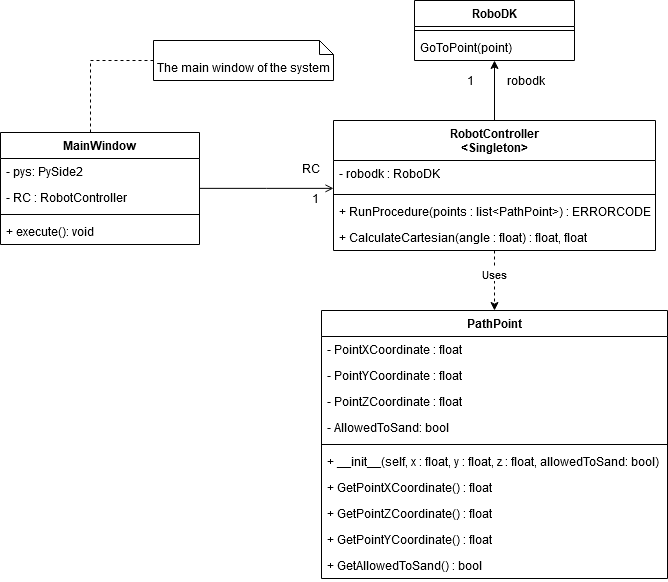


Figure 18: Class diagram

Figure 19: Sequence diagram

## 6.2 Requirement 2

*As client I want to be able to manually confirm if the paint detection is accurate, so that possible mistakes can be prevented.* When the Vision algorithm has finished, the MainWindow will call the ConfirmationWindow where the result is shown. The user then has the option to either except or deny the image.

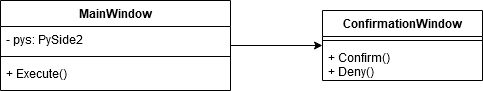
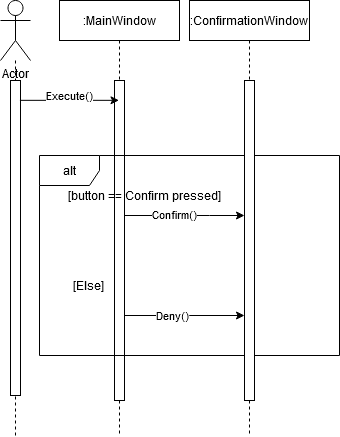


Figure 21: Sequence diagram

Figure 20: Class diagram

## 6.3 Requirement 3 & 6

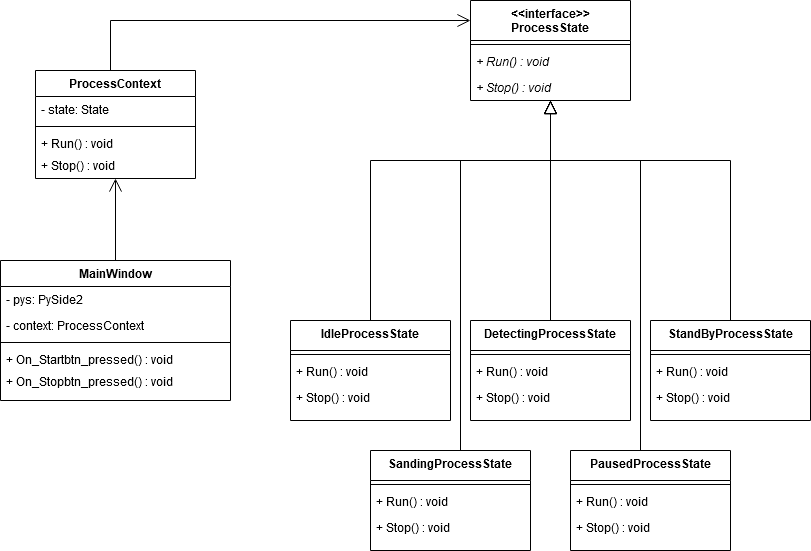
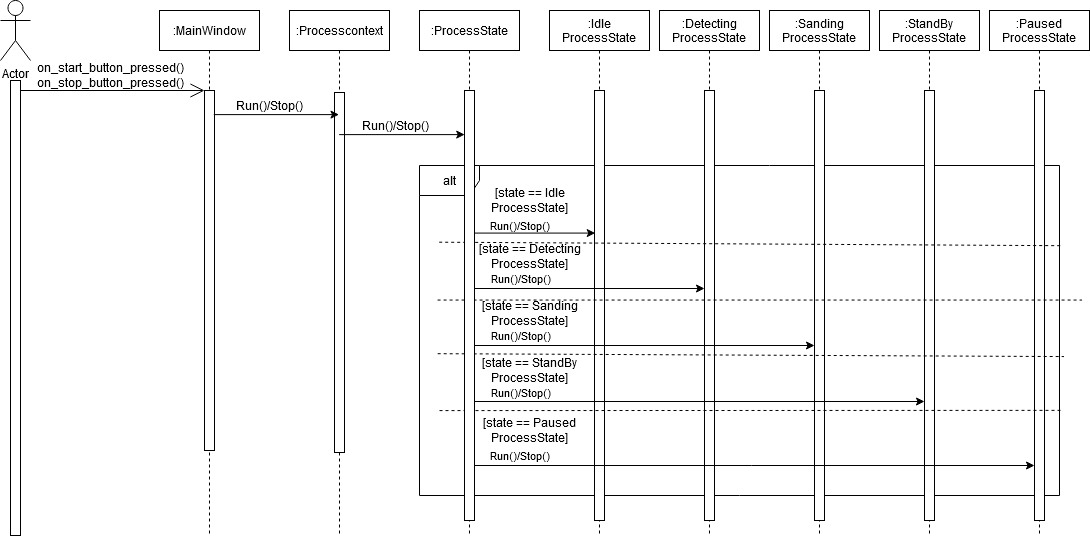
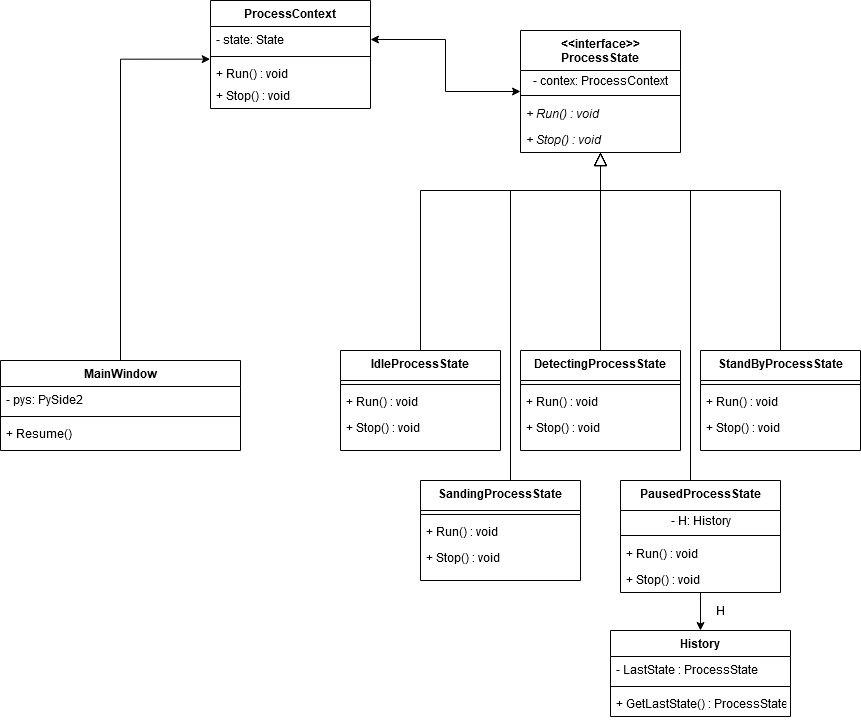
As client I want a start button, so that I can start the system and as client I want a stop button, so that I can stop the system when it is running both share the same design. MainWindow will call the processContext which contains the state the system is currently running in. The state can be any of the following states below ProcessState. This design is based upon the state design pattern.

Figure 22: class diagram

Figure 23: Sequence diagram

## 6.4 Requirement 5

As client I want that the robot is able to continue where it left of after it was stopped, so that unnecessary time loss can be prevented. It is assumed that the system is in the PausedProcessState when it was paused the user resumes the procedure and the PausedProcessState loads the original state and transitions to that state.

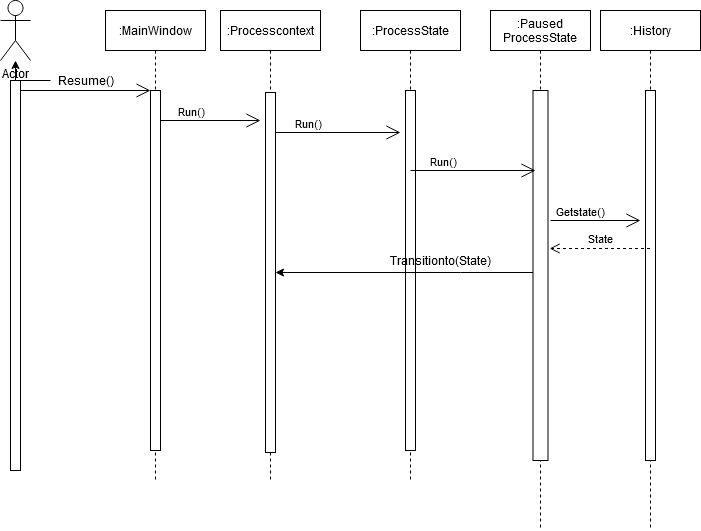


Figure 25: Sequence diagram

Figure 24: Class diagram

# 7. implemented results

Of the 9 requirements listed above the following have been completed.

1. As client I want the robot to be able to remove paint, so that the removal process can be automated. (Must)
2. As client I want to be able to manually confirm if the paint detection is accurate, so that possible mistakes can be prevented. (Must)
3. As client I want a start button, so that I can start the system. (Must)

These requirements have been tested.

The vision algorithm uses the described steps in chapter 5. It will then use this result to create a path that can be used by the robot controller. The path is created by moving a ellipse shaped kernel over the labeled image to determine whether a certain path is allowed or not. This creates a list of point that is used by the robotcontroller to move the arm over the object.

# 8. recommendations

## 8.1 Better lighting and camera equipment.

A higher resolution camera is needed to make a more exact distance calculation of the created path. A better lighting setup could be used to remove the extra glare that is visible in the object.

## 8.2 A better path algorithm

The path that is created now is very simple and doesn’t make the most optimal path to remove paint. It’s a simple block wave. This type of movement will miss some spots as it will ignore paint spots that are closer to non-paint spots. A more optimal path would be something that can move between different non-painted spots to remove the paint more accurately.

# bibliography

*Flat diffuse*. (z.d.). [Foto]. A Practical Guide to Machine Vision Lighting. https://www.ni.com/nl-nl/innovations/white-papers/12/a-practical-guide-to-machine-vision-lighting.html