HOCHSCHULE
HANNOVER
UNIVERSITY OF
APPLIED
SCIENCES
AND ARTS

Generating 3D Models of Glass Panels

Santeri Stigell

Bachelor's thesis
August 2021

Mechanical Engineering
Machine Automation

TIIVISTELMÄ

Hochschule Hannover
Hannover University of Applied Sciences and Arts
Konetekniikka
Koneautomaatio

SANTERI STIGELL:
Generating 3D Models of Glass Panels

Opinnäytetyö 48 sivua
Elokuu 2021

3D-mallien luominen läpinäkyvistä esineistä on merkittävä aihe mallien käytännön sovellusten vuoksi. Tässä opinnäytetyössä 3D-malleja käytettäisiin lasipaneelien pakkaamiseen. Alan aiempi tutkimus on myös melko vähäistä. Opinnäytetyössä esitellään ja tutkitaan lukuisia menetelmiä lasilevyjen 2D- ja 3D-mallien luomiseen.

Menetelmä, joka osoittautui toimivimmaksi 2D-reunamallien luomisessa, oli tietokonenäön käyttäminen ja kuvien käsittely Pythonilla yhdessä OpenCV -tietokonenäkökirjaston kanssa. 2D-reunamallien luominen onnistui pääasiassa muokkaamalla kuvia HSV:llä ja käyttämällä sitten Cannyn reunantunnistusalgoritmia muokattuihin kuviin. Nämä reunamallit muutettiin sitten STL-tiedostomuotoon antamalla niille kuvitteellinen z-koordinaatti matriisilaskennan avulla, jotta ne voitiin tuoda 3D-malleina esimerkiksi robotille.

Lasipaneelien todellisen 3D-mallin luomista tutkittiin analysoimalla valoa, käyttämällä röntgensäteentunnistusta, käyttämällä stereonäköä dispariteettikartan luomiseksi ja tarkistamalla z-koordinaatit robotin sekä lineaarisen asentoanturin avulla. Valokuvioiden analysointi, röntgensäteiden havaitseminen ja z-koordinaattien robottitarkastus ovat vaihtoehtoja, jotka ovat lupaavimpia kehitettäväksi lasin 3D-tunnistusjärjestelmäksi.

Hankkeen budjetti- ja aikarajoitusten vuoksi täydellistä 3D-mallia ei saatu aikaiseksi. Jatkoa ajatellen, T. Yamanaka, F. Sakaue and J. Sato: “Adaptive Image Projection onto Non-planar Screen Using Projector-Camera Systems” -tutkielma on arvokas resurssi valokuvioiden analysoinnin rakentamiseen. OptiTrackin Motive vaikuttaa myös hyvältä järjestelmältä 3D-mallien luomiseen.

Asiasanat: tietokonenäkö, 3D-malli, robotiikka, ohjelmointi, Cognex, röntgensäteily, lasilevy, esineen havaitseminen, Python

ABSTRACT

Hochschule Hannover
Hannover University of Applied Sciences and Arts
Mechanical Engineering
Machine Automation

SANTERI STIGELL:
Generating 3D Models of Glass Panels

Bachelor's thesis 48 pages
August 2021

Creating 3D models out of transparent objects is a significant subject due to the practical applications of the models. In this project, the purpose was to create 3D models for packaging glass panels. Previous research in the field is rather limited. In this thesis the numerous methods of creating 2D and 3D models of glass panels are introduced and explored.

The method that proved to work the best for creating 2D edge models was using computer vision and processing the images with Python in conjunction with the OpenCV computer vision library. Creating the 2D edge models was mainly achieved by modifying the images in HSV and then using a Canny edge detection algorithm on the modified images. These edge models were then turned into the STL file format by giving them a fictional z-coordinate by using matrix calculations to be able to import them as 3D models to, for example, a robot.

Creating the real 3D model of the glass panels was explored by analysing light, using X-ray detection, using stereovision to create a disparity map and by checking the z-coordinates using a robot and a linear position sensor. The options that have the most potential to be fully developed into a 3D detection system for glass are analysing light patterns, detecting X-rays and robotically checking the z-coordinates.

Due to budgetary and time constraints of the project, a complete 3D model was not achieved. For further research, the paper by T. Yamanaka, F. Sakaue and J. Sato: “Adaptive Image Projection onto Non-planar Screen Using Projector-Camera Systems” is suggested as an in-depth source on light patterns. Motive by OptiTrack is also promising for mapping contours in 3D spaces.

Key words: computer vision, 3D-model, robotics, programming, Cognex, X-ray, glass panels, object detection, Python

CONTENTS

ABBREVIATIONS

2D	Two-dimensional space
3D	Three-dimensional space
IR	Infrared, electromagnetic radiation around 700nm to 1mm in range
UV	Ultraviolet, electromagnetic radiation around 10nm to 400nm in range
PC	Personal computer
SSH	Secure Shell, a cryptographic network protocol, used to provide a secure channel in an unsecure network
HSV	Hue, saturation and value, used in colour-space changes

1 INTRODUCTION

The goal of this thesis is to research the ability to create 3D models of various shaped and sized glass panels. The project is done for Müller Industrietechnik GmbH, which is a company that has around 50 employees and that specialises in cabin components of a driver for aggregation, building, and municipal machinery. A sizeable part of the business of the company is selling glass panels for the cabins of different machineries.

Müller Industrietechnik GmbH has over 1000 different sized and shaped glass panels to sell and to eventually package to send to the buyers. Currently the glass panels are packaged manually by the storage workers.

The manual packaging process is a tedious and time-consuming job, and it is very taxing on the packaging workers wrists. The process is very repetitive in nature and could be automated using a robot. In addition to helping with the storage workers workplace wellbeing, a robot would free them to complete other tasks while the robot packs the panels. For the robot to package the panels, it first needs to have some outlines to follow. Creating these outlines is the goal of this thesis.

2 PROJECT DESCRIPTION

The glass panels to be packaged come in various shapes and sizes. Below are some examples of glass panels for John Deere series 3100 tractors (see figure 1).

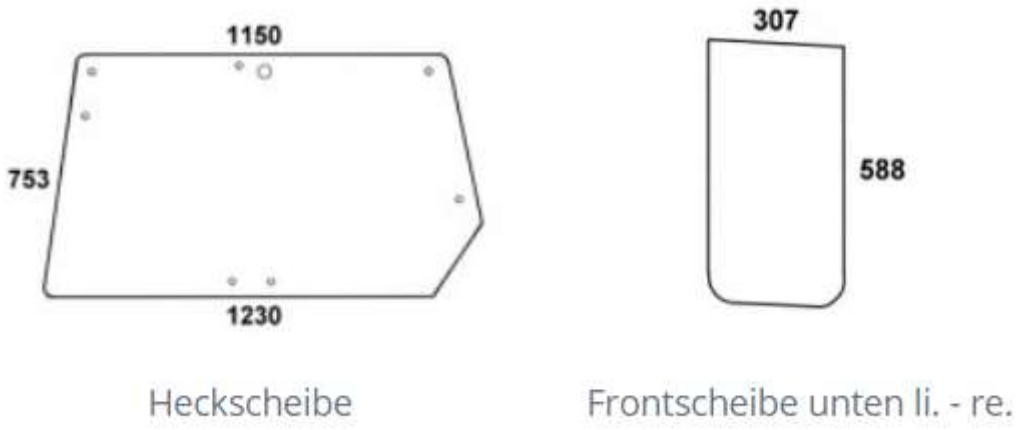

Figure 1. Glass panel examples. [1]

The glass panels have a thickness of 4–8 mm and noticeable 3D contours that can be seen in the picture below (see figure 2). The job of the packaging robot would be to apply the blue edge protecting packaging material to the sides of the glass panels. To do that, it needs a 3D image of the panels. Therefore, the goal of this project is to create 3D images of the glass panels.

Figure 2. Example of a packaged glass panel.

2.1 Difficulties of the project

The biggest challenge of the project is that there are over a thousand different models of glass panels. This means that the system must be very flexible, and it must be a universal solution instead of tackling a single type of glass panels.

Another difficulty point is the fact that glass is translucent. This makes certain solutions, like machine vision, difficult to implement and to make reliable in the process. However, some of the glass panels have a slight green tint on them, which makes them easier to work with.

This project also has a budget of a few thousand euros, which limits some of the equipment choices. For example, an X-ray detection system (see page 10) was not tested in this project because of budget limitations.

3 DETECTING GLASS PANELS IN 2D

Detecting glass is not an easy task. The nature of the settings of this project and the transparency of the glass panels varies from a slight green tint to clear. The main ways to detect glass are photoelectric sensors, X-ray scanners and machine vision. In the following chapters, their pros and cons will be discussed to find the best one for this project.

3.1 Photoelectric sensors

Photoelectric sensors detect differences in light intensity to perceive the presence, absence, or distance of target objects. Photoelectric sensors are divided into three types. First, thru-beam sensors have a separate transmitter and a receiver. Second, retroreflective sensors have a transmitter and a receiver on the same unit and a reflector which directs the light back to the receiver. Both the thru-beam and the retroreflective sensors work by detecting objects that interrupt the light coming to the transmitter. The third type of sensors is a diffuse reflective sensor. It works similarly to the retroreflective sensor, but instead of having a reflector that directs the light back at the receiver, it requires an object to do so. [2] The types of photoelectric sensors are illustrated on figure 3 below.

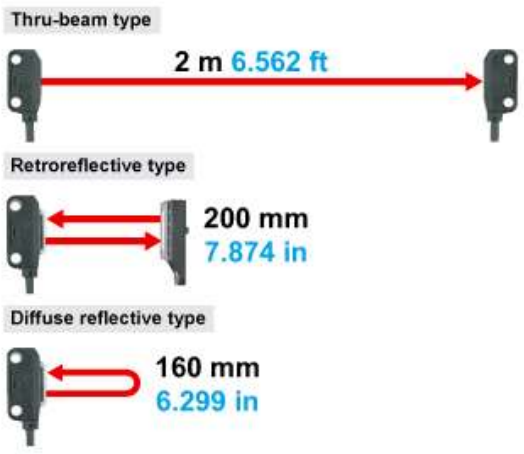

Figure 3. Photoelectric sensor types. [3]

Of the three types of photoelectric sensors, the one best suited for detecting glass is the retroreflective sensor [4]. While the retroreflective sensors do come in light curtain configurations, they lack the second dimension of detection. Furthermore, they only detect the presence, absence or distance to an object and thus are not suitable for this kind of application. In other words, none of the options were used in this project.

3.2 X-ray detection for 2D

X-ray detection relies on the penetrating attributes of high-energy electromagnetic radiation. X-ray machines function on a relatively similar concept as the thru-beam photoelectric sensor mentioned earlier, requiring a transmitter and a receiver but on a much higher energy level. A big difference in contrast to the photoelectric thru-beam sensor is the fact that the X-ray can create 2D visualizations from objects. [5] Figure 4 below models a simplified X-ray imaging system.

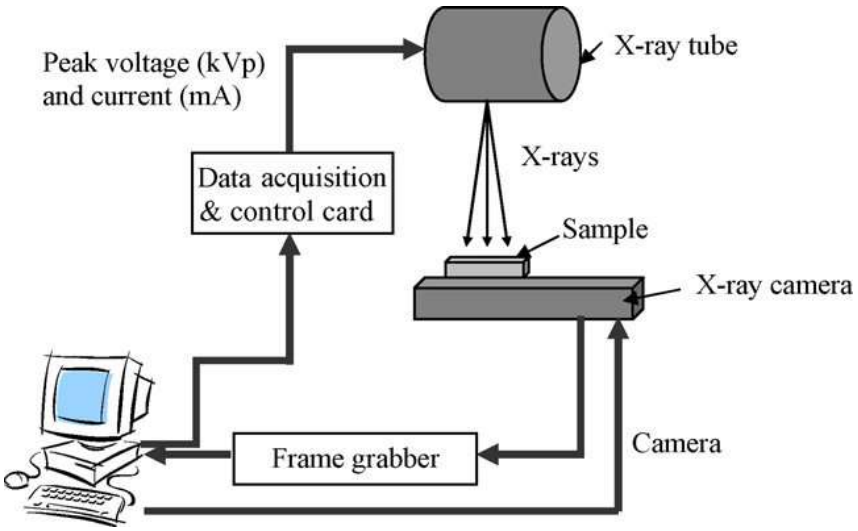

Figure 4. Schematic of an X-ray imaging system. [6]

The most ordinary types of X-ray detectors being used are medical X-ray machines and airport security X-ray machines. In industry, X-rays are also being used to find for example glass and other materials inside of food objects [7] or to find cracks or flaws in materials [8]. Figure 5 below shows a picture of a Nikon X-ray Inspection System and a picture taken with the system.

Figure 5. Nikon X-ray Inspection System and an X-ray of an engine. [9]

X-rays would be a very reliable option for the detection of glass panels in this project because of its lack of reliance on material transparency. However, X-ray detection is, relatively to other methods listed here, very costly, and would also require high voltage power supplies to generate X-rays [7]. For these reasons, X-ray detection was also not tested for this project. The theory of the subject will, however, be further explored in chapter 6.2, due to its possible use of creating 3D models.

3.3 Machine vision

The third option for detecting glass panels is machine vision. Machine vision is a flexible technology which allows machines to perceive the world using one or more cameras. It is effective at, for example, identifying objects, pattern matching, and guiding robots [10] Machine vision is a subset of computer vision that is used in industrial applications, like this one. This means that all machine vision is computer vision, but not all computer vision is machine vision, so the names can be used interchangeably in this context. [11]

The greatest difficulty while working with machine vision is that the glass panels are translucent, or even completely transparent. Clear objects are not as easy to pick up as solid objects with a clear contrast to their surroundings. Another problem is lighting. Ambient lighting required to get a good image of the glass panels might also create their own reflections which might interfere with the machine vision algorithms.

Machine vision is, however, a relatively cheap option, since it only requires a camera – a basic webcam is enough to get started. Also, most of the glass panels have a slight green tint, which creates a sufficient contrast to be seen by a camera. The edges of the glasses also have a slight colouration, which might be picked up by the cameras (see figure 6). For these reasons, machine vision is the best place to start testing.

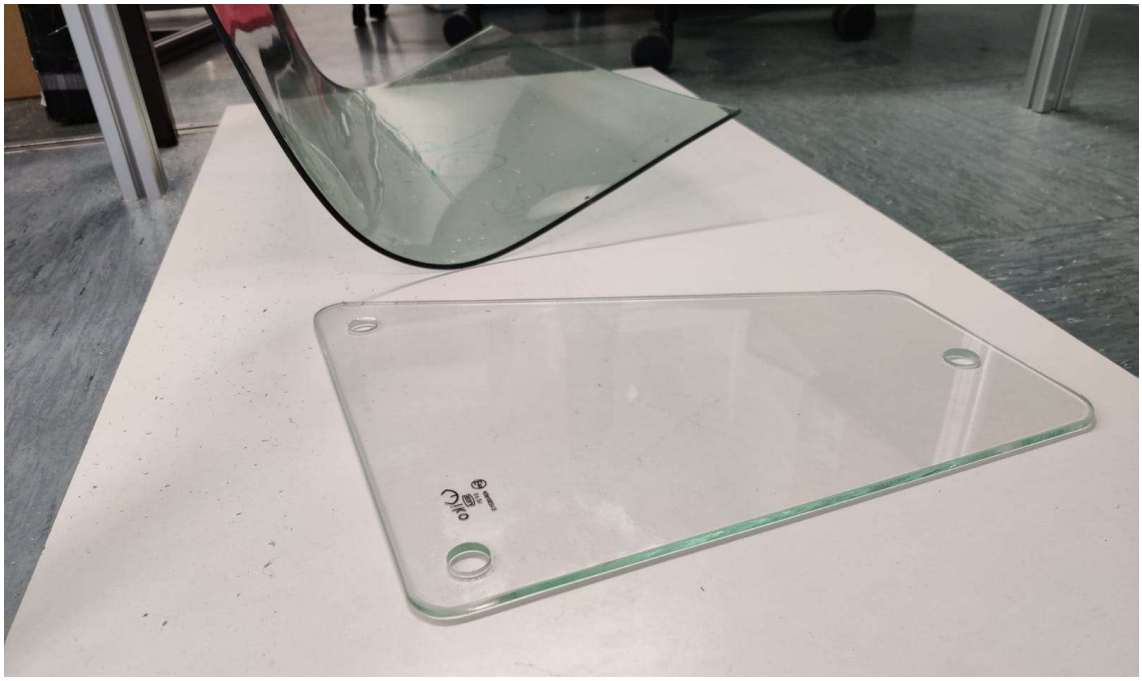

Figure 6. Colouration of the glass panels.

4 TESTING MACHINE VISION METHODS

Based on the reasons listed before, it is concluded that machine vision is the best option to begin testing to find the 2D shapes of the glass panels. In this chapter, the different options for 2D machine vision will be introduced and the testing process explained.

4.1 In-Sight Explorer by Cognex

In Sight Explorer by Cognex is an easy way of getting into machine vision. It requires no previous programming knowledge, which makes it a very powerful tool for machine vision applications using predetermined objects. For example, part counting operations, presence/absence operations and simple math and logic operations are very easy to use with In-Sight Explorer [12]. On the left in figure 7 is the basic configuration of the software, which requires a feature to locate the part to be inspected before inspection tools can be used. More part inspection operations can be seen on figure 8 on the right.

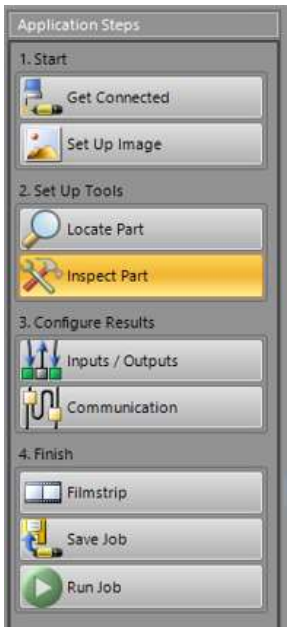

Figure 8. Basic configuration of the software.

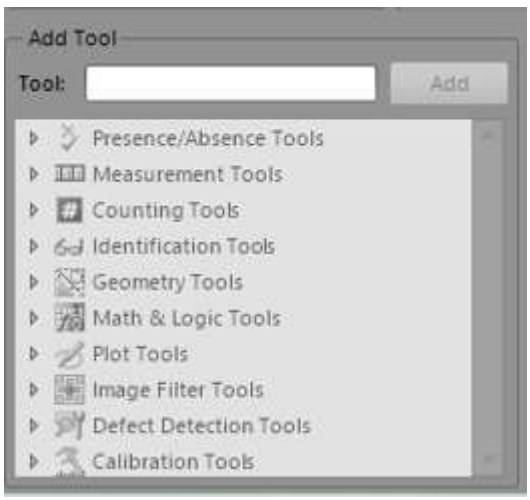

Figure 7. More part inspection tools.

4.1.1 Cognex camera setup

The setup for the Cognex system testing was quite simple. A Cognex camera was installed above a homogenously coloured surface using a truss system originally meant for stage lighting. This allowed for quick adjustments of the position of the camera. The surface chosen was white to allow for the best contrast between the glass panels and the surface.

The Cognex cameras that were available for the project were the 5100 and the 5403 models (see figure 9). The cameras are functionally very similar, but with different sensors. Both the models can record 8-bit greyscale images, but the image resolution on the 5100 is only 640 x 480 pixels with a 5,92 mm sensor, versus the 1600 x 1200 pixels with an 8,8 mm sensor of the 5403. The 5100 is capable of a maximum of 60 frames per second, while the 5403 reaches only a maximum of 14 frames per second. [13] Because the images taken with the 5403 were in a much higher resolution and wider field of view, the 5403 was chosen to conduct the tests.

Figure 9. Cognex 5403 machine vision camera. [14]

4.1.2 Testing In-Sight Explorer

As mentioned before, programming a machine vision solution in In-Sight Explorer starts with locating the part (see figure 10). The corner of the part was chosen over, for example, a straight edge, because a corner is a constant feature on all the glass panels and can be used to orient the inspection tools. Choosing the corner is better as opposed to choosing a straight line. A line can be found on either side of the corner, thus giving the inspection tools also two possible orientations. Having two different orientations would increase both the programming effort and the program execution time. Also, because the goal of this section is to create a 2D model of the edges of the panels, the corners would later have to be recognised anyway.

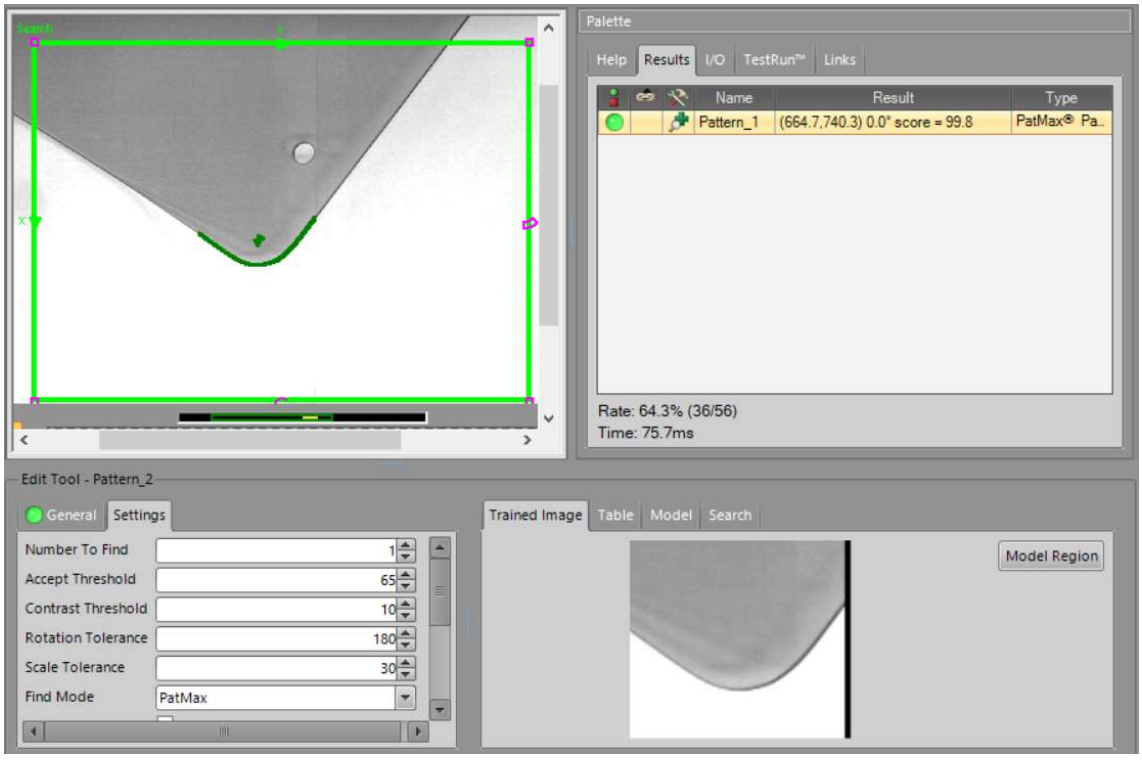

Figure 10. Locating the part in In-Sight-Explorer.

The pattern location tool functions by the user training models of the features to the program. The program then compares the trained model to the features of the image. The program then finds the closest matching feature to the model and gives it a score on how close they match with each other. The accept threshold of the location tool can then be modified to suit the individual case.

Generally, it is preferable to keep the accept threshold as high as possible, while still finding the wanted features. Because the corners come in various shapes and sizes in this project, the accept threshold was kept as low as possible while not picking up unwanted features.

After locating the part, various inspection tools (see figure 8) can be used. Most notably, edge and plot tools were used to find and draw the edges of the glass panels. The inspection tools must be linked to a location tool which gives them an orientation. The part inspection tools have a quite poor rotation tolerance setting. It means that the initial location tool must be very close to on-point. Otherwise, the inspection tools either fail or the inspection tools are found in unwanted positions. The latter case can be seen in figure 11, where the left edge tool is off point.

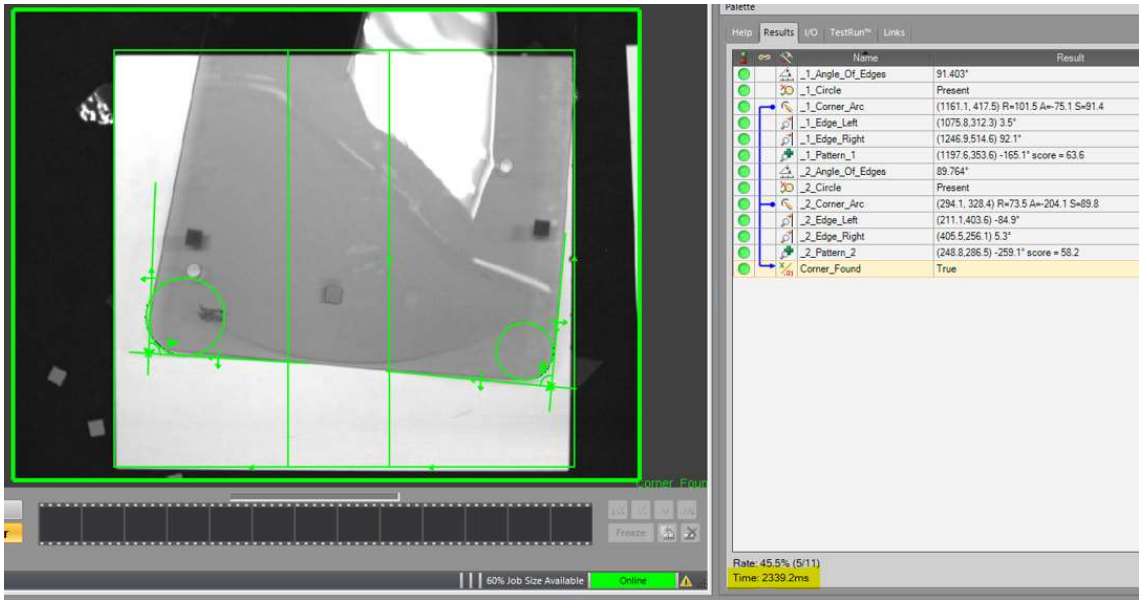

Figure 11. Detecting two corners and drawing lines.

4.1.3 Problems of the In-Sight system

The biggest problem of the In-Sight system proved to be the part location tool. Each part location tool can be taught in 10 different models, and because there are numerous different shaped and sized corners, not all corners from all the

glass panel models would be recognised. There is an option of adding more location tools, which would allow 10 more models to be taught per location tool, but this teaching process is a difficult and very time-consuming job.

Having additional location tools would also require the inspection tools to be copied to each location tool, which significantly increases the programs processing time. In figure 11 it is shown that while only detecting two corners with two different part location tools, the processing time is already over two seconds. At least two more location tools would have to be added to detect all four corners, which would already make the processing time 4,5 seconds. This is due to the program being processed in the camera itself, which does not have good processing capabilities.

However, these problems could possibly be fixed. The problem of having to teach multiple corners can be mediated by using a smart camera that Cognex also provides. The smart camera is designed for deep learning, which could be used to teach various corners manually, and then the program would learn what a corner is and detect the rest on its own. [15] Unfortunately, due to its costs, this was not available for this project. The processing time problem could be fixed, or at least made significantly better, if a computer is used to process the program.

In short, where a regular, non-deep-learning In-Sight Explorer system fails, is flexibility. As stated earlier, In-Sight Explorer works best when using predetermined objects, but when working with objects of unknown shapes and sizes, it is not the best choice.

4.2 Python and OpenCV

The Cognex In-Sight Explorer machine vision approach seemed to function as a concept, but it lacked flexibility. The solution to this is to create a machine vision program out of scratch. C++, MATLAB and Python are the best programming languages for machine vision. Of these three, Python was chosen, because it is

especially easy to learn for beginners. Also, it is the most used programming language, so it has a lot of ready-made resources online as well as powerful tools for visualizing. [16]

Python is a high-level programming language, meaning that it is a programming language with a big emphasis on readability. It also automates or even hides completely some areas of computing, making the development of a program simpler than a low-level programming language. Python is also a modular programming language, which means that there are numerous libraries that can be added to the core programming language to make it complete additional tasks. [17]

The most important one of these libraries for this project is the computer vision library called OpenCV. OpenCV, or Open Source Computer Vision Library, includes several hundreds of computer vision algorithms ranging from image processing to 3D reconstruction [18]. With OpenCV, a photo of a glass panel can be processed to an output image with 2D information that the robot can read.

4.2.1 Raspberry Pi setup

To take the photos of the glass panels, an independent Raspberry Pi system with a camera was used in a “headless” configuration. Raspberry Pi is, in short, a 86 mm long and 57 mm wide computer that runs on a Linux based operating system which makes it like any other PC running Linux, albeit with less processing power [19]. Running the Pi headless means that the Pi is not connected to a keyboard, mouse or a monitor but instead accessed using another computer through SSH. SSH is a protocol that allows connecting to another computer remotely. In practice, SSH opens up a window on the controlling computer, which shows the screen of the controlled computer.

Running the Pi headless through SSH means that the camera and glass panel setup can be separate from the computer that is running the image processing code. The reason for the image processing was done on a computer other than the Pi was simply to cut down on the testing time because of the tiny size of the Pi that makes it very slow at processing complex image processing tasks.

The camera that was used on the Pi was the Pi NOIR camera which has no infrared blocking filter. The camera can see both infrared and ultraviolet light instead of just the normal visible spectrum of a light. In normal daytime imaging this means, however, that photos taken with the camera seem washed out (see figure 12). [20] A regular Pi camera with an IR-filter could have worked as well, but the Pi NOIR gave the added option of using infrared light to, for example, sense heat.

Figure 12, Pi NOIR vs RasPicam comparison [20]

The camera was installed in a stable position above a white surface to create as much contrast as possible between the glass panels and the surface. As seen in figure 13, the camera is attached to the middle of the wooden panel facing the glass panel.

Figure 13. Setup for the Pi NOIR camera.

4.2.2 Image processing

The Python coding process was started with no previous coding experience and was done mostly by following the guides from other people and then modifying them to fit this project. The references are marked in the code according to which guide was used as a base to build up on.

The code starts with changing the normal coloured image to HSV. HSV is an abbreviation of hue, saturation, and value. Hue can be adjusted to modify the colour scheme, and to block out the unwanted ones, saturation can be adjusted to find the right strength of hue and value to change the brightness range (see figure 14) [21].

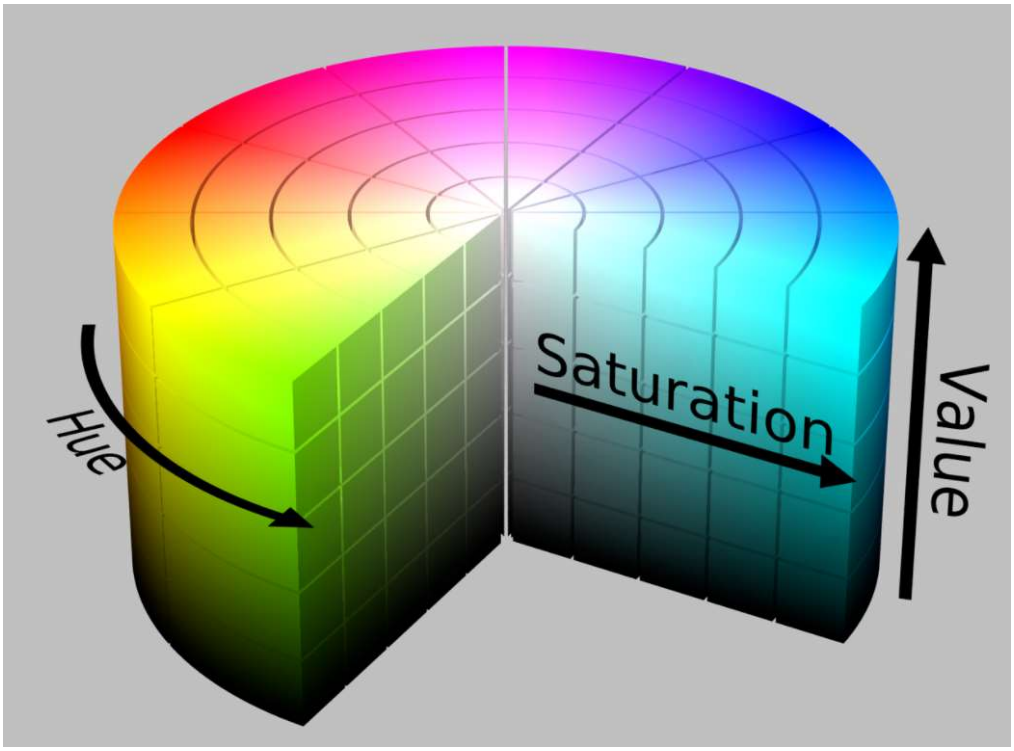

Figure 14. Hue, saturation, and value demonstrated. [22]

A “while” loop is used to keep the code repeating until the break key “q” is pressed (see figure 15). This in conjunction with trackbars means that the image can be adjusted “live”. The changes made to the locations of the trackbars can be seen instantly on the image, which makes finding the right parameters easier.

```
# https://sodocumentation.net/opencv/topic/6099/edge-detection
while True:
    frame = cv2.imread(img)
    imgHSV = cv2.cvtColor(frame,cv2.COLOR_BGR2HSV)
    h_min = cv2.getTrackbarPos("Hue Min","TrackBar for edges")
    h_max = cv2.getTrackbarPos("Hue Max","TrackBar for edges")
    s_min = cv2.getTrackbarPos("Sat Min","TrackBar for edges")
    s_max = cv2.getTrackbarPos("Sat Max","TrackBar for edges")
    v_min = cv2.getTrackbarPos("Value Min","TrackBar for edges")
    v_max = cv2.getTrackbarPos("Value Max","TrackBar for edges")
    #the trackbars modifies the mask
    lower = np.array([h_min,s_min,v_min])
    upper = np.array([h_max,s_max,v_max])
    mask = cv2.inRange(imgHSV,lower,upper)
    #this is used to colour the mask with the original images colours
    #otherwise it would be black and white
    imgResult = cv2.bitwise_and(frame,frame,mask=mask)
    #shows the image
    imgResized = cv2.resize(imgResult, (960, 540))
    cv2.imshow("Edge image to be saved",imgResized)
    cv2.waitKey(1)
    #press q for a second to save the changes
    if cv2.waitKey(1) & 0xFF ==ord('q'):
        break
```

Figure 15. Code for HSV processing. [23]

The original image, that is taken with the Pi NOIR camera before any image processing, can be seen in figure 16. In figure 17, hue has been changed to a point, where most light is blocked, but the edges are still visible.

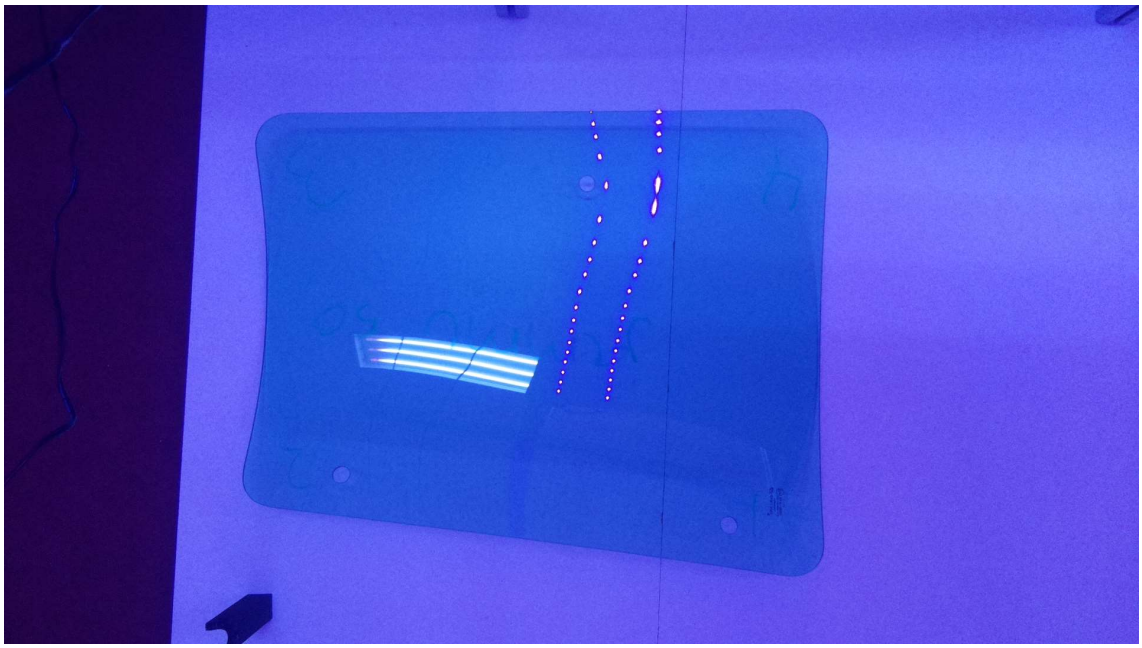

Figure 16.Original image taken using Pi NOIR.

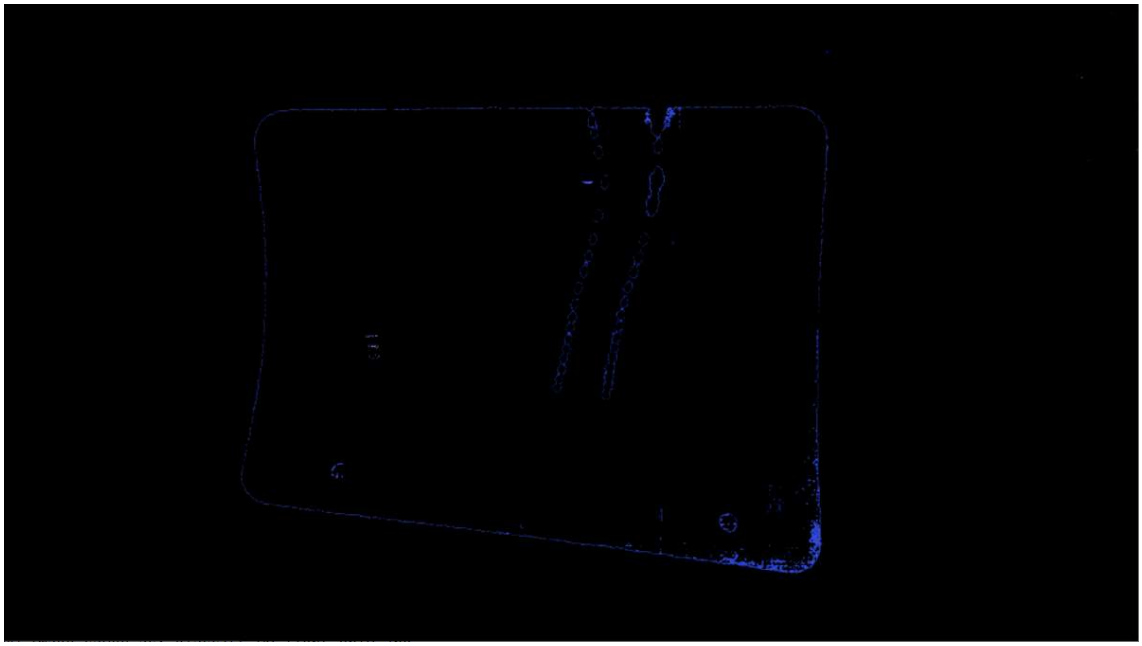

Figure 17. Image after hue adjustments.

Next, the saturation can be modified to make the effect of hue a little more powerful. To finish the HSV processing, value can be adjusted to make the light specs disappear completely. After this the image is saved by pressing “q”.

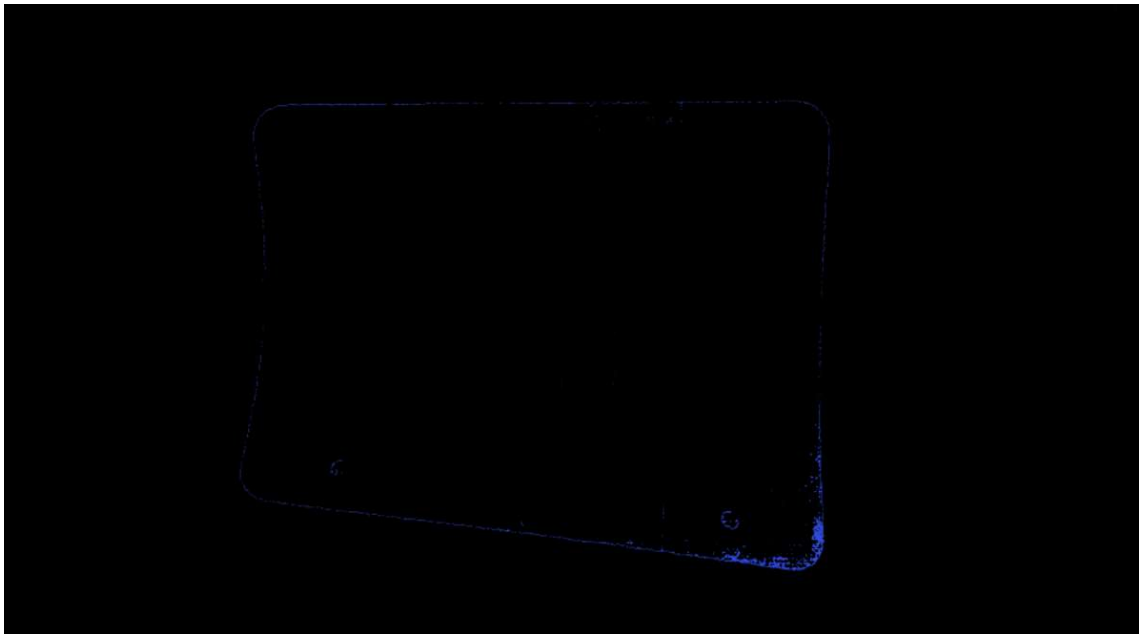

Figure 18. Finished HSV adjustments for the best non-edge suppression.

After the HSV processing, most of what is left on the image are the edges. Even though the edges seem very faint in figure 18, the image processing is successful because the dim edges will be modified further in the next process. There are some more specs left especially where the holes are in the panel, but they should not matter in the long run.

4.2.3 Edge detecting

After the image has been adjusted with HSV, it goes through more image processing and an edge detection tool. First, the image is converted from colour to greyscale with cvtColor, then a gaussian blur is applied to it which takes in nearby colours of the pixels and gives the average colour of the pixels as the output [24]. After this, Canny can be used to detect the edges of the image.

Canny is an important part of the code since it makes the edges of the glass panels very clearly visible, so that they can be made to a 2D surface model. To

use the Canny filter, the appropriate minimum and maximum values must be found. These are used to determine if edges found by the Canny filter are really edges and which are not. The maximum value is the limit for something called “sure-edges”, and the minimum value is the limit for what can be considered an edge. The lines in the area between the minimum and maximum values are classified as edges if they are touching a “sure-edge” and discarded if they are not. [25] As shown in the figure 19 below, the line C is classified as an edge because it is touching line A that is a “sure-edge”, and line B is discarded.

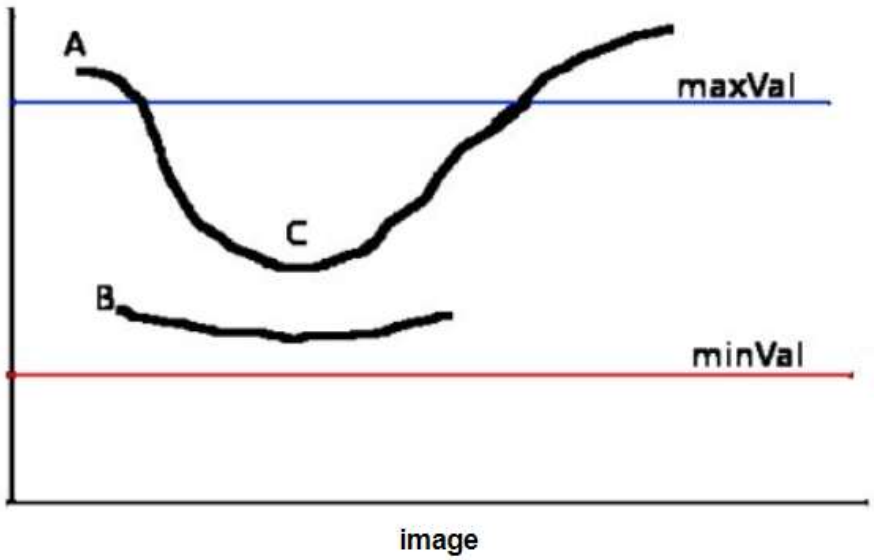

Figure 19. Canny edge detection algorithm’s working principle. [25]

After creating the edge model using Canny, the image is then eroded and dilated to isolate individual elements and to join others (see figures 20 and 21). This makes the edge more connected to create a better 2D surface model while also reducing the count of small pixel groups that might have been left behind from all the other filtering. [26]

```
#https://www.youtube.com/watch?v=WQeoO7MI0Bs
#edge recognition and more image processing
imgGray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
imgBlur = cv2.GaussianBlur(imgGray,(7,7),0)
imgCanny = cv2.Canny(img,100,120)
kernel = np.ones((5,5),np.uint8)
imgDialation = cv2.dilate(imgCanny,kernel,iterations=1)
imgEroded = cv2.erode(imgDialation, kernel,iterations=1)
```

Figure 20. More image processing and edge detection. [27]

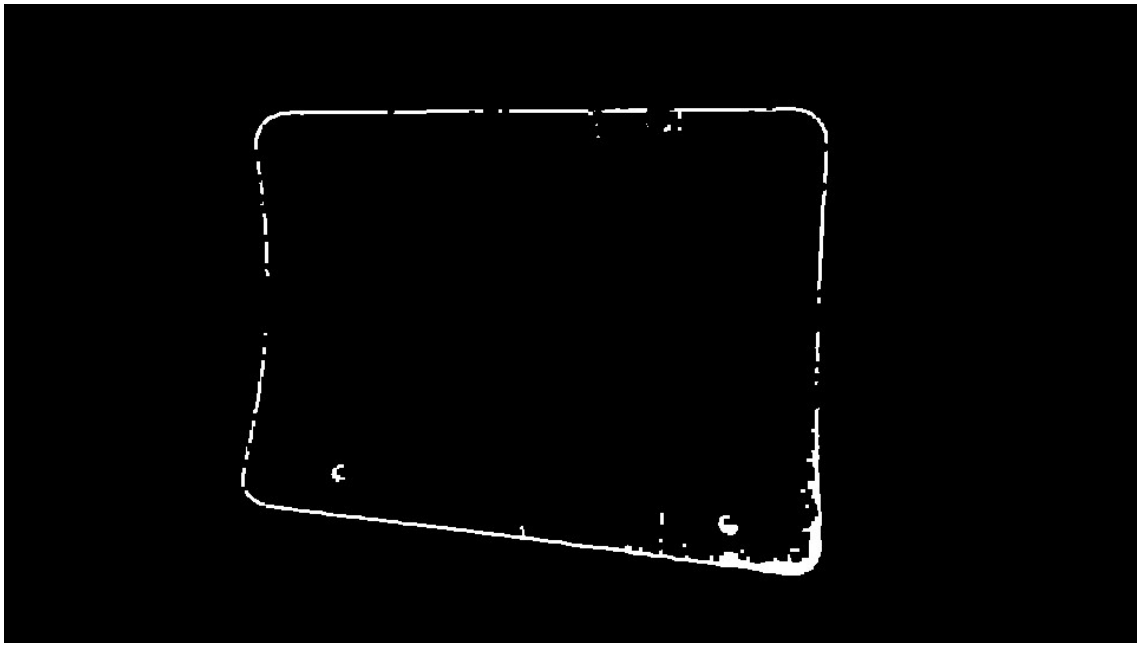

Figure 21. Glass edge model after Canny, ready to be made into a 2D surface model.

4.2.4 Creating a 2D surface model

After the image has been processed to the state seen in figure 21, it is ready to undergo the stl-file conversion sequence. This sequence finds the pixel intensity of every pixel in the image to determine the thickness to be used in the surface model. Then some matrix calculations are used to create the 2D surface model (see figure 22). [28]

The stl-file is really a 3D model, but because the y coordinates of the model do not match the real y coordinates of the glass panel, it is not a finished 3D model, and thus it was named a 2D surface model to help differentiate between the two (see figure 23). The whole point of converting the edge model to a 2D surface model is to allow it to be imported to other places, like to the robot to check the real z-dimension, for example.

```
#https://spltech.co.uk/how-to-create-a-3d-model-of-a-photo-using-python-numpy-and-google-colab-part-i/
vertices=np.zeros((nrows,ncols,3))

for x in range(0, ncols):
  for y in range(0, nrows):
    pixelIntensity = imageNp[y][x]
    z = (pixelIntensity * max_height) / maxPix
    vertices[y][x]=(x, y, z)

faces=[]

for x in range(0, ncols - 1):
  for y in range(0, nrows - 1):
    # create face 1
    vertice1 = vertices[y][x]
    vertice2 = vertices[y+1][x]
    vertice3 = vertices[y+1][x+1]
    face1 = np.array([vertice1,vertice2,vertice3])

    # create face 2
    vertice1 = vertices[y][x]
    vertice2 = vertices[y][x+1]
    vertice3 = vertices[y+1][x+1]

    face2 = np.array([vertice1,vertice2,vertice3])

    faces.append(face1)
    faces.append(face2)

print(f"number of faces: {len(faces)}")
facesNp = np.array(faces)
# Create the mesh
surface = mesh.Mesh(np.zeros(facesNp.shape[0], dtype=mesh.Mesh.dtype))
for i, f in enumerate(faces):
    for j in range(3):
        surface.vectors[i][j] = facesNp[i][j]
# Write the mesh to file
surface.save("glass"+num+"_surface.stl")
```

Figure 22. Matrix calculations to create the surface model. [28]

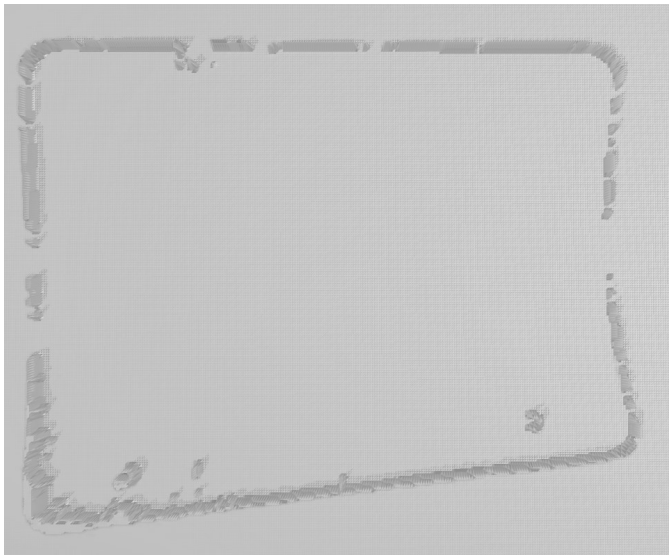

Figure 23. Finished stl surface model of the glass panel.

5 ANALYSING LIGHT TO FIND 3D CONTOURS

After the 2D surface model is determined, the third dimension can be added. Analysing light can be a valid option to finding the 3D features of the glass panels. Both methods described in the next few chapters revolve around finding the angles of the glass in z-direction, one by using reflection and the other by using refraction to measure the lateral shifts in the glass.

5.1 Reflection of light

Reflection is the phenomenon of light bouncing off an object. Especially from shiny surfaces, like glass, the light will reflect at the same angle as it hits the surface. There are two different types of reflection, which are specular and diffuse reflection. Specular reflection refers to the situation when light hits a flat, smooth surface and bounces off in the same pattern as it hits the surface. The reflection that happens when light hits an uneven surface is called diffuse reflection. Both reflection types are introduced in figure 24 below. [29]

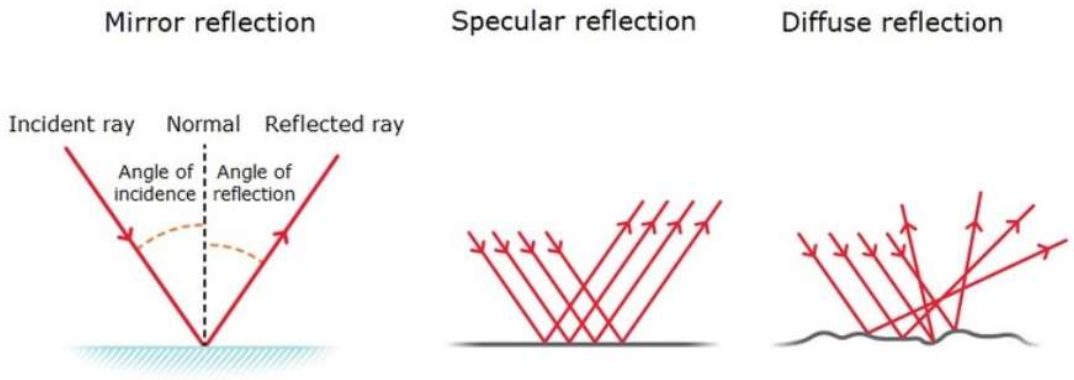

Figure 24. Different types of reflections of light. [29]

The light gets reflected in predictable manners, be it specular or diffuse in nature. This fact could be used to find the 3D contours of the glass panels and will be tested in the next few chapters.

5.1.1 Light pattern setup

The setup of checking the reflections was very similar to the process described in chapter 4.2.1. This time, the focus is on the ultraviolet lights instead of the edges of the panels. The UV lights were placed in a pattern around the camera to shine down onto the glass and to reflect to the camera in a specific pattern (see figure 25).

Figure 25. UV light pattern.

5.1.2 Pictures with UV lights

The setup worked as intended, and the UV light patterns are clearly visible in the photos. When taking a picture of a flat glass panel with no 3D contours, the pattern was unchanged (see figure 26). When the glass panel had 3D contours, the light pattern would change according to the shapes (see figure 27). On the figures some reflections of ambient light can be seen. They are a different colour than the UV lights though, so if needed, they can be easily filtered out using the same HSV process, as in chapter 4.2.2.

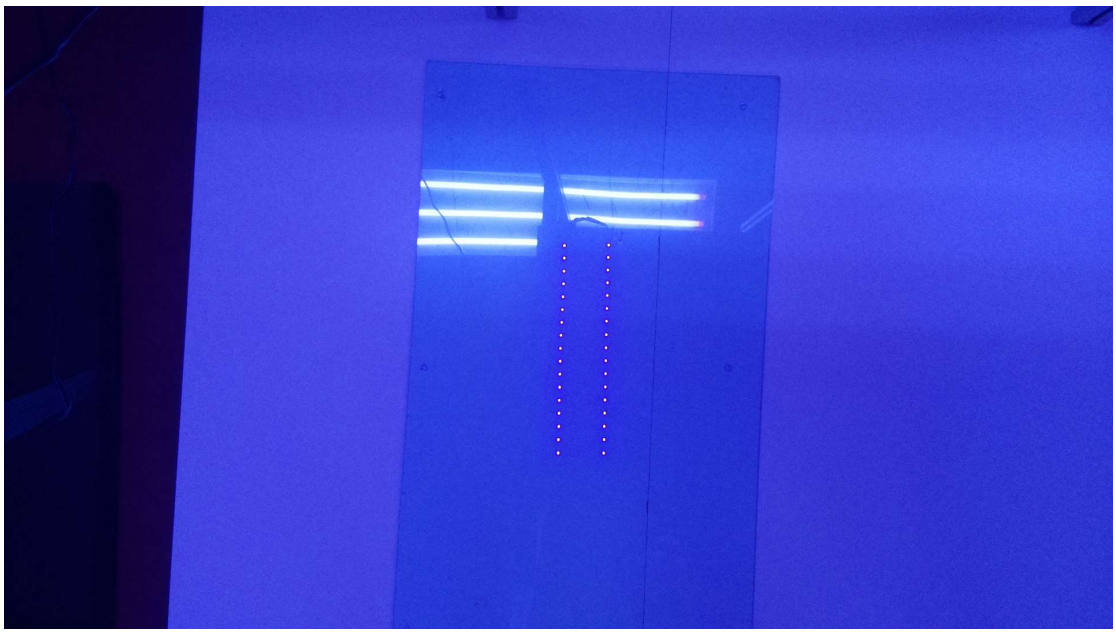

Figure 26. Glass panel with no 3D contours.

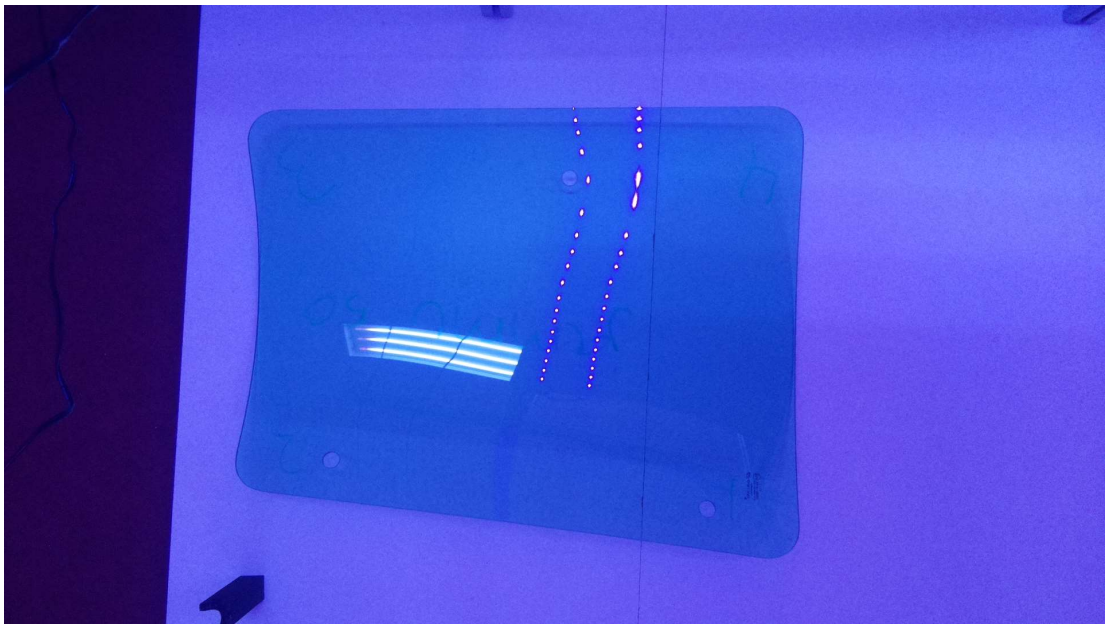

Figure 27. Glass panel with 3D contours.

5.1.3 Analysing the UV patterns

As demonstrated in the previous chapter, the pattern remains unchanged when reflected off a straight surface and is changed when reflected from a surface with 3D contours. This deviation could be used to apply the concept proposed in the paper by T. Yamanaka, F. Sakaue and J. Sato: "Adaptive Image Projection onto Non-planar Screen Using Projector-Camera Systems" [30].

In the paper, a solution is proposed on how to project a 2D image to a 3D screen using a camera (see figure 28). Using epipolar geometry and matrix calculations, the group managed to adapt the picture being projected to fit the 3D screen (see figure 29). In this project, the UV lights on the glass function similarly as image of the projector on the screen. The idea of the paper could thus be used to analyse the 3D contours of the glass and then to create a 3D model.

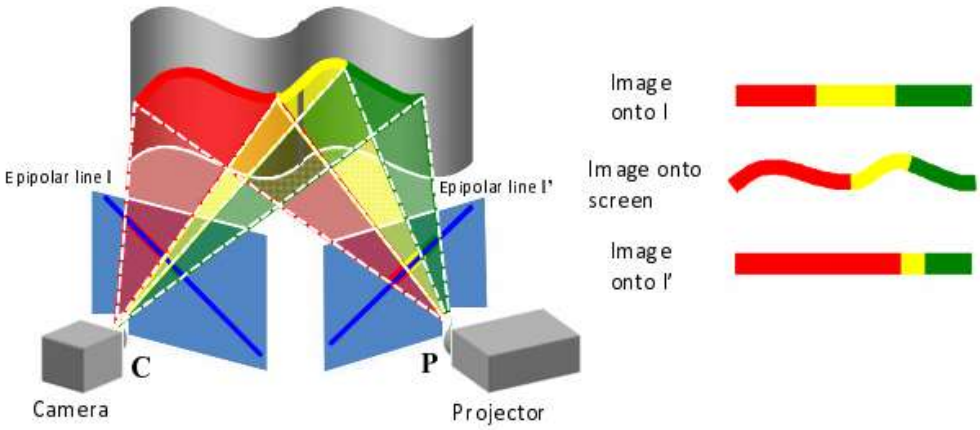

Figure 28. Projecting an image to a 3D screen. [30]

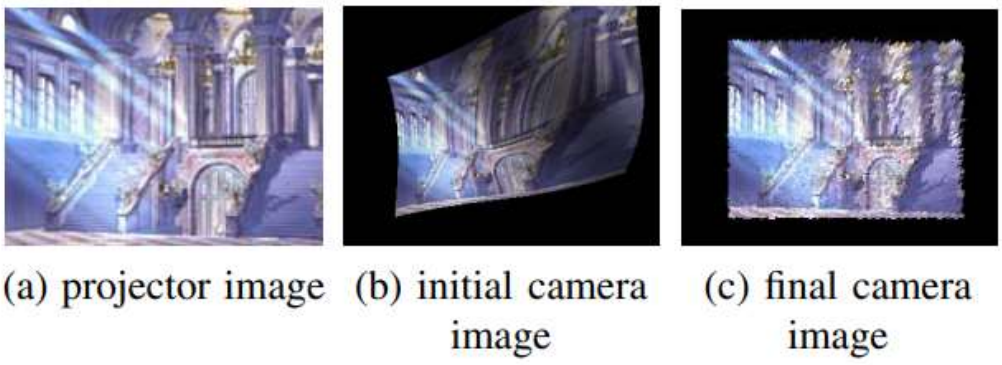

Figure 29. Adapting the projected image to a 3D screen. [30]

However, even though the process was explained in the paper, it was not implemented in this project. This is due to the complexity of coding all the matrix functions required for the idea of the paper to function and due to the time constraints of this project. With more time and with someone with a background in linear algebra, the idea of the paper could be used for the application of this project as well.

5.2 Lateral shift of light

The lateral shift of light could be used to determine the angle of the glass and therefore also form a 3D picture of the glass panels. It can be calculated with formula 1, where *d* is the amount of lateral shift in millimetres, *t* is the thickness of glass slab in millimetres, *i* is the angle of incidence and *r* is the angle of refraction (see figure 30). [31]

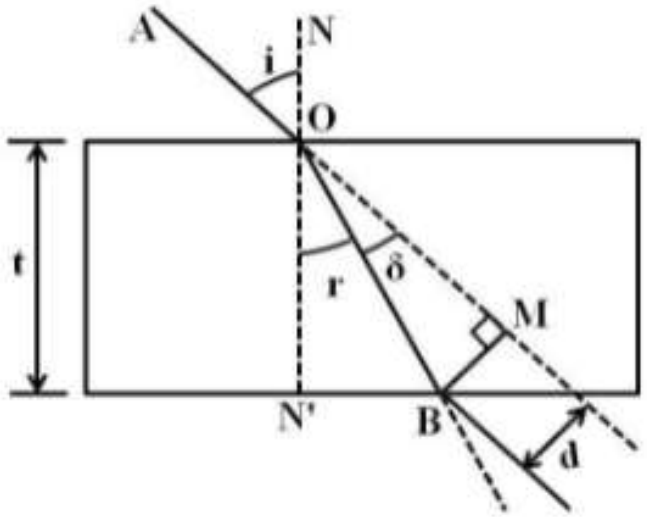

Figure 30. Lateral shift in a glass slab. [32]

To calculate the value for *r*, Snell's law can be used written as formula 2. In Snell's law, n_1 is the refractive index of the first substance, θ_1 is the angle between the normal vector and the incoming light, n_2 is the refractive index of the second substance, and θ_2 is the angle between the normal vector and the outgoing light. [33]

$d = \frac{t * sin(i - r)}{cos(r)}$	Formula 1, lateral shift
$n_1 * sin(\theta_1) = n_2 * sin(\theta_2)$	Formula 2, Snell's law

5.2.1 Lateral shift setup

The lateral shift could be used by shining a light source with the help of a robot in a 90-degree angle in relation to the table where the glass panels are kept and then observing with a camera mounted to the robot where the laterally shifted light ends up. When the light moves from air to the glass, the light shifts towards the normal of the edge of the glass marked in figure 30 with *N*. However, with a 90-degree angle, it is already at the normal, so there would not be any lateral shift [34]. Therefore, it could be determined that the glass panel is horizontal.

If the glass panel has an angle, the light will shift towards the normal causing a lateral shift, marked in figure 30 with d, which could be measured and then calculated into the angle of the glass (see figure 31). Afterwards, the calculated angles could be added to the 2D surface model with the robots coordinates data, making it a model 3D.

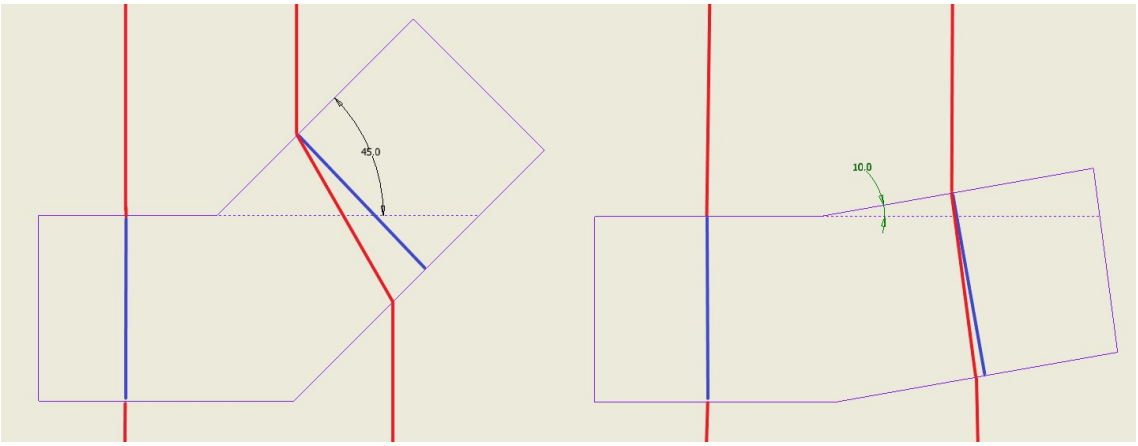

Figure 31. Visualized lateral shift in 45-degrees and 10-degrees

5.2.2 Calculations of lateral shift

To calculate the lateral shift, the following values are used. With a red light with a wavelength of 589 nanometres, the reflective index of typical window glass is 1.516 and air is 1.000293 [35]. Using Snell's law, the value for *r* can be calculated.

The glass panels have a thickness of 4 to 8 millimetres. With these values, formula 1 and formula 2 can be used to calculate the lateral shift of the laser as follows:

$r = sin^{-1}\frac{1.000293 * sin(45)}{1.516} = 27.803$
$d = \frac{6mm * sin(45 - 27.803)}{cos(27.803)} = 2.005mm$

These numbers diminish even more with a smaller angle of entry to the glass:

$r = sin^{-1}\frac{1.000293 * sin(10)}{1.516} = 6.577$
$d = \frac{6mm * sin(10 - 6.577)}{cos(6.577)} = 0.361mm$

With these findings, it can be determined that either the process of measuring the movement of the laser would have to be incredibly accurate, or that lateral shift cannot be used to generate the 3D images in this application.

6 MORE 3D DETECTION METHODS

Due to analysing light not generating the hoped results, the methods of detecting 3D features were changed. In the following chapters, stereovision, X-ray detection and robotic measuring will be discussed, because they are the next best choices to be used in this project.

6.1 Binocular stereovision

Stereovision is a machine vision solution used to create, for example, a depth map. Binocular stereovision means that there are two cameras to sense the object from different viewpoints, much like human eyes. By using the principle of triangulation, a depth map can be created. [36] Next, it is demonstrated how binocular stereovision was tested to find out whether it is suitable for this project or not.

6.1.1 Stereovision setup

The setup for testing stereovision was very similar to all the other machine vision experiments. The biggest differences were the use of two independent Raspberry Pi camera setups and the alignment of the cameras.

The alignment of the cameras was a crucial step to get reliable results. Aligning the cameras was completed by installing them both on the same support frame with a distance between each other. Figure 32 demonstrates the setup.

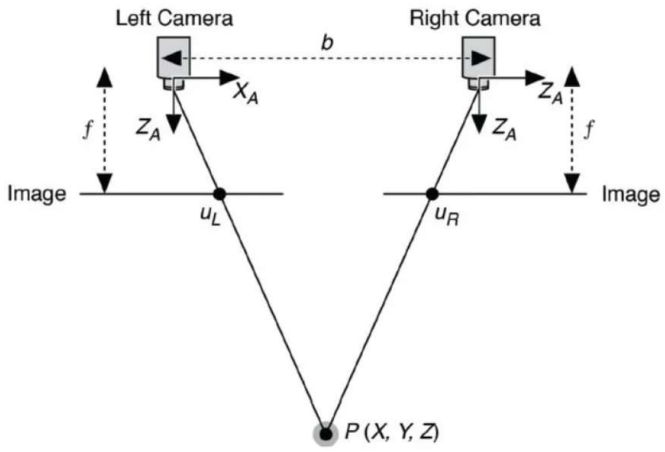

Figure 32. Binocular stereovision system. [37]

To align the cameras properly, some test photos were taken and compared to each other. The cameras were then adjusted until the images were as close in line to each other as possible (see figure 33). A narrow piece of background was used for these experiments because the corners of the background can be used as anchor points to make fine adjustments to the images. For example, the slight rotational error can be fixed easily later in the code.

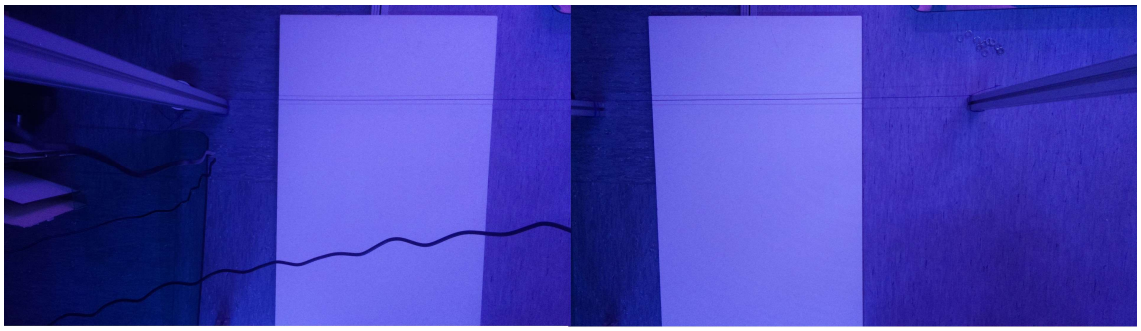

Figure 33. The images of both the cameras.

6.1.2 Stereovision code

To analyse the images using the stereovision algorithm, the images first had to be cropped down to only include the background. This was accomplished by taking the pixel locations of the corners of the background and using a cropping function. The cropped images were then changed to greyscale. Figure 34 shows both cropped greyscale images.

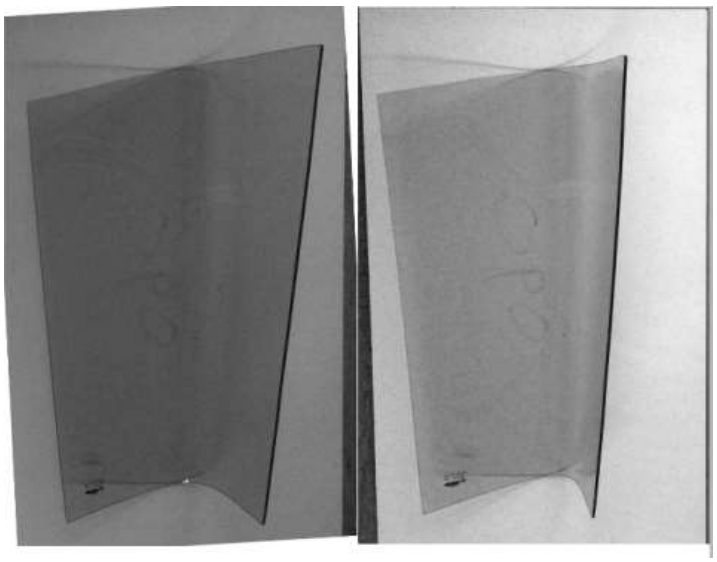

Figure 34. Cropped greyscale images.

After the images were prepared, a stereovision algorithm could be used (see figure 35). The most important of the parameters are “number of disparities”, “block size” and “uniqueness ratio”. “Number of disparities” affects the accuracy of the disparity map and should be kept as high as possible. It also shifts the image right, so it must be compensated with “minimum disparity” which is to be half and opposite to number of disparities to keep the image centred. “Block size” is directly correlated with how big the images are and how far the cameras are from each other: when increased, the images get smoother. “Uniqueness ratio” is a post-filtering step which filters out pixels that are too similar in both original pictures. [38] After the compute function is used, the result is a disparity map (see figure 36).

```
#https://docs.opencv.org/4.5.0/dd/d53/tutorial_py_depthmap.html
stereo = cv.StereoSGBM_create(minDisparity=-30,
        numDisparities=60,
        blockSize=3,
        disp12MaxDiff = 10,
        uniquenessRatio = 6,
        speckleWindowSize = 2,
        speckleRange = 20)

disparity = stereo.compute(gray1,gray2)
```

Figure 35. Stereovision algorithm. [39]

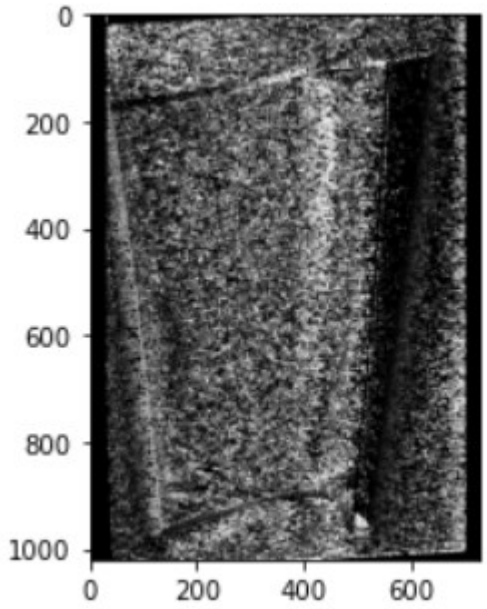

Figure 36. Disparity map.

The disparity map functions by showing areas that are greatly different in darker colours as well as areas that are more like each other in lighter colours. The disparity map displayed in figure 36 can then be used to make a depth map by calibrating the different values into real world distances. The basic idea behind making a disparity map into a depth map is that things that are further away from the cameras are more like each other in the two images, than things closer to the cameras.

6.1.3 Problems of stereovision

It is important to also note that the disparity map has a few notable problems. One is that there is a lot of background noise, which could not be filtered out. The background noise is most likely caused by slightly different perception of light by the cameras (see figure 37). The images were taken simultaneously, so it seems like the problem is in the equipment.

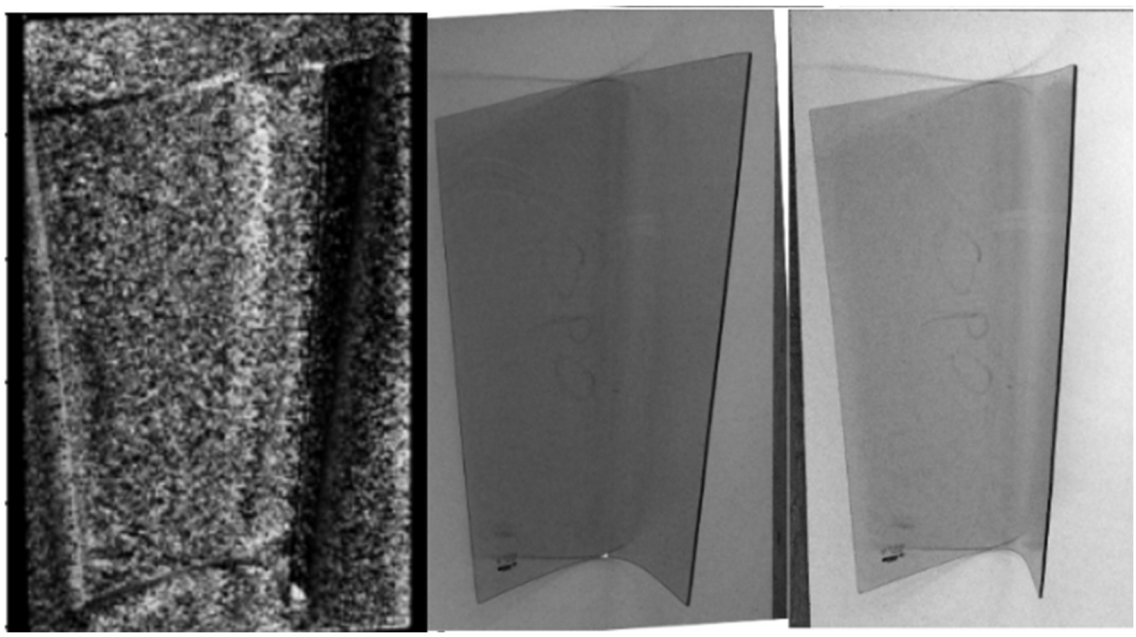

Figure 37. Comparison between the grey glass panels and the disparity map.

The other problem is the inconsistencies in the disparity map. Edges that are as far away from the cameras, like the top left corner, are different: one side is black, the other is grey. This is most likely due to the cameras being so far away from each other. However, the distance between the two cameras is needed for the glass panels with less noticeable 3D contours. The images below were taken with the same setup and the same algorithm parameters were used, but the 3D contours of the glass panels are hardly visible in the disparity image.

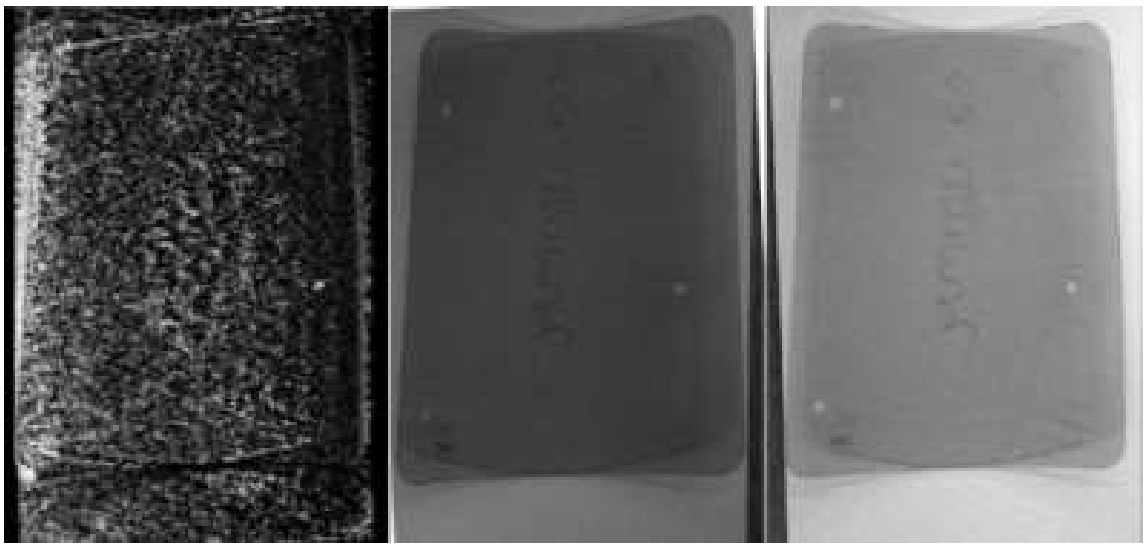

Figure 38. Another comparison set.

All in all, stereovision does not seem like the most reliable solution to find the 3D contours. With better equipment, like sharper and more uniform cameras, some of the problems could be fixed. The problem of sensing hardly noticeable 3D contours while also not getting bad information from the panels with big contours could be fixed by having different settings for the distances between cameras. However, changing the settings is not a simple process, and it would also require changes in the code, which makes it a very slow fix.

6.2 X-ray detection for 3D

As mentioned earlier in chapter 3.2, X-ray detection systems are reliable ways of detecting materials and creating 2D images of them. In this chapter, the ways how X-ray imaging works are examined further to find out how it could be used to create 3D models.

X-ray imaging is based on the penetration of X-ray photons, which is affected by the thickness, density, and the atomic number of an object [40]. Attenuation length is used to describe the penetration of the X-ray photons expressed in units of distance. It is measured by calculating the distance where the probability of a particle passing through the object is one divided by the Euler's number. Simply put, attenuation length can be found when only 36.8 percent of the photons pass through the object. Attenuation length is also correlated with photon energy: the higher the energy, the higher the penetration (see figure 39). [41]

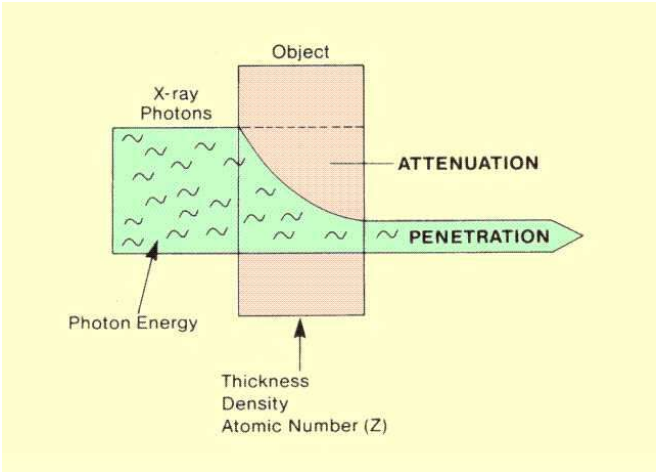

Figure 39. X-rays penetration in relation with an objects thickness and density [40]

The density, atomic number, and the thickness of a glass panel is homogenous. The only factor that can change is the angle of the glass in comparison to the direction of the X-ray beam. This results into a change in the thickness the X-ray photons must penetrate (see figure 40).

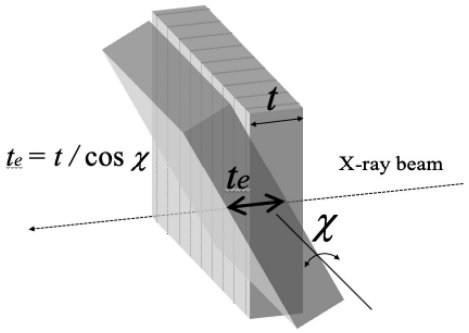

Figure 40. Penetration of an X-ray through an angled surface. [42]

The change in thickness of penetration will create different shades of grey on the 2D X-ray image. This can be exploited to create 3D images of the glass panels.

To prove that the concept could work, an online calculator from Henke can be used to find the attenuation lengths for the glass panels [43]. Most commercially available X-ray spectrometers have a range from about 0.6–60 keV, which puts a limit on how much energy can be used [44]. A typical example of window glass that is mostly composed of silicon dioxide has a density of 2.5 kg/m^3 [45]. Inserting these numbers to the online calculator gives out a graph shown in figure 41.

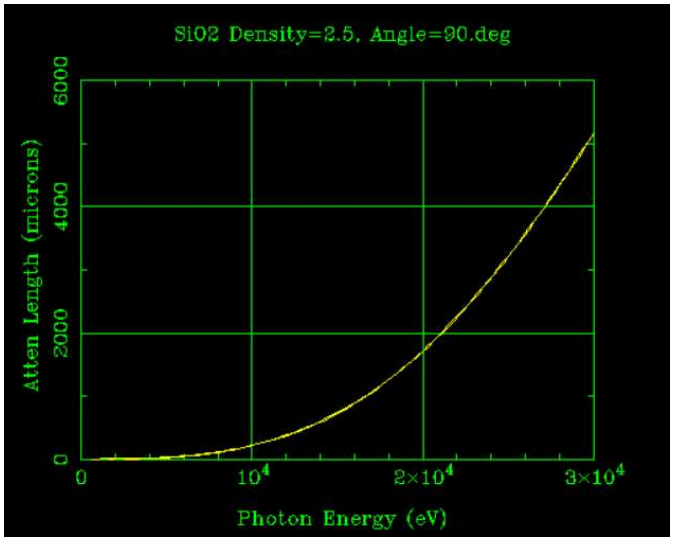

Figure 41. Attenuation length for window glass. [43]

As seen in figure 41 before, it can be determined that at typical X-ray spectrometer energies, the glass panels can be comfortably detected. This could be used to, for example, turn a regular airport baggage X-ray scanner into a 3D model scanner, provided that the airport scanner is sensitive enough to notice the small changes in attenuation lengths.

The X-ray scanner would create 2D greyscale images with the darkness based on the thickness or, in this case, the angle of the glass panels. The images could then be inserted to a Python code, which makes them into 3D. The Python code functions similarly as the one explained in chapter 4.2.4, but with no edge recognition creating a complete 3D model instead of only an edge model. Below in figure 42 is an example of a 16-step horizontal greyscale turned into a 3D model.

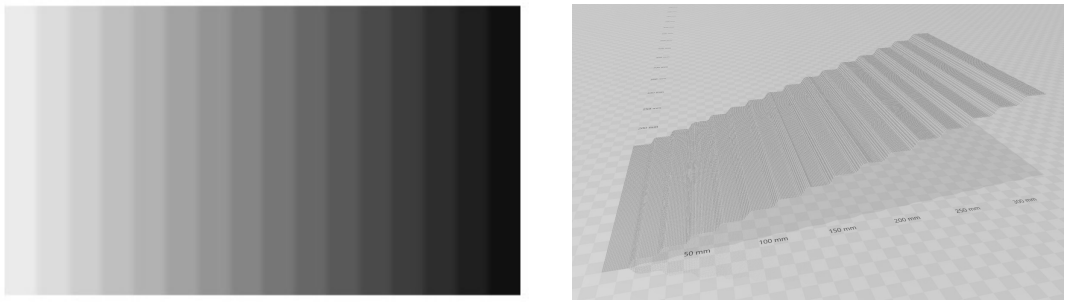

Figure 42. Grayscale to 3D model example. [46]

There are also ready made 3D X-ray scanners on the market, but these are significantly more expensive compared to typical airport 2D X-ray scanners. Most of the 3D X-ray scanners on the market also have too small of a working area to fit most of the glass panels, while the airport scanners are designed to fit bulky objects. [47] [48]

6.3 Robotic measuring

With the 2D edge image created and converted into a stl-file in chapter 4.2.4, the robot could be operated. The general idea is to have the glass panel probed with a linear position sensor to measure the z location of the edges of the glass panels. The linear position sensor would have a roller in the end of it to allow for low

friction movement on the glass panels surface. The linear position sensor would also have a spring to help the sensor return to its normal position (see figure 43).

Figure 43, 200 mm linear position sensor with a return spring. [49]

The robot will first start to drive the position sensor against the glass, which will compress the sensor when it touches the glass panel. This could then be measured and stopped when the position sensor reads 100 mm. It is important for the sensor to be halfway compressed in the beginning of the process, so that the robot can sense shapes that are downwards in the glass instead of only shapes that come upwards.

When the spring is compressed halfway, the robot can then be instructed to drive along the edges of the model using the stl-file and to take the measurements from the sensor, for example, every 50 mm. Every time the sensor measures, the robot would also be adjusted in height so it would not crash into the glass panels.

The path of the robot could be recorded and combined with the data of the sensor to create a line in 3D space where the tool point has been. This line, that is in 3D space, is now the new 3D edge model. The robot could also be programmed to drive across the whole glass panel, from side to side, through the entire width of the panel to gain a complete 3D model of the panel. However, for this application, knowing only the edge positions is enough. Running the robot just once around the glass panel is also much faster than running it back and forth through the whole width of the panel.

7 CONCLUSIONS

The goal of this thesis was to first create 2D models of the glass panels and then to add the third dimension to the models. Creating 2D models was theorized to work with X-ray detection methods and proven to work with machine vision using Python and the OpenCV library. However, creating 3D models turned out to be a more challenging task.

Out of the five methods for creating 3D models or glass panels, three have a good chance of working. Creating 3D models should be possible using X-ray detection, analysing light patterns, and by robotically measuring the z-coordinate. Due to budgetary and time constraints, these options were not explored until reaching the desired goal.

Robotically measuring the z-coordinate using a linear position sensor is presumably the easiest way to get to work. Using the 2D surface models created with machine vision the robot can be programmed to measure points along the edge of the glass panel for the z-coordinate. The z-coordinate information can then be combined into the xy data of the 2D surface model to complete the 3D model. The downside of robotic measuring is the fact that it might be slow depending on the required measurement frequency. However, this can be mitigated by only measuring the first new glass plate of a series and using the same 3D model for all panels of the same form instead of measuring every panel as they are packaged.

X-ray detection is also a good option for creating the 3D models. The benefits of X-ray detection are that it can be implemented into the process easily by using conveyor belts and that the 3D model creation process can run simultaneously while another glass panel is being packaged. X-ray detection also works as a standalone solution, not requiring machine vision like robotic measuring. The difficulties in X-ray detection come with choosing the right equipment, which might also be pricy.

Analysing light patterns is the most difficult of the three to develop into a fully fledged solution. The matrix calculations required to calculate angles to the 2D image are very difficult. This solution would have to be further explored by someone with a background in linear algebra and programming.

For further research, the paper by T. Yamanaka, F. Sakaue and J. Sato: “Adaptive Image Projection onto Non-planar Screen Using Projector-Camera Systems” is good to get deeper into light patterns. Motive by OptiTrack is also promising for mapping contours in 3D spaces.

SOURCES

[1] Fahrzeugglas. Müller Industrietechnik GmbH. Read 5.7.2021. https://mueller-industrietechnik.com/produkte/fahrzeugglas/john-deere/

[2] What is a photoelectric sensor? Keyence. Read 5.7.2021. https://www.keyence.co.uk/ss/products/sensor/sensorbasics/photoelectric/info/

[3] Ultra-compact Photoelectric Sensor EX-20 Ver.2. Panasonic. Read 3.8.2021. https://www3.panasonic.biz/ac/ae/fasys/sensor/photoelectric/ex-20/index.jsp

[4] What is a Photoelectric Sensor? Tri-Tronics. Read 3.8.2021. https://www.ttco.com/what-is-a-photoelectric-sensor

[5] J. Tyson & E. Grabianowski. How Airport Security Works. 20 June 2001. HowStuffWorks.com. https://science.howstuffworks.com/transport/flight/modern/airport-security.htm

[6] P. Wreckler. 9.2007. Schematic of X-ray imaging system. ResearchGate. https://www.researchgate.net/figure/Schematic-of-X-ray-imaging-system_fig6_248449554

[7] X-Ray in Food Inspection. Food Facts for Healthy Choices. 2.1.2013. https://www.eufic.org/en/food-production/article/the-use-of-x-rays-in-food-inspection

[8] Industrial Radiography. United States Environmental Protection Agency. Read 8.8.2021. https://www.epa.gov/radtown/industrial-radiography

[9] Industrial X-ray/CT Inspection Systems. Nikon. Read 8.8.2021. https://www.nikon.com/about/technology/product/xray-ct/index.htm

[10] A Deep Dive Into Machine Vision Systems And Their Uses. L.A.C. Conveyors Automation. Read 15.8.2021. https://www.lacconveyors.co.uk/machine-vision-systems-provided/

[11] Machine Vision vs. Computer Vision — What's the Difference? Appen. 31.5.2021. https://appen.com/blog/computer-vision-vs-machine-vision/

[12] In-Sight Explorer Software Features. Cognex. Read 15.7.2021. https://www.cognex.com/products/machine-vision/2d-machine-vision-systems/in-sight-vision-software

[13] In-Sight Standard Vision System Specifications. In-Sight® 5000 Series Vision System Installation Manual. Read 15.7.2021. https://www.cognex.com/support/downloads/ns/1/11/35/51xx_54xx_ss.pdf

[14] Cognex In-Sight 5403 Hi-Res Sensor w/PatMax® (IS5403-11). Vision-Supplies EU. Read 15.7.2021. https://eu.vision-supplies.com/cognex-in-sight-5403-hi-res-sensor-wpatmax

[15] In-Sight D900 Vision System. Cognex. Read 15.7.2021. https://www.cognex.com/products/deep-learning/in-sight-d900

[16] M. Chatterjee. 29.6.2021. What is Computer Vision? Know Computer Vision Basic to Advanced & How Does it Work? My Great Learning. https://www.mygreatlearning.com/blog/what-is-computer-vision-the-basics/

[17] What is Python? Executive Summary. Python. Read 4.8.2021. https://www.python.org/doc/essays/blurb/

[18] Introduction. Open Source Computer Vision. Read 4.8.2021. https://docs.opencv.org/4.5.2/d1/dfb/intro.html

[19] Operating system images. Raspberry Pi. Read 4.8.2021. https://www.raspberrypi.org/software/operating-systems/

[20] M. Murray. 15.12.2019. RASPBERRY PI NOIR VS NORMAL CAMERA. The Geek Pub. https://www.thegeekpub.com/16220/raspberry-pi-noir-vs-normal-camera/

[21] Hue, Value and Chroma. Vita North America. Read 15.8.2021. http://www2.vitanorthamerica.com/products/shade-management/color-theory/understanding-color-overview/hue-value-and-chroma/

[22] HSV color solid cylinder. Wikimedia Commons. Read 15.8.2021. https://commons.wikimedia.org/wiki/File:HSV_color_solid_cylinder.png

[23] Edge detection. SO Documentation. Read 4.8.2021. https://sodocumentation.net/opencv/topic/6099/edge-detection

[24] Smoothing Images. Open Source Computer Vision. Read 4.8.2021. https://docs.opencv.org/4.5.2/d4/d13/tutorial_py_filtering.html

[25] Canny Edge Detection. Open Source Computer Vision. Read 4.8.2021. https://docs.opencv.org/3.4/da/d22/tutorial_py_canny.html

[26] Eroding and Dilating. Open Source Computer Vision. https://docs.opencv.org/3.4/db/df6/tutorial_erosion_dilatation.html

[27] Murtaza's Workshop - Robotics and AI. LEARN OPENCV in 3 HOURS with Python | Including 3xProjects | Computer Vision. Youtube. https://www.youtube.com/watch?v=WQeoO7MI0Bs

[28] A. Cachada. 15.10.2020. How to create a 3D model of a photo using Python, NumPy and Google Colab. Spltech. https://spltech.co.uk/how-to-create-a-3d-model-of-a-photo-using-python-numpy-and-google-colab-part-i/

[29] Reflection of light. Science Learning Hub. 27.8.2020. https://www.sciencelearn.org.nz/resources/48-reflection-of-light

[30] T. Yamanaka, F. Sakaue and J. Sato. 26.8.2010. Adaptive Image Projection onto Non-planar Screen Using Projector-Camera Systems. https://ieeexplore.ieee.org/document/5597793

[31] M. Gayathri. Lateral Shift. Read 27.7.2021. http://kea.kar.nic.in/Outreach/Physics/Geometrical_Optics.pdf

[32] Lateral shift due to glass slab. Toppr. Read 27.7.2021. https://www.toppr.com/ask/content/story/lateral-shift-due-to-glass-slab-49247/

[33] Snell's law. Britannica. Read 27.7.2021. https://www.britannica.com/science/Snells-law

[34] Refraction of light. Science Learning Hub. https://www.sciencelearn.org.nz/resources/49-refraction-of-light

[35] Tekniikan kaavasto, page 132, Tammertekniikka, Bookwell Oy 2016

[36] K. Lu, X. Wang, Z. Wang and L. Wang. 8.7.2011. Binocular stereo vision based on OpenCV. https://ieeexplore.ieee.org/document/6138147

[37] A Guide to Stereovision and 3D Imaging. Tech Briefs. 1.10.2012. https://www.techbriefs.com/component/content/article/tb/pub/features/articles/14925

[38] K. Sadekar. 5.4.2021. Depth perception using stereo camera (Python/C++). LearnOpenCV. https://learnopencv.com/depth-perception-using-stereo-camera-python-c/

[39] Depth Map from Stereo Images. Open Source Computer Vision. Read 27.8.2021. https://docs.opencv.org/4.5.0/dd/d53/tutorial_py_depthmap.html

[40] P. Sprawls. Radiation Penetration. Sprawls Educational Foundation. Read 9.8.2021. http://www.sprawls.org/ppmi2/RADPEN/

[41] V. Thomsen, D. Schatzlein and D. Mercuro. 1.9.2005. Tutorial: Attenuation of X-Rays By Matter. Spectroscopy. https://www.spectroscopyonline.com/view/tutorial-attenuation-x-rays-matter

[42] M. Ando, R. Gupta, A. Iwakoshi, J. Kim, D. Shimao, H. Sugiyama, N. Sunaguchi, T. Yasa and S. Ichihara. 1.11.2020. X-ray dark-field phase-contrast imaging: Origins of the concept to practical implementation and applications. European Journal of Medical Physics. https://www.physicamedica.com/article/S1120-1797(20)30309-4/fulltext

[43] X-Ray Attenuation Length. Henke. Used 9.8.2021. https://henke.lbl.gov/optical_constants/atten2.html

[44] R. Jenkins. 2003. X-Ray Analysis. Encyclopedia of Physical Science and Technology (Third Edition). https://www.sciencedirect.com/topics/engineering/x-ray-photon

[45] Physical Properties of Glass. Saint-Gobain. Read 9.8.2021. https://uk.saint-gobain-building-glass.com/en-gb/architects/physical-properties

[46] VideoForge Pro - Pattern Descriptions. Calman Resource Center. Read 9.8.2021. https://kb.portrait.com/help/videoforge-pro-pattern-descriptions

[47] 130 KV2D /2.5D Röntgenanlage für Elektronik. X-ray Lab. Read 10.8.2021. https://xray-lab.com/2d-2-5d-roentgenanlage-fuer-elektronik/

[48] Airport Security X Ray Machine. Aliaba. Read 10.8.2021. https://www.alibaba.com/showroom/airport-security-x-ray-machine.html

[49] Position Transducer with return spring potentiometric up to 200 mm, IP54. Novotechnik. Read 15.8.2021. https://www.novotech-nik.com/pdfs/TEX_spring_US_e.pdf