Laser assisted optical gap detection

Thomas Sund Mjåland
Sigurd Oddtrøen Strand

February-May 2019

Title:
Laser assisted optical gap detection

Candidate numbers:			
Date:	**Subject code:**	**Subject:**	**Document access:**
19/05-2019	IE303612	Bachelor thesis	Open
Study:	**number of pages/attachments:**		**Bibl. Nr:**
Automation	82/2		

Task giver(s)/Supervisor(s)
Task given by Currence Robotics Supervisor Ottar L. Osen

Task/Summary
Machines are used more and more in warehouses to move products. More traditional machine-vision has difficulties distinguishing between packages decorated with complex color patters and stacked closely together. The goal of this project is to create a system that can find the boundary between boxes, and allow localizing single packages in a stack which can then be retrieved/picked.

Contents

Acronyms

HSV (**H**ue, **S**aturation, **V**alue) is an alternative way to represent colors made to more accurately reflect how the human eye perceives color than the RGB model. HSV separates the color (Hue) from the intensity (Saturation) and brightness (Value). 17, 72

RGB (**R**ed, **G**reen, **B**lue) is a way of representing colors numerically as a mix of the three base colors Red, Green and Blue. 17

Glossary

algorithm a process or set of rules to be followed in calculations or other problem-solving operations, especially by a computer. 7, 69, 78

artificial intelligence Artificial intelligence is the teaching of robot to calculate a "best solution" for any input, based on weights the machine will consider.. 80

aspect ratio ratio of a digital camera sensor's width to its height. 5

big O notation Big O notation is a way of representing the complexity of an algorithm in relation to the size of the input. $O(n)$ means the computation time increases linearly with the input data, $O(n^2)$ means doubling the input data would quadruple the computation time.. 25

datasheet a document that summarizes the performance and other technical characteristics of a product, machine or component. 5

extrinsics parameters that describes the cameras position and rotation in space. 50

focal length the length from the camera sensor to the principal point. Often given in mm or pixels. 51

fourier transform The fourier transform of a signal or image gives information of the frequency component. For an image it shows how often patterns or gradients repeat.. 80

intrinsics describes the cameras inner parameters, such as the focal length, principal point and radial distortion. 50

laser mask The laser mask is an image of 0 and 1 values, where 1 means the pixel at that position is part of the laser line in the image.. 21

logical image A logical image is a binary image, an image consisting purely of 0 and 1. It is often used in image processing to represent where a shape or feature is in another image.. 9

photon a individual ray of light in the electromagnetic spectrum. 69

pixel A pixel is the smallest unit of a digital image or graphic that can be displayed and represented on a digital display device. 53, 72

principal point A principal point refers to the point on the camera sensor surface, bounded by a line that intersects the centre of the camera lens. 51, 53

reflectance the measure of the proportion of light or other radiation striking a surface which is reflected off it.. 72

topology The topology of a shape or object is a description of the shape, irrespective of any deformation, but not tearing or gluing. This means a square and a circle have the same topology, but a toroid and a sphere do not, as the sphere must have a hole torn, and the toroid needs a hole glued together for them to be equal.. 9

vertex a point created by an intersection of two curves, lines or edges. 51

Chapter 1

Introduction

1.1 Thesis

In the course automation engineering at NTNU, the degree is finished with a final 20 point bachelor thesis, to finalize the 3 year class. The thesis is built upon a pre- project report, which builds the foundation of the thesis task and goals.

1.2 Background

Distinguishing decorated packages stacked closely together from each other is not handled well by many computer vision systems that use 3D cameras or 2D cameras alone. Being able to reliably pick out a single package from a stack is important for automation systems handling such materials. Especially for semi-automated systems such as the one being developed by Currence Robotics, which cannot use a fully automated system where the position of packages can be accurately recorded and known. In these cases the system needs to be able to accurately locate them itself, and be able to handle a wide range of package positions and orientations.

1.3 Issue

The thesis ”Laser assisted optical gap detection” is given by Currence Robotics, an automation company that wishes to improve logistics through technology and innovation. They currently work with HI Giørtz, a wholesale company that owns a number of intermediate storage locations, which currently are fully operated by humans. There is now a desire to find a way to automate these storage sites without needing to rebuild the buildings to allow full automation. This is to be done by designing a robot that will be able to work alongside humans, and that does not require any significant infrastructure to operate.

1.4 Approach

The initial purpose of the project was to create a system that would be capable of handling transparent products, and products wrapped in plastic. We gathered data on the type of plastic commonly used for wrapping products for transport, and the reflectance of said plastic. We then used our knowledge of image processing as well as taking inspiration through articles, such as a thesis done by Josef Anzengruber on triangulation, to plan and build our own solution to the problem. This was later narrowed down to detecting gaps between regular cardboard boxes due to the scope and technical difficulties of the task. We also got help from the team Currence Robotics and Hi Giørtz, both by being given access to a Giørtz storage site for testing the solution, and suggestions and guidance during the project.

Chapter 2

Theory

2.1 Camera properties

2.1.1 Field of view

The field of view (FoV) describes the the amount of area captured by the image sensor, and the wide angle of the camera lens. A more concrete definition of the FoV splits the term into sub classes

- Angular Field of View (AFOV)

  Most commonly, the AFOV is what everyone relates to when they hear the term "Field of View". It is the total angle span shot from the camera lens, and is most often referred to as the magnitude of the angle if not specified.

- Vertical Field of View (VFOV)

  The Vertical Field of View is the vertical distance measured in metres across the plane projected by the camera. It is a direct relationship between the angular field of view and the working distance of the camera setup.

- Horizontal Field of View (HFOV)

  Horizontal FOV is typically used in discussions of FOV as a matter

of convenience, but the sensor aspect ratio must be taken into account to ensure that the entire object fits into the image, where the aspect ratio is used as a fraction (e.g. 4:3 = 4/3). While most sensors are 4:3, 5:4 and 1:1 are quite common too [2]
(When a datasheet for a digital camera sensor lists the "FOV" it is the maximum "Angular Horizontal Field of View, if not specified".)

$$\text{Horizontal FOV} = \text{Vertical FOV} * \text{Aspect ratio}$$

2.3.

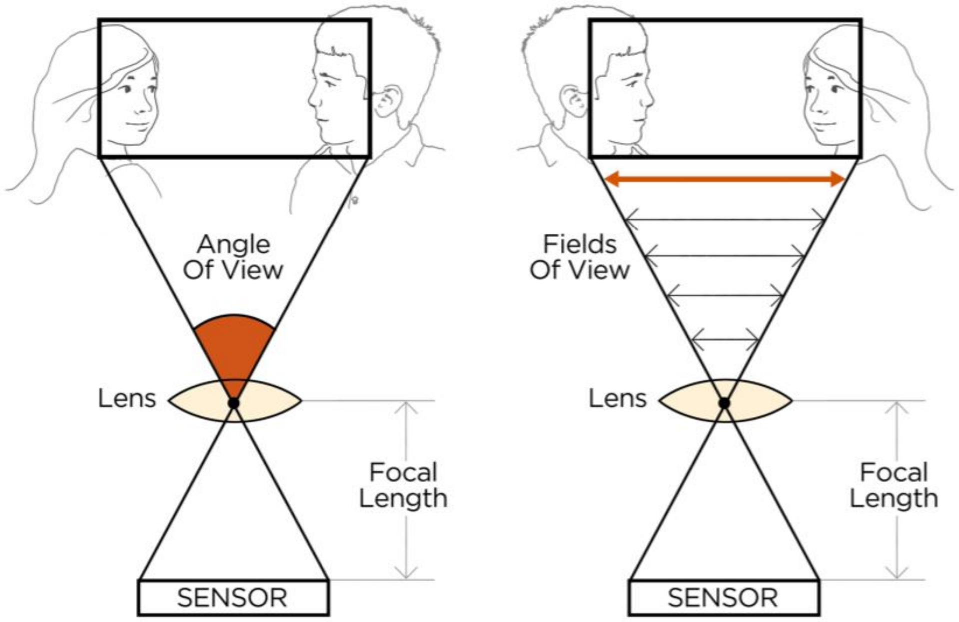


Figure 2.1: AOV and FOV of the same image

2.1.2 Camera parameters

A camera has many parameters that are either hard coded or can be modified to give different image results. Web-cameras often change these values themselves by default to try to give as good quality images as possible.

We will here only list a short summary of the parameters relevant for this project. For a more in-depth explanation of how the parameters are used in this project, see chapter 4.2.

Exposure	How long the camera sensor is exposed to the incoming light. A long exposure time leads to more light hitting the sensor, and therefore a higher value reading in the sensor.
Contrast	Contrast is how large the difference between colors is. To increase the contrast in an image, the camera essentially stretches the colors to increase the "distance" between the pixels in color space.
Sharpness	Sharpness is how "sharp" the edges in the image should be. If the sharpness is increased, the camera will attempt to find edges in the image that are smooth, and make them sharper.

2.1.3 Radial Distortion

As there is no “perfect lens”, any digital camera will always be subject to **radial distortion** to some extent. Luckily there are algorithms to correct it, but there will always be a minor flaw, as the corrected image will only be an **estimation**.

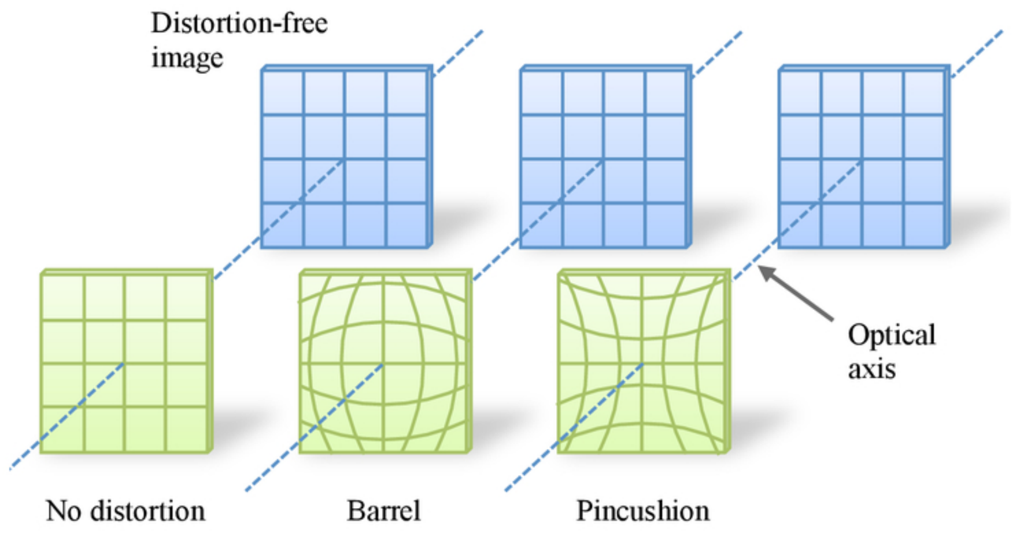


Figure 2.2: Radial distortion about the optical axis

Radial distortion can be thought of as an uneven magnification spreading in a circular pattern, originated from the image centre. The “Barrel distortion” is the most common distortion, where the centre seems magnified and the edges are compressed by it. Pincushion is the opposite effect, where the centre is seemingly pushing away from you, as the name implies, a cushion compressed by a resting head.

2.2 Camera calibration

Camera calibration is the process of estimating intrinsic and/or extrinsic parameters. The intrinsic parameters describe the cameras internal parameters, such as the focal length, radial distortion and principal point. The extrinsic parameters describe the cameras position and orientation in space.

2.3 Image processing

Image processing is any technique that allows one to extract some information from a digital image. The topic is too broad to be covered in depth here, but the main techniques we applied in this project were the following.

2.3.1 Thresholding

Thresholding is the act of isolating all pixels who's value is within a certain range. This can be used for things like isolating and removing a greenscreen in an image, and in our case we used this to isolate the reflection of the laser line from the rest of the background.

2.3.2 Line detection

Line detection, as the name implies, finds pixels in an image that together form a coherent line in the image. There are multiple methods of accomplishing this, with the most common being "Houghs transform" and "Probabilistic Houghs transform", the latter being a faster version of the first by looking only on parts of the image, at the expense of some accuracy.

2.3.3 Convolution

Convolving is the act of taking a square pattern, a “kernel”, and iteratively compare it to all areas of the image. This can be used to find areas in an image that matches said pattern, or for doing mathematical operations such as derivating, integrating or smoothing an image. This together with thresholding are the most basic and most used techniques in image processing.

2.3.4 Skeletonization

Skeletonization is finding the “skeleton” of a logical image. The skeleton is in a sense a thin version of the shapes in the image, thinned down to 1 pixel wide lines, while maintaining the topology of the shape, as can be seen in figure 2.3.

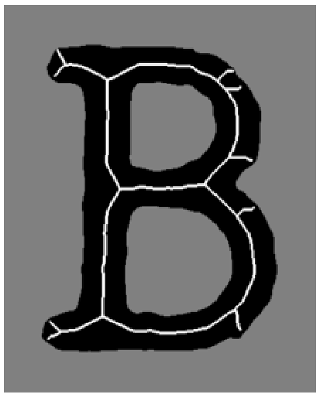


Figure 2.3: Example of skeletonization of an image depicting the letter “B”. Image courtecy of wikipedia

2.4 Triangulation

Triangulation in geometry and trigonometry, is measuring known points and angles to determine points in space, lengths and angles, with the help of comparing right triangles.

Chapter 3

Materials

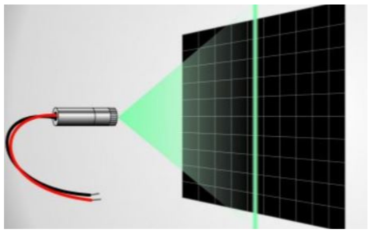
3.1 Webcam

Figure 3.1: Logitech C270 USB HD Webcam
logitech.com

640 x 480 USB.2 web camera.

3.2 Laser

(a) Laser Tip



(b) Line Laser example

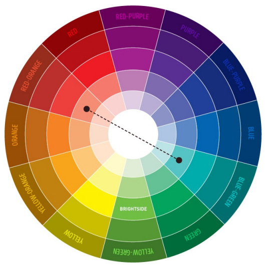
Figure 3.2: LLM Line Laser Module GREEN
lasershop.de

Line lasers project a line, rather than a point, in front of them.

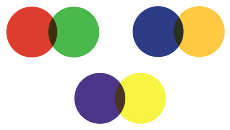
There were 3 laser used throughout the project.

1. A green 25mW 505nm laser line module with a 120 degree FOV.
2. A blue-violet 25mW 405nm laser line module with a 120 degree FOV.
3. A blue-violet 80mW 450nm laser line module with a 90 degree FOV.

The reason for changing the initial green laser to a blue-violet laser was because the green laser worked very poorly against some colors, such as red. Likely a result of combining complimentary colors. And also against black surfaces, where the reflection became very weak.



The blue-violet laser worked much better overall on all colors it was tested against, and was also better reflected off of packaging plastic. The reason for why this worked better is mostly due to no complementary colors of the ultraviolet part of the laser. The result appeared poorer to the human eye, but it is possible the camera is more sensitive to this range of electromagnetic light.



While changing the wavelength of the laser made the images clearer, it still had struggles with transitions from dark to light materials, as can be seen in figure 3.3. This was why the third laser was bought. With $\frac{80mW}{25mW} = 3.2$ times more power (5dB), and a narrower FOV, meaning the light is more concentrated leading to an additional $\frac{120^{\circ}}{90^{\circ}} = 1.33$ times (1.25dB) more power, giving us a total of 4.26 times (6.3dB) more power. This leads to the reflected laser light being even more powerful, allowing us to drown out even more of the background light (see 4.2). This also reduced the spread seen when directing the laser at a white surface, it became possible to transition from a black to a white surface without the line becoming spread out and noisy, as can be seen in figure 3.4.

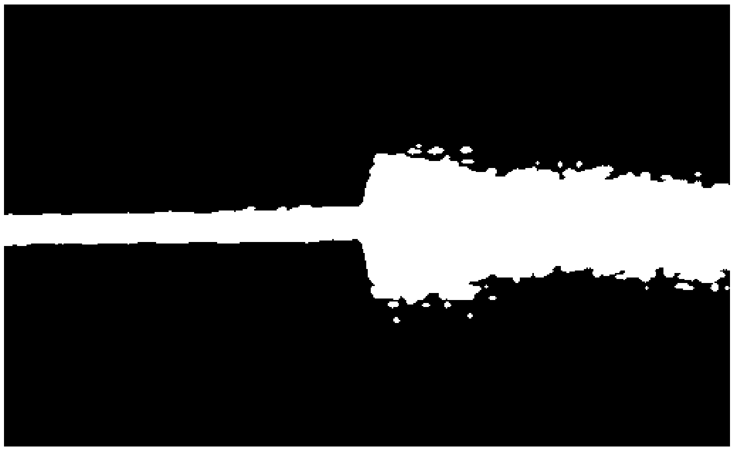


Figure 3.3: Laser line on black and white surface

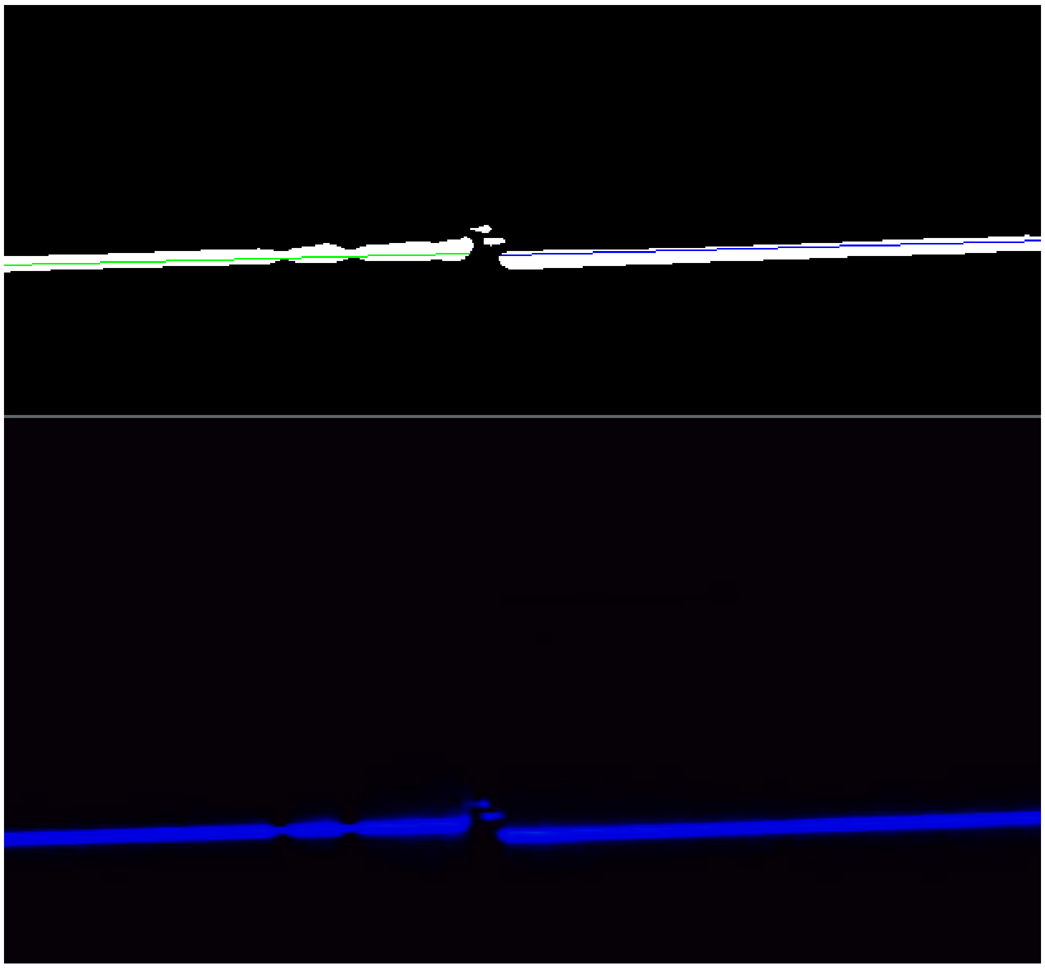


Figure 3.4: Mask and raw image when using more powerful laser on the gap between white and black box

3.3 Chessboard calibration sheet

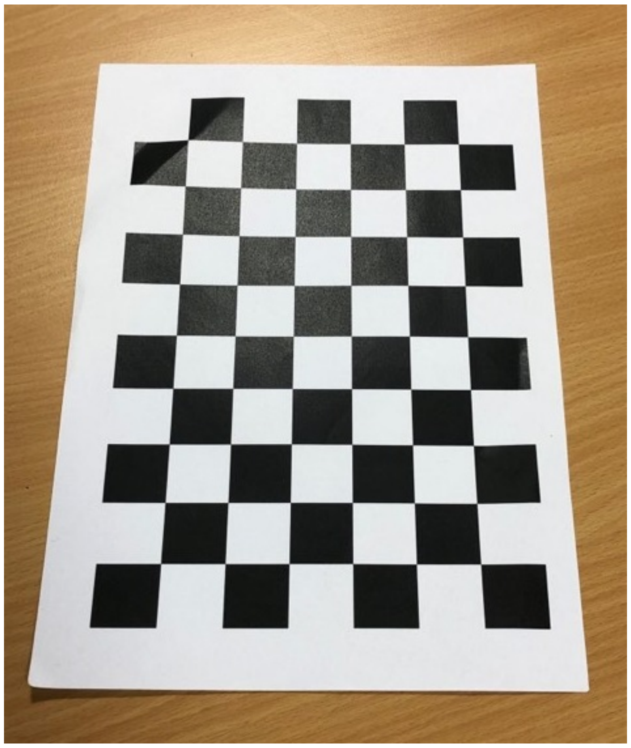


Figure 3.5: Paper sheet of a definite square chess pattern

Traditionally ink printed sheet of paper with a pattern of 25mm x 25mm chessboard squares. Tool for calibrating digital web cameras with Zhang's method.[5]

3.4 Software

Program/application	Version
Python	3.6.5
pip	9.0.1
OpenCV	4.0.0.21
numpy	1.16.1
Matlab	9.3.0.713579 (R2017b)
Git	1.20.1
Latex	LaTeX2e

Table 3.1: The programs and features used in the project and their version

Chapter 4

Method

4.1 Line-based gap detection

The initial approach to solving this problem was to first take a regular 2D color image pointed at the reflection of the laser. We would threshold it to isolate the reflection of the laser in said image, and then use line detection to find connected segments of the laser, and look for discrepancies such as the line being split. This could then be expanded to detecting areas where the line was curved and so on.

In the initial tests at using thresholding to isolate the laser reflection in an image, we used the green laser and a HSV threshold on a photo with a weak Gaussian blur applied. This showed that while the laser could be picked out even at a distance, it was very susceptible to noise and colored objects, as seen in figure 4.1. We decided to use HSV over RGB because HSV has the advantage of separating color and intensity, meaning we could easily threshold a specific range of color, at any intensity.

This lead us to explore something that had already been done by the team at Currence, namely modifying the camera parameters such as the brightness and exposure. see section 4.2.

Finding the appropriate camera parameters and threshold values was done with a little tool we wrote ourselves that allowed us to modify the camera parameters and threshold values while viewing the live video feed and the

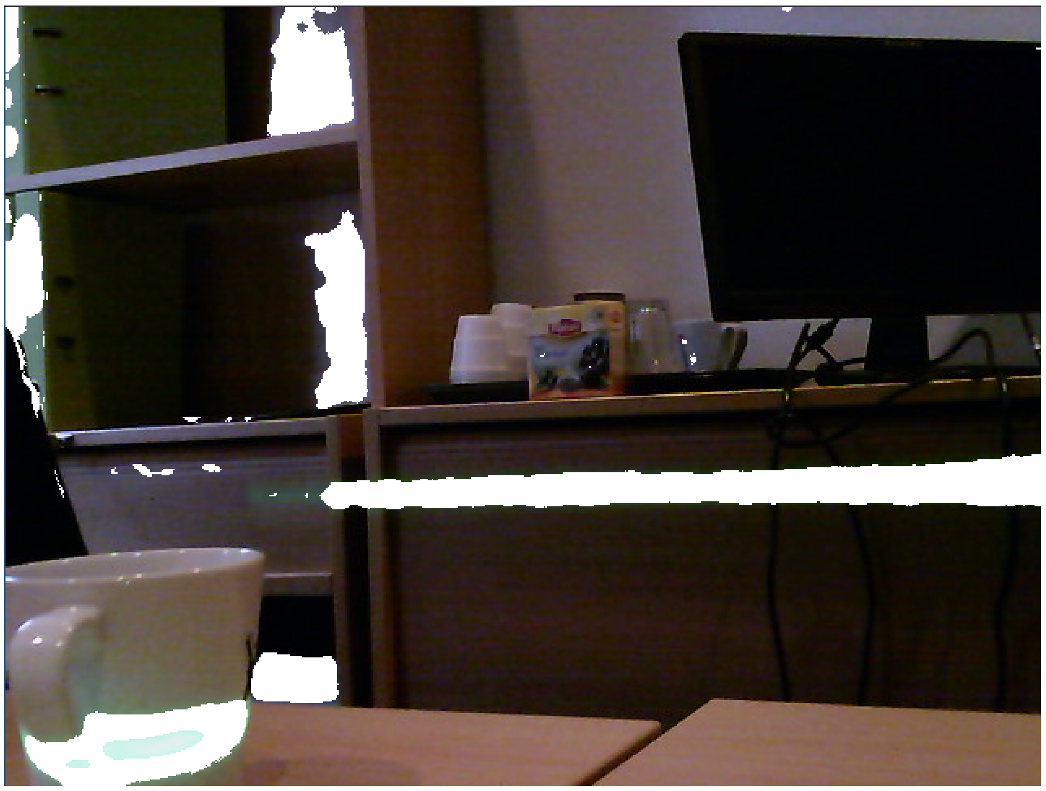
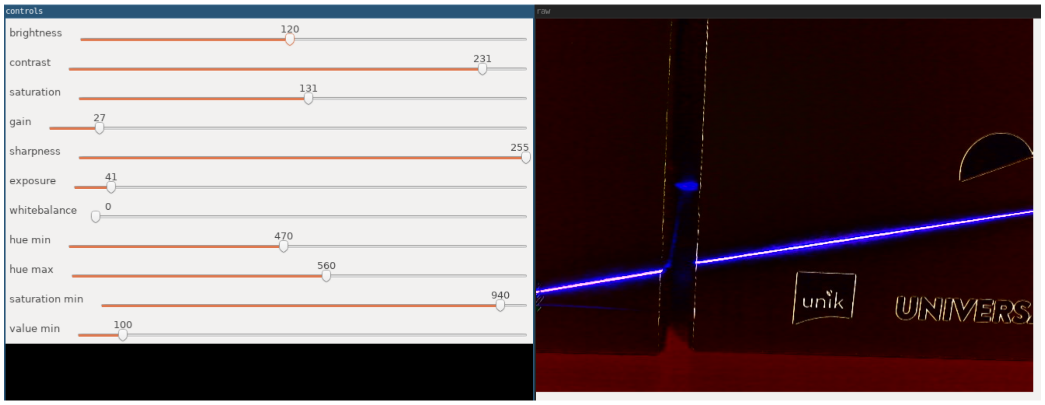


Figure 4.1: First thresholding test with green laser

result of the threshold (see figure 4.2). This was done to find the best values for both parameters and thresholds which were then hard coded into the test program for future testing. This means that the system will be somewhat sensitive to changes in ambient light and especially to changes in the intensity or color of the laser, but automatically adapting to these changes was decided to be outside the scope of this project.

Then we found the skeleton of the image to find the center of the laser, and then applied a Hough transform to find the lines in the image. Using the normal Hough transform gave us promising results, see figure 4.3, though determining where a line started and ended was difficult. Because of this, we switched to the probabilistic Hough transform, which not only finds lines but also their start and end position when using the OpenCV implementation. This also gave the added benefit of being significantly faster than the non-probabilistic version.

We did have troubles with the resolution of the hough transform, it was unable to detect lines of certain angles, and it would often return multiple almost identical lines, only differing by a slight angle, this was a problem in both implementations of the hough transform and can be seen in figure 4.3. The latter was mended by finding the shadows of the lines cast down on the x-axis, and when the shadow of two lines overlapped by more than a certain value, then the shorter one of the two is removed, this method will be explained more in depth later.


controls

brightness 120

contrast 231

saturation 131

gain 27

sharpness 255

exposure 41

whitebalance 0

hue min 470

hue max 560

saturation min 940

value min 100

Figure 4.2: Program that allows camera parameters and threshold values to be modified live.

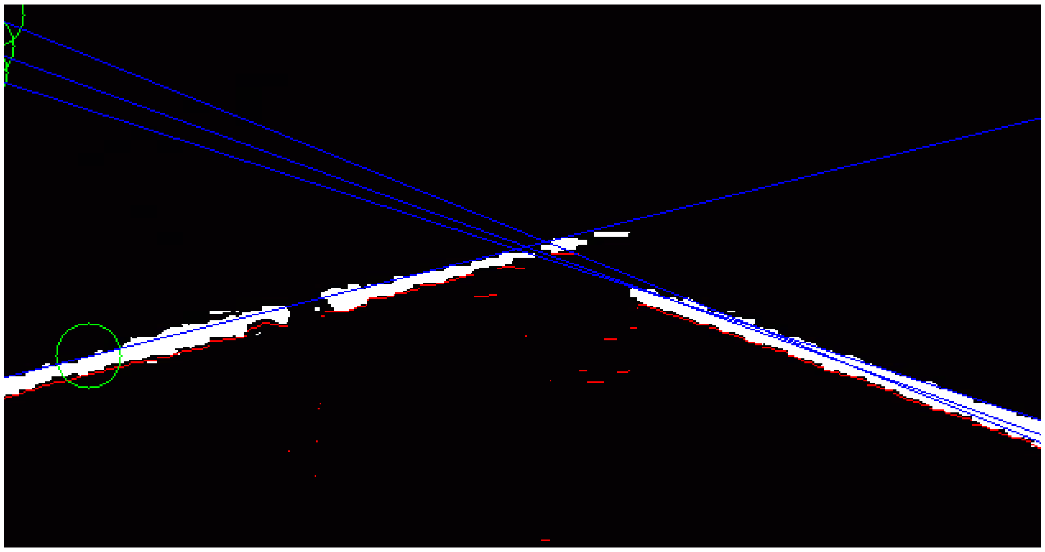
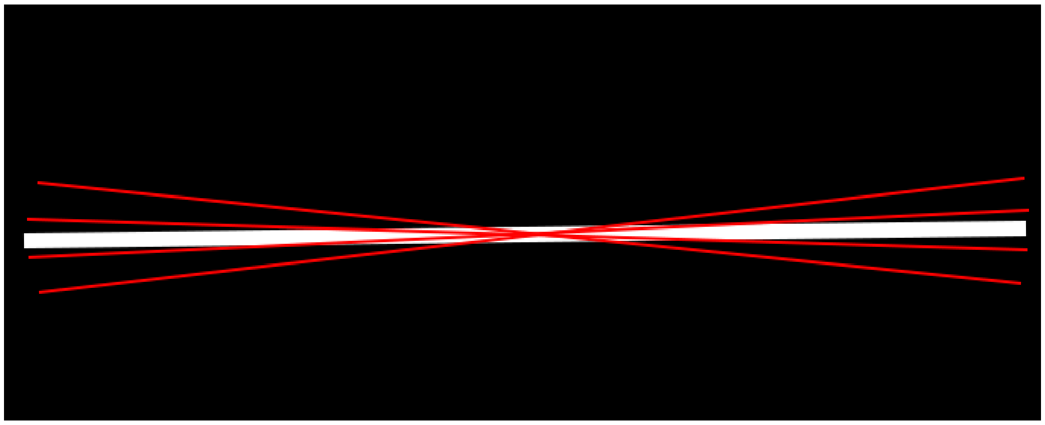


Figure 4.3: Hough transform of corner image

The former issue was more difficult to solve. Since the Hough transform looks for lines with specific angles, it cannot scan for every angle possible. There would be a practically infinite number of angles to test. So it quantizes the angles and checks the different angles in discrete steps. But if we have a line in the image that's in-between those steps, it won't be correctly detected, demonstrated in figure 4.4a. Increasing the angular resolution didn't seem to have an effect after a certain point, so we ended up simply dilating the skeleton to make the lines thicker before finding the lines (figure 4.4b). In turn it caused us to abandon the skeletonization method all together, and simply apply the laser mask directly. This meant we traded accuracy to ensure we had results (see figure 4.5, and they were deemed to be close enough for our purposes.


(a) visualization of hough transform angle quantization. The white line is the laser and the red lines are lines tested for by the hough transform

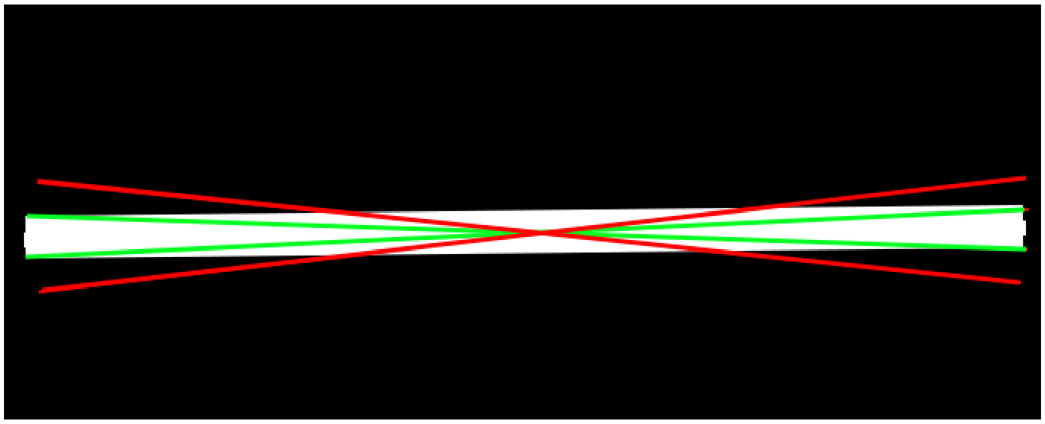

(b) Same example as figure 4.4a but the laser is dilated. The green lines are lines detected by the houghtransform

Figure 4.4

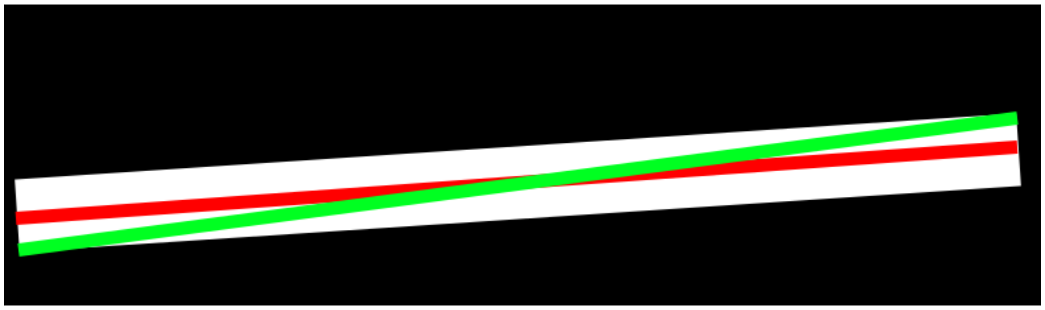


Figure 4.5: Example of hough transform on laser mask. Red is the desired line, green is the line actually found

4.1.1 LSD-based line detection

An alternative to the Hough transform for finding lines in an image is simply called "Line Segment Detector" or LSD for short. It has the advantage of speed and not being as restricted on what angles are tested for as the Hough transform, but tends to find only shorter line segments if the line in the image is uneven. We attempted to use the OpenCV implementation of this method as an alternative to the Houghs transform, but in order to use it for our purposes we needed to solve some of its limitations.

The LSD algorithm is meant to work on thin lines in images, often found by first running edge detection on the image. We were more interested in finding the center line of the laser reflection, and so we once again took the skeleton of the mask produced by the threshold process. Because this skeleton often is quite uneven (see figure 4.6) the LSD often found smaller line segments on this skeleton instead of one long line following the entire length of the skeleton

Since our system needs to find one or two long lines in the image, and the LSD detects mostly short lines, as shown in figure 4.7, we needed to be able to stitch together the segments into longer lines, and to do this we would have to find line segments close to each other and pointing in the same direction, and then merging them, and this had to be done until we are left with the long full lines we are looking for.

We started doing this the naive method, starting with a set of all the lines, then comparing every line in the set against every other line, and seeing if

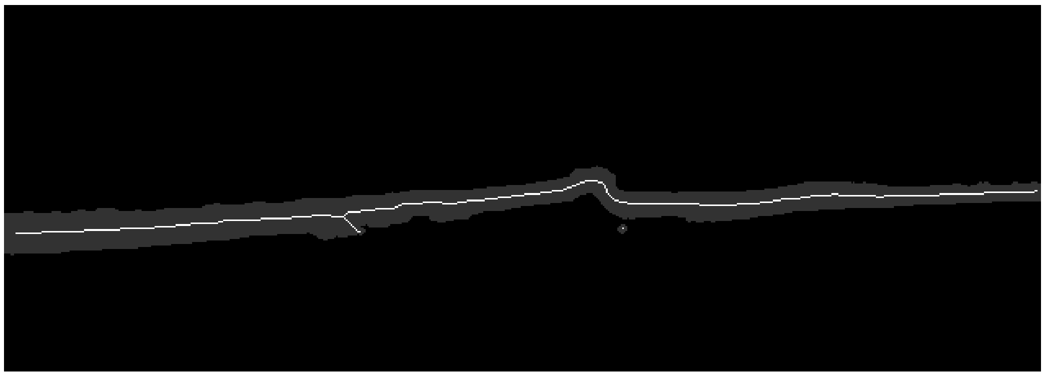


Figure 4.6: Example of skeleton of laser mask

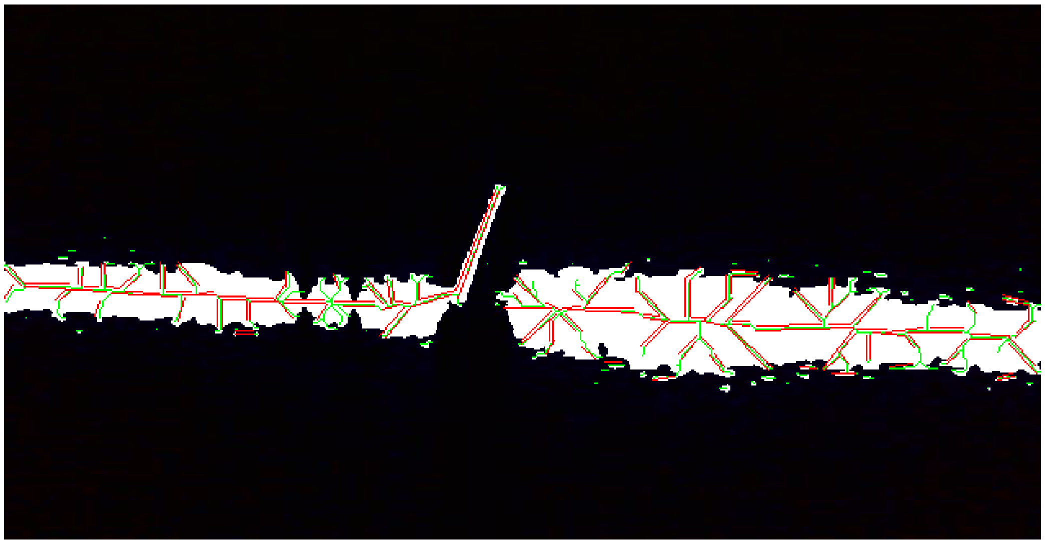


Figure 4.7: raw output from the LSD algorithm on top of laser mask and skeleton. Red lines are lines found by the algorithm.

any of their endpoints are within a certain distance. If so, then we check their angles and see if they are pointing in more or less the same direction. If two lines are found that meet these criteria, then we create a new line from those two, and append it to the set of all lines.

This method was written in pure python and ran on a single CPU core, it's complexity in big O notation is $O(n^2)$, but since we only need to compare every line *i=0..n-1* against every line *j=i+1..n*, we reduce it to $O(\frac{n^2}{2})$, though it's still has an exponential complexity. In addition, we had to repeat this process until no lines were found that could be merged that had not already been merged, and since we then iterated the same list again and again we ended up testing lines that had already been tested before again and again, so when the lineset approached 1000 or more lines, the method would spend the majority of the time comparing lines already compared and wasting time, often using multiple minutes to complete the calculation.

The process was optimized in two ways, the first was doing bulk calculations run closer to machine code rather than in python by using numpy arrays. Calculating the distance between line endpoints could be done as a massive matrix operation by splitting the set of lines P into the X and Y vectors P_x and P_y, and finding the distances D as $D = (P_x - P'_x)^2 + (P_y - P'_y)^2$ (4.1). The resulting nxm matrix D then contains the squared distance from every point n to every point m in the set P, with zeros where $n = m$. This improved execution time dramatically as the calculations were done in a lower level language and being distributed among all the available CPU cores.

$$\begin{bmatrix} Px_0 \\ Px_1 \\ Px_2 \end{bmatrix} - \begin{bmatrix} Px_0 & Px_1 & Px_2 \end{bmatrix} = \begin{bmatrix} Px_0 - Px_0 & Px_0 - Px_1 & Px_0 - Px_2 \\ Px_1 - Px_0 & Px_1 - Px_1 & Px_1 - Px_2 \\ Px_2 - Px_0 & Px_2 - Px_1 & Px_2 - Px_2 \end{bmatrix} \tag{4.1}$$

The other improvement was to reduce the amount of redundant calculations. Say the function to merge lines was $F(S)$ where S is a set of lines and the function returns the a set with the new lines found by merging lines in the given set S. What we were doing was running $S[n+1] = S[n] \cup F(S[n])$ until $S[n+1] == S[n]$ ($F(S)$ returns an empty set), but we modified the function to instead be $F(S1, S2)$, where only lines in $S1$ and $S2$ are compared and merged, and the function then returns the newly found lines. That meant we could instead do:

$$N[n+1] = F(S[n], N[n]) \cup F(N[n], N[n])$$
$$S[n+1] = S[n] \cup N[n]$$

Where S is the set of old lines, and N is the set of new lines. So every iteration we compare the new lines to the old set and to itself, then we can append them to the old set, this way all lines in the old set will already have been compared to all other lines in the old set. This changed the complexity from $O(\frac{n^2}{2})$ to $O(k * m)$ where in most cases $k << m$ and $k + m == n$, leading to another significant increase in speed.

Despite all our optimizations, it still ran unacceptably slow, and the results never turned out as expected. The lines were not stitched correctly together to form the long lines we were after, instead it tended to end up simply giving clusters of lines. This can be seen in figure 4.8, and at this point it was decided that too much time had been spent exploring this path without yielding any results, and so it was ultimately dropped.

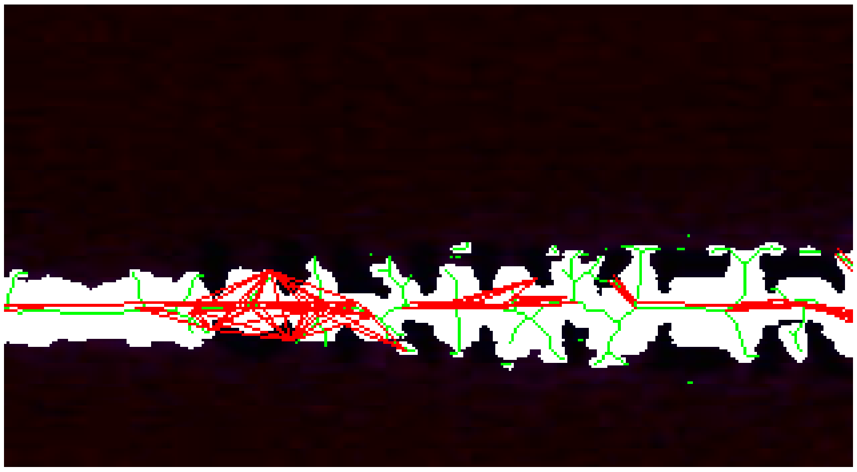


Figure 4.8: Result of lineset stitching function. White is laser mask, green is image skeleton and red is the resulting lines.

4.1.2 Line data evaluation

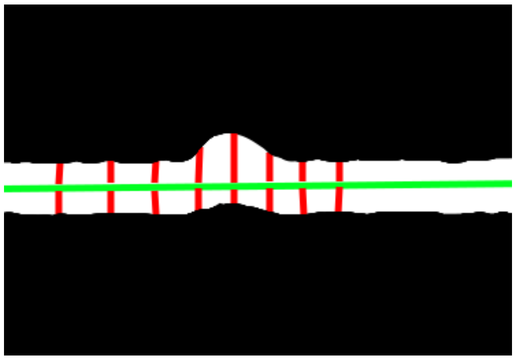
At this point we reverted back to using the probabilistic Hough transform to detect lines, mainly due to the method being proven and reliable, and despite its shortcomings it was deemed the best choice. Now with a method of detecting lines we proceeded to examine what information could be extracted from the lines.

Center variation

The first, and somewhat promising method, was to tune the hough transform so that it would detect lines across gaps, so that if there was a line in the image with a gap, it would detect this as just one long line. We would then move along this line, and at a set interval, measure the distance from the line to the top and bottom of the laser mask.

The idea here was that where there was a gap, there would be a bump up, and as such, the distance from the center of the line to the top would increase and to the bottom would decrease, as seen in figure 4.9

One of the advantages of this method was that one of the big problems with the laser mask at this time, the fact that the laser was more spread when


(a) Illustration of ideal gap

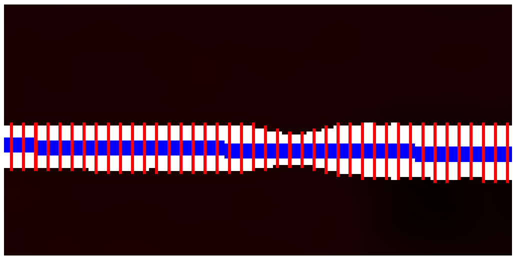

(b) Method in action on decorated surface without gap

Figure 4.9

hitting bright surfaces, and the laser mask would grow thicker when going from a dark surface to a white bright surface, could be ignored. This was done by first combining the measurements into a single vector

$D[n] = |M_{up}[n] - M_{down}[n]|$

Because the laser would grow thicker in both directions, up and down, when hitting a bright surface, this calculation would effectively remove it's influence, and leave us with a number representing how much the laser is shifted in relation to the center-line for this position.

Evaluating this data can be done directly by thresholding this vector with an appropriate value, but to make it more robust to varying conditions and camera resolutions, we can take advantage of the fact that most of the laser line will not have a gap. If we first find the mean average of the vector D, we can then find the variance V as;

$V[n] = (D[n] - AVG_D)^2$

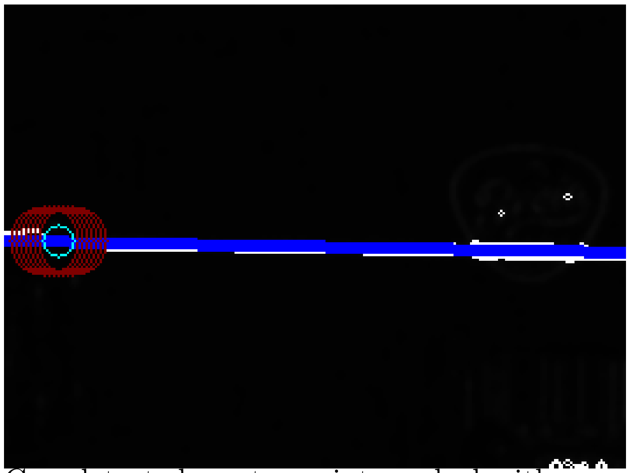
We can then threshold the V vector instead of the raw data in D. This makes it more resistant to varying conditions, as this will effectively scale the threshold if the measurements has higher variation. This could be due to lighting conditions or camera differences.

The areas where the variance is above the threshold is stored in a list for further processing, but we are still missing one important part. If the line in the laser mask has a gap, then the center-line found will cross it. The measurements done at those points will show a distance of 0 from the center line to the top and bottom. These points are also added to the list of important points for further processing.

At this point there could be many points of interest found, but many of these are simply clustered around gaps. In order to determine the number of gaps and their position, we needed to be able to group the points together based on distance.

The grouping algorithm is quite simple. It iterates over all the points, for each point it checks for points within a set distance that are part of a group, if it can't find one, it creates a new group ID and assigns it to the point. If it has two groups within reach, then it joins the group with the lower index. The other group is deleted, and all its member points are added to the first group. This means a single pass over the point list is sufficient to group all points.

This then enabled us to reduce the data down to a set of points centered on their respective gaps, as seen here;

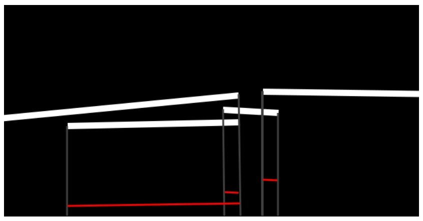

Gap detected, center-point marked with cyan circle

There was also done some testing in ignoring groups with fewer than a certain number of points to filter out points caused by noise, as well as merging results over time and only accepting points and groups persisting over multiple frames. These methods improved accuracy in some cases where intermittent noise was present, but did not solve the inherent problems;

The laser mask extraction at this point gave very poor results. This was because of the poor quality camera, weak green laser and insufficient image filtering. The laser mask would often be very rough, with a uneven, noisy edges. This would lead to the skeletonization function often struggling. This meant that the lines detected by the Hough transform weren't always perfectly centered in the laser mask. This method was therefore at the time deemed too unstable to be used in practice. However, near the end of the project this was reconsidered, see 6.9.1.

Line filtering

Due to the line detection algorithm often giving multiple partially overlapping lines, we needed a way to filter out only the best ones. We do this by comparing all the lines, and when two lines overlapped, the shorter one is removed. This ensured that in the cases where multiple candidate lines are found, only those spanning the entire length of the laser mask is kept. In overlapping we mean that the lines projected down onto the x-axis overlap, this was chosen because the laser line should always be horizontal in the image, and so the lines following the laser should also be horizontal. An example of how this method works is shown in figure 4.10.


(a) Lines are projected onto x-axis. Overlap is marked with red

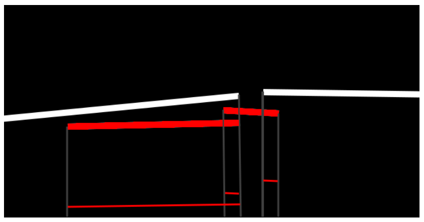

(b) Where overlap happens, the shorter of the lines is removed. Removed lines are colored red

Figure 4.10

Line count

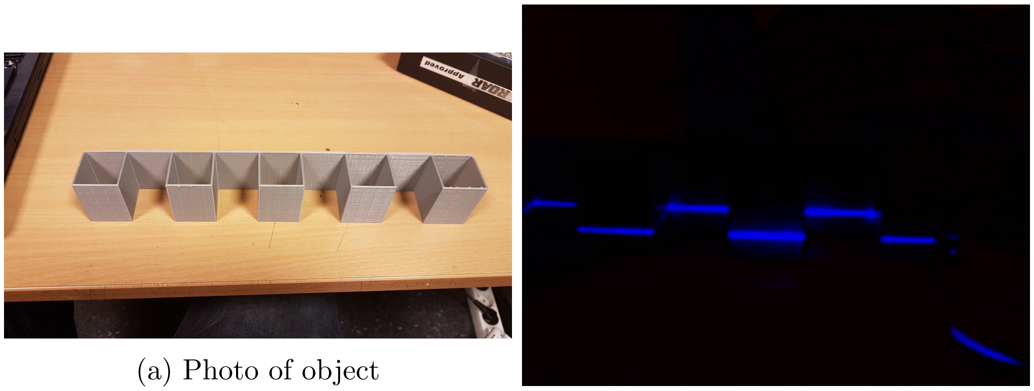
Once only the major lines are left in the image, we pick out the two longest ones. The idea behind this is that the image should only contain either one gap, or no gaps. Therefore we should either have two lines, a line on either side of the split, or a single line spanning most of the image if there is no gap.

If we find that there either is only one line left after removing all the overlapping lines, or that the longest line spans more than 90% of the image, then this indicates that the laser line isn't broken.

On the other hand, if we find two lines, and neither covers the majority of the screen, then we have a few other tests we can perform on them.

Line angle and shift

The distance from the laser to the object determines where on the image the reflection ends up, with more distant reflections being further up on the image. This means that two boxes standing side by side, with one box being slightly further back than the other, will have a shift in the line. This can be seen in figure 4.11. This same property also leads to the laser lines being angled if the boxes are angled.



(a) Photo of object

(b) Laser reflection off of object

Figure 4.11

We wanted to make use of this fact, as boxes most often are cubes with mostly flat surfaces. If the line is angled or shifted this strongly indicates that we're seeing two packages that aren't placed neatly next to each other.

To do this we take the lines, which are until this point defined as a set of coordinates, and convert it into a more mathematical representation $y = a * x + b$. This is easily done by finding the gradient $a = \Delta y/\Delta x$, and then using one of the two endpoints P to find the offset $b = P_y - atan(a) * P_x$, which simply finds at what height the line intersects the Y-axis.

We can now check if the gradient differs by a significant amount, we first calculate the angle of the gradient, $angle = atan(a)$, and we compare the angles of the two lines. Through testing we found that when two lines was erroneously found on the same package, their angles were never more than 1 or 1.5 degrees, so when comparing angles we set a threshold of 2 degrees for it to be considered valid.

We did the same for comparing the shift between the two lines, a difference of more than 10 pixels is considered valid.

Line evaluation result

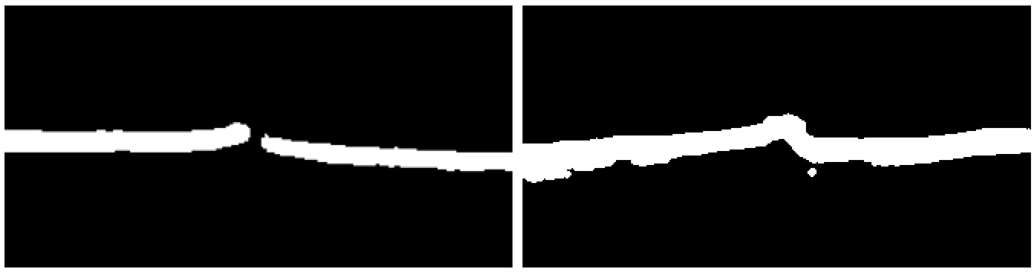
The line evaluation method returns a set of tags that apply to the image it was given.

NOLINE	No lines found in image
FULLLINE	only a single line found, spanning the entire image
ENDED	only one line found, which ends abruptly
BROKEN	two lines found
ANGLED	two lines found, they are angled on each other
SHIFTED	two lines found, they are shifted in relation to eachother

Notice that none of the features actually prove that there is a gap, they just strongly or weakly indicate that there might be one. This is because of the inherent unreliability in the line detection algorithm, and the fact that varying package types, colors and lighting affects how well it's able to pick out what we're interested in. Interpreting these tags and performing the multiple checks necessary to make sure isn't handled by the method itself.

4.1.3 Convolution based gap detection

We focused largely on the line-based method for most of the duration of the project, and had yet to attempt any feature extraction using convolution (see chapter 2.3.3). But after taking and examining a range of test images, it was apparent that the gaps in the images could be grouped into two main categories. The ones where the packages are sufficiently distanced from each other, leading to the laser line appearing split on the image. And the second being where the packages are so close that the gap only manifests itself as a small "bump" on the laser line in the image. See figure 4.12.



(a) Gap manifested as a gap in the laser line (b) Gap manifested as a bump in the laser line

Figure 4.12: The two dominant types of gaps

We then realized that for there to be a gap in an area of the image, a set of features would be present. For the case where the laser line is broken, you would have two edges, a left and a right edge, where the line stops and then starts again. For the bump, you'll have a horizontal line that intersects the laser right where the bump is, and the inverse is true if the laser has a upwards bump on the bottom. Se figure 4.13;

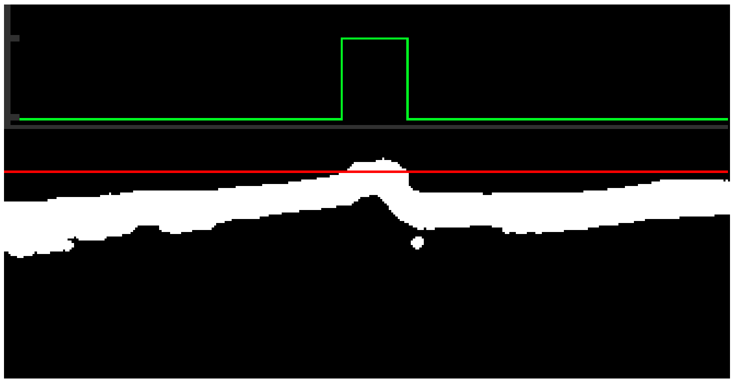


Figure 4.13: Where a bump is present, a line can be drawn that intersects with the bump, but not with the line

By finding these different features in the image we could then look for areas of the image containing all the features associated with a specific gap type.

We started by designing convolution kernels that would be able to detect the different features we were looking for. The simplest was the edge detecting kernel, it can be seen in figure 4.2. This would give a positive response only in the areas where there's a relatively straight vertical left edge, and the same kernel with the x axis reversed detects the right edge.

$$\begin{bmatrix} -2 & 0 & 1 \\ -3 & 0 & 2 \\ -4 & 0 & 3 \\ -5 & 0 & 4 \\ -4 & 0 & 3 \\ -3 & 0 & 2 \\ -2 & 0 & 1 \end{bmatrix} \tag{4.2}$$

We also made kernels to detect the bumps and the inverted bottom bumps, these kernels were scaled to work with a specific camera resolution and object distance, but the basic idea is as follows;

$$\begin{bmatrix} -1 & 0 & 0 & 0 & 0 & 0 & -1 \\ -1 & 0 & 0 & 1 & 0 & 0 & -1 \\ -1 & 0 & 0 & 0 & 0 & 0 & -1 \end{bmatrix} \tag{4.3}$$

The idea being that in order for the kernel to trigger, there would have to be a protrusion up from the laser that reaches the positively weighed center without touching the negatively weighed edges, this is shown in figure 4.14.

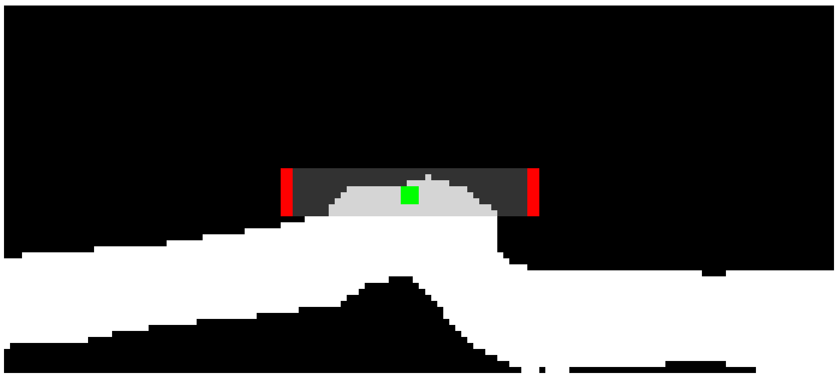


Figure 4.14: Bump kernel triggering on bump in laser mask

To detect the upwards bump on the bottom of the laser line, we used the same kernel, but inverted

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & -6 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.4}$$

Notice that the negative center is scaled to be equal to the sum of the positive edges, this ensures the kernel always will give zero or less when the center is triggered. The idea is otherwise the same as for the top bump kernel, but looking for an inverted bump instead. See figure 4.15.

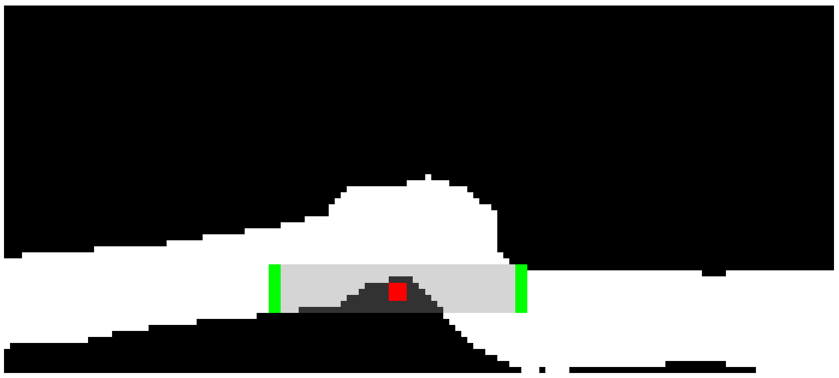


Figure 4.15: Inverted bump kernel triggering on inverted bump in laser mask

4.1.4 Convolution data evaluation

These different convolutions give a set of matrices with how large a response the kernel gave for the area around each pixel. To help us with finding areas containing all the different features associated with the gaps, we first down-sample the matrices, dividing them into overlapping blocks and calculating a new value from this block.

Three down-sample functions were tested. Minimum, maximum and average. Taking the minimum, maximum and average value of each block as the new value in the down-sampled matrix. The problem with minimum and average was that since most of the matrices would be negative or zero, except in the small area where the feature is located. This meant that both methods would for the most part return negative values. This was mended by clipping the values in the matrices, setting all negative values to zero. This mean that minimum would now almost always return zero for all blocks, while average would return small, but positive values.

The thinking behind using average was to mend the problem the maximum function brought with it. Maximum would lead to the value of the block being high if even just a single entry in it was high, meaning that a large area with high response would be treated equally to a small area of high response, meaning noise causing erroneous responses would be an issue. average would take this into account, but the values could then range widely. In the end, it was found that maximum was easier to get working reliably, and this method was therefore chosen.

After down-sampling the response matrices, they were combined into the result matrices. This was done in two ways, each looking for a different type of gap. For the split type gap, the response matrices for the left and right edges and for the split kernel. The result matrix is the minimum of these three feature matrices. This means that if an area has a left edge response and a split response, but zero or low right edge response, then the value would be zero or low, limited to the lowest feature response. This is also done for the bump-type gap, where the same procedure is done with the bump and dimp feature matrices.

The two result matrices now contain higher values where a gap of their respective type is suspected to be. These matrices are combined into one using the maximum function to get a final result matrix containing where any type of gap is suspected to be. When scaling this matrix and overlaying it on the input image, as shown in figure 4.16, it is easy to see where in the image the gaps are.

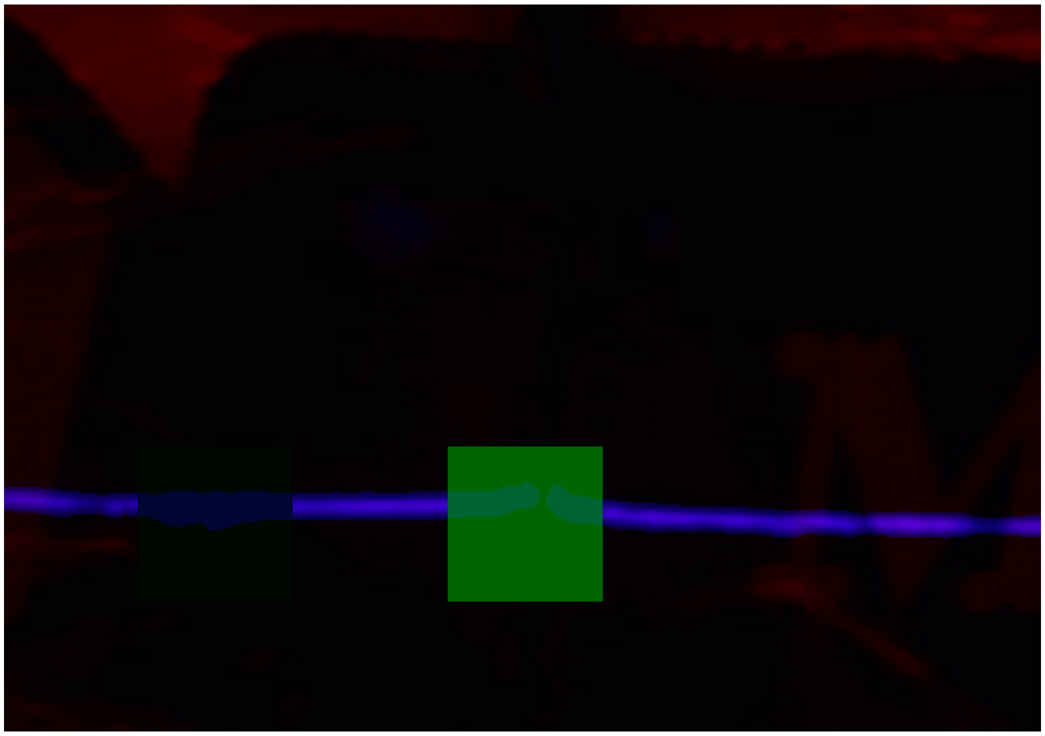


Figure 4.16: Overlaid result of convolution gap detection

Evaluating results

The results from the convolution gap detection can be used directly to locate where in a picture a gap is located, but as the main objective of the project was narrowed down to simply detecting the precense or absense of a gap, this was never implemented. Instead we convert it into a yes or no answer. This is simply done with a simple threshold. If any part of the result matrix is above a certain threshold, then this means it has detected a gap somewhere in the image. We also include the maximum value of the result matrix as part of the evaluation result, as the magnitude of this value indicates how sure the method is that what it has found truly is a gap.

It is important to note that this method is very sensitive to noise, as random noise is likely to trigger the different kernels randomly, and if the noise triggers multiple kernels then this will be perceived as a gap. This can be seen in figure 4.17. Uneven lines, and spots caused by reflections and similar can cause this.

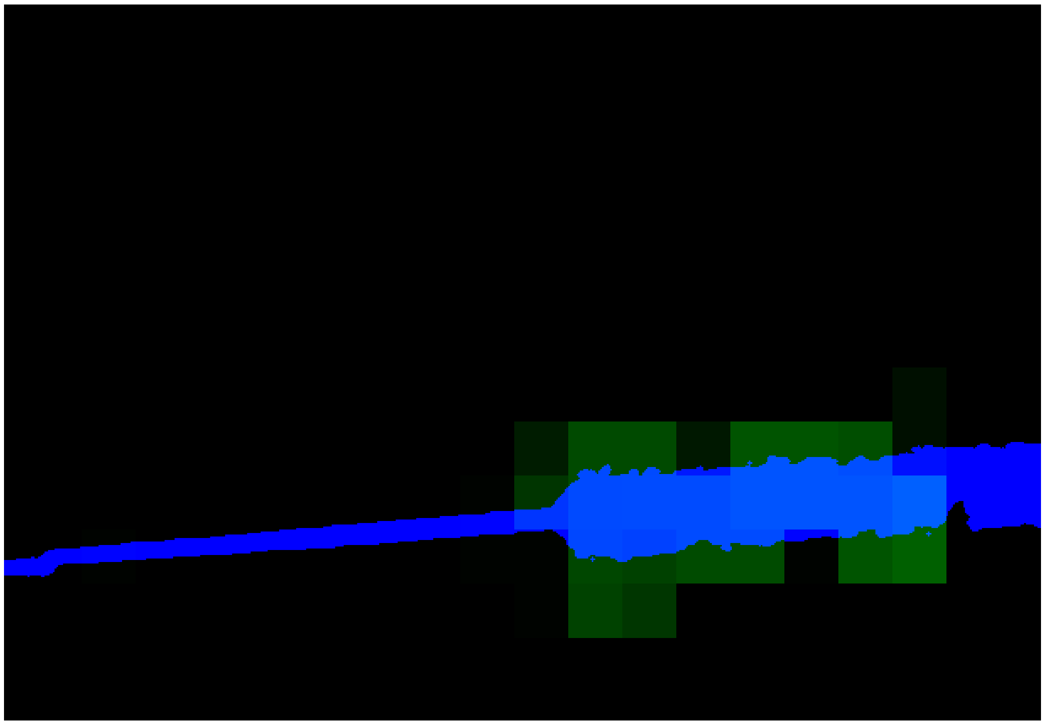


Figure 4.17: Convolution gap detection applied on image where box color goes from black (thin line) to white (thick spread out line)

4.1.5 Combining results

After running the two methods, the line based and the convolution based method, we are left with two very different kinds of result. The line based result, which is a collection of tags, and the convolution result which is a value reflecting how certain the method is that there is a gap. To combine this into a single result we needed a method of weighing the results. We decided to go for a simple rule based system.

We decided to threshold the convolution result, and say that a value over 70% means it thinks there's a gap, and if below then it doesn't. We then used the following rules:

- noline & unlikely mask → certain, no gap
- noline & likely mask → certain, no gap
- fulline & unlikely mask → certain, no gap
- fulline & likely mask → halfcertain, gap
- ended & unlikely mask → halfcertain, gap
- ended & likely mask → certain, gap
- broken & unlikely mask → halfcertain, gap
- broken & likely mask → certain, gap
- angled & unlikely mask → halfcertain, no gap
- angled & likely mask → certain, gap
- shifted & unlikely mask → halfcertain, gap
- shifted & likely mask → certain, gap

These rules are constructed purely through intuition, as the focus was more on the methods themselves. For a production system, more testing and optimization of this ruleset is recommended.

4.2 Camera parameters

Up to this point we had been using the default camera parameters for white balance, exposure time and focus, which are intended to give the best quality colored images possible. What we want is for the camera to only pick out the most powerful sources of light, such as the reflection of our laser, and ignoring everything else.

The exposure, or exposure-time, is a measure of long the camera sensor is subjected to the incoming light. The longer the exposure time is, the more light is received by the sensor. Once a part of the sensor has received a certain amount of light, it maxes out, which means it's over-saturated. Normally the camera adjusts the exposure automatic to get as good an image as possible, meaning the image is as bright as possible, without any part of it becoming over-saturated. But this means the powerful laser reflection, which covers a small portion of the image, will become over-saturated while the camera gathers sufficient light for the rest of the image.

By lowering the exposure, we can make sure that light from ambient sources don't have time to significantly trigger the sensor, while the bright laser reflection makes a significant imprint. This, drowns out all light sources that aren't bright enough.

In addition we can tweak the contrast and sharpness, to further increase the difference between the bright blue laser and the dark background. Importantly we also have to disable auto-exposure and auto-whitebalance, otherwise the camera will automatically adjust these parameters, which would lead to inconsistent and unpredictable results. We want to manually control the exposure, and keep the default whitebalance value.

The difference between the laser and the background is greatly increased with this method, as can be seen in figure 4.18. This makes thresholding the image far easier and less susceptible to noise and blue-colored objects, which otherwise would be hard to separate from the laser and would lead to poorer results during the gap detection.

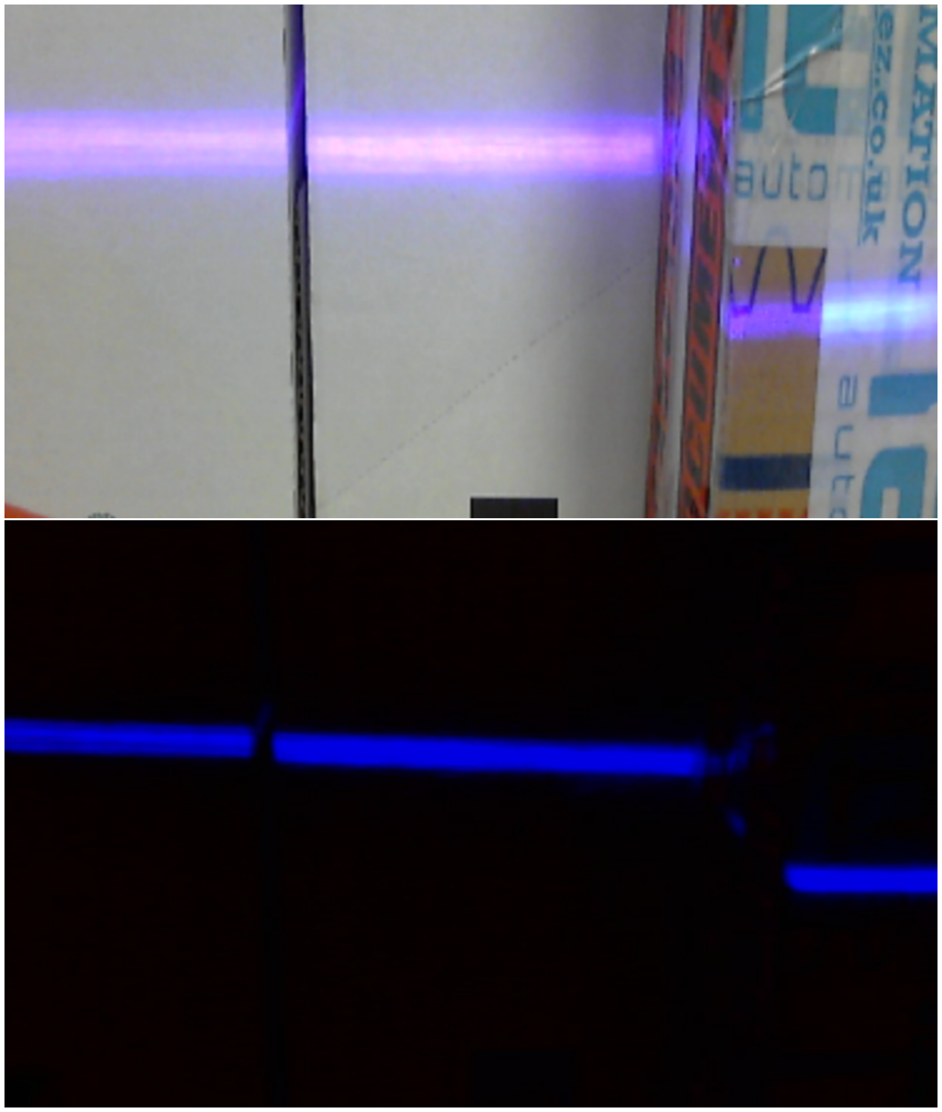


Figure 4.18: Normal image vs image taken with parameters tweaked to enhance laser visibility

4.3 Setup

4.3.1 Laser

The lasers used require a 12V to 24V voltage source to operate, and as we wanted to be able to move and test the system in an actual storage facility environment, we powered it with a 14.8V Li-Po battery pack. We wired the laser directly to the battery except for a mechanical switch on one of the lines to make it easier to turn on and off as needed, see figure 4.19.

The battery, a "Turnigy 14.8V 4.5Ah“ battery pack would in theory be able to power a 25mW laser continuously for over 100 days, and a single charge was enough to last the entire duration of the project. The bundle seen to the right on figure 4.19 is just a part of the adapter cable used to connect the battery.

It's worth mentioning that a separate, dedicated, battery isn't necessary for this system in any way, and in practice it would be getting it's power from the same energy source as the rest of the robot. Presumably a central battery bank.

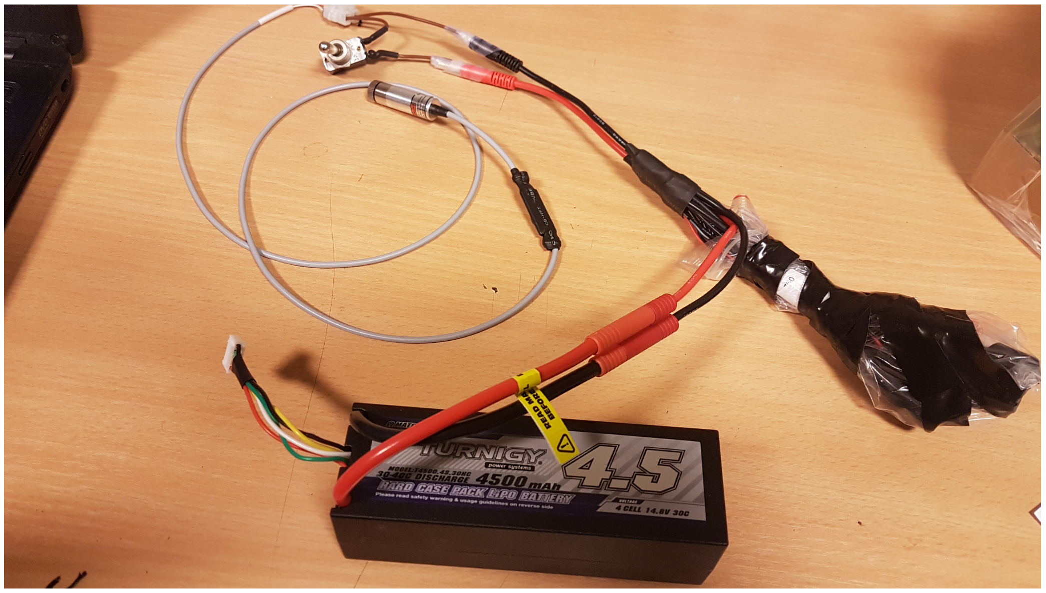


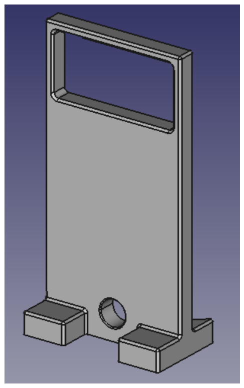
Figure 4.19: laser and battery setup

4.3.2 Holders

For testing

The initial setup for testing during development consisted of a webcamera and laser mounted on a 3D printed holder to keep them at a known distance from each other, shown in figure 4.20b.

We had one earlier iteration of said holder shown in figure 4.20a, which was designed for another camera we were considering using, this camera however was unsuited due to it not being possible to modify it's parameters (see section 4.2). The other camera we chose had an hinge and a mounting contraption, we designed the holder to hold onto the mounting part, allowing the pitch of the camera to be easily adjusted, though this meant we wouldn't know it's pitch angle with any accuracy


(a) Initial design

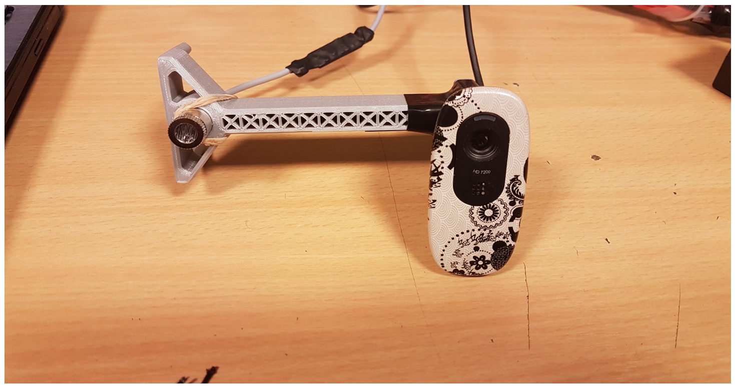

(b) final design

Figure 4.20: Holders

This holder was used for the entire duration of the project for all testing with the system separate from the robot.

On the robot

The robot used an intel D435 camera , a 3D and 2D camera in one. This camera is from before mounted with screws low in the front of the robots arm. In order to use our system we needed to move the camera up some distance, and add a holder to hold the laser module.

Our first and second iteration can be seen in figure 4.21. The holder is designed to fit on the camera mount already in use on the robot, and it moves the camera forward and upwards, moving it forwards was necessary to avoid collisions with other parts of the robot when rotating the arm head. It has holes for hex nuts on the bottom and holes at the top for screws to fasten it to the robot and the camera to it respectively. It also has a built in clamp on the bottom where the laser module can easily be fastened and removed. The two iterations are functionally the same except for the first one using less material, it was replaced with the second one as it was too weak, and would easily deform and break.

The second mount was used to test the viability of the system. In particular, to check that the camera and laser could be mounted with a sufficient distance to each other, without having to go above the profile of the robot arm. When

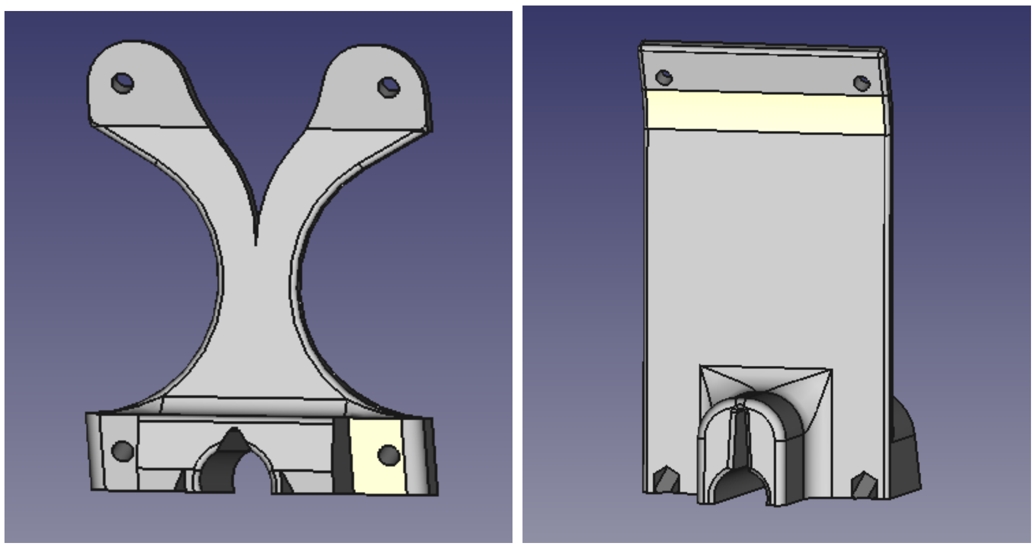


Figure 4.21: The two iterations of the holders to be mounted on the robot

using the mount with roughly 10cm between the camera lens and the laser, it still didn't protrude over some of the other parts of the arm, and so didn't increase the vertical footprint of the arm. See figure 4.22.

This mount was later removed by the team at Currence, and instead of having a holder mounted in the existing camera mount, the existing part with the old camera mount was removed and a new part was designed that had mounting holes to add the camera in the old position, as well as the new higher up position needed by our system, and with a clamp for the laser, similar to the one on our designs. This new holder was used for the rest of the project and during the full-scale tests. The holder can be seen mounted on the robot in figure 4.23.

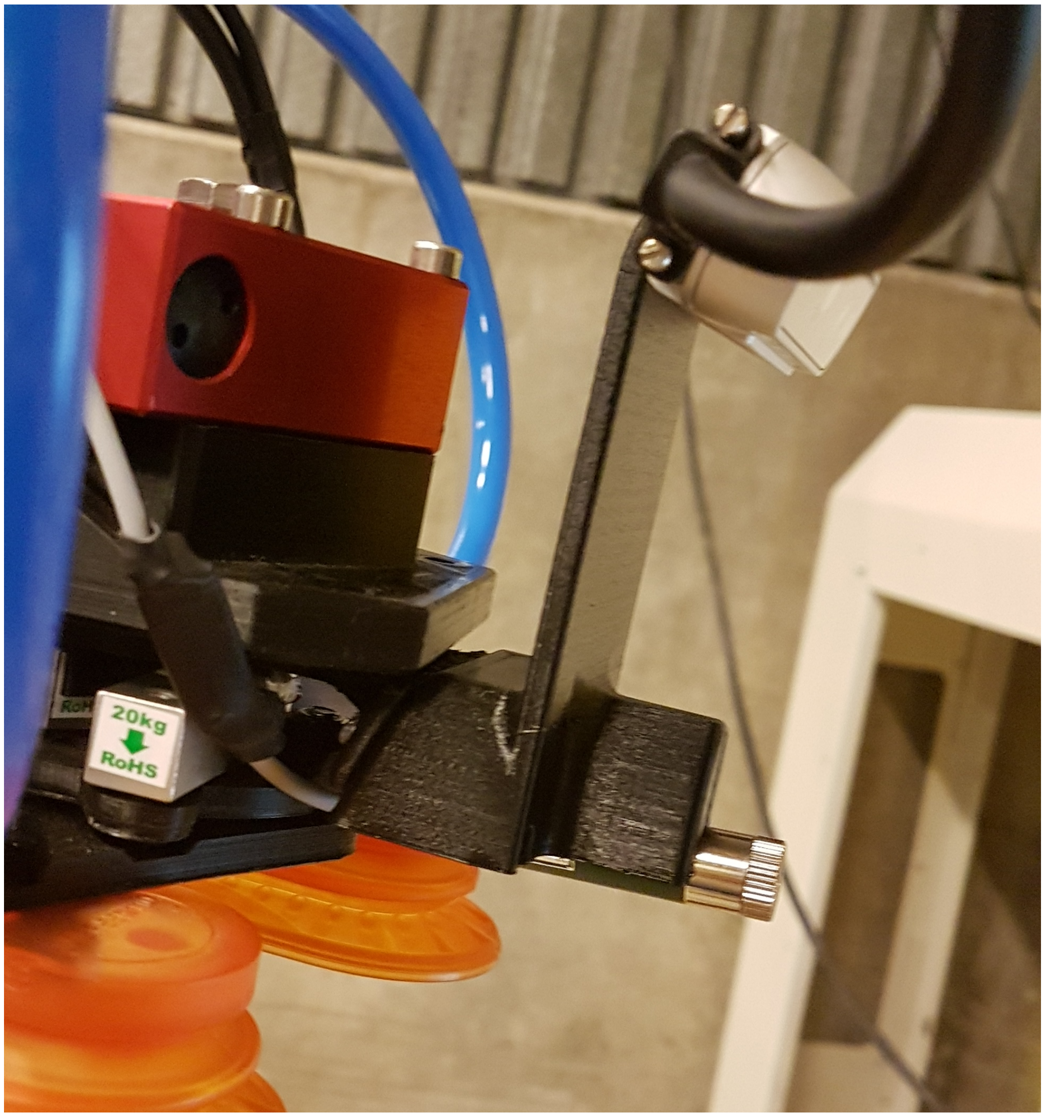


Figure 4.22: Holder mounted on robot, with both laser and camera attached.

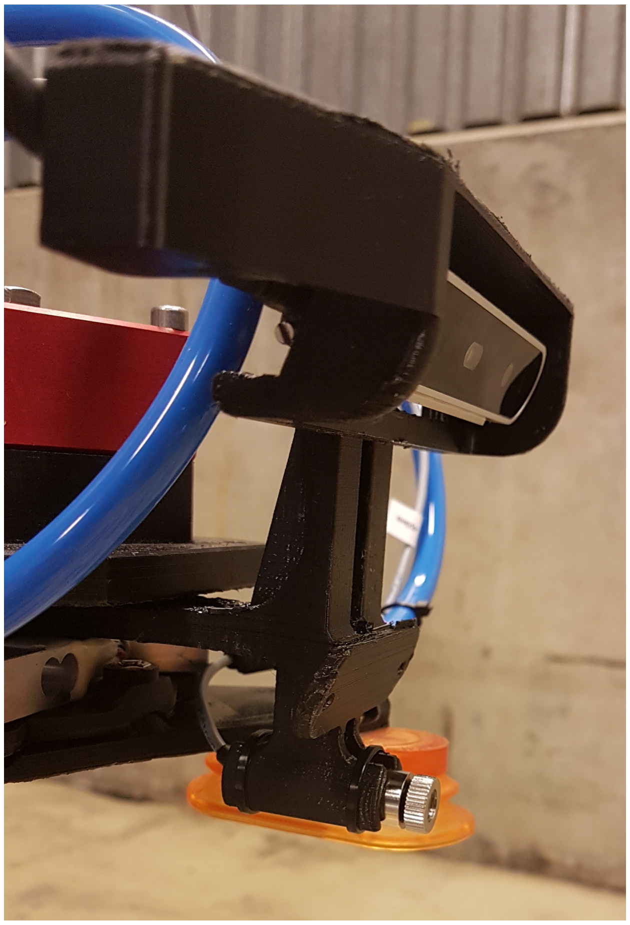


Figure 4.23: Mount for camera and laser built into robot head.

4.4 Image Processing

4.4.1 Calibration

We used Zhengyou Zhang's camera calibration technique, where you snap pictures of a printed chessboard pattern with defined square lengths, at multiple angles. Zhang's method presumes the squares are perfect, (of equal sides and uninfluenced by radial distortion) and knows the lengths of the sides as input.

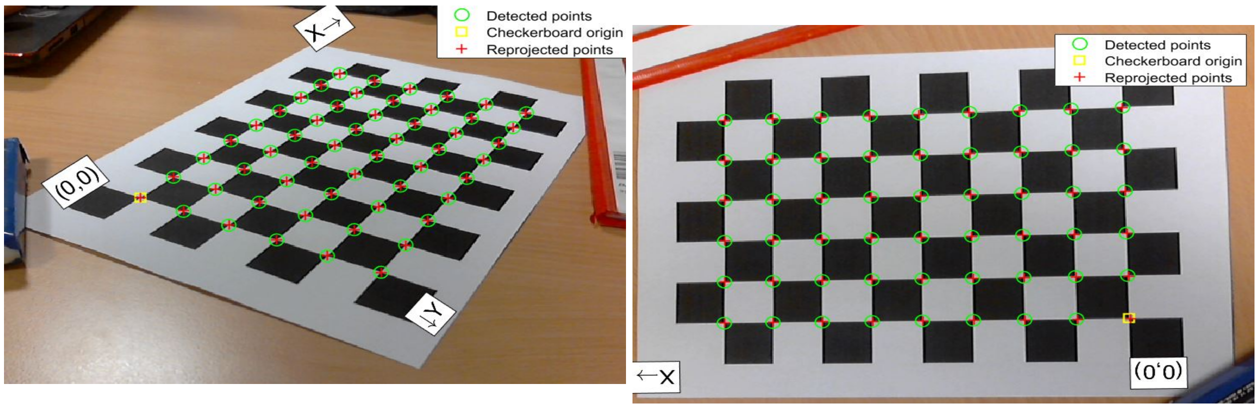
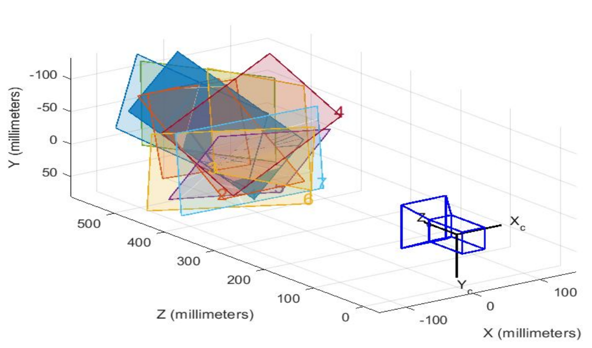


Figure 4.24: Two of many test images

Varying the camera orientation relative to the printed chessboard plane, is essentially the same as varying the position and orientation of the plane with respect to the camera, stationary in space. Thus, the size of the squares and the planes of the entire board can be compared, and triangulated towards the cameras focal point, to make precise predictions of the cameras intrinsics values (Focal length, Principal point and Radial distortion), and extrinsics values. [3]


(a) Camera-centric view

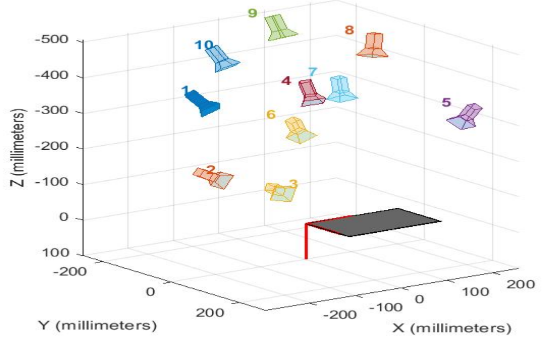

(b) Pattern-centric view

Figure 4.25: Ten test images in matlabs extension: camera calibration tool

4.4.2 Triangulation

The triangulation of the working distance calculation is based upon the known physical constrains, such as the angle where the cameras optical axis and the real life horizontal plane create a vertex, the focal length of the digital camera sensor (in pixels), the principal point of the camera sensor (in pixels) and the vertical distance between the camera and the laser.

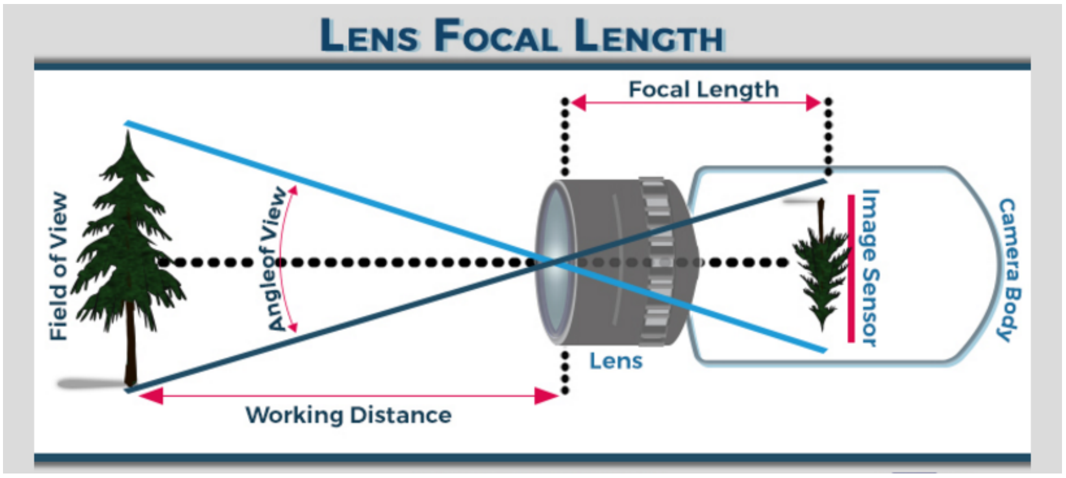


Figure 4.26: similarities inside and outside the camera body
superiorcctv.com

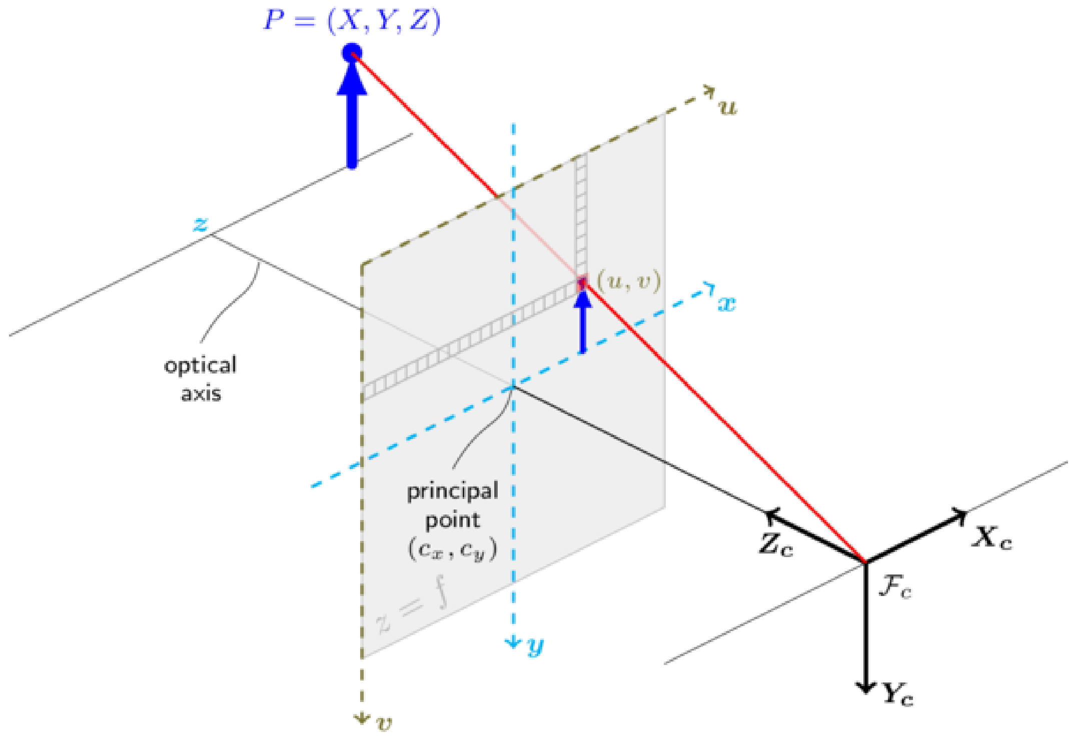


Figure 4.27: 3D representation of a real world coordinate **P**, as viewed from the camera facing Z_c

Considering the 2D projection of the yz- plane in 4.27, Any value for **Y** in **P(Y,Z)** will yield a specific angle, because the value line fired through the plane is fixed (Laser mount) 4.22. Hence the point will not be a line originated from origin as in 4.27 but from an offset from F_c with a fixed angle through the uv- plane 4.27

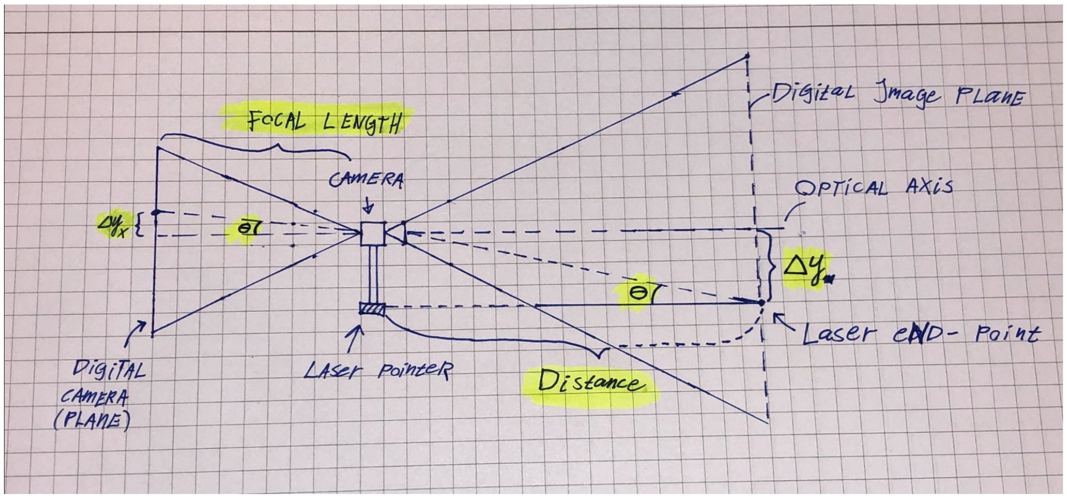


Figure 4.28: Designed plot of the working distance solution

From the camera calibration in 4.24, we will have obtained the focal length in the y- direction (in pixels), from the captured image we will have the principal point and the distance $\triangle y_x$, we can derive the angle Θ highlighted in the left side of 4.28.

$$\Theta = \arctan(\frac{\triangle y_x}{\text{Focal Length}}) \tag{4.5}$$

This angle is mirrored as illustrated in 4.26 when an image passes through a pinhole. δy on the right hand side of 4.28 is evidently the distance between the digital camera and the laser pointer. Comparing the two right triangles in 4.28 yields

$$tan(\Theta) = \frac{\triangle y_x}{\text{Focal Length}} = \frac{\triangle y}{\text{Working Distance}} \tag{4.6}$$

Comparing (4.5) and (4.6), we are left with a single free variable which is the height of the pixel in the centre of our view, (which evidently is the δy_x , and our output will be the horizontal distance from the camera and/or laser pointer, to the end point of the laser.

$$\text{Working distance} = \frac{\triangle y \cdot \text{Focal Length}}{\triangle y_x} \tag{4.7}$$

given the chosen point is in the centre column of the image.

Chapter 5

Results

5.1 Triangulation

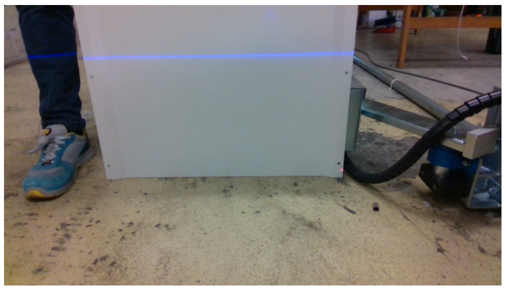


Figure 5.1: Test image, 82cm

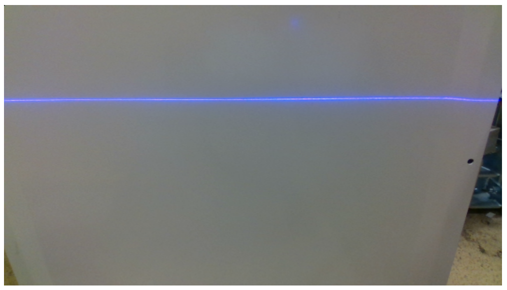


Figure 5.2: Test image, 36cm

To determine the horizontal distance of a point in a pixel plane from a digital camera, some information needs to be known to us. The distance between the digital camera and the laser beam projector, and the angle between the cameras optical axis and the two dimensional line developed by the laser.

With these constrains, any point in the pixel plane can be driven down to a single definite distance from the camera to an end- point by the laser pointer, and we managed to do so with a precision of +- 4cm. This fault increases as distance is increased, and would need optimization.

5.2 Gap detection

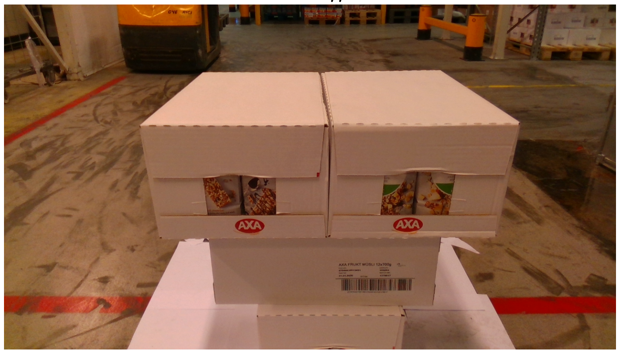
Due to some mechanical problems and time constraints, only a few tests were done with the system running on the actual robot. The system was merely added onto the side, with the gap detection being run and the result logged, but the output was not actually used for anything. There were therefore conducted a series of tests afterwards, with the system alone in a controlled environment.

5.2.1 Robot tests

The following entries are the test run done the 16th of May 2019, where the robot was directed to a shelf where it would locate and pick packages. Due to a power failure some test data was lost, and of the 4-5 tests run, only two were usable. There was also an error in the logging of some of the data, so only rudimentary data was logged.

These tests serve more as a proof-of-concept, that the system can function in a practical setting.

Test #1


Normal photo of boxes from robot perspective

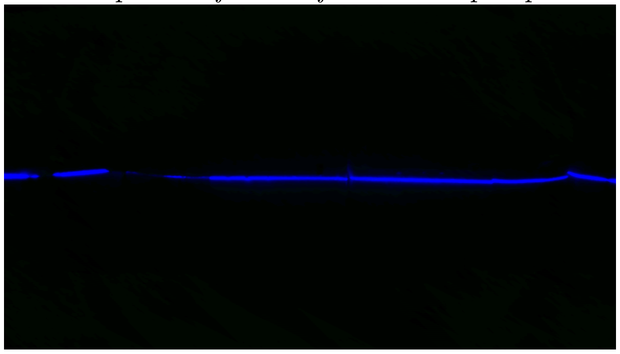
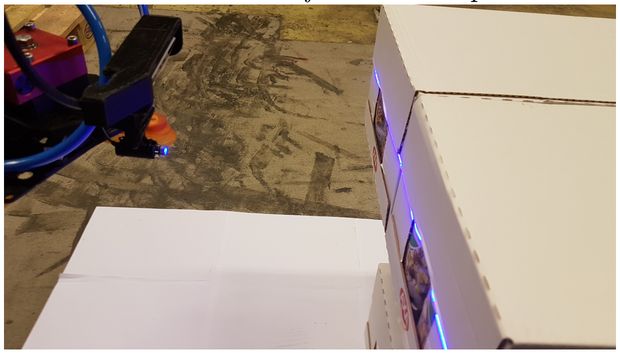


Photo taken with modified camera parameters


Profile of system in action

For this case, the system reported detecting a gap with a certainty of 50%

Test #2

Normal photo of boxes from robot perspective

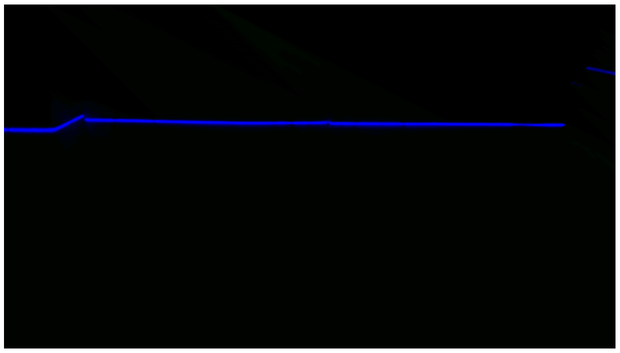
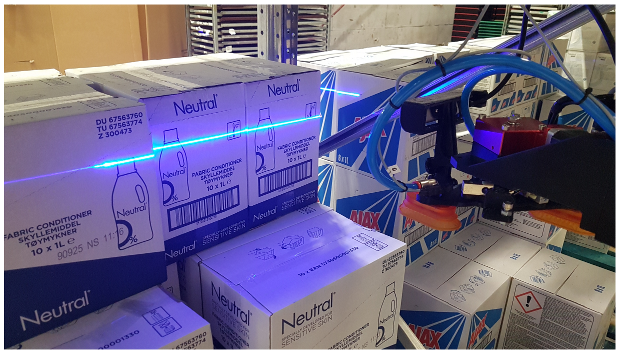


Photo taken with modified camera parameters


Profile of system in action

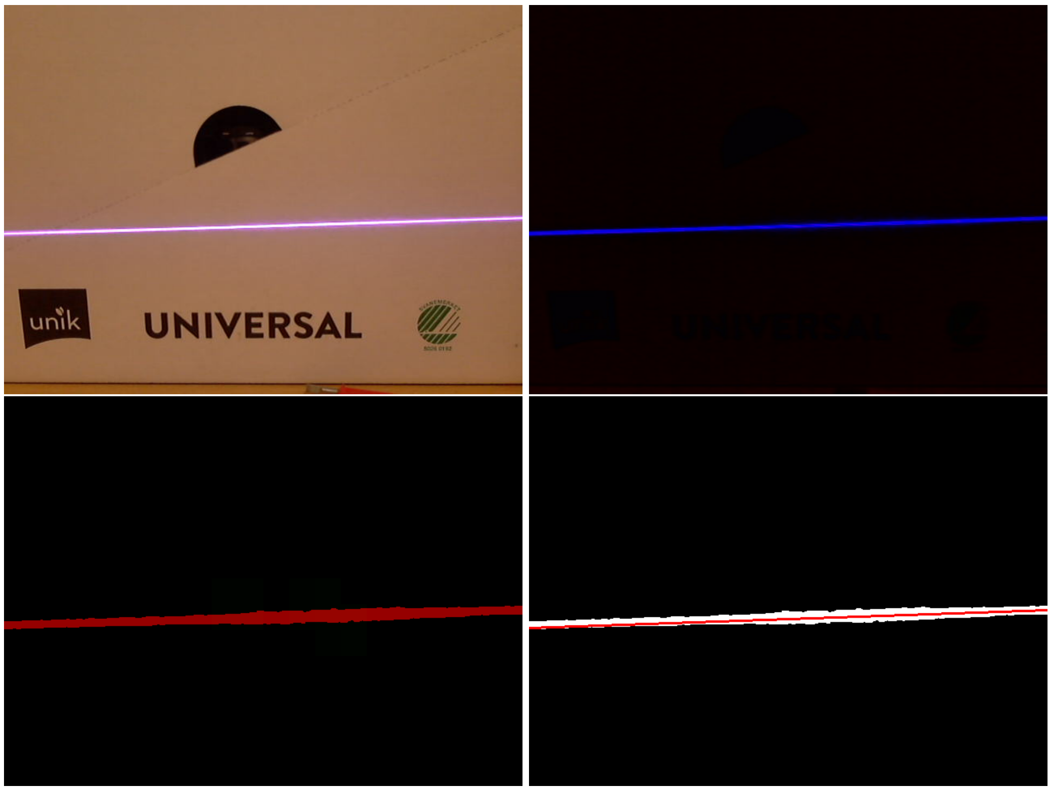
This test also resulted in a gap detected with a certainty of 50%.

5.2.2 Lab tests

Due to the time and technical problems, the systems performance wasn't possible to ascertain from practical tests. A series of tests in a controlled environment with the testing rig used development was performed. The purpose of these tests are to show the strengths and weaknesses of the system by demonstrating it's performance in edge cases.

The results are displayed as follows, a normal colored image, image taken with tweaked camera parameters, the mask and convolution method response, mask overlaid with lines found by the line method.

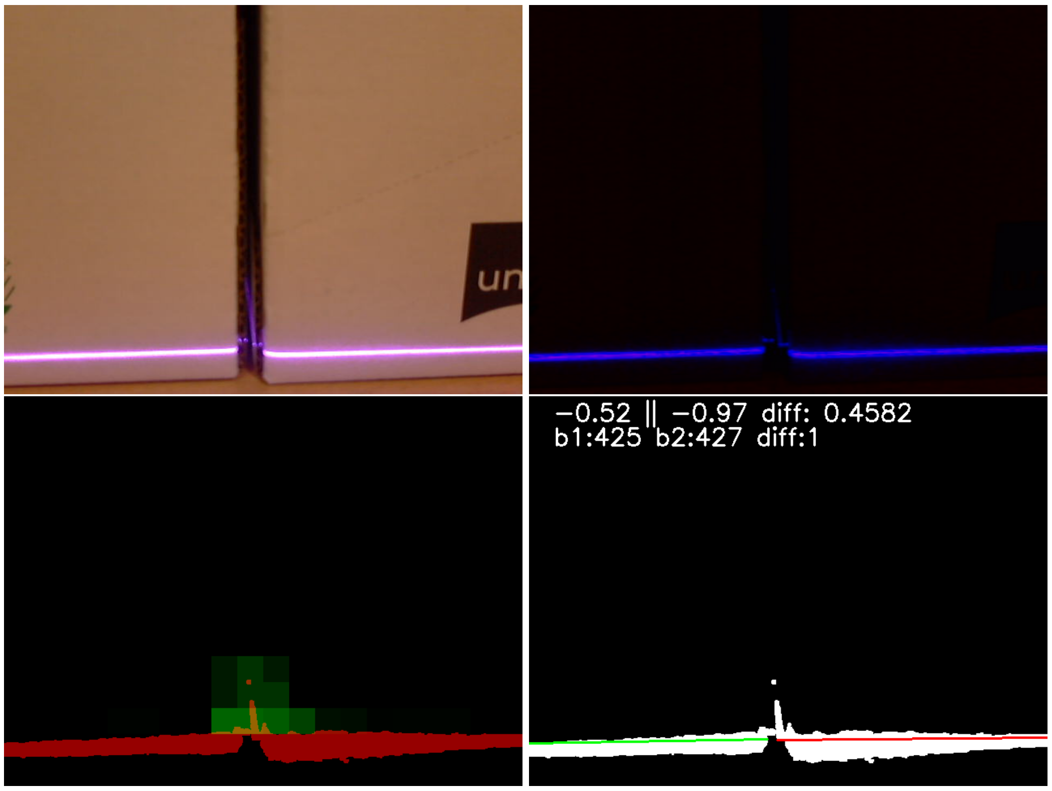
Test #1 - No gap, white box



We see the mask is nice and clean. The convolution method doesn't trigger anywhere, and the line method finds only a single line spanning the entire image.

Expected answer	No gap
Line tags	Full line
Prediction	No gap
Confidence	100%
Prediction correct	Yes

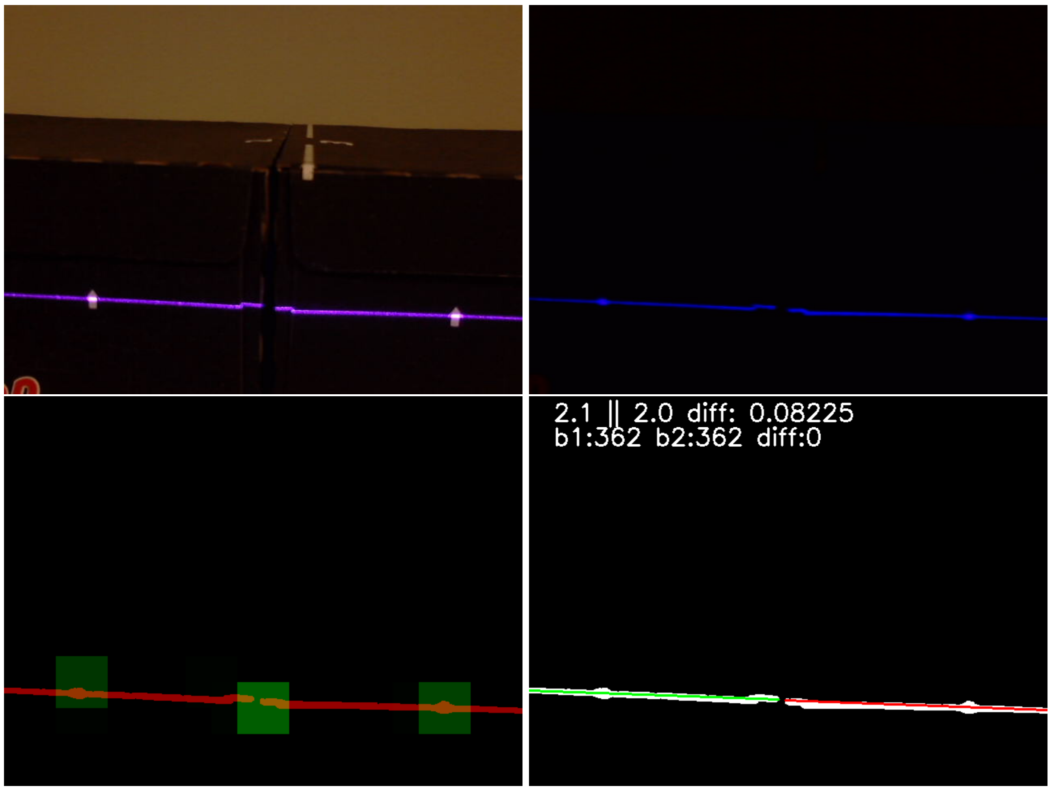
Test #2 - Gap, white boxes



We see the mask is nice and clean. The convolution method only triggers around the gap, and the line method finds the two lines, split by the gap. The line method finds that the angular and height difference is 0.46°and 1 pixel respectively.

Expected answer	Gap
Line tags	Broken line
Prediction	Gap
Confidence	100%
Prediction correct	Yes

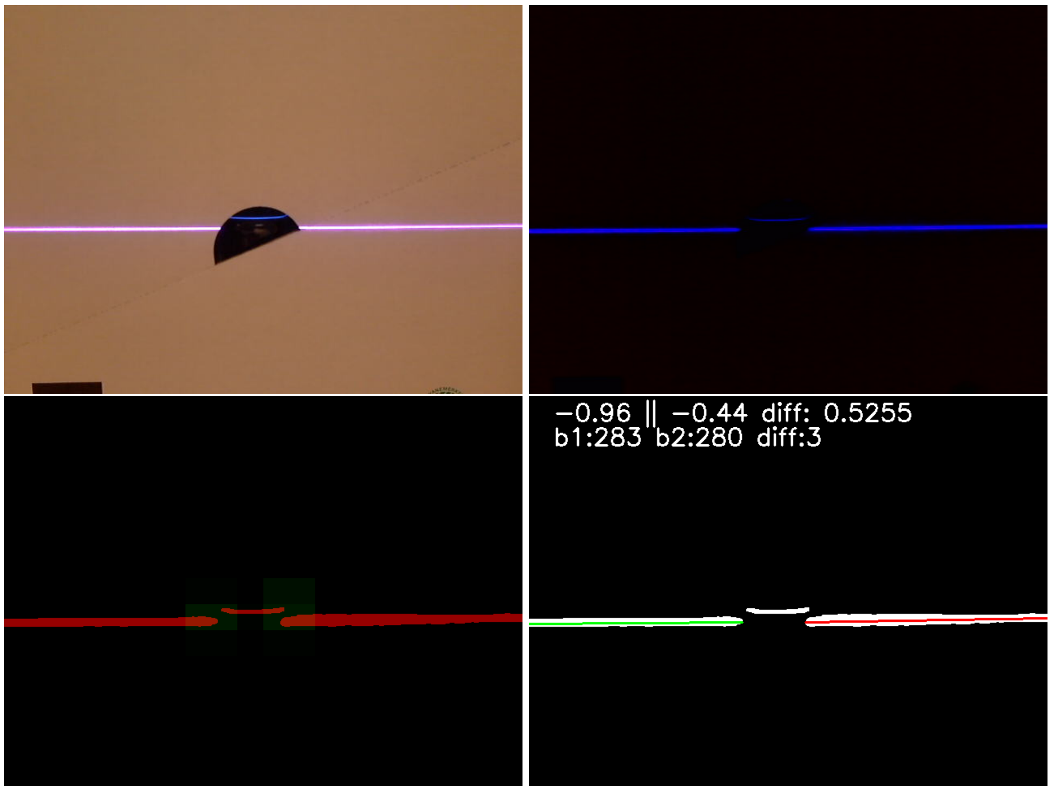
Test #3 - Gap, black boxes



We see the mask has some bulges where the box had white decorations, and a little step before the gap due to the construction of the boxes. The convolution method triggers the most around the gap, but also at the bulges. The line method correctly finds the two lines, split by the gap. The line method finds that the angular and height difference are 0.08°and 0 pixels respectively.

Expected answer	Gap
Line tags	Broken line
Prediction	Gap
Confidence	100%
Prediction correct	Yes

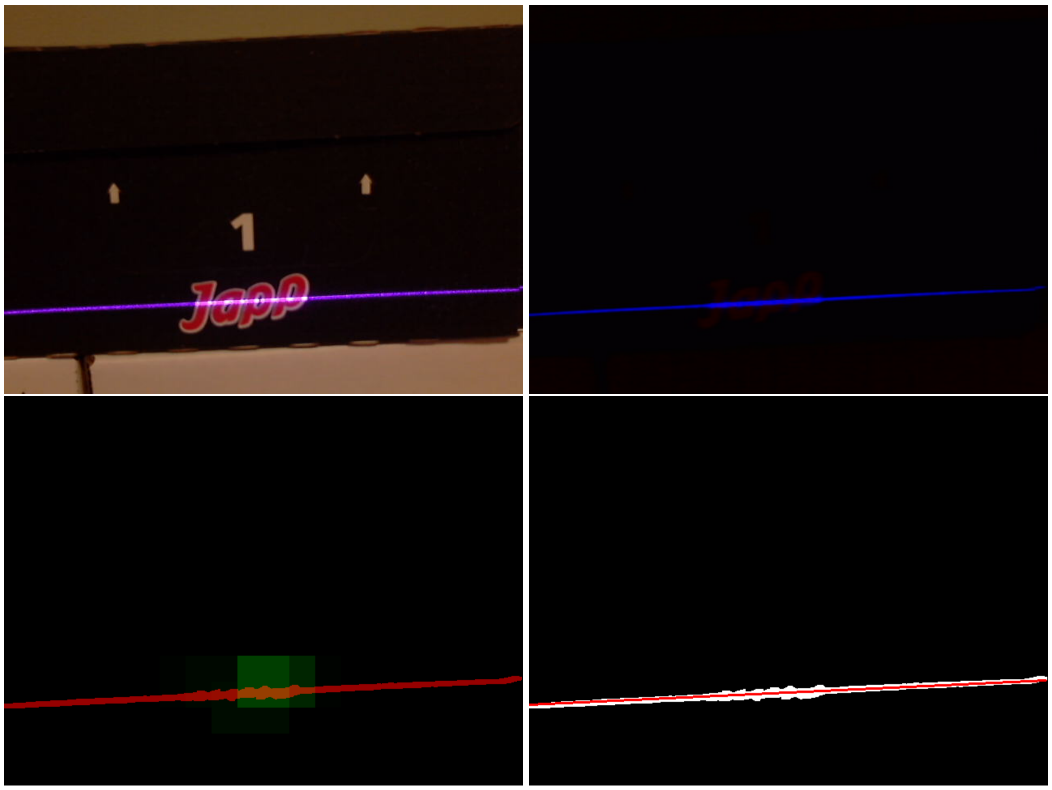
Test #4 - Hole, white box



We see the mask is clean, but shifted up where the laser goes further in due to the hole. The convolution method triggers a little, but not strongly enough to indicate a gap. The line method finds the two lines, split by the hole in the box.

Expected answer	No gap
Line tags	Broken line
Prediction	Gap
Confidence	50%
Prediction correct	No

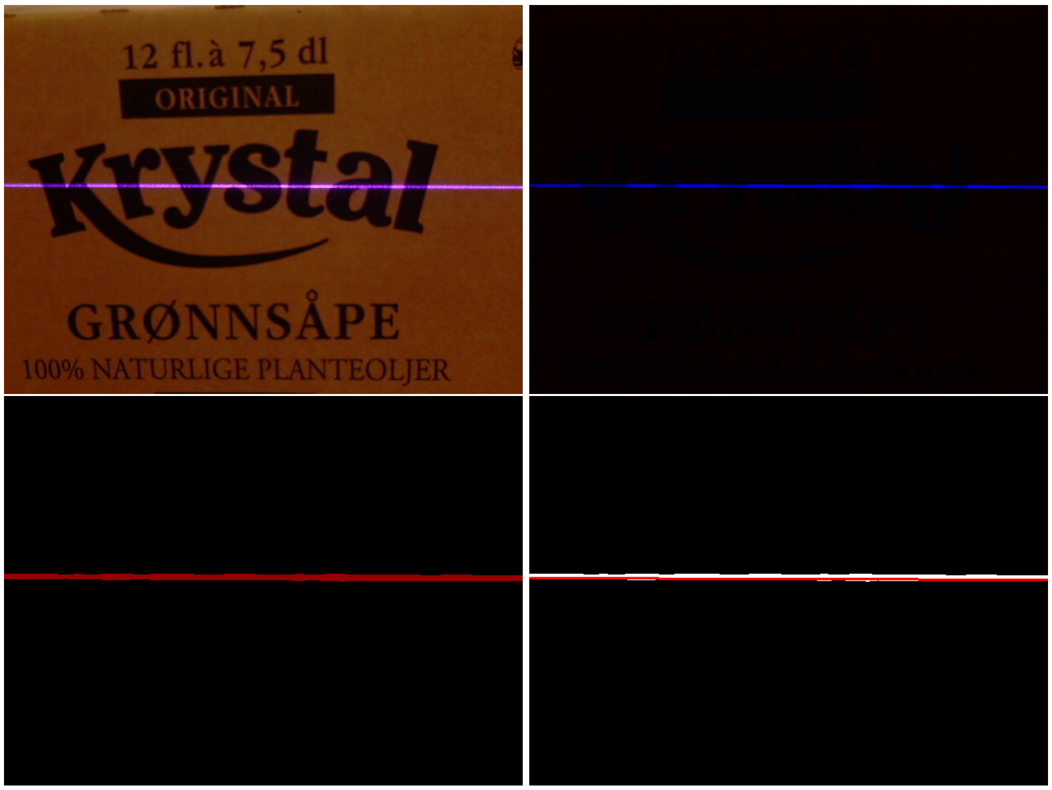
Test #5 - Pattern, black box



We see the mask is clean except where the laser crosses the pattern, where the mask bulges in tune with the pattern. The convolution method triggers significantly, likely enough to indicate the presence of a gap. The line method however, finds a single line spanning the entire image.

Expected answer	No gap
Line tags	Full line
Prediction	No gap
Confidence	100%
Prediction correct	Yes

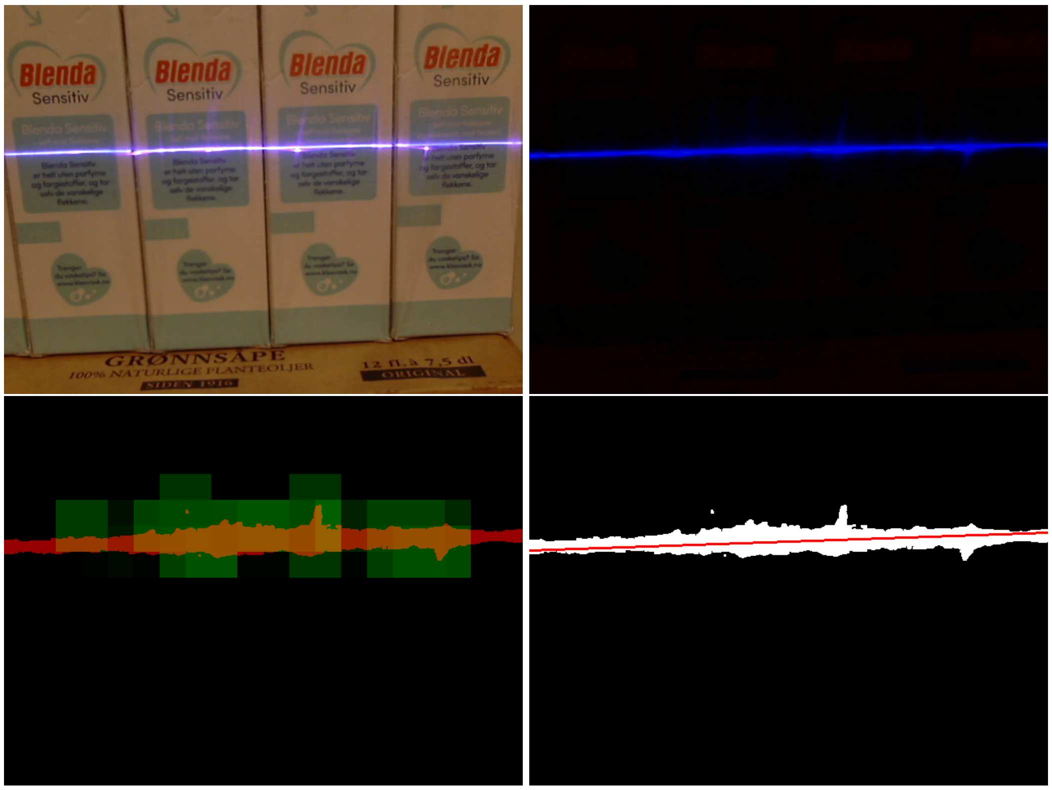
Test #6 - Pattern, white box



In the parameter tweaked image, we can see the strength of the reflection varying as the laser moves across the darker pattern, but the mask ends up mostly unchanged by this. The convolution method does not appear to trigger at all, and the line method finds only a single line spanning the entire image.

Expected answer	No gap
Line tags	Full line
Prediction	No gap
Confidence	100%
Prediction correct	Yes

Test #7 - Multiple boxes, plastic wrapped



Due to the high reflectance and the uneven surface of the plastic, we get many "false" reflections, in this case leading to the mask having many parts sticking out. This noisy mask leads to the convolution method triggering strongly across almost the entire mask. The line method still does find a single line spanning the image despite the noisy input.

Expected answer	No gap
Line tags	Full line
Prediction	Gap
Confidence	50%
Prediction correct	No

Chapter 6

Discussion

6.1 Laser Safety

A transient increase of only 10 °C can destroy retinal photo-receptor cells. If the laser is sufficiently powerful, permanent damage can occur within a fraction of a second, meaning the eye won't have time to blink to protect itself. Sufficiently powerful lasers in the visible to near infrared range (400-1400 nm) will penetrate the eyeball and may cause heating of the retina, whereas exposure to laser radiation with wavelengths less than 400 nm and greater than 1400 nm are largely absorbed by the cornea and lens, leading to the development of cataracts or burn injuries. [4]

There are four main classifications of a laser based on its laser radiation hazard level; class 1 through 4

- Class 1 lasers are safe under all conditions. This includes direct exposure to the retina of the eye. Class 1 splits into a subclass: Class 1M. This class means as long as there are no magnifying optics between the source of the laser, no harm will affect the human eye, where as Class 1 can be amplified with optics and still pose no hazardous threat.
- Class 2 lasers are considered safe because of our blink reflex. Our blink reflex is fast enough close and therefore prevent any permanent damage. Class 2 lasers also splits into two subclasses in the same way, class 2 and class 2M. Where class 2M are marked safe as long as no optics are

focusing the laser beam.

- Class 3 lasers should be handled with care. They can damage the eyes with direct exposure (although the less hazardous subclass proposes a low risk of eye damage). It is perfectly safe to view the reflections of the laser if it bounces off a non-reflective material for both subclasses. It is also safe to view it indirectly with smoke to illuminate the beam. Protective goggles are advised if there is a chance of direct exposure to the beam. Class 3 is divided into two subclasses: Class 3R and 3B. 3R is the forgiving of the latter, who's restricted to 5 mW. 3B is restricted to 500 mW (continuous beam), and protective goggles are advised if there is a chance of looking directly at the beam. Diffuse reflections from rugged material such as paper is fine however.
  The lasers used in this project falls into this category (3B), as they were 25mW and 80mW.

- Class 4 lasers are the most dangerous/hazardous. A class 4 laser can burn the skin, or cause devastating and permanent eye damage as a result of direct, diffuse or indirect beam viewing, and ignite flammable material such as clothing.

6.2 Polarization

Light is an electromagnetic wave, and the electric field of this wave oscillates perpendicularly to the direction of propagation. Light is called unpolarized if the direction of this electric field fluctuates randomly in time. [1] A polarizing filter will absorb electromagnetic waves oscillating nonparallel to its texture.

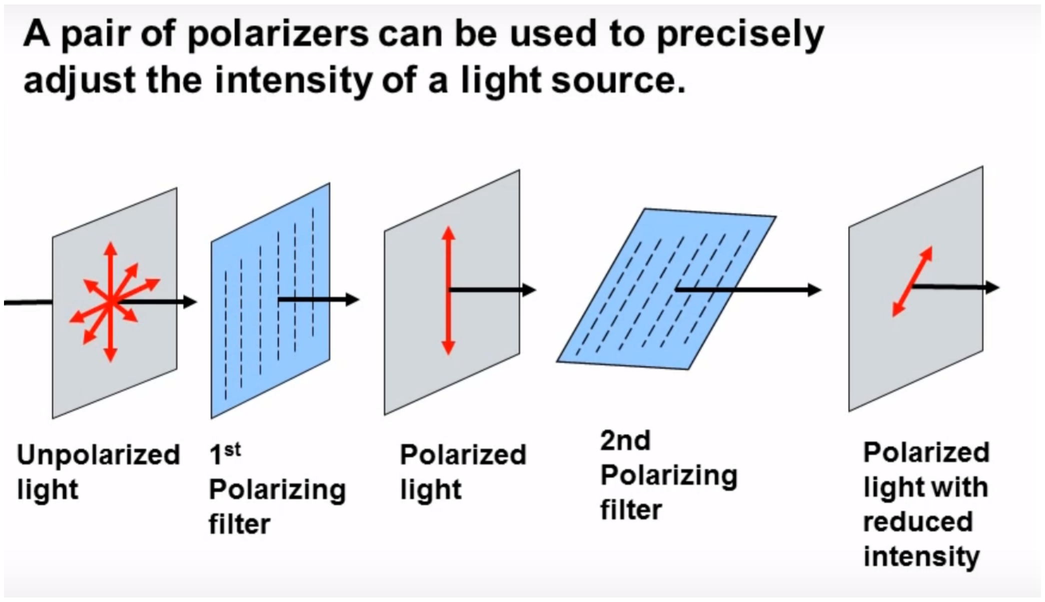

A pair of polarizers can be used to precisely adjust the intensity of a light source.
Unpolarized light
1st Polarizing filter
Polarized light
2nd Polarizing filter
Polarized light with reduced intensity

Figure 6.1: combination of polarizing filters

Filtering a source of unpolarized light leaves a polarized beam of light. Laser beams are polarized light, and can be filtered yet again to reduce its strength. The idea of bringing polarizing filters to the project is to minimize the light pollution caused by the laser, which illuminates the surrounding area of the laser line. The camera perceives this as noise, which in turn makes a less accurate model for the skeletonization 2.3.4 or convolution gap detection algorithm 4.1.3 method, regardless there will be more raw data and less noise.

However, as the polarized laser light hits the rugged material of cardboard, the light is split into several polarized and unpolarized photons. To maintain its current polarization upon reflection, a reflective material such as metal or a flat surface with silver coating (such as a cinema screen) has to be used.

We had hoped that the light would keep its polarization when reflected off of the cardboard. If so, then two cameras could be used, one with a filter and one without. The filtered camera would see the scene with the laser in it, the unfiltered camera would see the scene without the laser, as the laser would be completely filtered out. This would have meant that we would be able to compare the images to find the laser.

6.3 Project Planning

The course of the thesis spanned across four months and was carefully planned with risk factors and a self designed gant diagram to follow and stretch towards our intended goals for the thesis. However, as time progressed, it became apparent that the gap detection task became was far more demanding and time consuming that we expected. The gap detection algorithm for cardboard boxes was initially meant to be a stepping stone for the bottle gap detection, but ended up becoming the main purpose of the project.

6.4 Calibration

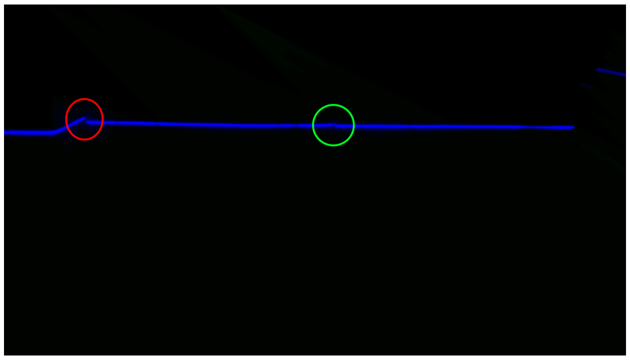
Recent years this has become the most common way to reconstruct and estimate the extrinsic parameters of any digital camera, and is easy to do yourself with minimal equipment, which is why we decided to use the Zhang's method. It is easy to use and requires minimal tools to utilize.

6.5 Triangulation

At first we experimented with methods of capturing test images of known distances to compare right triangles and depict a rate of change in pixel height and distance. However, as we realized the mounted camera angle is known and static, this method became more obvious, and never would the camera need to re-calibrate, and the distance to an object would be known solely based upon the pixel height of the point of interest.

6.6 Tests on robot

While the results from the testing on the robot were correct in both cases, they were both quite ideal situations. It is also important to note that the system was designed to receive a region of an image to test the presence of a gap in that region, while in this test the entire image was passed to the function. This means that for a case like test #2, the program wanted to known about the gap marked with green, but the function likely detected the gap marked with red instead:



This combined with the fact that so few tests were conducted, no data on how the system perceived the cases was collected, and the system was at no point tested against a case where there was no gap, means that these tests cannot be used to show the systems performance, nor reliability.

6.7 Tests in controlled environment

The controlled tests demonstrated the systems performance under ideal conditions with ideal packages, and also cases where the system struggles. These tests were done with a constant source of light, and with the system being tuned to work under these conditions, and might not accurately reflect how it would perform under the varying conditions it would experience in the field.

It is important to note that these tests were performed using a different camera than what's used on the robot, one with lower resolution and quality, which can explain eventual differences between the tests.

6.8 Issues

6.8.1 Reflectance of differently colored materials

On flat surfaces with little to no variation in color, the system performs well, detecting both the presence and absence of gaps. Once color patterns are introduced, the system starts struggling; the greater the difference between the color, the more difficult it becomes.

This is a result of the reflectance of the color white; as the white cardboard is reflecting the laser line in all directions more vigorously, the cardboard around the laser line is illuminated to an extent where the camera perceives it as part of the laser.

This problem has persisted throughout the development, as the only method developed to work around it, is to increase the HSV threshold for a pixel to be considered a part of the laser. However, when we increase the constrain in the Value (V-) component of the HSV to filter our the surrounding illumination, the constrain quickly becomes too strict, and filters out the entire laser line on a black cardboard surface, as its intensity is absorbed into the black pigment. 3.3

It has been discussed that this could in theory be alleviated by first taking a normal color image, and creating a local threshold value for determining the laser mask based on the color in the color image. This way, an area that is brightly colored can have a higher threshold, while a dark area can have a lower threshold, and in this way, in theory, the reflectance problem can be negated.

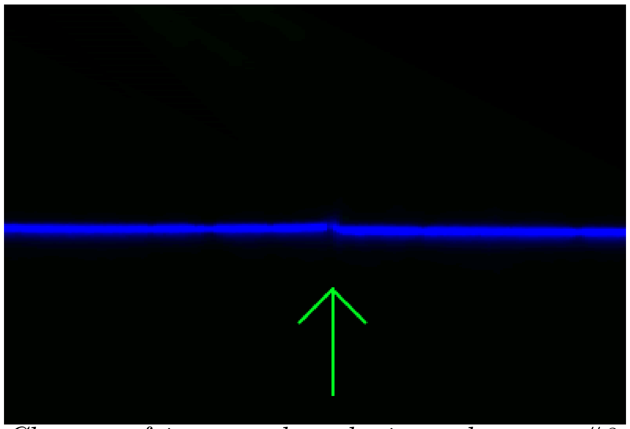
6.8.2 Variations in package designs

Packages that have structural differences from others, such as holes or protrusions, cause the line to be "broken", either shifted up or down in relation to the rest of the laser line. The line based method will therefore assume it is a gap. The convolution based method however, likely won't see this as a gap.

This illustrates one of the most damning problems with the line based methods, it's inflexibility. It has the advantage of being robust to noise in the laser mask, as the laser mask having an uneven surface is irrelevant to it, as all it cares about is whether it is able to draw a line through the core of the mask.

6.8.3 Tightly packed packages

When the packages are tightly packed, or when the camera is further away from them, the impact of the gap on the laser line is greatly reduced. This can be seen on robot test #2, where the package corners are very sharp, not rounded by design or by deformation from being handled. The gap is barely noticeable, and cannot easily be distinguished from the line deformations caused by the patterns on the packages.



Closeup of image taken during robot test #2, gap marked with green arrow

In cases like this, the laser mask is likely to jump this insignificant gap due to there being some blue color, albeit weaker, in the gap, leading to the laser mask showing a solid line. There is also not much of a bump on the top or bottom, which means that the convolution method might also fail to trigger on this gap, meaning that in all, the result would be that there is no gap in the image, and with great certainty.

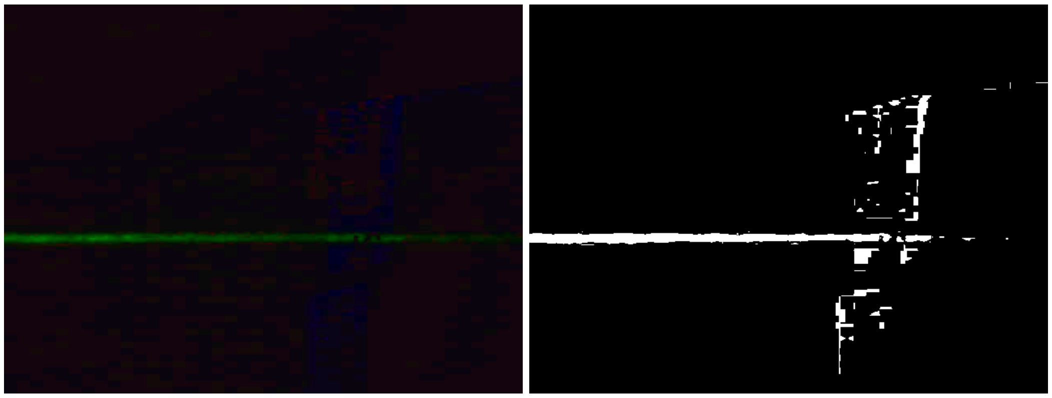
This is another problem that will likely require a better way of isolating the laser in the image to solve. A method that can detect this small deformation is likely to also trigger at other small deformations caused by variations in the package surface, color and so on.

6.9 Alternate approaches

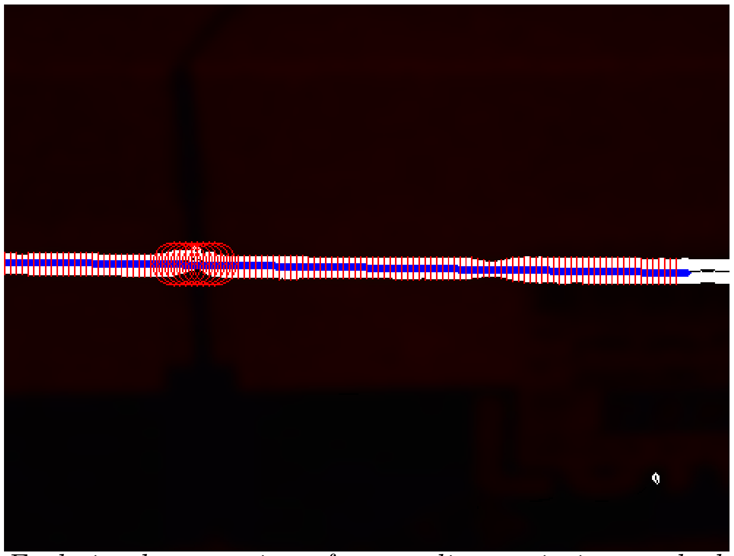
There are multiple approaches that either was put aside due to the complexity of implementing, limitations imposed by the hardware used at the time, or due to them not being thought of early enough to allow for them to be implemented.

6.9.1 Centerline variations

This method was attempted early on in the project, see 4.1.2. At that time the laser in use was the green 25mW laser, a laser that suffered from high and noisy spread when reflecting off of white surfaces. And weak and often broken up lines when reflecting off of dark surfaces, such as this;



This was also because our image filtering and thresholding was still in its early stages, leading to noisy and overall bad laser masks. So while the results were promising, and a simple implementation was made which gave good results:


Early implementation of center-line variation method. Red lines show distance from center-line to mask edge, circles mark areas of high variation

The biggest strength of this method can also be seen here, to the right the laser line is thinner due to it crossing a darker colored part of the box, but due to the distance from the center-line to the top and bottom being similar, this is ignored, only where the relationship between the distances to the top and bottom shift is marked as a possible gap.

With the improved camera, laser and thresholding tools, this method is one that we now wish had been explored more in depth, and is one that shows great promise.

6.9.2 Pattern Recognition

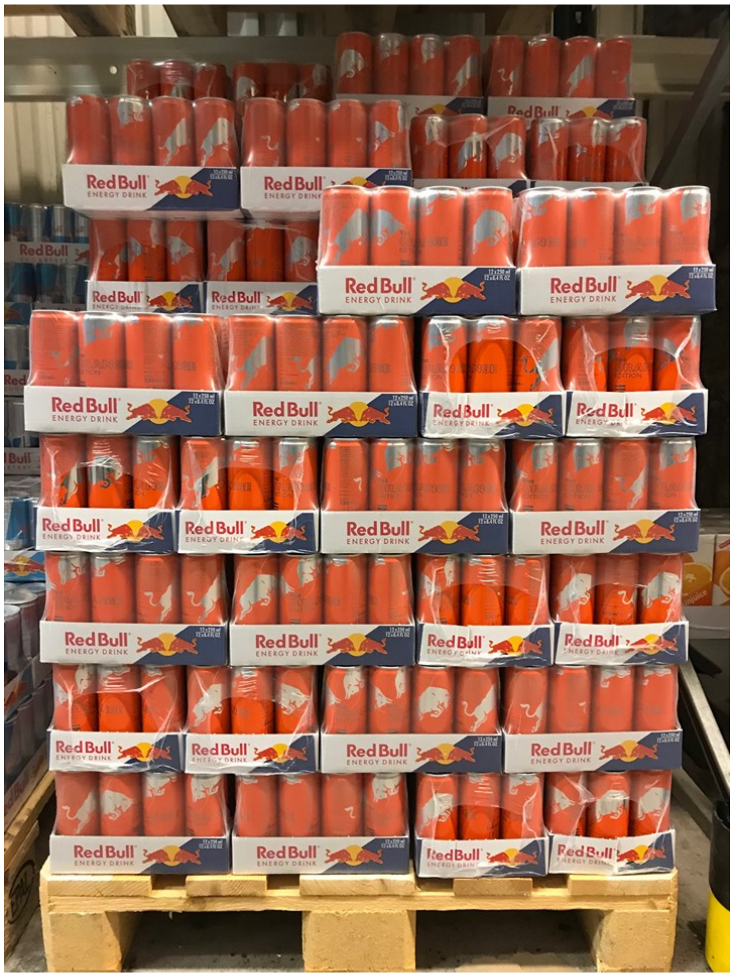


Figure 6.2: Cardboard pattern with redbul logo

Pattern recognition is a wide concept, and it could surely prove its benefits in this project. With the upcoming input from the teams database, they can withdraw the package dimensions which would yield the width, length and height of the packages.

For this specific example of red bull packaging 6.2, a single image of any row of red bulls will be hard to distinguish which curving of bottles determines the "edge" of the package.

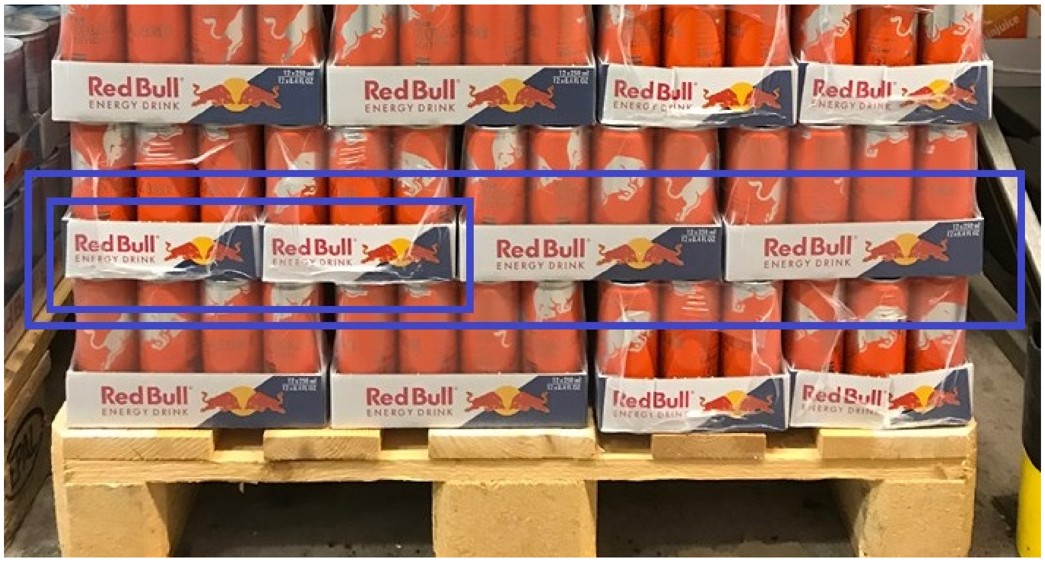


Figure 6.3: recognizable pattern

Provided the length and with of the packages highlighted in 6.3, we should be able to emphasize and distinctly decide which columns in our 2D image matrix who's bound to have a gap. This is done with counting the frequency at which the cardboard logo repeats itself. We would then know where we would expect the pattern to repeat itself with an algorithm, and detect any fractures in the repeating pattern.

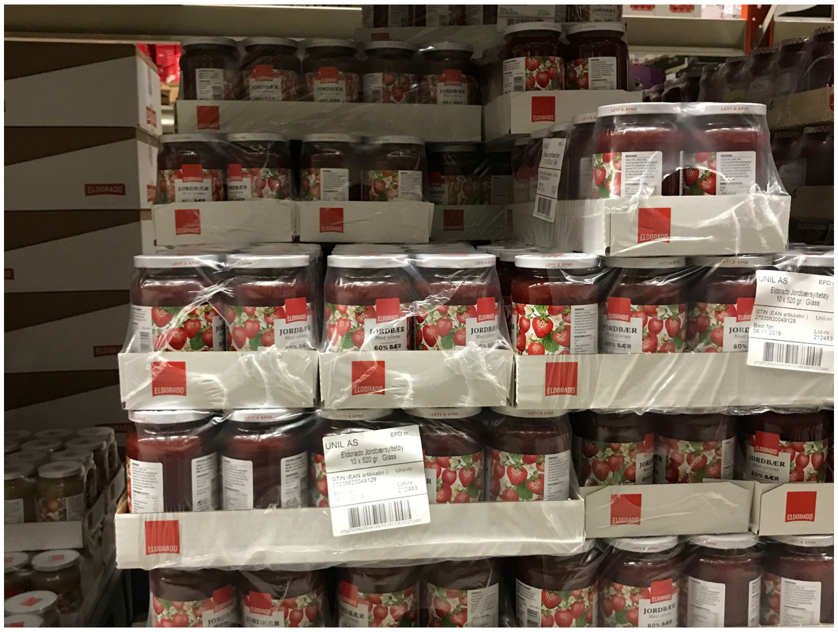


Figure 6.4: Though pattern

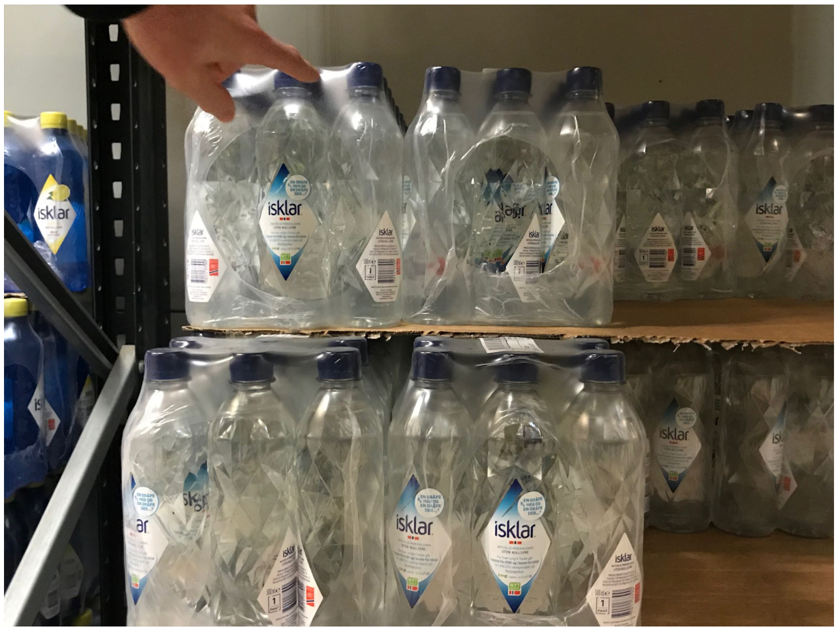


Figure 6.5: No pattern

This solution quickly grows in complexity as the variety in items are vast and seemingly random, as there can be single standing items, bulging shrink plastic covers and confusing pattern on the cardboard 6.4. For 6.5, the only apparent pattern is the true gap itself, which would in turn only be distinguished by any other method. Even if the pattern recognition would separate the water bottle bundles with the wide gap between the caps, the angle at which the camera will see the other gaps will differ greatly as the camera gets closer to the packages.

Pattern recognition can be achieved with the help of analyzing the fourier transform of the image, or the use of artificial intelligence.

Chapter 7

Conclusion

The objective of this project was detecting gaps in stacks of plastic bottles in plastic wrap, which was a major problem the Currence team had been facing. However, it became apparent that this task became far too ambitious once we started delving into the topic at hand.

The project has shown that the key to a reliable gap detection system is proper hardware. As any system attempting to detect such small features in such a noisy environment is bound to be very sensitive to noise. Most of the work done was in attempting to find methods that would work despite this noise, rather than pursuing better hardware that could reduce it.

Bibliography

[1] Edmund Optics Inc (2015). *Introduction to Polarization.* URL: `https://www.edmundoptics.com/resources/application-notes/optics/introduction-to-polarization/`.

[2] Edmund Optics Inc (2015). *Understanding Focal Length and Field of View.* URL: `https://www.edmundoptics.com/resources/application-notes/imaging/understanding-focal-length-and-field-of-view/`.

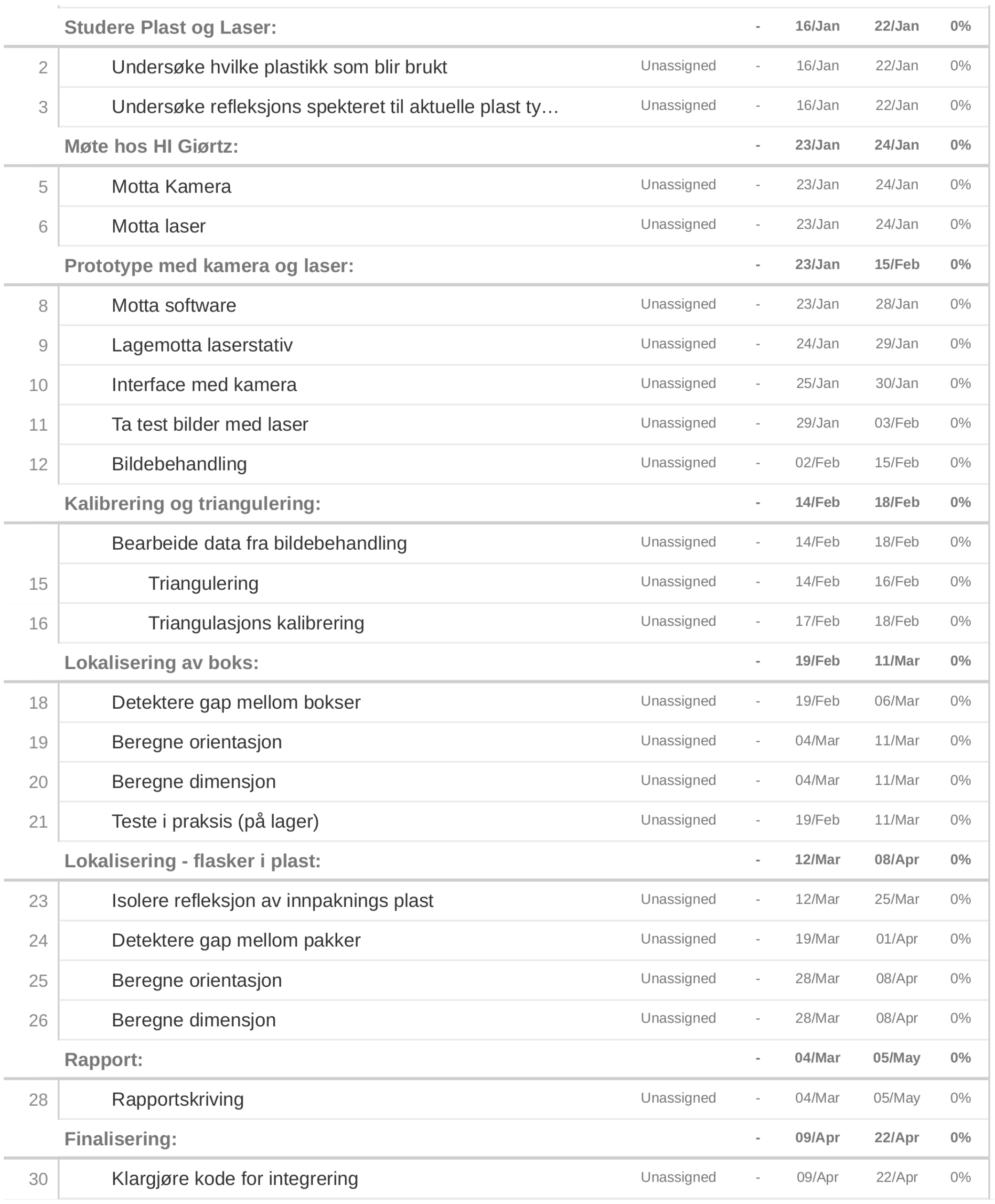
[3] Jean-Yves Bouguet. *CalTech's Matlab Calibration Toolbox.* URL: `http://www.vision.caltech.edu/bouguetj/calib_doc/`.

[4] Osama Bader; Harvey Lui. "Laser Safety and the Eye: Hidden Hazards and Practical Pearls". In: *Vancouver hospital & Health Sciences Centre* (1996).

[5] Zhengyou Zhang. "A Flexible New Technique for Camera Calibration". In: *Microsoft Research* (1998).

Attachments

Attachment 1	Gant List 7.1
Attachment 2	Gant Diagram 7.2



	Studere Plast og Laser:		-	**16/Jan**	**22/Jan**	**0%**
2	Undersøke hvilke plastikk som blir brukt	Unassigned	-	16/Jan	22/Jan	0%
3	Undersøke refleksjons spekteret til aktuelle plast ty…	Unassigned	-	16/Jan	22/Jan	0%
	Møte hos HI Giørtz:		-	**23/Jan**	**24/Jan**	**0%**
5	Motta Kamera	Unassigned	-	23/Jan	24/Jan	0%
6	Motta laser	Unassigned	-	23/Jan	24/Jan	0%
	Prototype med kamera og laser:		-	**23/Jan**	**15/Feb**	**0%**
8	Motta software	Unassigned	-	23/Jan	28/Jan	0%
9	Lagemotta laserstativ	Unassigned	-	24/Jan	29/Jan	0%
10	Interface med kamera	Unassigned	-	25/Jan	30/Jan	0%
11	Ta test bilder med laser	Unassigned	-	29/Jan	03/Feb	0%
12	Bildebehandling	Unassigned	-	02/Feb	15/Feb	0%
	Kalibrering og triangulering:		-	**14/Feb**	**18/Feb**	**0%**
	Bearbeide data fra bildebehandling	Unassigned	-	14/Feb	18/Feb	0%
15	Triangulering	Unassigned	-	14/Feb	16/Feb	0%
16	Triangulasjons kalibrering	Unassigned	-	17/Feb	18/Feb	0%
	Lokalisering av boks:		-	**19/Feb**	**11/Mar**	**0%**
18	Detektere gap mellom bokser	Unassigned	-	19/Feb	06/Mar	0%
19	Beregne orientasjon	Unassigned	-	04/Mar	11/Mar	0%
20	Beregne dimensjon	Unassigned	-	04/Mar	11/Mar	0%
21	Teste i praksis (på lager)	Unassigned	-	19/Feb	11/Mar	0%
	Lokalisering - flasker i plast:		-	**12/Mar**	**08/Apr**	**0%**
23	Isolere refleksjon av innpaknings plast	Unassigned	-	12/Mar	25/Mar	0%
24	Detektere gap mellom pakker	Unassigned	-	19/Mar	01/Apr	0%
25	Beregne orientasjon	Unassigned	-	28/Mar	08/Apr	0%
26	Beregne dimensjon	Unassigned	-	28/Mar	08/Apr	0%
	Rapport:		-	**04/Mar**	**05/May**	**0%**
28	Rapportskriving	Unassigned	-	04/Mar	05/May	0%
	Finalisering:		-	**09/Apr**	**22/Apr**	**0%**
30	Klargjøre kode for integrering	Unassigned	-	09/Apr	22/Apr	0%

Figure 7.1: Sub-goal with associated tasks

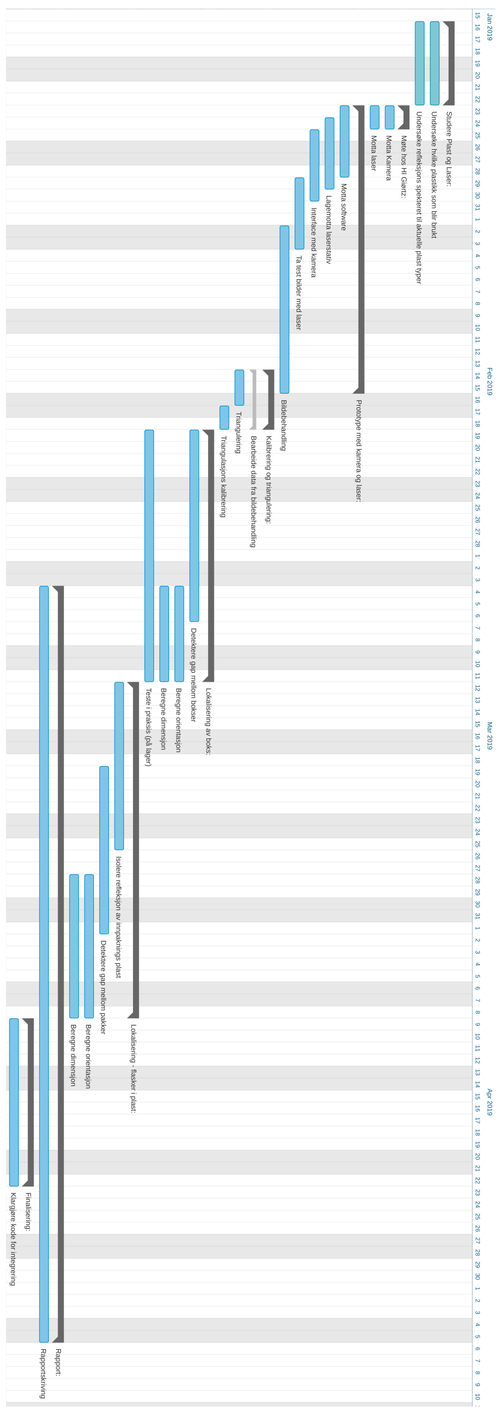


Figure 7.2: Time allocation for sub-goals and tasks