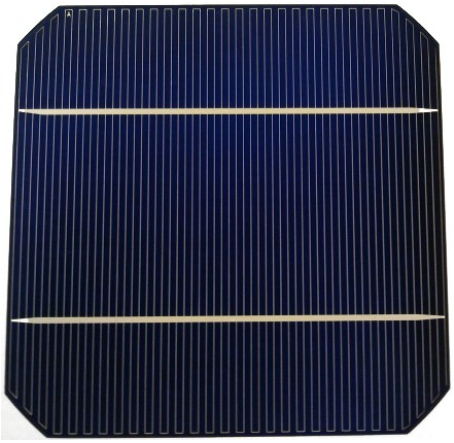
Vision-based edge tracking for area optimization

van den Dungen, M.B.R.

Award date:
2013

VISION-BASED EDGE TRACKING FOR AREA OPTIMIZATION

MARTIJN VAN DEN DUNGEN



Thesis

July 2013 – version 1.0

Confidential

"The most important thing is to keep the most important thing the most important thing."

— Donald P. Coduto

ABSTRACT

Advances in solar cell technology have dropped solar energy costs over the last 30 years. To keep this rate going solar cell manufacturers are required to keep reducing fabrication costs and improve efficiency.

One of the fabrication processes is edge isolation, which separates the active front from the rear of the cell. Small variations and faults in the outer contour of a wafer disturb this process, leading to defective cells. To overcome this, edge isolation takes place a fixed distance from the outer edge, wasting effective area.

Vision-based edge tracking allows the contours of wafers to be detected precisely and followed on the fly. Doing so, edge isolation can be performed closer to the wafer's edge and detected defaults can be corrected for, maximizing surface area and improving cell efficiency.

In this work we provide a method for vision-based edge tracking which reduces the distance at which isolation takes place from $1mm$ to $0.1mm$ on a 156x156mm cell, thus increasing efficiency by 0.47%.

ACKNOWLEDGEMENTS

I would like to express my gratitude to all those involved with this thesis over the past year.

Foremost, I want to thank Peter du Pau for all his help and encouragement. Your experience and patience have been a great help. You also inspired me to put the quote about importance at the beginning of this thesis, a valuable lesson.

Many thanks go out to Henk Corporaal, Zhenyu Ye, Mark Wijtvliet and Jan van Dalfsen. Our regular meetings always provided me with new ideas and feedback. Thank you very much for the opportunity.

Much appreciation goes out to my colleagues here at Roth & Rau. I've enjoyed spending time with you. I wish you all the best of luck and I hope to see you again in the future.

Most of all I thank my parents for all their support. I wouldn't be here without you.

CONTENTS

LIST OF FIGURES

Part I

INTRODUCTION

This work has been carried out at Roth & Rau, Eindhoven (formerly OTB-Solar) as part of the Embedded Vision Architecture (EVA) project. Roth & Rau provides solutions for solar cell technology.

In this first part of the thesis the problem at hand is stated and contributions are listed. Work relating to the various parts of this thesis have been summarized as well.

1 INTRODUCTION

Solar energy currently provides only a quarter percent of the world's electricity supply. This is changing. Over the last 30 years the price of capturing solar energy has dropped exponentially. This phenomenon has been dubbed Swanson's law (Figure 1), in imitation of Moore's law which suggests that transistor size halves approximately every 18 months [1, 2].

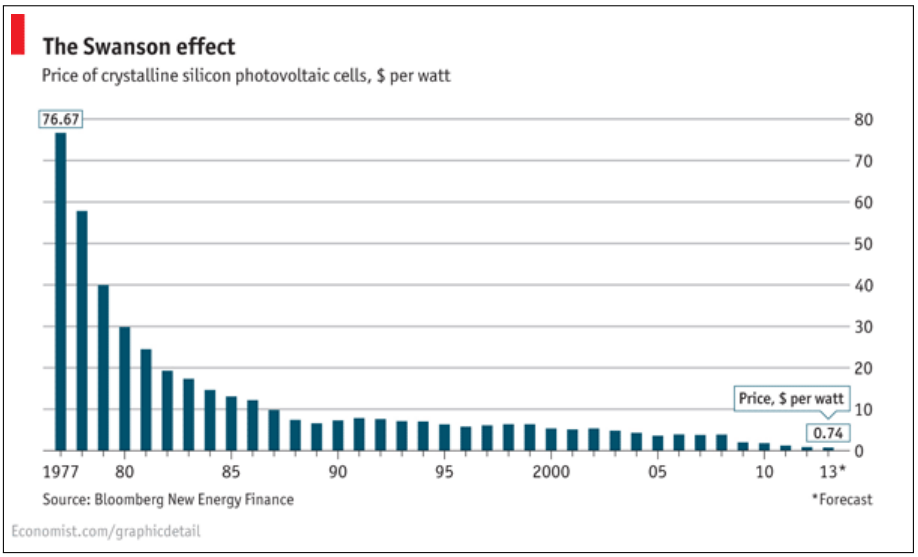


Figure 1: The Swanson effect.

There are two factors that drive this change. First, solar cell manufacturers are learning how to reduce the cost to fabricate cells. Second, the efficiency of solar cells - the ratio of electrical output of a solar cell to the incident energy in the form of sunlight - is continually improving.

Compared to conventional energy sources, solar energy is starting to get competitive. It is expected that between 2013 and 2020 prices for solar cell and conventional energy will match [3].

In this thesis we focus on efficiency of solar cells, specifically by maximizing the usage of active surface area of cells.

1.1 EDGE ISOLATION

In most industrial type solar cell processes the *edge isolation* is an important step. During processing the active front side of the cell becomes electrically connected to the rear, causing a shunt. Edge iso-

lation provides electrical separation between these. An example is shown in Figure 2.

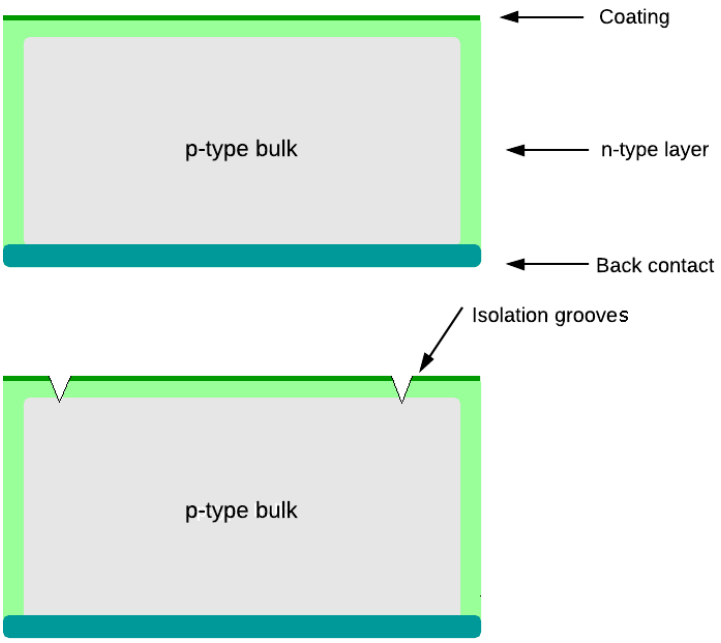


Figure 2: Edge isolation example.

Several methods are used in order to perform edge isolation. A common technique is to remove this shunt by plasma etching of the wafer stack. Other techniques are grinding of the wafer's edge using sandpaper and cutting of isolation trenches by either milling or laser [5].

All options have their pros and cons but the most important aspect is that isolation takes place around the entire contour of the wafer. Any gaps in the isolation will lead to a short-circuit situation and therefore losing efficiency of the cell.

1.2 PROBLEM STATEMENT

During fabrication there is a possibility of small defects appearing in the wafer's edge. These defects can disturb the edge isolation process. A notch in the edge, for example, could cross the position at which electrical separation by laser would occur, leaving a short-circuited wafer. Figure 3 shows a portion of a wafer's edge (dark) on a light background with a notch in it, possibly leading to an isolation fault.

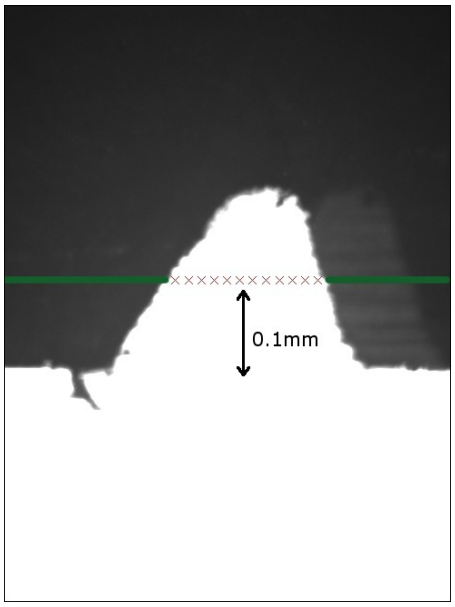


Figure 3: Isolation fault.

To overcome this problem edge isolation takes place at a fixed distance from the wafer's outer edge. In most processes this fixed distance is 1mm to make sure small defects cause no shunt, wafers with larger faults are discarded due to structural integrity issues. A disadvantage of this method is that it wastes a significant portion of surface area that is perfectly useful.

The major goal of this thesis is to position the edge isolation closer to the outer contour of the cell to maximize effective surface area and improve efficiency.

1.3 FUNCTIONAL REQUIREMENTS

Several requirements have been proposed that should be met in order to successfully bring the project to a close.

For one, processing of wafers with dimensions of 156x156mm should be processed within 1.5s as a typical plant has a throughput of 2400 wafers/hour.

Secondly, the current 1mm distance from the edge on which isolation takes place should be reduced to 0.1mm.

The maximum edge faults that need to be corrected for are 1mm. Wafers with larger defects are discarded due to structural integrity.

A 28µm wide line will be printed along the wafer's edge, it may never happen that the continuity of this line is broken, either by faults in printing or the crossing of a fault in the cell.

To find the maximum efficiency gain of the reduction we can consider a 156x156mm wafer with no faults and 20% efficiency. The efficiency gain is then given by Equation 1.

$$\Delta\eta = \frac{A_n - A_o}{A_o} * 20\% \tag{1}$$

Where η is the efficiency gain and A_n and A_o are the new and old effective surface areas respectively. A_n and A_o are simple area calculations given by the following equations (with w and h the wafer width and height):

$$A_n = (w - 2 * 0.1)(h - 2 * 0.1) A_o = (w - 2 * 1.0)(h - 2 * 1.0) \tag{2}$$

Filling in values we obtain an efficiency gain of 0.47%. Such gains are considered significant in solar cell development. Figure 4 illustrates the area gain.

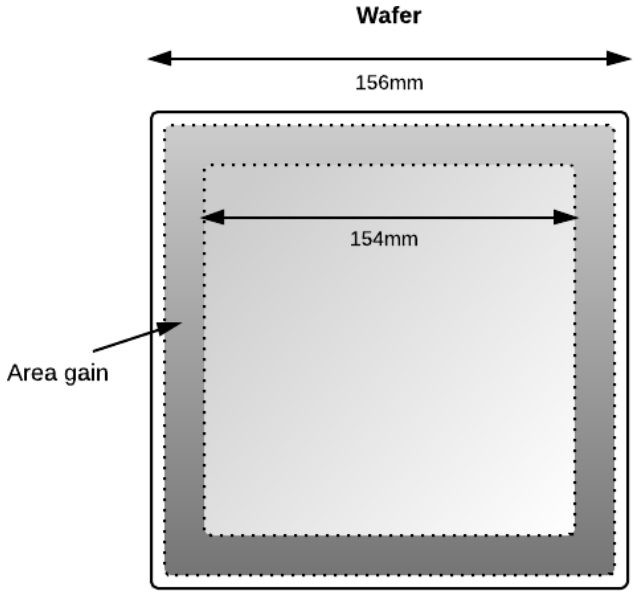


Figure 4: Area gain.

The tasks of the thesis are limited to system analysis, motion and vision. Actual printing falls outside the scope of the project.

1.4 CONTRIBUTIONS

Time and accuracy are an important trade-off in many industrial process. For the visual tracking of the wafer's edge it is no different and such a trade-off must be analyzed to find the most optimal case.

As a number of system parameters will influence the speed and accuracy of edge tracking analysis incorporating all these parameters

is useful. Both for current system parameters as well as an analysis tool for future modifications.

Some system parameters are fixed, others are more flexible. It is crucial to know what parameters will break the stated functional requirements and to what extent.

The main contributions of the research conducted are then:

- Time versus accuracy trade-off in a visual tracking application.
- An analysis of visual tracking task given system parameters.
- Enumeration and quantification of system limits breaking functional requirements.

1.5 ORGANIZATION

The remainder of this thesis is organized as follows. Chapter 2 will describe related work. In Chapter 3 an analysis is given of the system and its parameters. Vision algorithms used for detecting edges in images can be found in Chapter 4. Vision is a prerequisite for visual tracking which is in Chapter 5.

Integration of developed components will be discussed in Chapter 6. Results of the tracking application are then analyzed in Chapter 7. Finally a conclusion and a description of future work are given in Chapter 8.

2 RELATED WORK

To perform analyses in this thesis and develop algorithms, work related to different aspects of the project have been researched. These are described below for three different themes: general, vision and tracking.

2.1 GENERAL

Pearce[4] analyzes the technical, social and economic benefits and limitations of photovoltaic energy production and how a shift from fossil fuel consumption to solar energy will affect costs. Although not directly related to the work in this thesis it is an interesting paper and it shows the importance of new energy solutions.

In a case study, Ye et al.[6] show that in high speed visual servoing, exposure and/or readout time can be the bottleneck in the vision pipeline, depending on the lighting condition, surface type and image size. Moreover, they state that time predictability is a critical requirement of visual servoing in industrial environments. Real-time guarantees should be met on the algorithmic level, hardware implementation and camera interface. Keeping these aspects in mind we analyze the framerate, image size and vision processing time to find an optimal solution regarding tracking the wafer's edge.

The details and importance of locking the memory used by real-time processes is given in [7]. Locking memory is crucial in our application as even small latencies from memory IO can disrupt the control which in turn may lead to faulty wafers.

2.2 VISION

Heath et al.[8] compared various edge detection algorithms. They found that no method exists to qualify a single algorithm as performing best. Different techniques work best in different contexts. In order to find algorithms that perform well in edge detection we keep this in mind and analyze the images and detected images in them. Intensity values and noise present provide a good measurement of how edge detection should be performed.

Probably the most widely used edge detector is the one by Canny[9]. Canny's aim was to develop an algorithm that has:

- Good detection: real edges must be marked.

- Good localization: marked edges should be as close as possible to the ones in the image.
- Minimal response: no false edges should be marked, real edges should be marked only once.

Several steps of the Canny algorithm are used for edge detection. In each step we try to find the most optimal parameters for our case.

Pavlidis[10] describes an algorithm for tracing contours in images. We use this algorithm to trace detected edges in images. When doing so we find a continuous edge segment in each image.

2.3 TRACKING

Hutchinson et al.[11] provide a tutorial on visual servo control. In this work they guide the reader through the concepts involved in visual servoing from different fields and supply references for further reading for each subject. It is a useful work for getting a grasp of the aspects involved in visual control.

Spong et al.[12] use polynomials to generate time-optimal trajectories. By choosing an appropriate polynomial and setting up a system of linear equations a trajectory with bounds on time derivatives such as velocity, acceleration and jerk can be obtained. We use their analysis to estimate how our system will react when faults are detected in the wafer's edge and correcting motion needs to be performed.

A different approach towards planning trajectories is by filtering techniques which alter an infeasible trajectory into a feasible one. Bianco and Gerelli[13] use filtering to obtain feasible trajectories. Because an infeasible trajectory is obtained by edge measurement we use filtering to create a trajectory the system is able to follow and at an offset from the wafer's edge.

Part II

SYSTEM DESIGN

The problem at hand requires careful analysis in order to meet the goals that have been set. In this section such analysis is performed.

System parameters influence if and to what extent goals can be met. Algorithms for motion and vision are detailed as well as their integration into the system.

3 SYSTEM ANALYSIS

In this chapter the experimental setup that is used during the thesis will be described. Knowing system parameters, we can analyze crucial aspects of the project.

First an overview of the system is given, followed by a discussion of boundary conditions regarding printing. Afterwards analysis is performed varying system parameters to find limits on what is achievable and how they affect the functional requirements that were set out.

3.1 SYSTEM OVERVIEW

The PiXDRO LP50 Inkjet Printer is an entry level R&D inkjet system. It is designed for evaluation and development of inkjet materials as well as research and development of inkjet processes and applications. Figure 5 illustrates major components of the LP50.

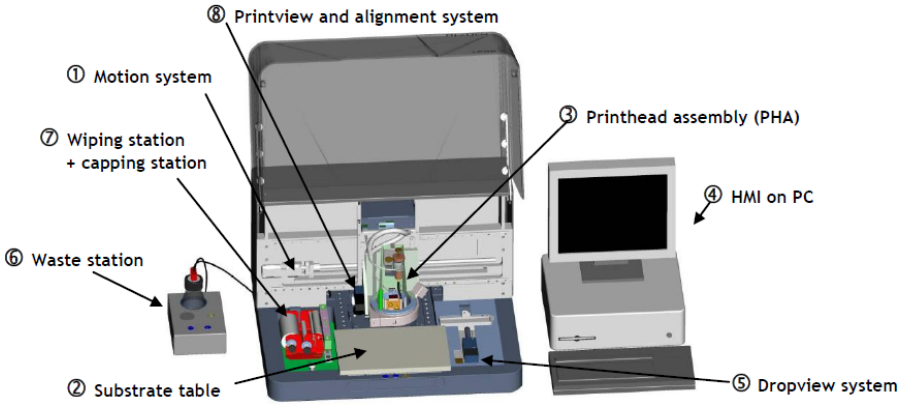


Figure 5: LP50 components.

An LP50 available at Roth & Rau will be used for experiments during the thesis. Most important components are the motion system (x- and y-directional movement), the camera (prerequisite for vision) and the host PC (computation).

A modification was made to the motion system by replacing the conventional motion controllers present by high-end motion controllers from ACS. These are controllers have more features than the conventional ones present in the LP50 system, allowing on-the-fly generation of trajectories and logging of data during motion. An overview of the entire system is given in Figure 6.

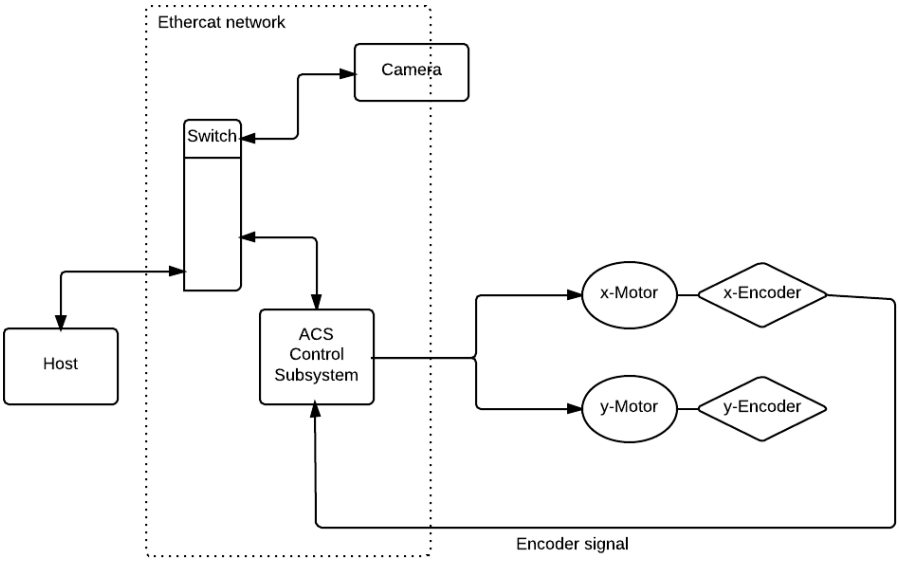


Figure 6: LP50 overview.

An important property of the ACS subsystem is the use of Ethercat for communication, Ethercat provides real-time guarantees which are of importance when implementing a control system.

The host will act as a setpoint generator for the motion system. It will do so by fetching images from the camera and computing desired positions. These positions are fed to the ACS subsystem which will steer the system to them. The ACS subsystem receives feedback from the motor encoders, closing the control loop.

Table 3 gives the maximum velocity and acceleration for the axes of the system.

AXIS	VELOCITY (mm/s)	ACCELERATION (mm/s^2)
X	200	1200
Y	400	1500

Table 1: Axes limits.

3.2 THE PRINTING TASK

The LP50 system printhead and camera are mounted in a fixed position, alongside each other. This configuration poses several boundary conditions when trying to print a line alongside the contour of the wafer.

Suppose the camera is located to the right of the printhead and the camera passes over a side of the wafer in positive x-direction. In this case the camera can fetch images, the host can compute coordinates and the ACS subsystem can control the motion in order to move around intrusions.

When moving in negative x-direction a problem occurs. The printhead passes over the wafer before the camera does hence potential intrusions cannot be corrected for by the control loop. Both cases are illustrated in Figure 7.

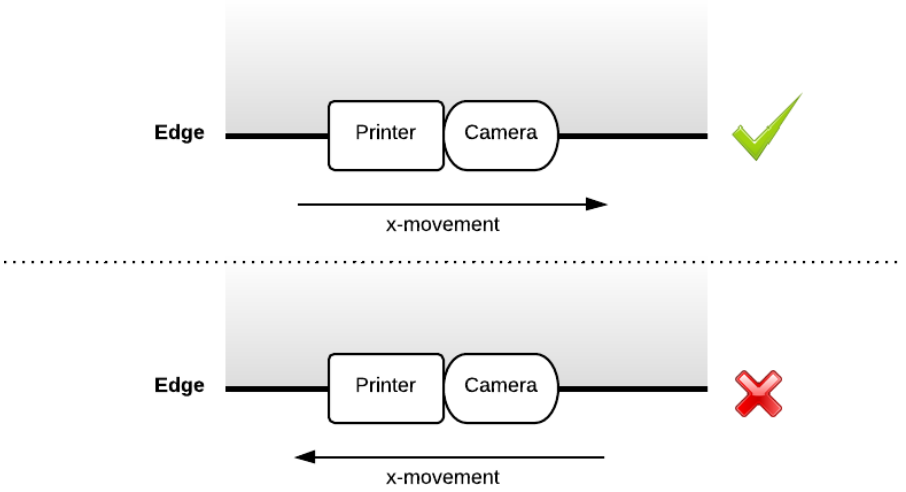


Figure 7: Printing problem.

Similar but equally problematic things happen when moving in y-direction, only now the printhead is always next to the camera instead of in front of it. This case is shown in Figure 8.

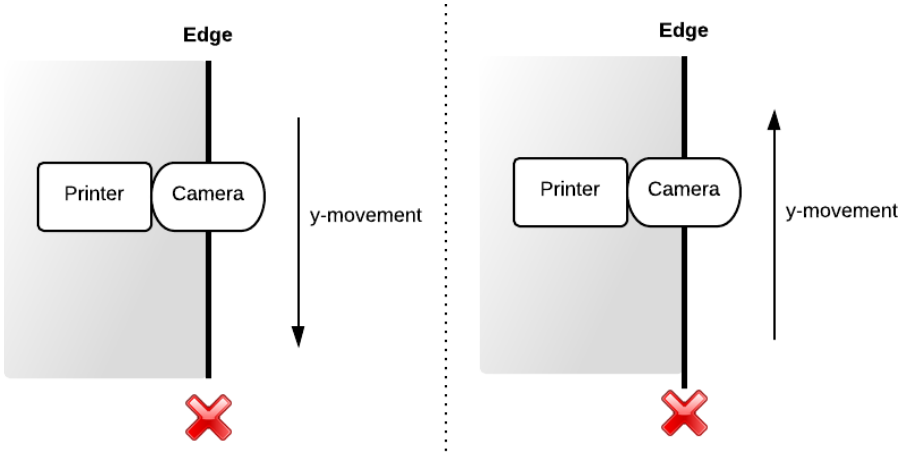


Figure 8: Printing problem.

An way out of this dilemma would be to first track the wafer with the camera and store all edge positions. When finished these positions can be fed to the motion controller, after accounting for the offset between camera and printhead. Doing so would incur a large timing penalty since the system would have to travel along the wafer's contour twice.

Another method would be to place two cameras on either side of the printhead. The printhead has the ability to rotate by 100°. When

a corner of the wafer is reached the printhead can be rotated by 90°, thus always making sure there is a camera in front of the printhead. A top-view of such a printhead with cameras is shown in Figure 9.

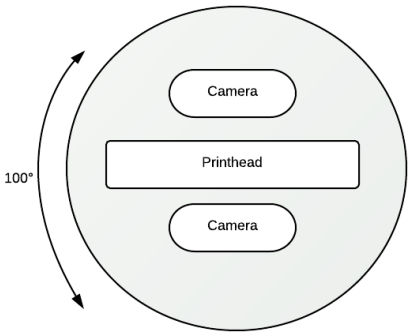


Figure 9: Printing problem.

This method would be attractive to use in future reseach.

3.3 FRAMERATE AND FOV BALANCE

Capturing images using the camera while tracking the contour of a cell, it must never happen that a portion of the edge is missed. The velocity at which the side can be tracked therefore depends on the field of view (FOV) and the framerate of the camera. The field of view is a camera parameter that can vary between 1x1px (minimum) and 656x492px (maximum). The framerate in turn depends on the chosen framesize: a larger framesize means a larger transfer delay of images to the host and therefore a lower framerate.

A balance must be found between FOV and framerate such that boundary conditions are fulfilled. No portion of the edge can be missed when the wafer's edge is scanned.

Maximum velocity

To find the velocity at which an edge can be tracked in x-direction, we keep the size of frames in y-direction at maximum (656px) and vary the size in x-direction and velocity such that images do not overlap nor have gaps between them. This is illustrated in Figure 10.

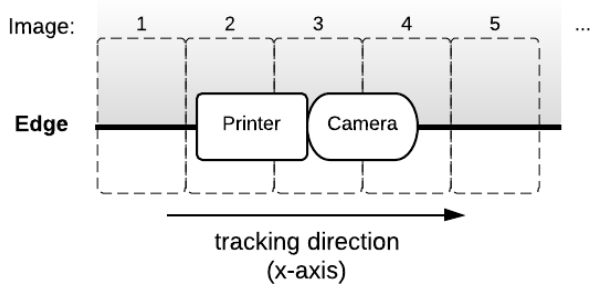


Figure 10: Tracking task.

The framerate for every framesize is a statistic that can be gathered from the camera. The width of each image is then given by:

$$w = n * s_p \tag{3}$$

Where $n \in [1, 492]$ is the width of the image in number of pixels and s_p is the width of a pixel in μm ($3\mu m$ for our camera). The velocity at which there is no overlap between frames nor any gaps between frames is then given by:

$$v = w * f_c \tag{4}$$

Where f_c is the framerate in frames per second. Figure 11 shows the framerate of the camera as a function of image width (a), distance per frame (Equation 3) (b) and the maximum velocity given image width (Equation 4) (c).

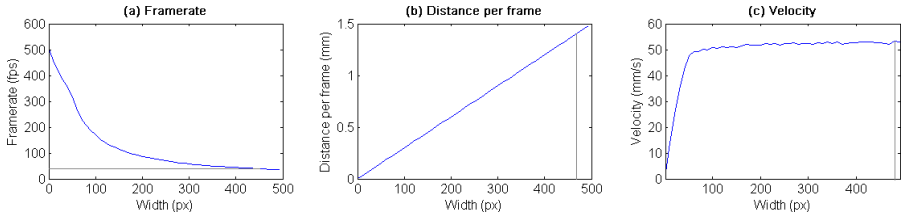


Figure 11: Velocity as function of camera rate.

The maximum velocity in x-direction is $53.3mm/s$, which is significantly lower than the system maximum of $200mm/s$.

Maximum defect

Given the maximum framesize of 656x492px and the pixel size of 3x3μm we find that the field of view is 1.97x1.48mm. Considering that our goal is to print a line $0.1mm$ from the outer wafer edge, thus keeping the position of $0.1mm$ from the edge at the center of the camera, the maximum faults we can detect is only $\frac{1.97}{2} - 0.1 =$

0.88mm. This variation is smaller than outlined in the functional requirements, meaning that the largest defects we are interested in can not be tracked.

Vision rate

Finding feature points (edges) in an image streamed by the camera must finish before the next image comes in. When this constraint is not satisfied delays add up in the control loop and correcting for variations in the edge will not happen in time. The computation time for the vision algorithms - discussed in Chapter 4 - is measured for varying image width. These results are shown in Figure 12 together with the inverse computation time set out against the camera framerate.

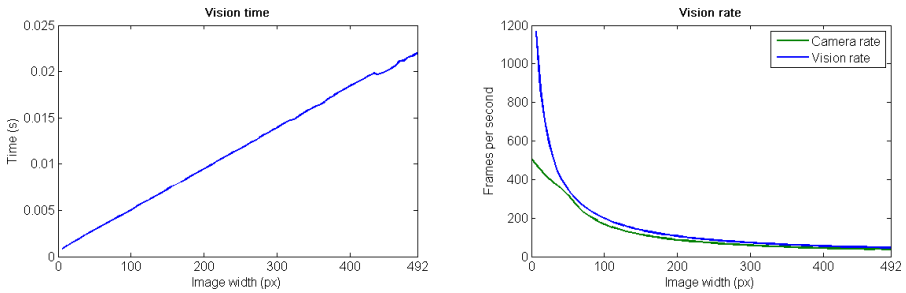


Figure 12: Vision rate.

From these results we gather that the vision rate is always higher than the camera rate which satisfies the constraint on vision computation time.

3.4 CORRECTION ANALYSIS

It must never happen that the outer contour of the wafer is crossed while printing a line along it. Given the limits on velocity and acceleration of the system as listed in Table 3 we can analyze if and how we are able to correct for the maximum variation of 1mm in time. In time means that a correcting motion has completed before the printhead reaches the point where the defect occurs. Figure 13 shows such a defect with a correcting motion.

Correcting motion

Suppose a wafer's edge is tracked in x-direction at some velocity and a 1mm intrusion is detected, how far in advance does it need to be detected? And how long does it take to execute a correcting motion in y-direction, given the constraints on velocity and acceleration?

Figure 13 illustrates a correcting motion the y-axis could make when a fault occurs in the edge while traversing the edge in x-direction.

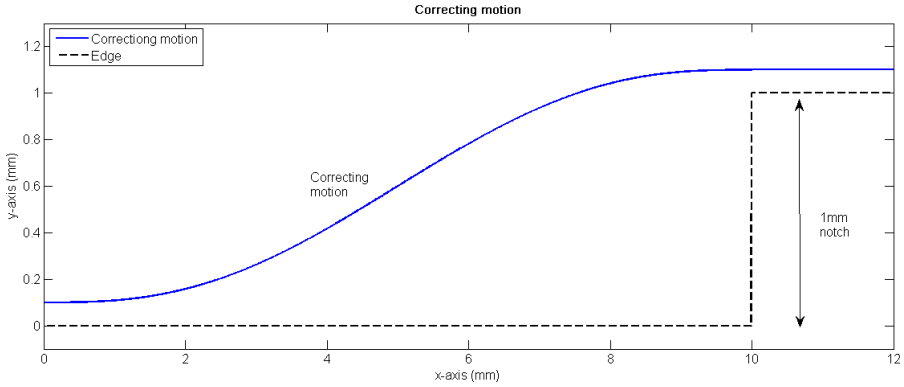


Figure 13: Correcting motion.

We can generate a trajectory for the correcting motion and perform analysis on it to find the required data. A trajectory can be described by a polynomial function of degree n in the form

$$q(t) = a_0 + a_1 t + a_2 t^2 + \ldots + a_n t^n, \tag{5}$$

with $t \in [t_I, t_f]$ where t_I indicates the initial time instant and t_f indicates the final time instant. The $n+1$ polynomial coefficients a_i can be determined while satisfying a number of required constraints. Coefficients can be found by solving the system of linear equations:

$$\mathbf{Ta} = \mathbf{b}, \tag{6}$$

Where **T** is the Vandermonde matrix, **a** contains the unknown polynomial coefficients and **b** the constraints the polynomial should satisfy.

In order to obtain trajectories with continuous acceleration there need to be constraints on the initial and final acceleration as well as position and velocity. This leads to a fifth degree (quintic) polynomial with the following constraints:

$$q(t_I) = q_I, \quad q(t_f) = q_f, \tag{7}$$
$$\dot{q}(t_I) = v_I, \quad \dot{q}(t_f) = v_f, \tag{8}$$
$$\ddot{q}(t_I) = a_I, \quad \ddot{q}(t_f) = a_f. \tag{9}$$

Solving Equation 6 with these constraints we obtain the following coefficients:

$$\begin{aligned}
a_0 &= q_I \\
a_1 &= v_I \\
a_2 &= \frac{1}{2}a_I \\
a_3 &= \frac{1}{2t_f^3}[20q_f - (8v_f + 12v_I)t_f - (3a_I - a_f)t_f^2] \\
a_4 &= \frac{1}{2t_f^4}[-30q_f - (14v_f + 16v_I)t_f - (3a_I - 2a_f)t_f^2] \\
a_5 &= \frac{1}{2t_f^5}[12q_f - 6(v_f + v_I)t_f + (a_f - a_I)t_f^2].
\end{aligned} \tag{10}$$

As the initial and final velocity and acceleration and initial position are zero (correcting movement stops when target position is reached), we can substitute for these and obtain the following non-zero coefficients:

$$\begin{aligned}
a_3 &= 10\frac{q_f}{t_f^3} \\
a_4 &= -15\frac{q_f}{t_f^4} \\
a_5 &= 6\frac{q_f}{t_f^5}.
\end{aligned} \tag{11}$$

Substituting these coefficients in the characteristic polynomial given in Equation 5 and taking the first and second derivative we obtain equations for position, velocity and acceleration.

$$\begin{aligned}
q(t) &= 10\frac{q_f}{t_f^3}t^3 - 15\frac{q_f}{t_f^4}t^4 + 6\frac{q_f}{t_f^5}t^5 \\
\dot{q}(t) &= 30\frac{q_f}{t_f^3}t^2 - 60\frac{q_f}{t_f^4}t^3 + 30\frac{q_f}{t_f^5}t^4 \\
\ddot{q}(t) &= 60\frac{q_f}{t_f^3}t - 180\frac{q_f}{t_f^4}t^2 + 120\frac{q_f}{t_f^5}t^3
\end{aligned} \tag{12}$$

There is a maximum acceleration the system can achieve. Knowing that we want q_f to be $1mm$ we can calculate what the maximum value of t_f is that does not break the acceleration limits. Acceleration is maximal at the point where its derivative (Equation 13) equals zero.

$$\dddot{q}(t) = 60\frac{q_f}{t_f}t - 360\frac{q_f}{t_f}t + 360\frac{q_f}{t_f}t^2 \tag{13}$$

Equating this to zero and solving for t gives the following solutions:

$$\dddot{q}(t) = 0 \quad \Rightarrow \quad t = \frac{1}{6}(3t_f \pm \sqrt{3}t_f) \tag{14}$$

Now substitute this solution back in the equation for acceleration and set equal to the maximum acceleration ($1500mm/s^2$):

$$1500 = 60\frac{q_f}{t_f^3}[\frac{1}{6}(3t_f - \sqrt{3}t_f)] - 180\frac{q_f}{t_f^4}[\frac{1}{6}(3t_f - \sqrt{3}t_f)]^2 + 120\frac{q_f}{t_f^5}[\frac{1}{6}(3t_f - \sqrt{3}t_f)]^3 \tag{15}$$

Solving Equation 15 for t_f finally gives the value $t_f = 62.0ms$ which means it takes the y-axis $62.0ms$ to perform a correcting motion for a defect of $1mm$. Figure 14 shows the generated trajectory graph and accompanying velocity and acceleration.

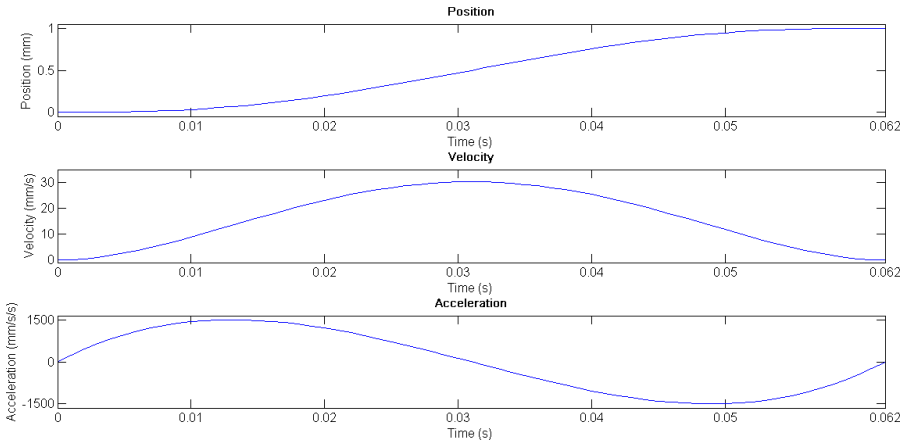


Figure 14: Polynomial trajectory.

Using full image frames the maximum velocity the x-axis can move at without missing edges is $52.8mm/s$. Therefore the x-axis will have traveled $0.062 * 52.8 = 3.27mm$. This is a larger distance than the width of images which means that we cannot directly act on detected faults. In other words: there is a lag between the vision and motion processes. Because the printhead will be at least a couple of centimeters behind the camera this is not necessary a problem regarding printing. It does however mean that for every side of the wafer there is an additional time cost of $62.0ms$ which is $0.25s$ in total, a significant portion of our total processing time of $1.5s$.

Taking no overhead into account (such as moving to a new position after a corner has been passed), we can calculate the total processing time for a wafer. Given the maximum velocity of $52.8mm/s$ and wafer size of $156x156mm$ the total processing time is $11.8s$, which is far longer than the proposed $1.5s$ for a plant with a throughput of 2400 wafers per hour.

Area loss

Defects in the edge cost area. Additionally, moving around them costs area because time and space are needed to move to the correct posi-

tion. Exactly how much area is wasted when a maximum-size defect occurs can be calculated from the analysis above.

First we replace the value of t_f we found by the distance the x-axis will have traveled while the correcting motion takes place (3.27mm). This results in the correcting motion trajectory being a function of x-displacement instead of time.

Substituting the new t_f (3.27mm) and old q_f (1mm) in Equation 12 and integrating we find the following result:

$$\int_0^{3.27} (0.285t^3 - 0.131t^4 + 0.016t^5)\,dt = 1.61mm^2 \tag{16}$$

This area is actually lost twice since the opposite movement takes place after the default. Figure 15 presents the area loss graphically.

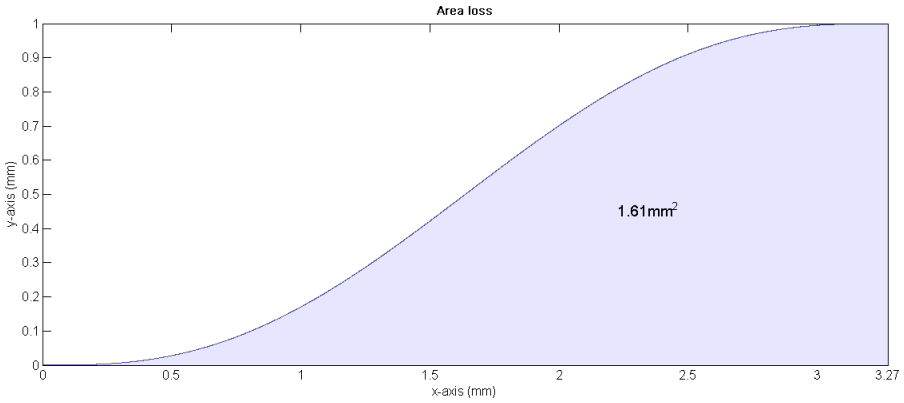


Figure 15: Area loss.

Printing frequency

Printing a line is performed by firing droplets of ink from a printhead. Although actual printing falls outside the thesis scope it is useful to consider the effect of correcting motion on printing.

There can never be a gap between droplets. Therefore the frequency at which the printhead fires droplets is determined by the displacement in x- and y-directions and the size of droplets. No defects in the edge means there is only a constant x-velocity and in turn a constant firing frequency.

The number of drops per displacement is then given by the total displacement divided by the diameter of a droplet:

$$n = \frac{\sqrt{\Delta x^2 + \Delta y^2}}{\varnothing}. \tag{17}$$

The time in which these n droplets need to be fired however only depends on the x-velocity and is therefore constant:

$$t = \frac{\Delta x}{v_x}. \tag{18}$$

Finally, the actual printing frequency is then simply the number of drops per second:

$$f_{print} = \frac{n}{t}. \tag{19}$$

Figure 16 shows the printing frequency for a wafer side of 156mm with a 1x1mm notch right in the center. The velocity was set to 52.8mm/s and droplet diameter to 28µm.

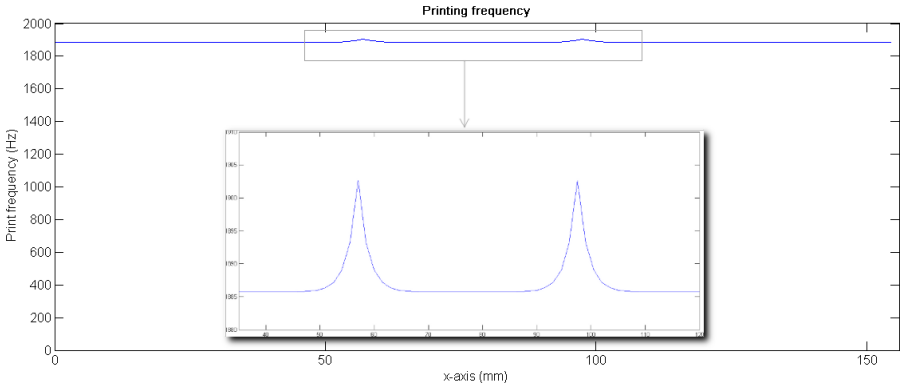


Figure 16: Printing frequency.

Because the slope of trajectories in the analysis given above is quite flat the printing frequency does not vary significantly ($< 1\%$) from which we can conclude that printing frequency does not pose a problem and can be neglected.

3.5 CONCLUSION

Some limitations were observed that will have an effect on the functional requirements that were set out.

First of all, due to the printing limitations only one side of the wafer can be used properly for experimentation. This is not necessarily a problem because other sides will behave similarly, only hardware modifications are needed before they can be verified.

Another system component that gives limitations is the camera. The framerate is too low to be able to move at maximum velocity without missing edge information. The field of view is also too small to see the maximum 1mm edge variations.

The limit on acceleration induces extra processing time. This extra time eats at the already small amount of time allocated for edge isolation.

4 VISION

Feature detection, a subfield of computer vision, concerns itself with extracting features from image data such as blobs, corners and edges[14].

The features we are interested in are the points on the boundary between the wafer and background regions, i. e.: edges. These edges are characterized by a sharp change in brightness. An optimal edge detector should posses the following properties:

- *good detection* - real edges should be marked.
- *good localization* - edges should be marked as closely as possible to their real location.
- *minimal response* - any real edge should be marked only once, image noise should not create false edges.

Given an input image, the goal is to obtain a one pixel thick continuous edge segment that separates the outer wafer contour from the background. Figure 17 shows an example input image on the left and the envisioned result on the right.

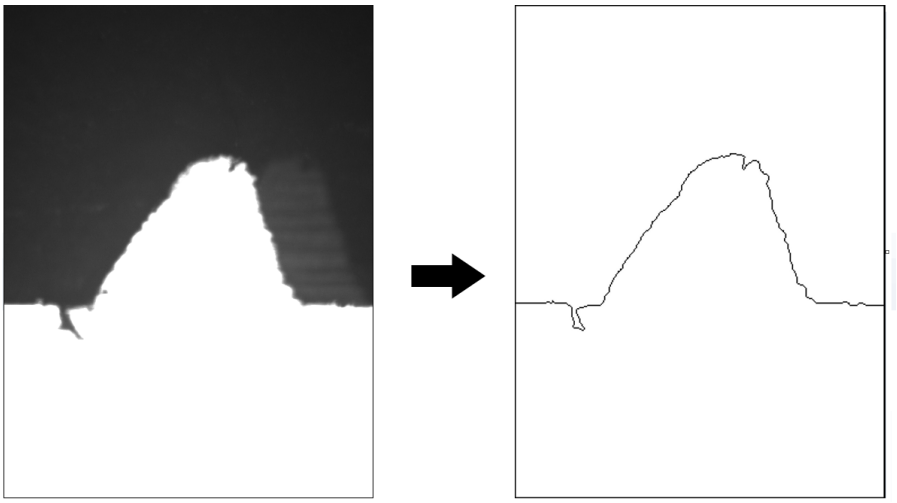


Figure 17: Vision goal.

In this chapter the algorithms for obtaining the desired result will be discussed. How this result fits in the visual tracking task is addressed in Chapter 5.

4.1 BACKGROUND CONSIDERATIONS

Wafers are placed on a substrate carrier. As we are interested in the contours of wafers the contrast between the wafers edge and the carrier itself is of importance.

Figure 18 shows a portion of the wafers edge (dark) on a bare substrate carrier. It can be seen from this image that the background is not uniform in colour and shows a lot of image intensity variations. These might disturb the measurement of the actual edge.

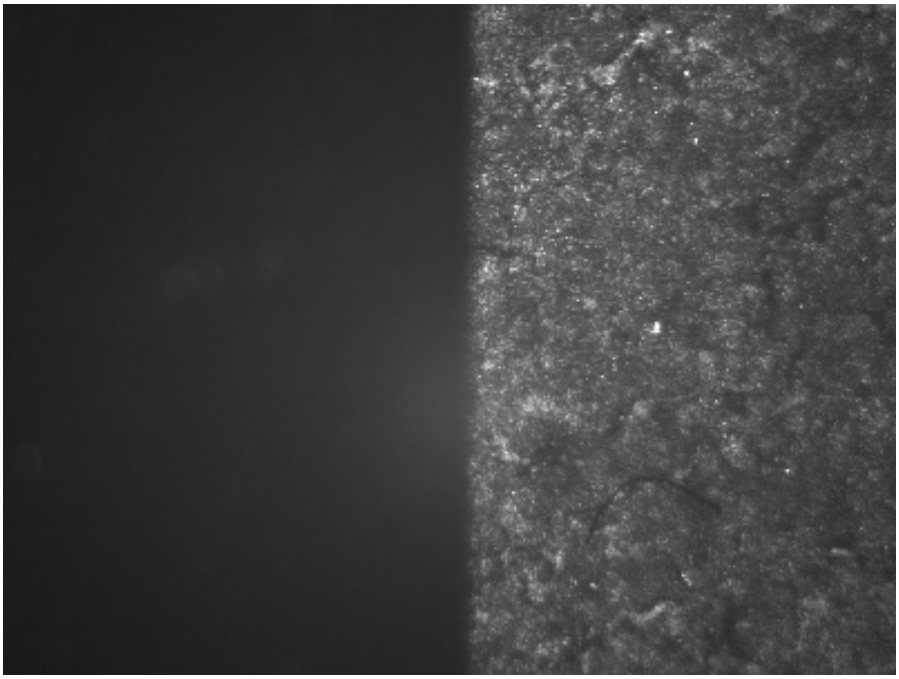


Figure 18: Carrier background.

To prevent this, different surfaces have been placed underneath the wafer. Figure 19 shows the same portion of the edge with a white paper background (left) and an aluminum background (right). This figure also shows the intensity plots for a single horizontal line.

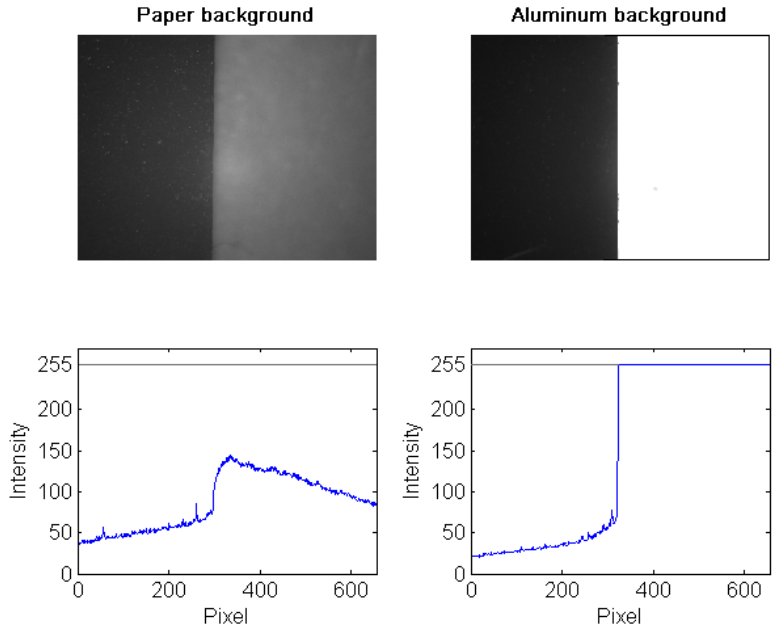


Figure 19: Background alternatives.

The step in intensity between wafer and background is much higher for the aluminum background than for the paper background. Another thing of note is the dropping off of intensity after the step in the paper background. The light source used is non-uniform across the image. This is not an issue for the aluminum background due to its high reflectivity. In fact, the intensity is saturated at its maximum value where no wafer is present. This means there is too illumination. Lowering illumination however results in small streaks of the aluminum tape becoming visible, because this might lead to detection of false images we choose to over-illuminate instead.

An reflective aluminum background has the best properties for edge detection and is therefore used throughout the rest of this thesis.

4.2 INPUT FILTERING

In image processing an input image is treated as a two-dimensional signal on which standard signal-processing techniques are applied. A common processing technique is noise reduction. Electronic recording devices such as cameras are subject to various sources of noise. This noise can cause sudden intensity deviations which leads to detection of false edges.

A Gaussian smoothing is a well-known method for reducing high-frequency components in signals. Figure 20 shows an artificially generated one-dimensional signal of a step function with added noise. By convolving this input with the Gaussian bell curve shown in the middle of this figure the output (bottom) is a smoother version of the

input signal. A side-effect is that the step is also slightly smoothed out.

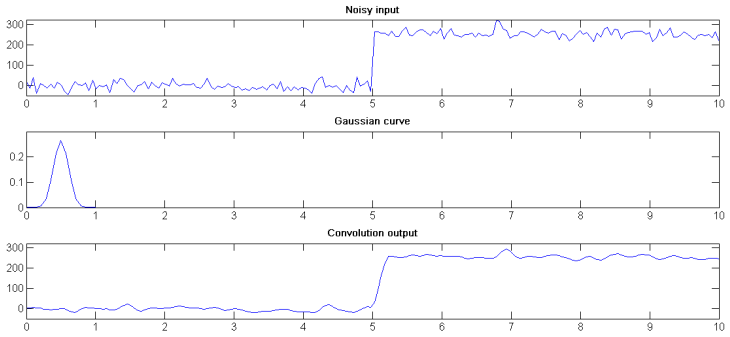


Figure 20: 1D Gaussian filter.

For the input images from the camera we can use a two-dimensional Gaussian function for convolution:

$$G(x,y) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{20}$$

In Equation 20 x and y are the distance from the origin pixel in the horizontal and vertical axis respectively and σ is the standard deviation of the distribution.

In general, more smoothing improves detection of edges, while it hurts localization. Given that pixels are 8-bit grayscale values there is a range of $[0, 255]$ which is also the maximum intensity step. By experimentation a σ of 1.5 preserves the edge while filtering out the disturbing high-frequency components. Because at a distance of more than 3σ from the origin the contribution of pixels becomes negligible, the (square) radius of the filter has been set to 5. This gives the following Gaussian convolution kernel:

$$\frac{1}{271}\begin{bmatrix} 4 & 8 & 9 & 8 & 4 \\ 8 & 15 & 18 & 15 & 8 \\ 9 & 18 & 23 & 18 & 9 \\ 8 & 15 & 18 & 15 & 8 \\ 4 & 8 & 9 & 8 & 4 \end{bmatrix} \tag{21}$$

Convolving the sample input image on the left of Figure 21 with this kernel results in the output on the right.

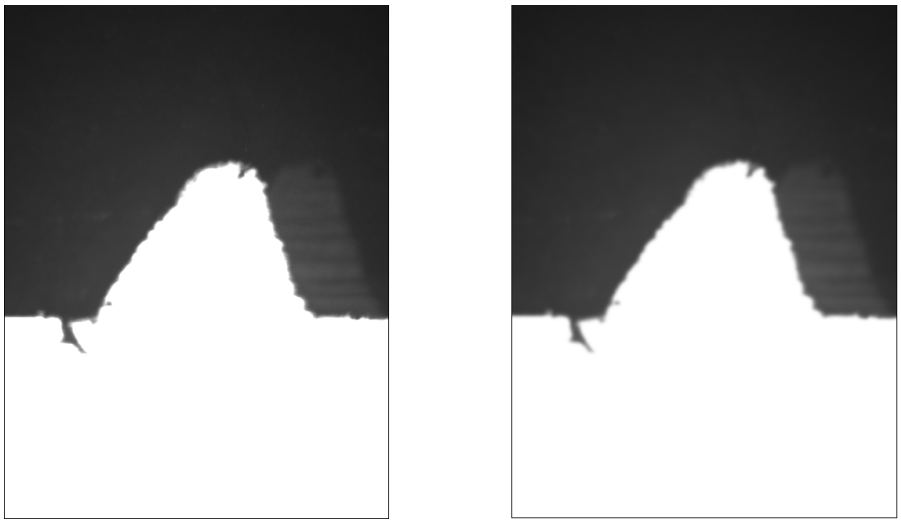


Figure 21: Gaussian blur.

With noise reduced the next step in edge detection can take place.

4.3 IMAGE GRADIENT

Figure 22 again shows an intensity plot with intensity step. In order to find where the edge (step) is exactly located we can take the derivative of this signal. The result is shown at the bottom of the same figure.

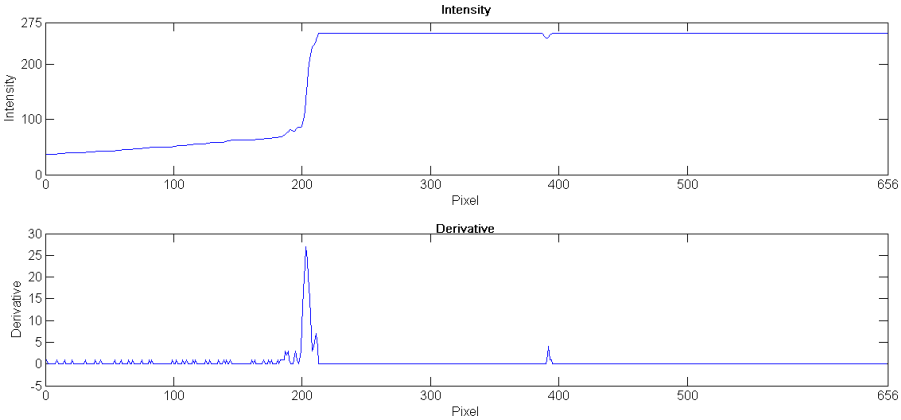


Figure 22: 1D derivative.

Images represent a signal in two dimensions and we are interested in finding edges in both these directions. Using the Sobel operator we can find horizontal and vertical derivatives and combine them to obtain the *image gradient*.

The Sobel operator is a discrete differentiation operator. To find the derivatives convolution takes place with the input image and the Sobel kernels (Equation 22 and Equation 23) for the two dimensions.

$$G_x = \begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix} * A \tag{22}$$

$$G_y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * A \tag{23}$$

Where $*$ denotes the two-dimensional convolution operation and A is the input image.

The image gradient consists of both a magnitude (G) and a direction (Θ), these are computed using the following equations:

$$G = \sqrt{G_x^2 + G_y^2} \tag{24}$$

$$\Theta = \arctan(\frac{G_y}{G_x}) \tag{25}$$

For every pixel in the image these represent the strength of the edge at the pixel's position and the direction in which the edge is oriented at that point.

Figure 23 shows the horizontal and vertical gradients (G_x and G_y) at the top and the image gradient magnitude (G) at the bottom.

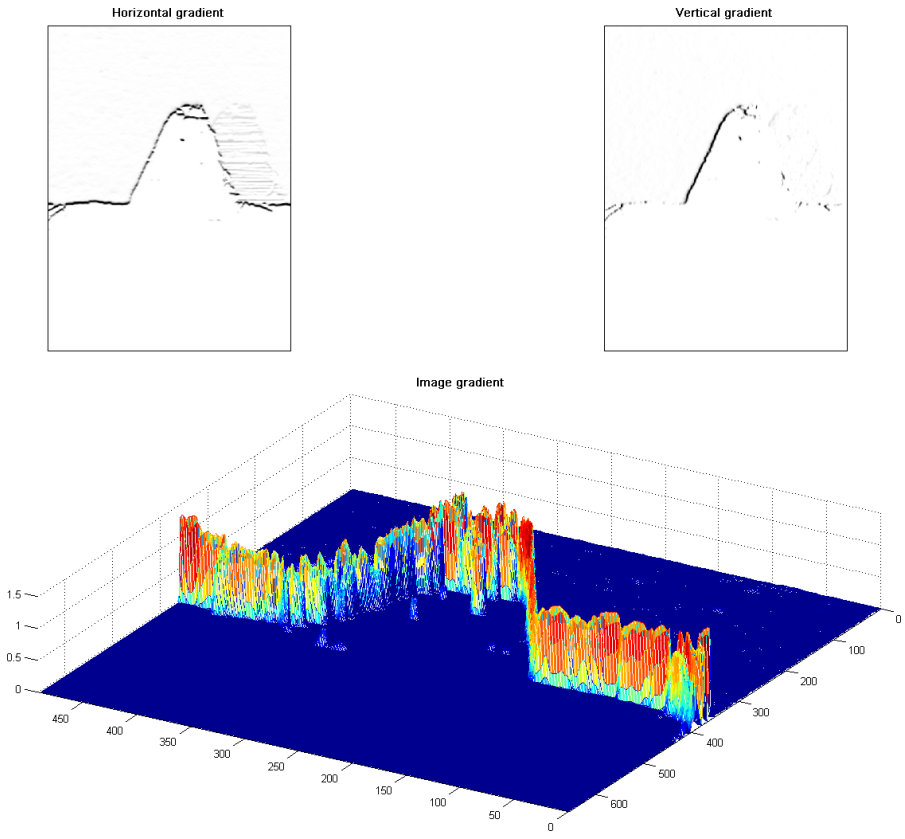


Figure 23: Image gradient.

In this figure we can clearly see the edge we meant to locate.

4.4 EDGE THINNING

By thresholding the image gradient the strongest edges are filtered out and a binary image is obtained. Figure 24 shows the image gradient (left) and the result of thresholding (right).

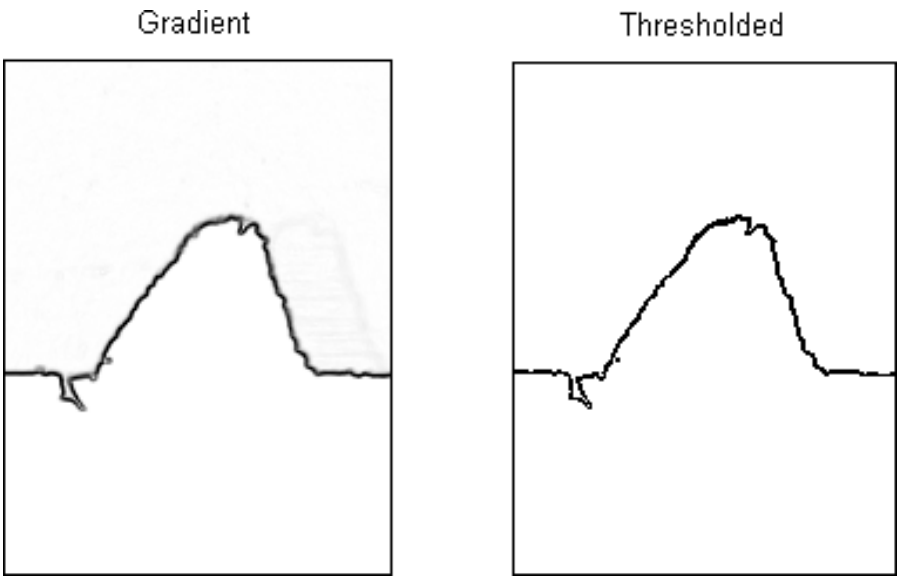


Figure 24: Thresholded gradient.

This result does not meet the goal: a one-pixel thick continuous edge.

To meet our goal, contour tracing is used. Finding a start point in the image (black pixel) and using Pavlidis' contour tracing algorithm we can find consecutive points until we have a complete trace of the edge segment.

In order to find a starting point for tracing it is useful to obtain some information about the nature of the edge segment that is being traced. By looking at the colours of the corners of images we know in what way the edge runs through an image. E.g.: a white top-left and bottom-left corner and black top-right and bottom-right corner means the edge runs from top to bottom. We can distinguish ten such cases. In case 0 the image is entirely white and there are no edges. The other nine cases are illustrated in Figure 25.

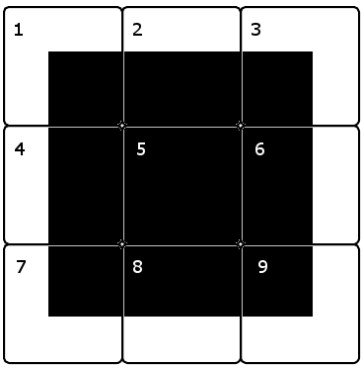


Figure 25: Edge image cases.

Knowing the case of the current image a starting point can be found. E.g.: in case four the search is started at the top-left pixel. We

advance through pixels in x-direction until a black pixel is located which is a starting point.

From a starting point and an initial direction in which to search for the next edgel the Pavlidis' contour tracing algorithm works as follows: look at the three pixels in front from the current direction. If the right pixel (priority 1) is black take a step forward and a step to the right (thus changing direction 90 degrees clockwise) and mark the new edgel. If the right pixel is not black but the middle one is then take a step forward and mark the new edgel. When only the left pixel is black then take a step to the left and then to the right (direction is not changed) and mark the new edgel. When none of the three pixels is black then change direction 90 degrees counter-clockwise and continue the search. These four cases are illustrated in Figure 26.

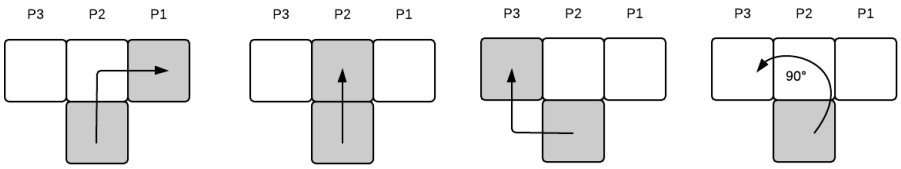


Figure 26: Edge image cases.

The search is ended when a trace has been found throughout the image. Taking case four again as example, the algorithm stops when the bottom row of pixels is reached.

It can happen that the starting point is a pixel belonging to a false edge. In such a case the trace will end up at the starting point, in which case tracing is stopped and from the previous starting point a new search is carried out to find a new starting point.

The end result of the contour tracing is a one-pixel thick continuous edge segment that has been illustrated in Figure 17.

5 MOTION PLANNING

Having developed the required vision algorithms for detecting obstacles in the wafer's edge, we can now plan motion for avoiding them. The vision results give a description of the complete wafer contour. This description can be transformed to generate a trajectory the system must follow. Constraints on velocity and acceleration as well as the fact that the wafer's edge must never be crossed are taken into account while generating the trajectory.

This chapter will describe how world coordinates are obtained from images and how they are used to plan required motion. The chapter ends with a proof on guarantees that constraints are met.

Integration of motion and vision in the system will be discussed in Chapter 6.

5.1 COORDINATE ESTIMATION

Edgels (commonly used term for referring to edge pixels) that have been marked in camera images have a x- and y-value, referring to their position in the image. As the ACS motion system accepts world coordinates for steering towards any position a transformation is needed.

Figure 27 illustrates a portion of the wafer being imaged by the camera. As an image is fetched, the current carrier position is requested. The green dot in the image on the right is located at the center of the image. This point is the current carrier position.

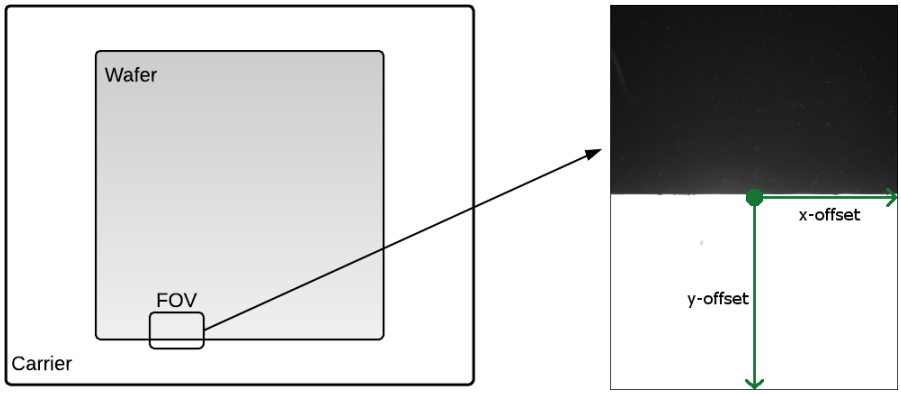


Figure 27: Image position.

By calculating the offset of an edgel from the center and knowing the carrier position as well as pixel size we can find the exact position of each edgel:

$$\begin{aligned} x_e &= (x_i - x_c)s_p + x_w \\ y_e &= (y_i - y_c)s_p + y_w \end{aligned} \tag{26}$$

In Equation 26 x_e and y_e are the world coordinates of an edgel with a position in the image given by x_i and y_i. x_c and y_c are the center position of the image in pixels. s_p is the size of a pixel (assuming width and height are the same). x_w and y_w are the current carrier position in world coordinates.

5.2 TRAJECTORY GENERATION

Having obtained world coordinates from detected edge segments we can generate a trajectory the system is able to follow. The objective is to create a smooth trajectory that complies with the maximum acceleration of the system and never intersects the detected edge.

The approach used for trajectory generation is by filtering. By applying appropriate filters we can alter the infeasible trajectory obtained by edge detection into one that is feasible.

The edge can be shaped in such a way that for a given x-position there exist multiple y-values. Moving then at a constant velocity in x-direction it's not possible to correct for such shapes. Therefore we create a bijective mapping of the input. For any x-position we take the maximum y-value belonging to it as we require the largest intrusion to be tracked. An example for a single image is shown in Figure 28.

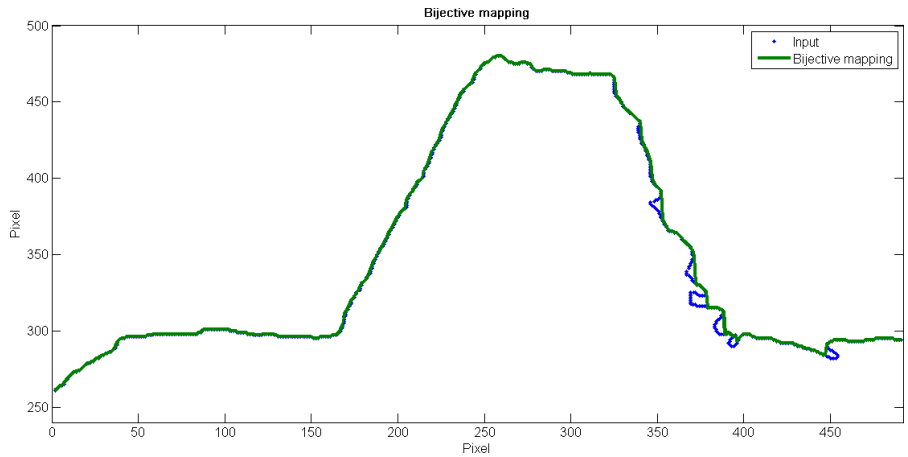


Figure 28: Bijective mapping.

A filter is used to widen detected edge intrusions. By widening intrusions correcting motion can start before the real intrusion is encountered thus giving enough time to reach the height of the intru-

sion before the printhead has reached that position. Figure 29 shows the filter used.

The difference equation for this filter is given by Equation 27. This filter simply takes the maximum value of a number of previous and future inputs as well as the current input. Because the goal is to print a line $0.1mm$ from the edge this value is added as well.

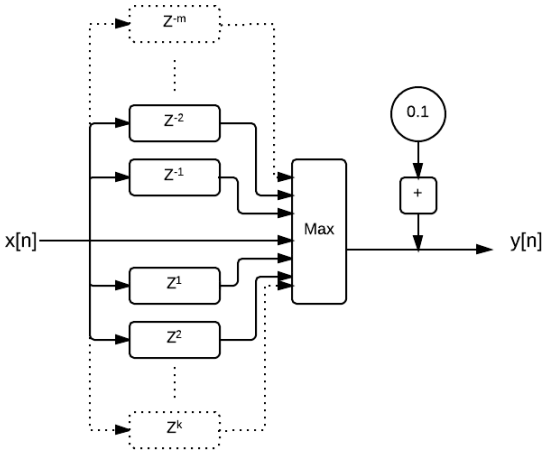


Figure 29: Thickening filter.

The size of the filter, given by m and k determines how much intrusions are widened and therefore how much time correcting motion has so the edge will never be intersected. Note that this filter uses future inputs which is possible since a portion of the edge is known in advance.

$$y[n] = 0.1 + max(x[n-m], \ldots, x[n-2], x[n-1], x[n], x[n+1], x[n+2], \ldots, x[n+k]) \tag{27}$$

Figure 30 plots the response of the filter to a $1mm$ intrusion.

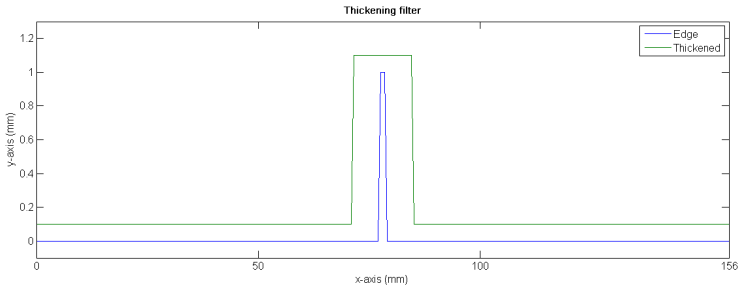


Figure 30: Thickening filter.

To attain a smooth trajectory, the thickened edge can be low-pass filtered. Using Equation 28 results in the response to the thickened edge as shown in Figure 31.

$$y[n] = (1 - \alpha)y[n - 1] + \alpha x[n] \tag{28}$$

The term α in this equation determines the weights of the previous output and current input. Lowering this value attenuates the slope of the response to a step whereas increasing it steepens the slope.

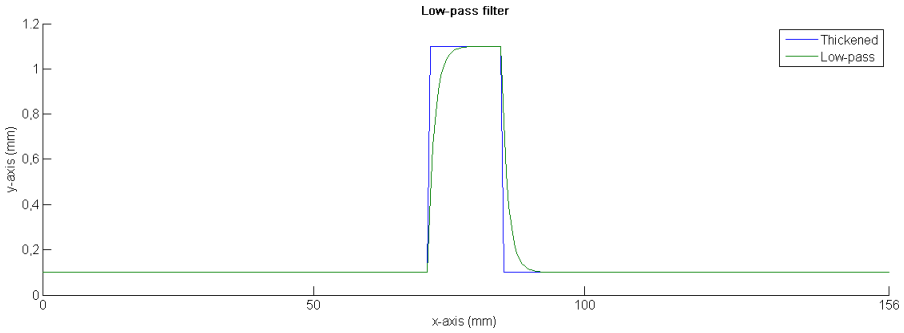


Figure 31: Low-pass filter.

This filters output still shows sharp changes in the bottom left-hand and top-right hand corners of the intrusion. For a smooth trajectory the output of the first low-pass filter is again filtered, but this time starting from the right. Equation 29 is the filter used, where y_d is the output of the double-sided filter and y_l the output of the one-sided filter.

$$y_d[n] = (1 - \alpha)y_d[n + 1] + \alpha y_l[n] \tag{29}$$

The output of this filter, together with the original input and thickened edges, are shown in Figure 32.

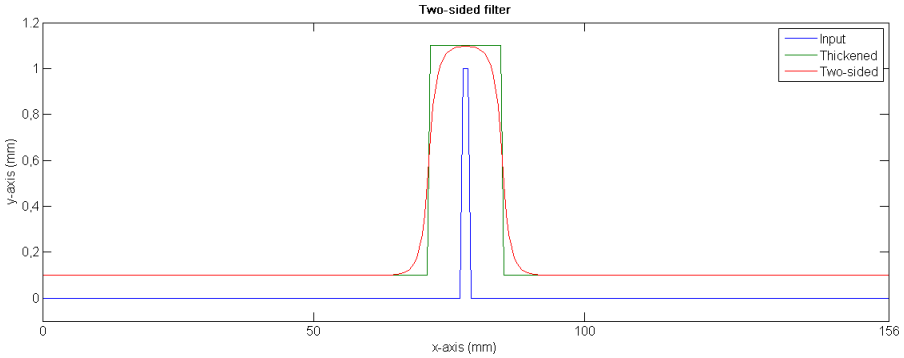


Figure 32: Double-sided filter.

This is a smooth trajectory that can be fed to the motion system.

5.3 FILTER GUARANTEES

As stated before, the trajectory is subject to constraints. We now take a look at what is required to meet these constraints.

First we take a look at how the left-sided filter from Equation 28 reacts to the Heaviside step function.

Knowing the equation we can calculate the following output as response to the Heaviside function:

H(x)	F(x)
H(0)	0
H(1)	α
H(2)	$\alpha + \alpha(1 - \alpha)$
H(...)	...

Table 2: Filter response.

This series can be formulated given the following summation:

$$F(x) = \sum_{i=0}^{\infty} \alpha * (1 - \alpha^i) \tag{30}$$

Using the rules of summation we can solve this to be:

$$F(x) = 1 - \alpha^n \tag{31}$$

The output of this equation is plotted in Figure 33. Note that this is the same response as the left-sided equation from Equation 28.

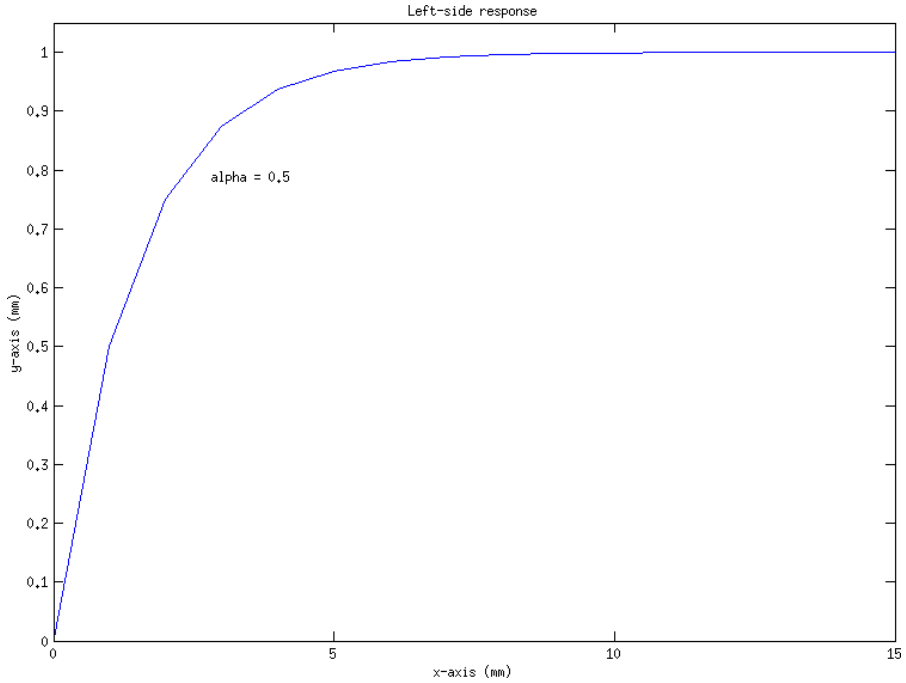


Figure 33: Left-sided response.

In a similar manner we can now find a response from the right-sided filter of Equation 29 with the left-sided equation as input. Only now we need to start in the future.

For some number of samples in the future (n) we know the output of $F(x)$ is:

$$\alpha * (1 - \alpha^n) \tag{32}$$

Knowing the equation for the filter we also know the output one sample before ($n - 1$):

$$\alpha * (1 - \alpha^{n-1}) + \alpha * (\alpha * (1 - \alpha^n)) \tag{33}$$

And again for $n - 2$:

$$\alpha * (1 - \alpha^{n-2}) + \alpha * (\alpha * (1 - \alpha^{n-1}) + \alpha * (\alpha * (1 - \alpha^n))) \tag{34}$$

By simplifying these equations we can see a pattern which is again a summation:

$$\sum_{i=1}^{\infty} \alpha^i (1 - \alpha^i) \tag{35}$$

To continue we take as an example α to be 0.5. For this value the summation above converges to $\frac{2}{3}$.

This value is the response of the filter to the Heaviside function at $H(0)$. Or in other words: where the step from 0 to 1 takes place.

On the left of $H(0)$ the value of $H(x)$ is always 0 and from our filter equations, with α set to 0.5 we know the value keeps halving every sample as we move farther to the left. This side can then be expressed as:

$$L(x) = \frac{2}{3}2^x \quad \text{for } x \leqslant 0 \tag{36}$$

Taking the left-hand limit we find that it converges to 0:

$$\lim_{x \to -\infty} L(x) = 0 \tag{37}$$

Which means that on the left-hand side of $H(0)$ the filter always converges to 0.

On the right side we again look at how the filter behaves using the known difference equations with $\alpha = 0.5$:

H(x)	G(x)
H(0)	$\frac{2}{3}$
H(1)	$0.5 * \frac{2}{3} + 0.5 * 1$
H(2)	$0.5 * (0.5 * \frac{2}{3} + 0.5) + 0.5 * 1$
H(...)	...

Table 3: Filter response.

Looking at this series we find it can be expressed as:

$$R(x) = \frac{2}{3}0.5^x + \sum_{i=1}^{x} 0.5^i \tag{38}$$

Which can be simplified to:

$$R(x) = \frac{2}{3}0.5^x + (1 - 0.5^x) \tag{39}$$

Again we can find limits:

$$\lim_{x \to +\infty} R(x) = \lim_{x \to +\infty} \frac{2}{3}0.5^x + \lim_{x \to +\infty} (1 - 0.5^x) = 1 \tag{40}$$

So on the right of $H(0)$ we find that the filter converges to 1. Figure 34 plots the response of our filter to the Heaviside function.

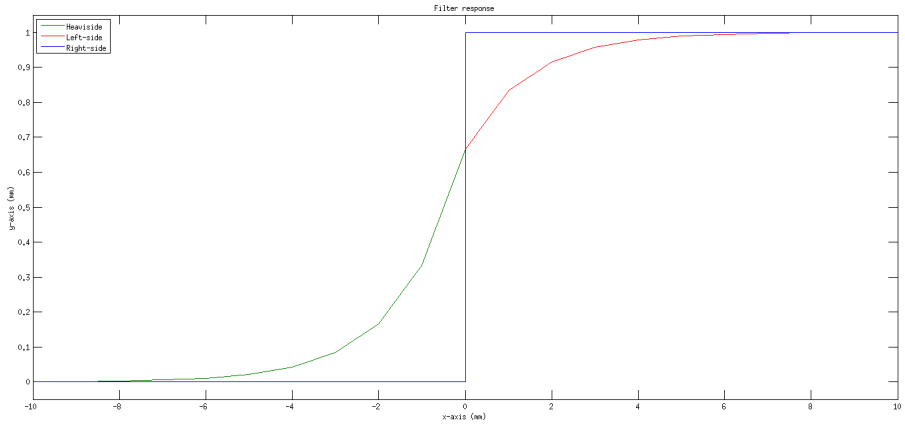


Figure 34: Filter response.

As the filter converges to 0 on the left and to 1 on the right we know it reacts as expected. Knowing this, proper values for m, k and α can be chosen in equations Equation 27 and Equation 28 such that convergence to position 1 happens before the printhead arrives at that location.

6 INTEGRATION

Having finished the required algorithms for both vision and motion it is time to look at implementation issues. First software design is discussed. Then real-time aspects of the developed system are discussed.

Motion blur is an effect that occurs when imaging while moving which can disrupt the process and is discussed as well.

Finally verification takes place on a sample input. Detailed analysis of results can be found in Chapter 7.

6.1 SOFTWARE DESIGN

The developed system consists of three independent subsystems. They are:

- *Camera:* supplies the host with images to be processed.
- *Host:* computational unit that calculates the edge positions from images and provides setpoints to the motion controller.
- *Motion controller:* receives setpoints from the host and steers the axes towards these positions.

These subsystems are interconnected by ethernet. The host acts as master and interfaces with the motion controller and camera to generate the required motion for edge tracking.

Figure 35 illustrates the system using a class diagram with EdgeTracker and Vision being the parts running on the host. Although the components in this diagram are not necessarily classes it serves to illustrate the system.

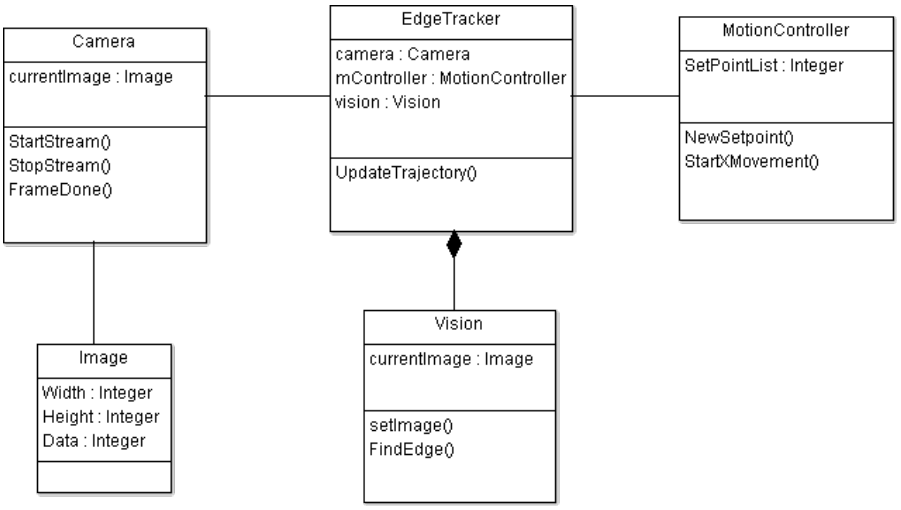


Figure 35: Class diagram.

The sequence diagram in Figure 36 illustrates the interfacing that takes place when tracking an edge. First the host starts the camera streaming function. Then two processes take place in parallel:

- x-*movement:* the host asks the motion controller to start moving in the x-direction at a constant velocity.
- *Correcting movement:* the camera streams images to the host. The host calculates edges and generates a trajectory from them. This trajectory is fed to the motion controller which controls the requested correcting motion in y-direction as not to cross the detected edge.

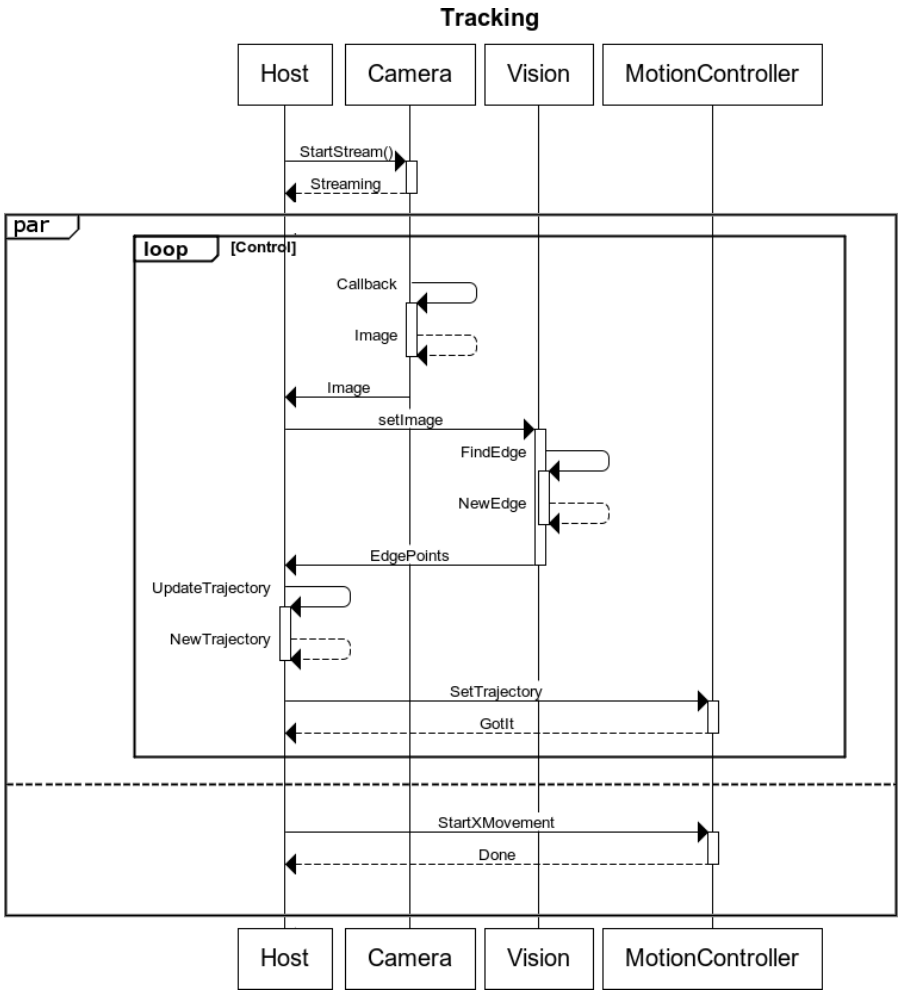


Figure 36: Sequence diagram.

When the motion controller notifies the host that x-movement is finished the host can finish the control loop at which point tracking a wafer edge is finished.

6.2 REAL-TIME ASPECTS

To successfully track the wafer's edge, some hard real-time constraints must be met. First of all, the vision process must finish before the next frame arrives from the camera. If this does not happen, delays will add up and the motion controller will react too late on inputs.

Secondly, the motion controller must always have a new setpoint ready. The host must always have supplied a new setpoint to the motion controller before the system has arrived at the previous setpoint.

The host system runs Linux with real-time patch applied. To ensure that the host process will always run when in ready state the priority is set to the highest level.

Page faults incur unwanted delays in the process. When a page is not present in memory IO activity takes place which cause additional

latency. Therefore, prior to performing the edge tracking all memory the process will use is pre-allocated. After allocation the real-time system call *mlockall()* is executed. This call locks all pages mapped to the address space of the calling process into main memory. Shared libraries the process utilizes are also entirely locked into memory.

Another cause of latency is hard-disk usage. Reading from or writing to disk can cause latencies of $20ms$ or more. As a new setpoint is generated every $28ms$ this latency is unacceptable. Any output thats needs to be written to disk is therefore stored in main memory and only written when the tracking task has finished.

6.3 MOTION BLUR

Motion blur is the smearing of moving objects in a still image. It results when the image being shot changes during recording either due to rapid movement or motion during long exposure.

This streaking is unwanted, the vision process requires a proper step in intensity for edges to be detected. A significant blur attenuates this intensity step thus hurting edge detection.

The amount of blur present in an image is then a function of the velocity (v) at which the substrate carrier under the camera is moving and the exposure time of the camera (t_e):

$$b = v * t_e \tag{41}$$

To verify this, the calibration plate in Figure 37 is placed on the carrier.

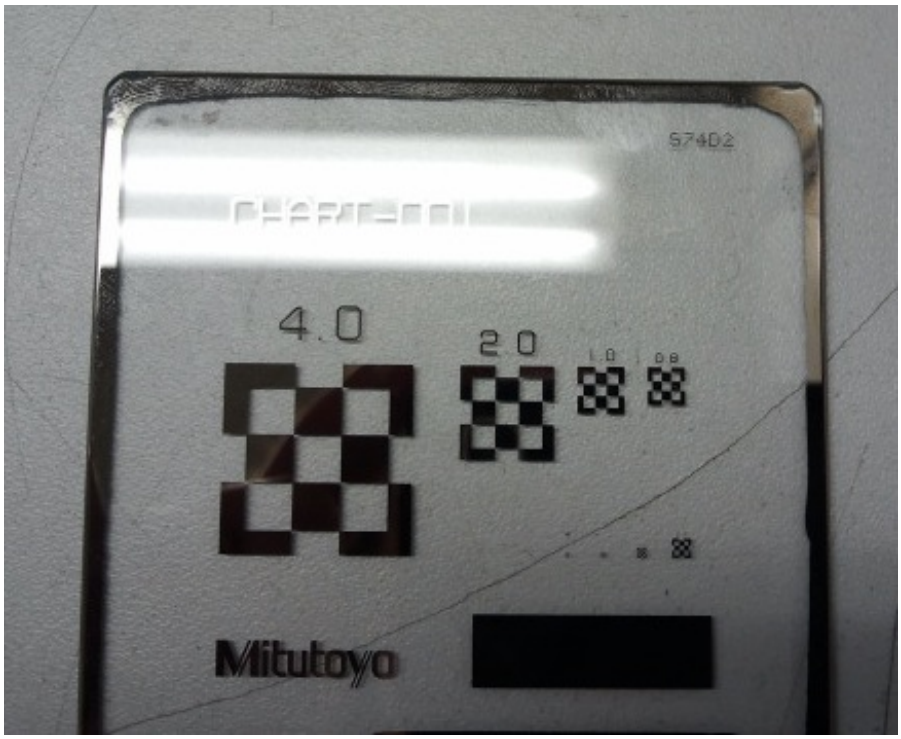


Figure 37: Calibration plate.

First an image is taken when the system is stationary. Then the system moves with a constant velocity and another image is taken. Figure 38 shows both images.

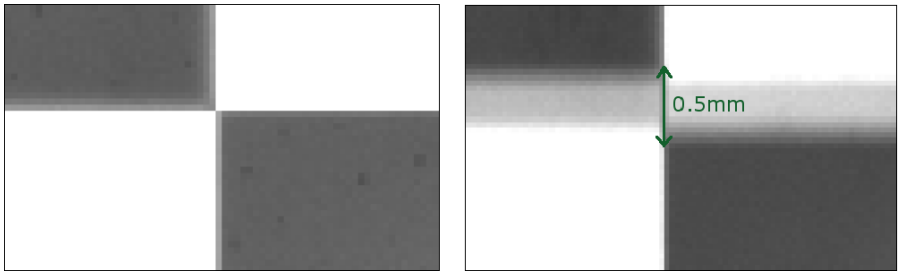


Figure 38: Motion blur.

With an exposure time of 1ms and a velocity of 50mm/s the blur should then be 0.05mm. Knowing the pixel size and counting them we find that this is indeed correct.

By reducing the exposure time the effect of motion blur can be reduced. When doing so, care must be taken that there is still enough light to exposure the area being captured. If not, the cameras CCD gain can be increased to strengthen present light. This will add noise to the image which can also disrupt the vision process. Figure 39 gives an example with very low exposure (25µs) and high CCD gain (18dB).

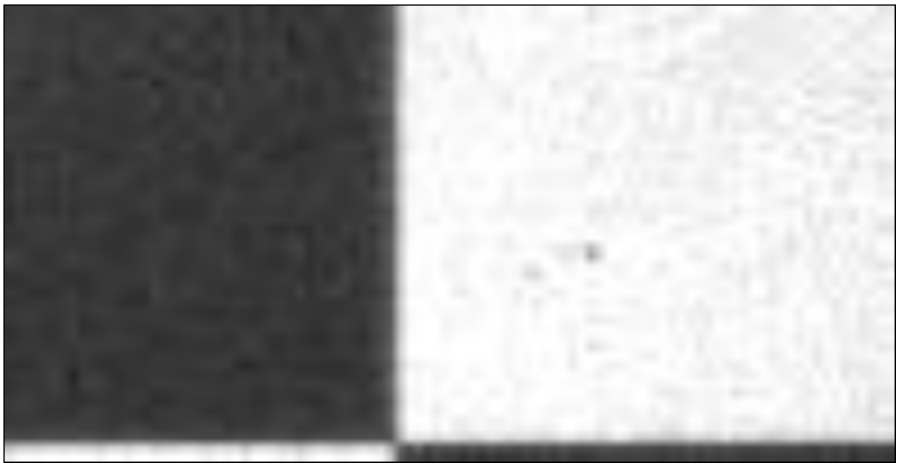


Figure 39: CCD gain.

This is an exaggerated case but it does clearly show this image has a lot of noise present.

Part III

RESULTS

Having discussed relevant algorithms for the visual tracking application, it is time to evaluate the developed system and look at the results.

An experiment performed with the system supplies data to be analyzed. After discussing results a conclusion is given as well as future work.

analyze results in terms of accuracy and speed, what kinds of errors and where do they originate

To find out how the developed algorithms perform an experiment is performed. A 156mm long piece of paper with a small notch cut out is placed on the substrate carrier with aluminum tape underneath it (see Figure 40). Paper is used as it allows for cutting out small pieces for verification of the system whereas removing a small piece of a solar cell wafer is more difficult. Both paper and wafers look the same when imaged with the camera.

Figure 40: Detected edge.

The system is then moved in a starting position such that the camera is placed on the right hand side of the paper, aligned with the bottom-right corner. Tracking is performed in negative x-direction because in our system the printhead will be placed on the right-hand side of the camera.

The vision process measures the edge in every frame it shoots and results in a single position per frame (the highest intrusion in each frame). When no paper is in view and no edges are detected, the y-value is set to 0. Figure 41 shows the output of the vision algorithm when the paper edge is tracked. As the process is started on the right hand side of the paper the initial value is zero. When the paper comes into view edges are detected and their exact locations stored. When movement has passed the left-hand corner there are no longer any edges and the y-values becomes zero again.

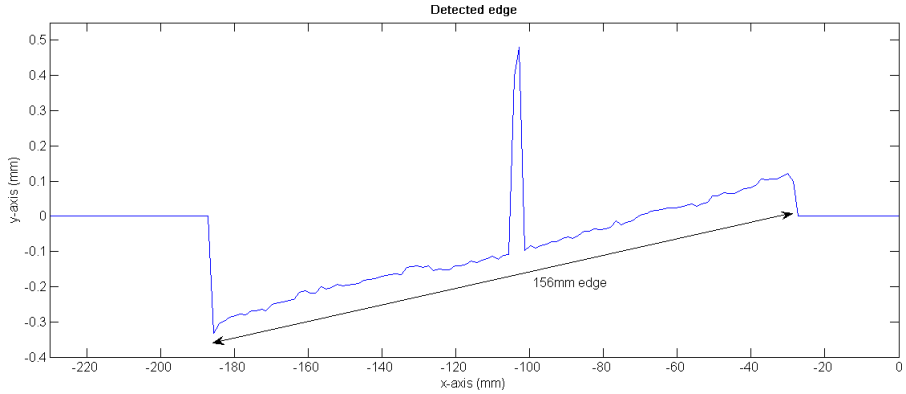


Figure 41: Detected edge.

Note that in this figure the scale of the x-axis is far larger than the y-axis scale. The notch might seem quite large but is actually quite small.

Now we take a look at the output of the thickening filter. This is plotted alongside the detected edge in Figure 42. As required, the thickened output is 0.1mm from the detected edge. On the very right there is a section where the output is zero. Because we don't know in advance where the next detected edge will be located we make sure the y-axis is pulled to position zero before correcting motion takes place.

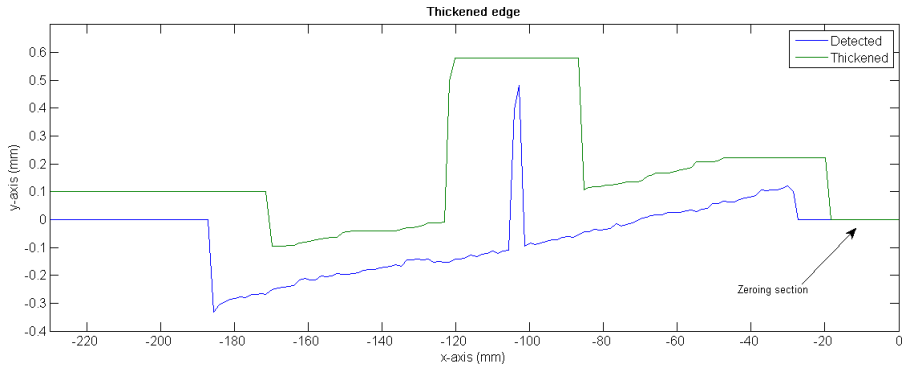


Figure 42: Thickened edge.

Having thickened the input, the trajectory for the correcting motion can be generated. Every time the vision process supplies a new point a new trajectory is generated on the fly resulting in one new setpoint for the motion controller. The output shown in Figure 43 are all the points that were fed to the motion controller.

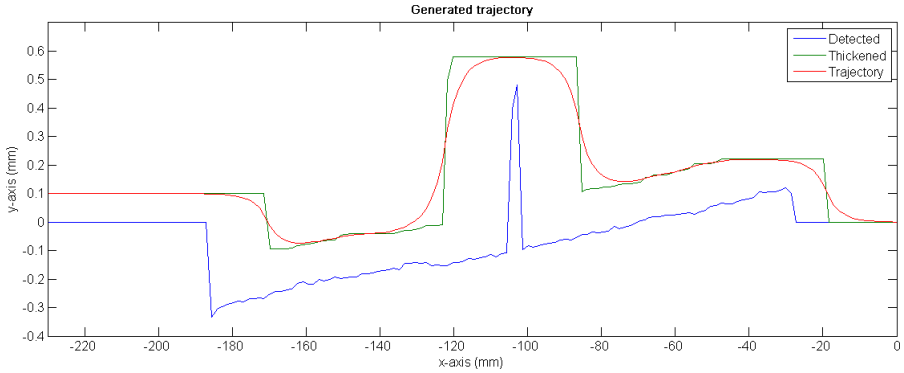


Figure 43: Generated trajectory.

As discussed in previous chapters, due to acceleration constraints the correcting motion can not be steep. The trajectory shown looks very steep but is actually quite flat because x- and y-position are scaled differently in the figure. Using equal scales, we would barely see a bump in the straight edge.

The motion controller can log performed motion. After a side of the wafer is tracked and both vision and correcting motion have finished this log can be retrieved. A plot of recorded motion is shown alongside the trajectory and tracked edge in Figure 44.

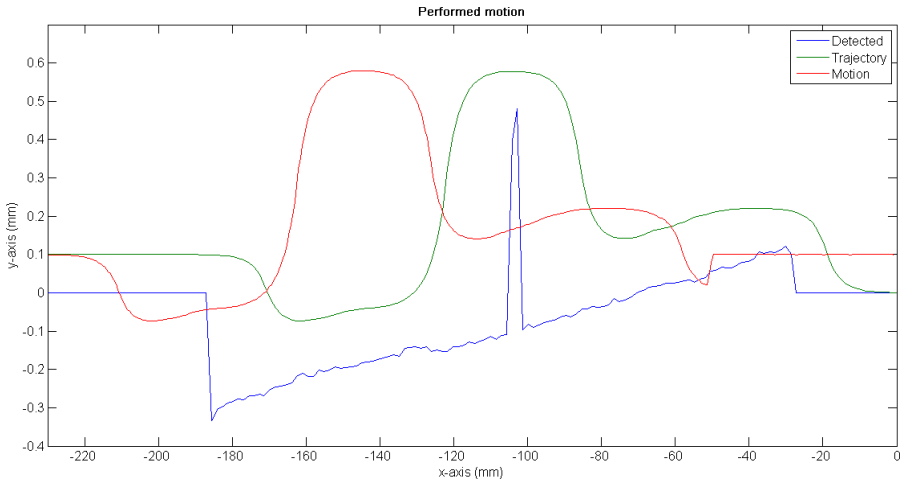


Figure 44: Performed motion.

The performed motion in this figure is shifted by 40mm. There is a lag between when the camera images a section and when the printhead will arrive at the same section which is why this lag is included in the process.

By shifting back the performed motion by 40mm and plotting it alongside the trajectory (Figure 45) we observe the controller follows the provided trajectory very well.

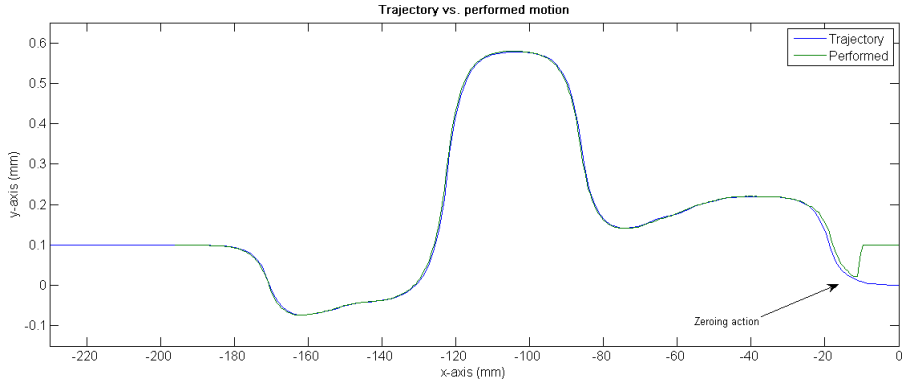


Figure 45: Shifted motion.

Again we see here the zeroing action at the start of the process. At the time the process started the y-axis position was at 0.1mm. After it is pulled back to zero, correcting motion starts.

We can conclude that the algorithms developed perform well on realistic input data. The trajectory generated is followed closely and no edge segment is intersected during the correcting motion.

8 CONCLUSION AND FUTURE WORK

8.1 CONCLUSION

In this thesis a method for vision-based edge tracking was developed. Vision and motion algorithms were developed, implemented and tested using the LP50 platform available at Roth & Rau.

From the results we can conclude that these algorithms perform according to the analysis given before implementation. A hard requirement is not crossing any part of the edge, we have developed algorithms and analyzed them in order to prove that the wafer's edge is indeed never crossed.

Despite the successful creation and testing of the system and fulfilling the requirement that the edge is never intersected, not all goals set out at the start of this project were met.

The field of view of the camera in the current system is too small to be able to see the full 1mm variation. Therefore variations of this size are also not trackable in the current setup.

The required processing time given a plant with a throughput of 2400 wafers per hour is 1.5s. Given that our maximum tracking velocity of a wafer's edge is 52.8mm/s and wafers are 156x156mm we have a processing time of 11.8s, not taking into account any overhead.

Although these goals are not met, the analysis provided in this thesis is used to obtain system parameters that will meet these goals. This means that the developed motion and vision algorithms are still valid, but hardware changes are a requirement to meet the final goals.

8.2 FUTURE WORK

In the first chapter of this thesis a number of functional requirements were stated. After analysis it was found that these requirements could not be met given the current system parameters.

A method was developed to find the most optimal solution given the system parameters and from the analysis and experiment developed it is possible to find modifications that meet the requirements.

The architecture of the LP50 system allows us to only track one side of the wafer. A possible solution was provided in Section 3.2 wherein two cameras are placed on both sides of the printhead. This solution looks promising and is a good candidate for future research.

As the field of view poses two problems, a new camera or modification of the camera is a must. In Section 3.3 we've shown that due to the limited field of view the full 1mm can not be detected and

tracked. Increasing the field of view would allow this. The second problem lies in the balance between framerate and field of view. As no edge points can be missed during scanning of the wafer's edge, the maximum velocity is limited. By either increasing the framerate or the field of view this velocity can be increased. The analysis in Section 3.3 can be used to find a camera that satisfies the requirements.

Acceleration is a major constraint. It is a must to start a correcting motion well in advance of a detected default in the wafer's edge, resulting in loss of area. The analysis in Section 3.4 can be used to find how much area is wasted given a maximum acceleration. Because infinite acceleration does not exist it will always be a constraint.

Finally, the analysis on filter guarantees in Section 5.3 should be extended to a general proof independent of parameter α.

BIBLIOGRAPHY

[1] G. Carr, "Sunny uplands - Alternative energy will no longer be alternative", The Economist, November 2012, *http://www.economist.com/news/21566414-alternative-energy-will-no-longer-be-alternative-sunny-uplands*

[2] R. Naam, "Smaller, cheaper, faster: Does Moore's law apply to solar cells?", Scientific American, March 2011, *http://blogs.scientificamerican.com/guest-blog/2011/03/16/smaller-cheaper-faster-does-moores-law-apply-to-solar-cells/*

[3] A. Vaughan, "Cost of solar energy will match fossil fuels by 2013, claims Solarcentury" The Guardian, May 2009, *http://www.guardian.co.uk/environment/2009/may/12/solar-energy-price-fall*

[4] J. Pearce, "Photovoltaics - a path to sustainable futures", Pennsylvania State University, September 2002.

[5] A. Hauser, G. Hahn, M. Spiegel, H. Feist, O. Breitenstein, J.P. Rakotoniaina, P. Fath, E. Bucher, "Comparison of different techniques for edge isolation", Max Planck Institute of Microstructure Physics, Germany, 2001

[6] Z. Ye, Y. He, R. Pieters, B. Mesman, H. Corporaal, P. Jonker *Bottlenecks and Tradeoffs in High Frame Rate Visual Servoing: A Case Study* Eindhoven University of Technology, Eindhoven, The Netherlands

[7] R. Reginelli, "Memory Locking", MontaVista Software Inc.

[8] M. Heath, S. Sarkar, T. Sanocki, K. Bowyer *Comparison of Edge Detectors: A Methodology and Initial Study* University of South Florida, Tampa

[9] J. Canny *A Computational Approach to Edge Detection* IEEE Trans. Pattern Anal. Mach. Intell.

[10] T. Pavlidis, "Algorithms for Shape Analysis of Contours and Waveforms" IEEE Trans. Pattern Analysis and Machine Intelligence, 1980

[11] S. Hutchinson, D. Hager, P. Corke *A Tutorial on Visual Servo Control* IEEE Transactions on Robotics and Automation, Vol. 12, No 5, October 1996

[12] M. Spong, S. Hutchinson, M. Vidyasagar *Robot Modeling and Control*, chapter 5: *Path and Trajectory Planning*

[13] O. Gerelli, C.G.L. Bianco *A discrete-time filter for the on-line generation of trajectories with bounded velocity, acceleration, and jerk*, University of Parma, Italy

[14] Wikipedia article on Feature Detection (Vision) *http://en.wikipedia.org/wiki/Feature_detection_(computer_vision)*

[15] Wikipedia article on Laplace distribution *http://en.wikipedia.org/wiki/Laplace_distribution*