Image Registration for Volume Measurement in 3D Range Data

Fernando Indurain Gaspar
Matthew Thurley
Lulea

May 2011

Abstract

At this report we will explain how we have designed an application based on morphological operators and the Harris method for corner detection to measure the volume of an element in a 3D image.

This application for mining industry will be able to measure the volume of rocks carried in the bucket of an excavator using a 3D image of it. This application will identify the vehicle using morphological operators and delimitate the bucket. Then it will find its corners using the Harris method combined with some extras we have designed in order to improve its accuracy and finally it will use a geometric transform to match this image with one image of an empty bucket to obtain just the rocks and measure its volume.

After testing the system in a test bench of 30 different images including some special cases we have obtained a fast time of execution and a final error always below one cubic meter.

Index

1. Introduction & Problem Background

The LKAB iron mine in Kiruna is the deepest and most modern iron mine in Europe and they are always trying to use technology to improve their productivity. In this report we will expose a method to help them reach this goal.

In the iron mining industry not all the ore extracted by an excavator can be used: the extracted ore contains waste rocks and desirable minerals. Excavators dig out the rock and load the material into the transport network. If the load is majority waste rock it will cause a reduction in productivity, both from carrying the waste rock and from further crushing and processing.

When mining iron ore, the difference in density between the iron ore (high density and therefore heavy) and the waste rock (low density) can be quite large. This indicator can be useful for the mining process: if the density is high it would mean that the ore has a high concentration of iron so we have to send it to process. On the other hand, if the density is low, it would mean that the ore is mainly waste rock so we should not carry it to the plant.

To calculate the density we need the mass and volume of the ore. The mass is already estimated from the hydraulic pressure of the excavator as it carries the load of material but the volume is much more complicated. In order to develop a system to estimate the density a laser scanner was installed in the roof of a mine tunnel at LKAB's mine in Kiruna, Sweden.

The goal of this report will be to explain a robust method to process these 3D images and obtain the volume of ore carried by the LKAB vehicle in "real time".

2. Research Topic

2.1 Data Collection

In order to have a better understanding of this report we will first give a fast overview of the data used.

Let’s start having a look to the Iron Mine. In Figure 1 we can see the evolution of digging along the time. In the right part of the image we can see as white lines the tunnels where the extraction process is being made.

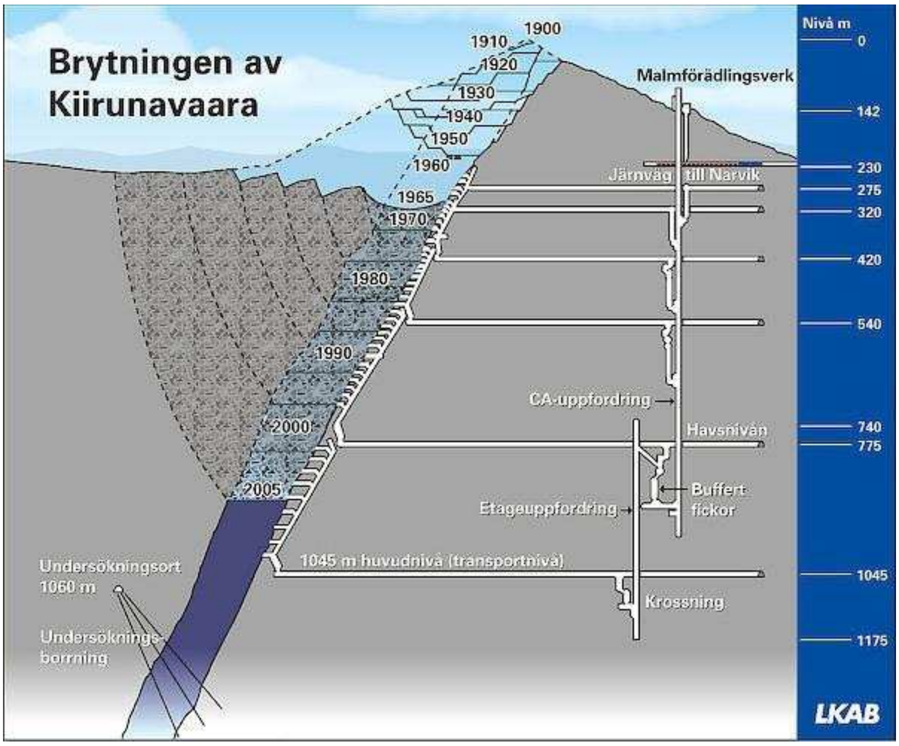


Figure 1: Cut of the Kiruna's mine

In Figure 2, we have a zoom of the extraction tunnels. In blue we can see the ore body where the excavator will extract the ore in the draw points. Then when the bucket is full enough it will drive to the shaft to deposit its charge, passing above the scanner in the measurement point. Finally it will drive back to the draw points. We have to comment that once the bucket is full the arm of the bucket adopt a fixed position so the bucket will be always at the same height when it pass under the scanner and the direction of the excavator will be always the same because it does not turn when it deposits its charge.

It is important too that an excavator will pass above the scanner every two minutes (more or less) so the method implemented must run in less than two minutes (for each image).

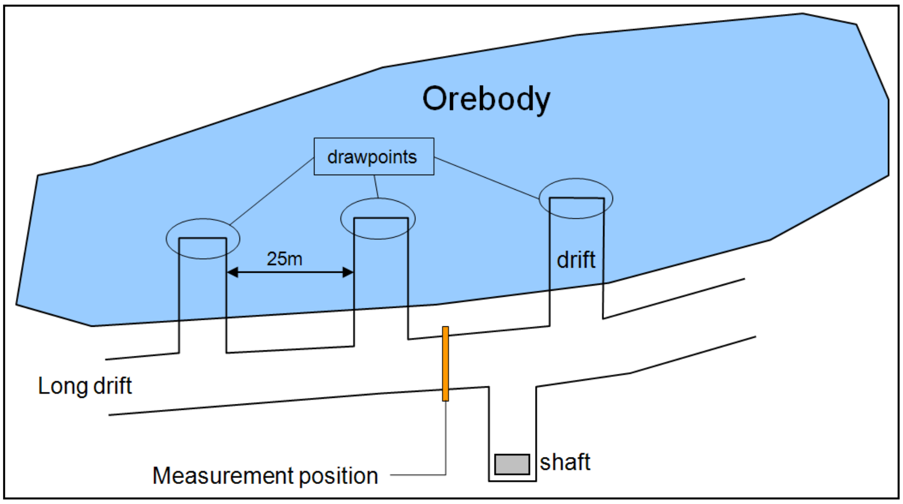


Figure 2: Extraction tunnel

LKAB has given us a big set of images to process. Thanks to previous work of Thurley [1], this image is traduced to a Z matrix where each component is the distance from that point to the scanner (if the coordinate point is too far from the scanner it will be assigned with a NaN entry). At this point we should comment on a few things: the vehicle is not stopped when it is scanned, it is moving under the scanner while the scanner sweeps from left to right so depending on the speed of the vehicle the size of the matrix may vary. If the vehicle is moving faster, it will pass under the scanner in a short time so the matrix will have less number of rows. If the vehicle is moving slower we have the opposite case so the matrix will have more rows.

For a better view, we will use the grey scale in the vehicle images where the black colour is used for the farther points and the white one is used for the closest.

To familiarize our reader with the excavator Toro2500E LHD we have included an appendix with its schematic (appendix - F).

2.2 Choosing a method

In order to solve this problem and after analyse it carefully we consider that the method we will implement will be composed by the next steps:

1. Identify LHD vehicle
2. Identify the bucket
3. Find the bucket's corners
4. Apply the geometric transform with an image of an empty bucket
5. Subtract the transformed image of the full bucket with the image of the empty bucket
6. Multiply the height of difference image with the correspond area of a pixel to obtain the volume of the rocks.

As they seen very defined steps the report will be structured using they and they will be explained in a more extensive way later.

The next chapter (chapter 3) will describe this method deeply with subsections for these six steps. The chapter 4 will describe the results obtained from the application of this method. In chapter 5 we will sum up with the conclusions of the results and some comments about them. Finally in chapter 6 we will establish the future work and research in order to improve the method and make it more reliable and efficient.

3. Method

At this chapter we will describe the method finally implemented and we will explain why the design decisions have been taken. We will make too a small brief of the design process and the troubles found in it but we will focus in the final method and why it is better.

As we said in 2.2 the method will be based in 6 steps:

1. Identify LHD vehicle
2. Identify the bucket
3. Find the bucket's corners
4. Apply the geometric transform with an image of an empty bucket
5. Subtract the transformed image of the full bucket with the image of the empty bucket
6. Multiply the height of difference image with the correspond area of a pixel to obtain the volume of the rocks.

3.1 Identify LHD vehicle

The first step is to identify the vehicle and discard any image that does not represent an excavator Toro2500E LHD. If our system would activate when every vehicle pass above the scanner it may result in error information, it may not be ready when an excavator Toro2500E LHD needs it and in short, it will be traduced in a loss of efficiency and money.

So this step is very important. If we are able to discard other vehicles we will guarantee robustly property in some way. In orther to do this we will analyse the width graph of the vehicle. This graph will show the width of the vehicle along its length.

3.1.1 The width graph

This graph has a really characteristical shape and it should be quite easy to analyse, so we decided to try to use this graph for both purposes: finding the bucket and identify the vehicle.

Before we calculate the graph of vehicle width along its length we will rescale the length of the image so that all the data sets have the same length. This will be important to standardize the unique properties of the graph in identification system. We will explain this point deeper later. We have to consider that this operation will introduce an error in the image's information because some rows or columns will be deleted or added.

The next step is to find the both margins of the vehicle for each row of the image. This will be done using the gradient of the grey-scale image for each row. A big peak should appear for the beginning of the vehicle as a sudden change of height. This will be result in the beginning column and the ending column of the vehicle for each row. To identify these sudden changes we will use a threshold of 300 units and -300 units. Their difference will result in the width of the vehicle.

The method to calculate the width of the vehicle will result in:

1. Rescale the image.
2. For each row of the image, calculate the gradient of this row (see figures 3, 4).
3. Identify the sudden change in the gradient graph as begging and ending column of the vehicle using two thresholds (We can appreciate the sudden peaks in figures 3, 4. Figure 5 shows the final result for all the rows in an image).
4. For each row subtract the begging column to the ending column to obtain the width for that row (see figure 6).

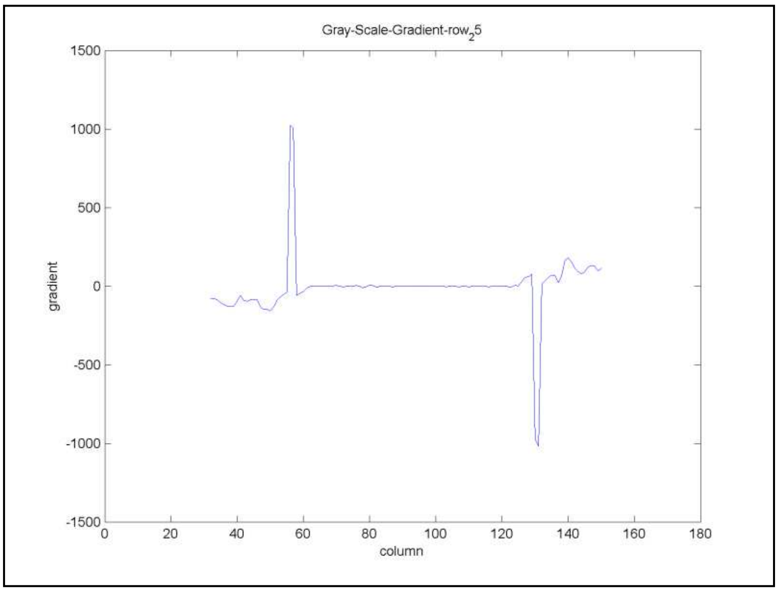


Figure 3: Gradient of grey scale image's row

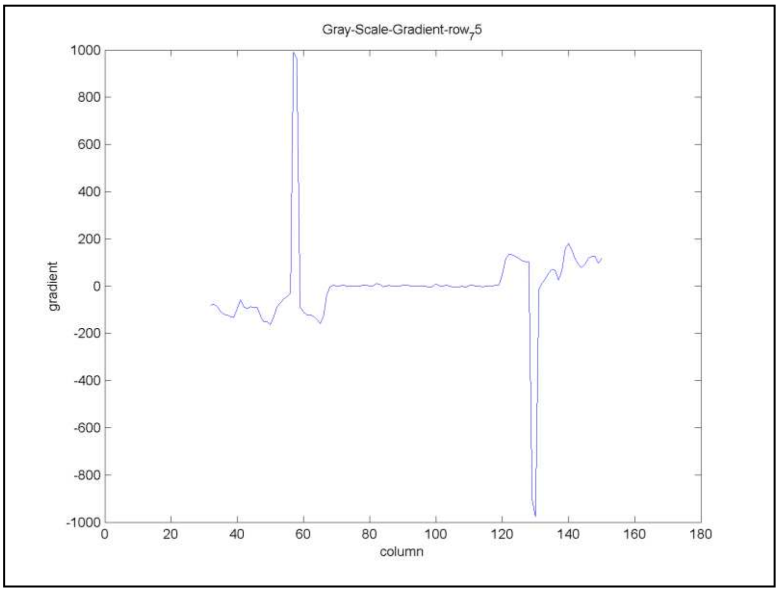


Figure 4: Gradient of grey scale image's row

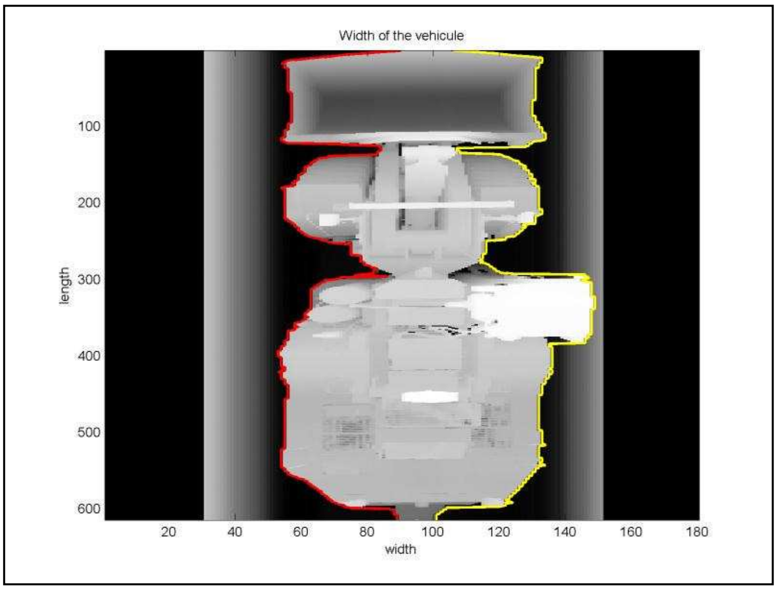


Figure 5 : Border detection: start column (red) and end column (yellow)

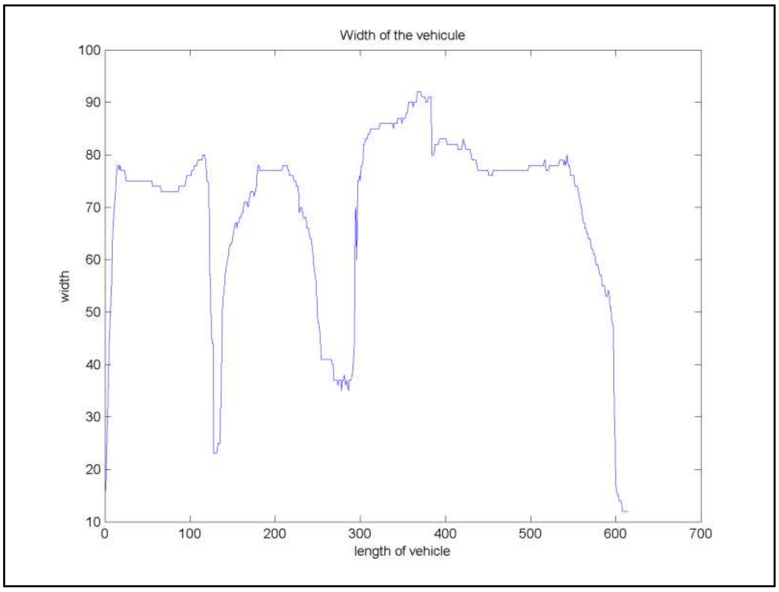


Figure 6: Final width graph

One of the challenges appeared during the testing of this algorithm was that in some images the vehicle is too close to the walls of the mine resulting in an absent of the mention pick (see figures 7, 8). These results in an error of finding one of the margins (see figure 9) that has a terrible impact in all the width graph treatment (see figure 10). Because of all this we decided to discard all the images that have this problem (during the testing we observed this phenomenon in just 1 of 30 images). For an industrial system a modified algorithm would be implemented to account for these cases.

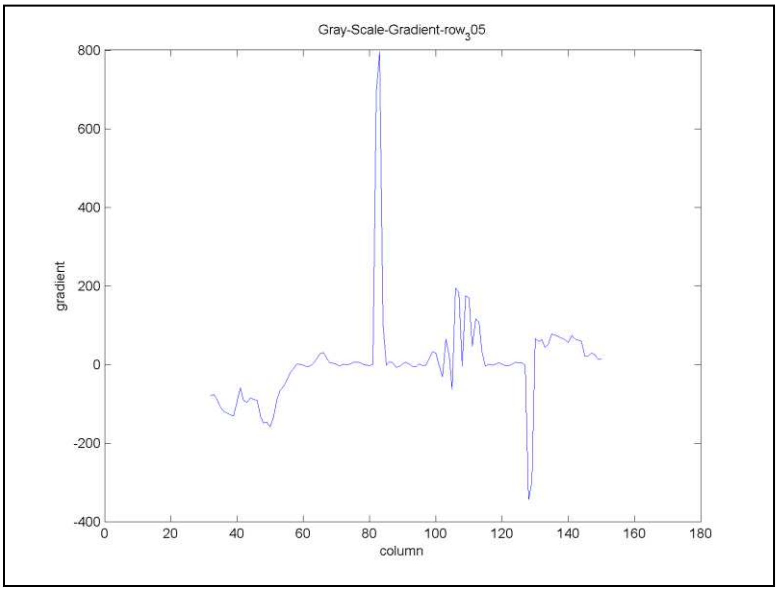


Figure 7: Abnormal row's gradient

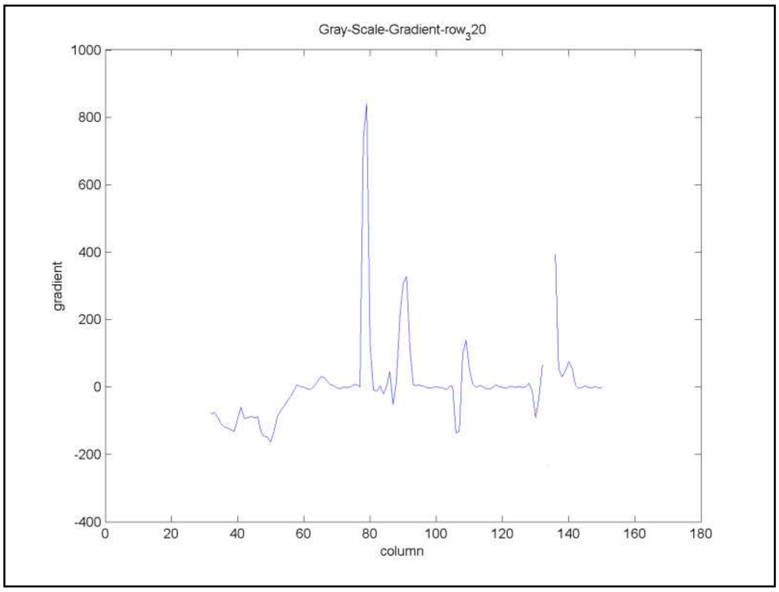


Figure 8: Abnormal row's gradient

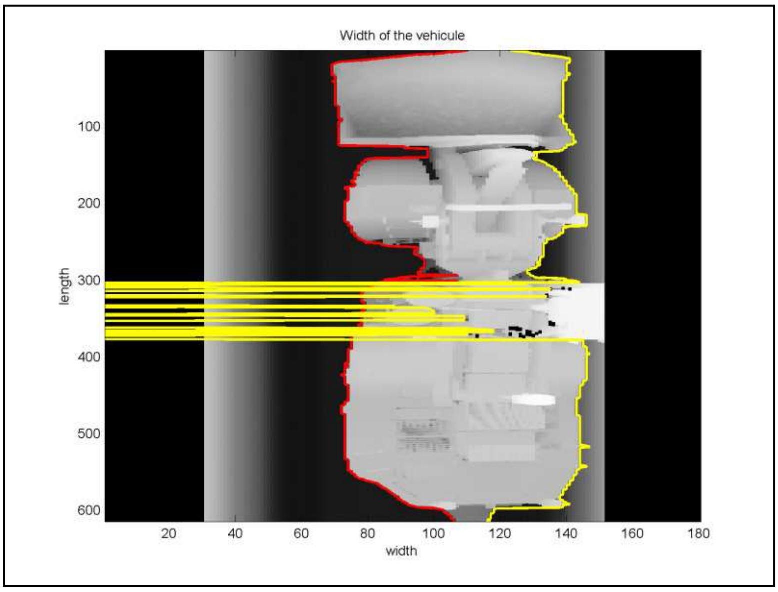


Figure 9: Error in border detection

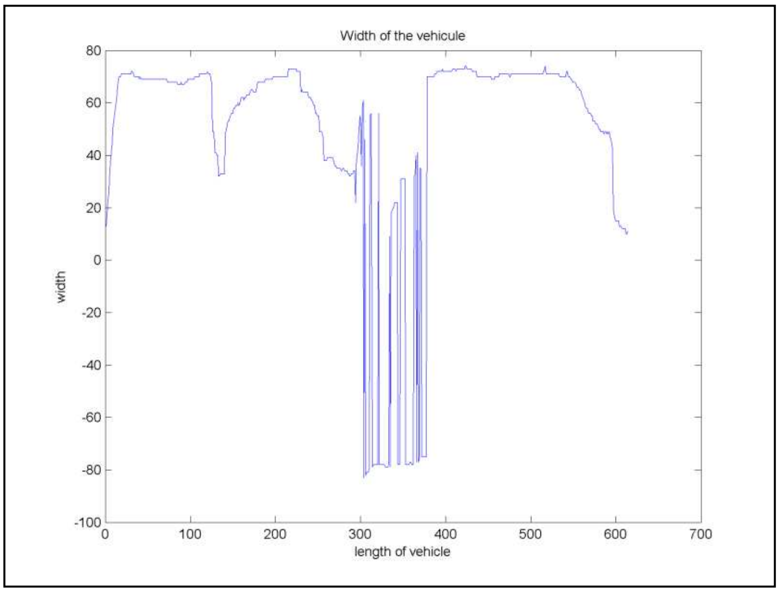


Figure 10: Error in width graph

3.1.2 Choosing a method

In order to identify the vehicle we can focus in different characteristics of the width graph and do it in different ways. Because of that we designed two different algorithms, one just based on the gradient of the width and the morphological operators (R. Dougherty [5], see APPENDIX-A for more information) and other one based on a fast fourier transform of the width and the used of its most representative frequencies.

At this point we will explain both algorithms and comment their results so we will be able to decide which one is better for our final algorithm.

3.1.2.1 Morphological Operators Algorithm

The algorithm consists on the next steps:

1. Apply an opening operator using a disk of radius equal to 30 pixels to the width vector. This results in a function more regular with a very small loss of information in most of the images. See figure 11, red function.
2. Calculate the gradient of the opened width. This will be useful to locate the minimum points. See green function in figure 11.
3. Apply a closing operator using a disk of radius equal to 10 pixels to the gradient's width vector. See pink function in figure 11.
4. We will use the closed gradient to identify the vehicle, looking for the 3 maximum points centred to 3 concrete points of the length: 0 pixels, 140 pixels and 290 pixels. These points correspond approximately, considering the start of the bucket as length '0', to the lengths 0 mm, 3170 mm and 6280 mm. We will give a tolerance of 5% (700.55 mm) to these positions.

In figure 12 we can see an expanded closed gradient function with its max peaks represented with red vertical lines. If vertical lines fall in these intervals of the known values plus or less 5% of the max length of the vehicle it will be considered identified. Otherwise it will be discarded.

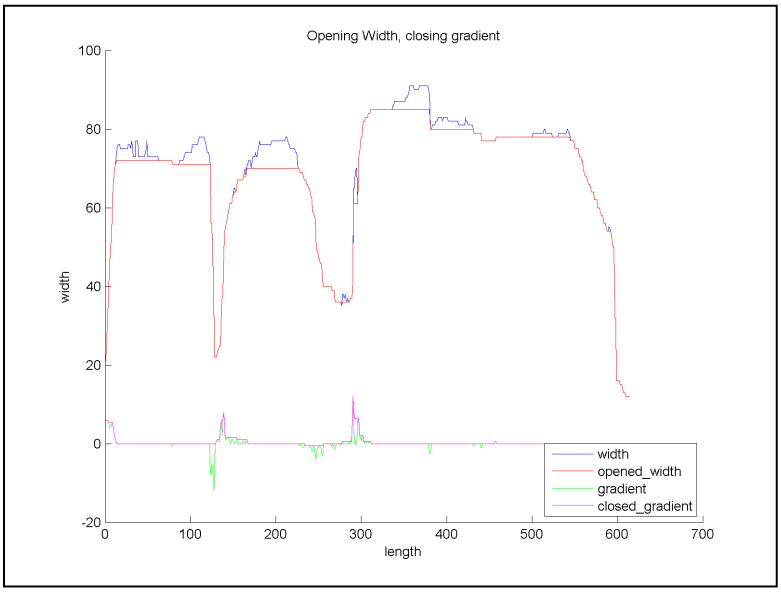


Figure 11: Width analysis

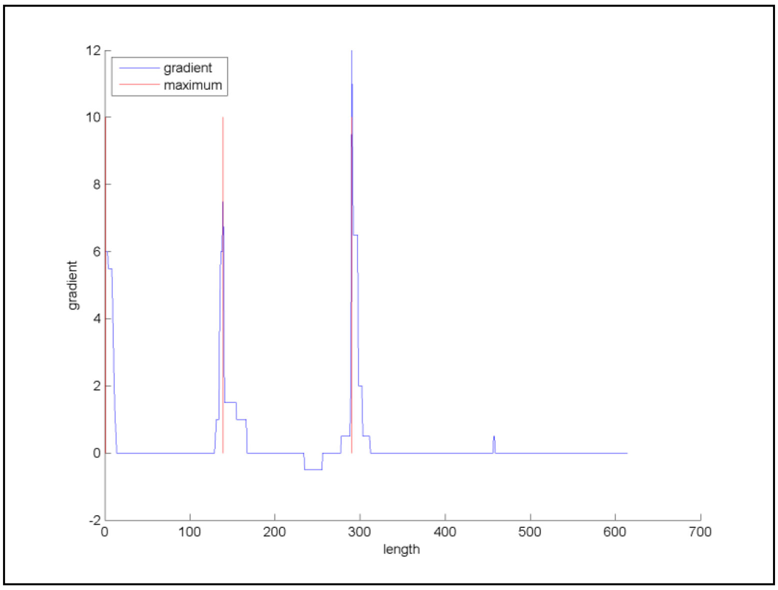


Figure 12: Gradient max detection

3.1.2.1.2 Results

After executing this method on our test bench of 30 images with different strange cases (see figures 13 and 14) using Matlab we obtained that 29 of 30 images (96.7%) were successfully identified. Only one image resulted in a false negative error (see figures 15 and 16) as the last maximum peak was out of range.

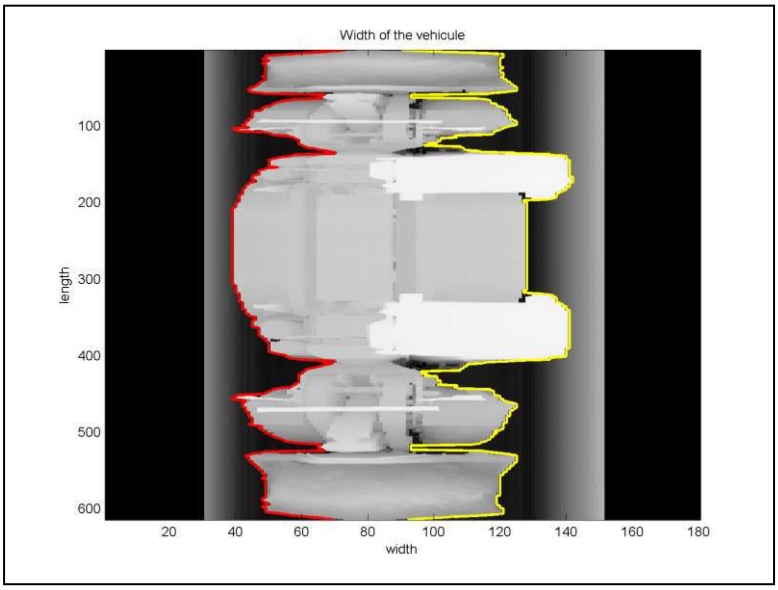


Figure 13: Toro moving straight and back again

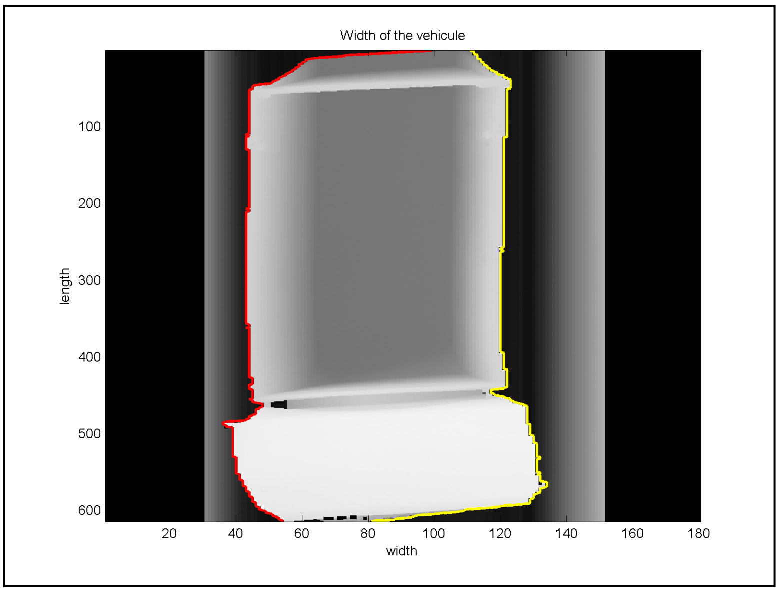


Figure 14: Another kind of vehicle

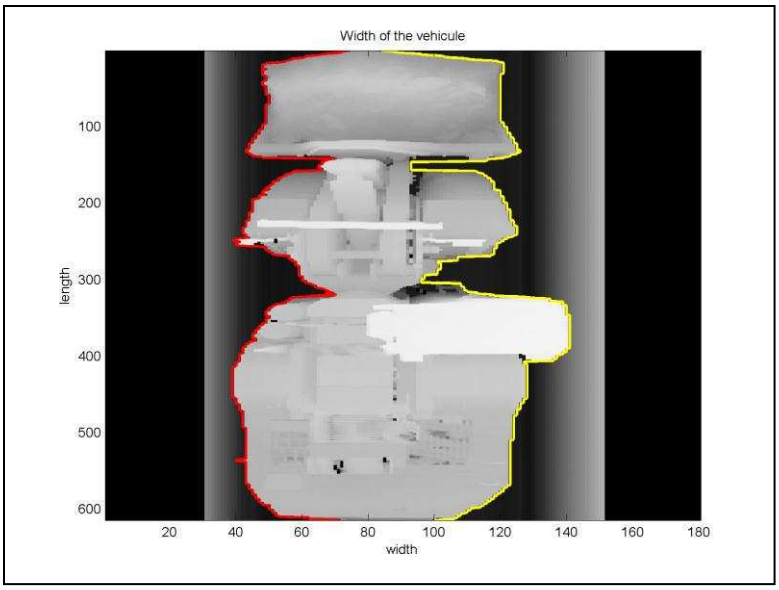


Figure 15: Border detection of false negative error image

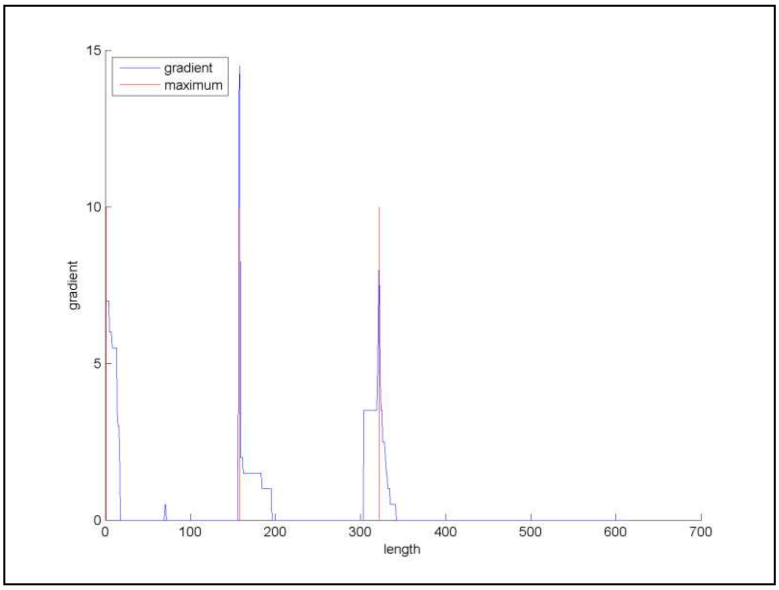


Figure 16: Gradient analysis of the false negative error image

During the development of this system we basically have to face two challenges:

1. The different size of the images.
2. The size of the structuring elements of the morphological operators.

The first one was a problem that had a big impact in the identification system. The images are implemented as matrixes with a variable number of rows depending on the speed of the vehicle while it was scanned. If the vehicle was slow it would result in bigger number of rows (see figure 17, y axis). If the vehicle was fast it results in a smaller image (see figure 18, y axis). Because of that, it was really difficult to fix the identification system, because we did not know where to look for the known values. To solve this we used a Matlab toolbox function, “resize”, that allow us to fix the image to a known value of rows and columns.

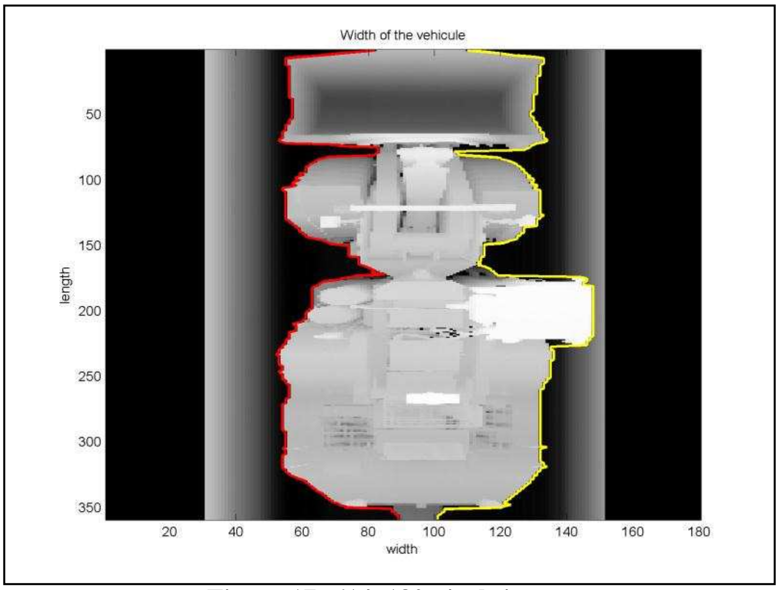


Figure 17 :614x180 pixels image

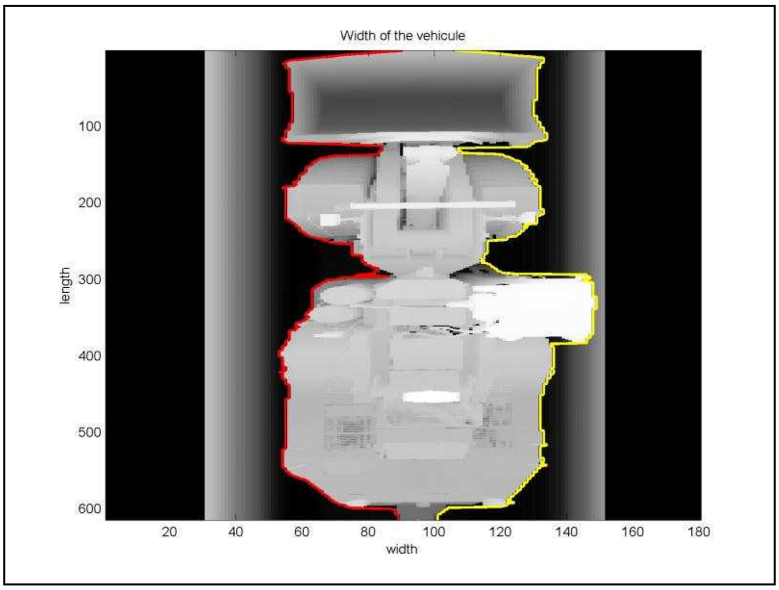


Figure 18: 359x180 pixels image

This solution is not perfect. Every time we make a resize of an image some information is lost. If we add rows we have to "create" information so the final result will be an approximation. If we remove rows some information it is obvious that we are losing information. Because of that we tried to resize the images to a big value to avoid this loose.

We have to say too that we accept the possibility that the vehicle speed up (or reduced its speed) during the scanning so the proportions of the vehicle will not be kept. At this point we decided to discard these images.

The other point was to choose a proper size for the structure elements in the morphological operations. We have had to make some test till we found proper values that simplify our job without deleting important information. The best way to choose the size should be a function related to the size of the vehicle. In this way it will be more flexible and it will be able to recognize other kinds of vehicles in the future.

Finally this is a small step very related with the bucket finding process because we will use the width graph too to reach that goal. Because of that we will analyse the execution time at the end of that step and we will choose between both different techniques (the morphological operator one and that one based on fast fourier transform) considering the results together of both steps.

3.1.2.2 Fast Fourier Transform Algorithm

In this section we will explain how we have used the FFT (fast fourier transform) to obtain an approximation of the width function and how we have used the most significant frequencies of the approximation to identify the vehicle. We will use the same image to show results as the one used for figures 11 and 12 so the reader will be able to compare them. This image will be used along all chapters' method to compare different algorithms.

The algorithm consists on the next steps:

1. Expand the length of the width graph to the next pow of two (the fft algorithm implemented in Matlab is more efficient when it works along a pow of two elements).
2. Calculate the fft of the width (frequencies and coefficients) along the new length. See figure 19.
3. Use the 20 frequencies with the biggest coefficients to rebuild an approximation of the width function. See figure 20.
4. We identify the vehicle comparing the bigger and more significant frequencies with known values. These values will be: 70,35,10,10,5,0,10,5,5 and their symmetric ones. We will accept a tolerance of 3.5 (5% of 70).

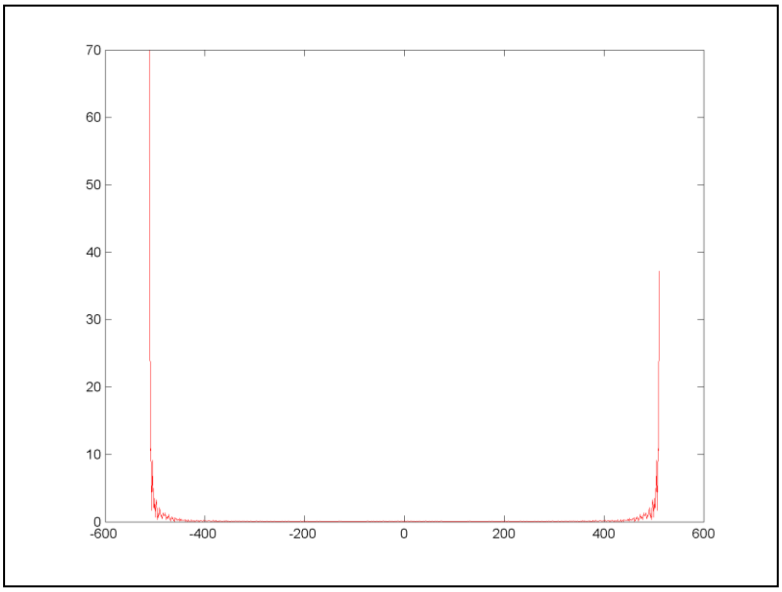


Figure 19: Frequency spectrum

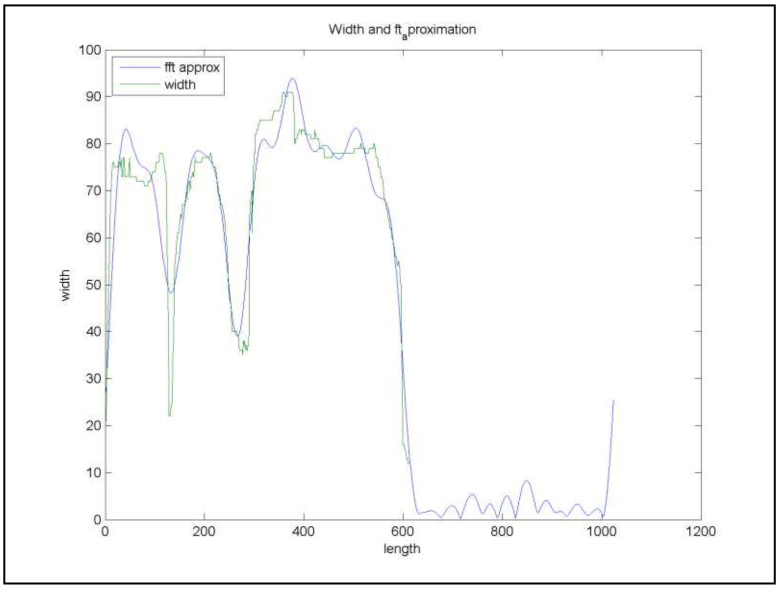


Figure 20: Width approximation

3.1.2.2.2 Results

After executing this method on our test bench of 30 images with different strange cases (see figures 13 and 14) using Matlab we obtained that 30 of 30 images (100%) were successfully identified.

During the development of this method we found two new challenges:

1. Number of points (frequencies) taken to have a good approximation.
2. The frequency spectrum should be symmetric.

We have to start saying that the result of applying the fft procedure to a function is a decomposition of the function in a sum of sines at different frequencies. Matlab will return these results as a matrix of two columns with the frequencies of the sines and the coefficients that multiply these sines.

Now we had to select the frequencies with higher coefficients (more significant) to build a good approximation of the width function making an inverse fast fourier transform (ifft) of the selected frequencies. In the end we choose 20 because with a small number of frequencies some irregularities appeared in the approximation that disturbs the whole method. See figure 21.

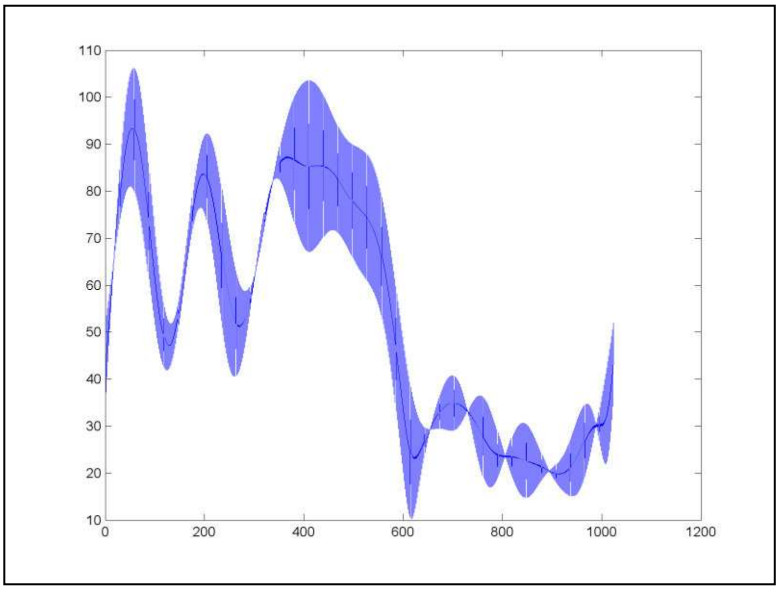


Figure 21: Width reconstruction using less than 20 frequencies

The other point is that the frequency spectrum should be symmetric and as we can see in figure 19 it is not. That was because the frequency vector range is from '-N' to 'N-1' (-N..N-1), not (-N..N) because we choose an odd number of frequencies.

We tried to expand the domain of the fft and the problem was reduced but not avoided and the execution time increased significantly. See figure 22 and compare with figure 19, one goes from '-600' to '600' and the other one goes from '-1500' to '1500'.

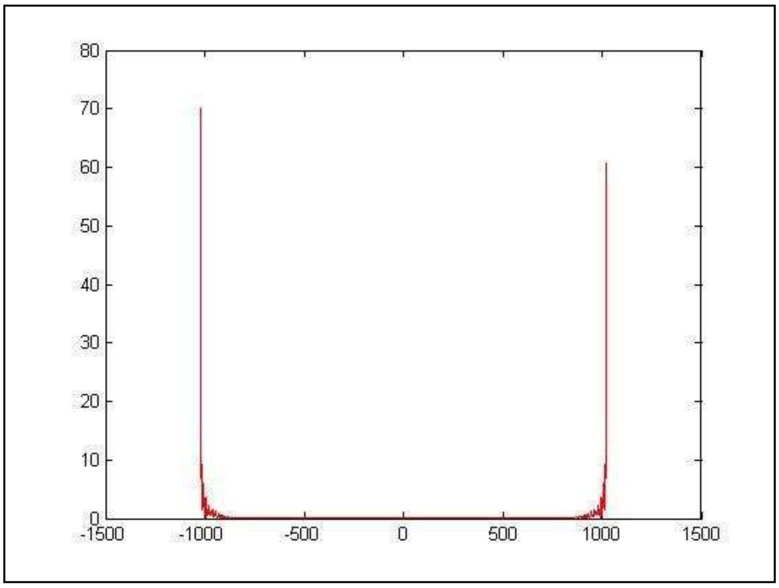


Figure 22: Frequency spectrum of an expanded domain

So to identify if it was a real problem we made a test to check if there was a loss of information:

1. As we said a component of the frequency spectrum does not seem to appear.
2. This should be traduced in a deformation of the reconstructed width function using all the frequencies because we are using less.
3. If we repeat the second step several times the deformation should be bigger in each iteration so we will repeat a great number of times (exactly the length of the vehicle times) the reconstruction of the width based in the previous approximation.

   width_approximation = ifft (fft (width_approximation))

After applying this method no changes were visible in the final width approximation (it was exactly the same as the original one) so there was no loss of information.

3.2 Filter the Bucket

As we said before this step is very related with the previous one: the width graph will be the key again in order to determine where the bucket ends along the length of the vehicle. Trying to continue the way explained in 3.1 we will develop both algorithms: the morphological operator’s and the based on fft one.

At this point after the comments and results of each algorithm it will be time to decide between them, according to their accuracy. We will consider too the time of execution in a determined computer. It will be a reference to compare both methods.

3.2.1 Morphological Operators Algorithm

We recommend to have a look to the 3.1.2.1 chapter because will continue adding steps to that algorithm in order to find the bucket. We recommend too having a look at the morphological operator’s theory.

To find the bucket we will add the next step to the algorithm:

- Apply an opening operator using a disk of radius equal to 10 pixels to the gradient’s width vector. With this step we will remark the first big minimum that will point out to the end of the bucket. See figures 23, 24.

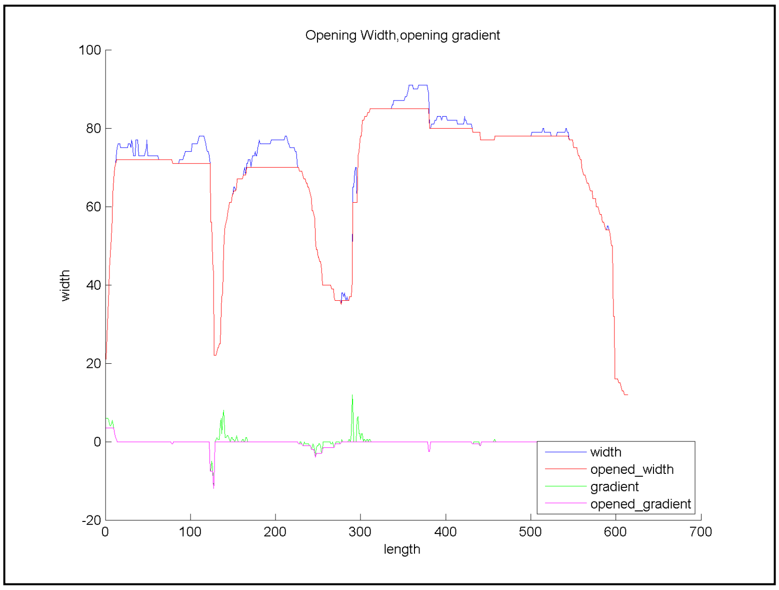


Figure 23: Width analysis for bucket filter

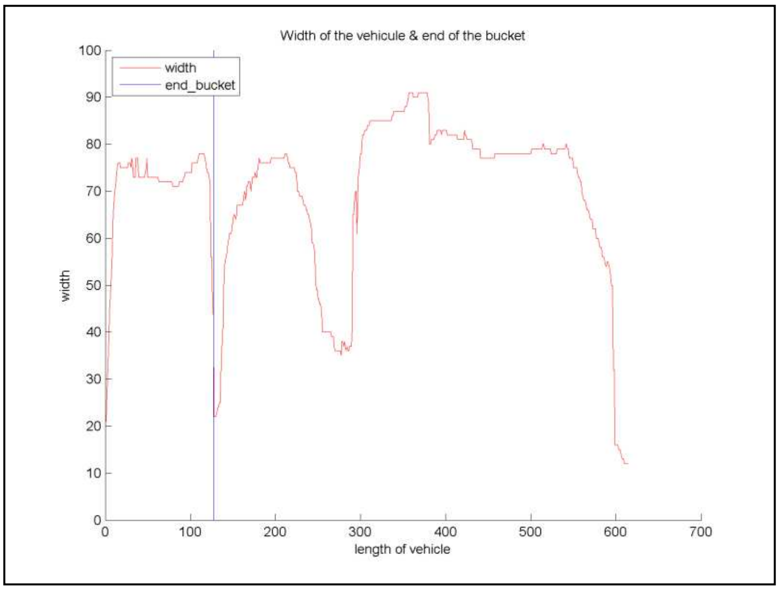


Figure 24: Width graph & bucket delimitation

3.2.1.2 Results

After executing the full method on 30 images of the test bench using Matlab 2010 running in an AMD Turion II X2 M500 (2,2 GHz) we obtained the next results:

- As we said before 29 of 30 images were successfully identified.
- 27 of 27 images well identified (100%) have their bucket well found.
- Time of execution for the 30 images: 513.653 seconds (17.122 seconds per image)

We have to say that we did not have additional complications to implement to bucket filter system as it was developed at the same time as the identification system.

Talking about the results we have reach an almost perfect bucket filter and its time of execution seems to be really good. We will have like two minutes per image to make the full analysis so although the program would run in a worst computer the time of execution should not raise too much.

3.2.2 Fast Fourier Transform Algorithm

We recommend having a look at the 3.1.2.2 chapter because we will continue adding steps to the algorithm explained in that section and we will use some results generated to continue the development.

In order to find the end of bucket we will add the next step:

- We will look for the first minimum of the width approximation. As the approximation should be composed of three big waves the end of the bucket should correspond to the minimum point where the first and the second waves join themselves. See figure 25.

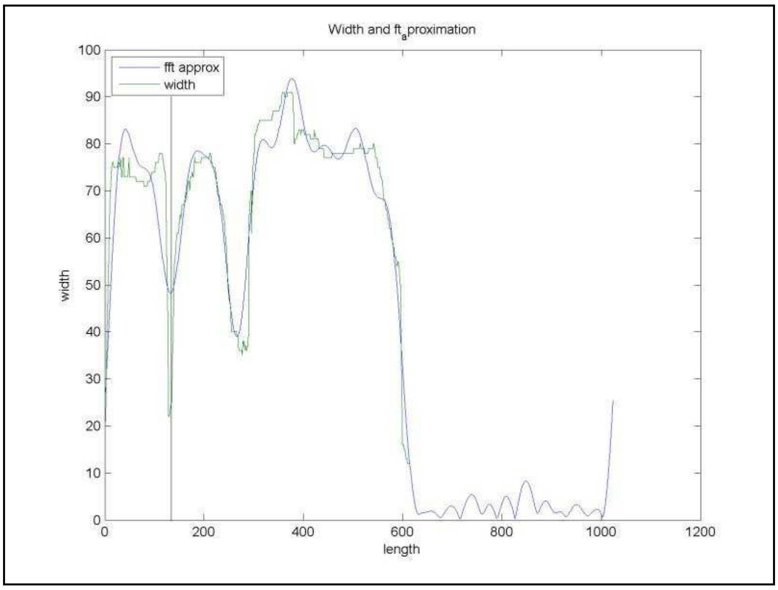


Figure 25: Width graph & bucket delimitation

3.2.2.2 Results

After executing the full method on 30 images of the test bench using Matlab 2010 running in an AMD Turion II X2 M500 (2,2 GHz) we obtained the next results:

- As we said before, 30 of 30 images (100%) were successfully identified.
- 8 of 28 images (28.6%) had mistakes or a big deviation finding the bucket. See figures 26 and 27.
- Time of execution for the 30 images: 671.133 seconds (22.371 seconds per image).

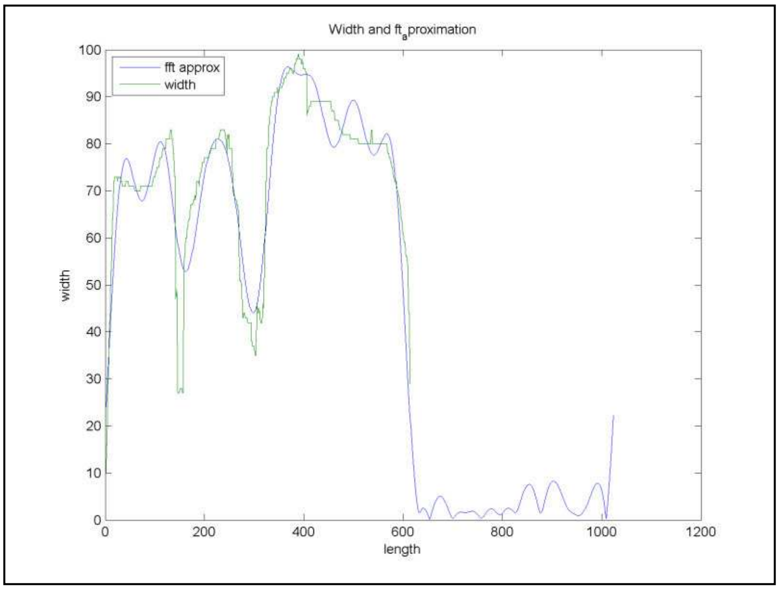


Figure 26: Width approx. of a bucket detection error

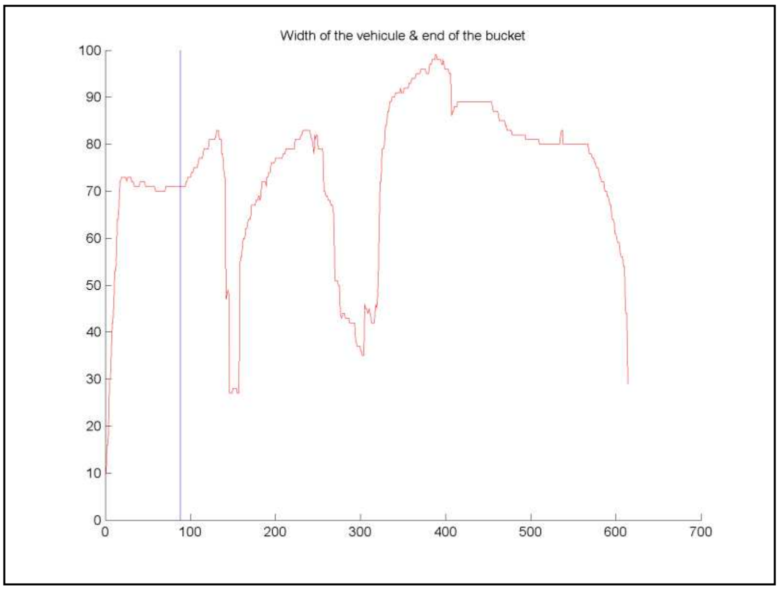


Figure 27: Bucket detection error

We notice that the error rate detecting the bucket is too high for an industrial system. Maybe the accuracy could be improved using some kind of threshold but we wanted to avoid using experimental data and as we can see in figure 26 this minimum can be quite big so maybe it would not be so effective.

The other point to mention is the execution time. Its absolute value does not give useful information but if we compare it to the execution time of the morphological operator algorithm we notice that it is almost a 25% bigger. This can be a bigger difference when we have to analyse all the vehicles that move under the scanner every day.

3.2.3 Conclusions

With the results of both algorithms it is easy to notice that the accuracy finding the bucket and the time of execution of the morphological operator algorithm makes up for its small error identifying the vehicle.

The small accuracy of the system based on fft is a real big problem. An important industry cannot afford an error of 25% on their productive process. The time of execution is bigger but maybe it is not a real problem if the rest of our system works fast.

In an early future if we success in improve this system it will be better because it analyse a very specific characteristic of the vehicle and it not depends so much in external elements as the structure elements of the morphological operators.

Because of all this information we decide to implement the system based on morphological operators for identify the vehicle and find the bucket in the image.

3.3 Corner detection

This is probably the most important section of the master thesis because the results of this algorithm will have a great impact on the final result. A small error finding the bucket's corners will result in a big error when the geometric transform is applied.

In order to detect the bucket's corners we will test two different algorithms: Hough Algorithm (Hough [2]) and Harris Algorithm (Harris [3]), one based on the detection of edges and other based on the detection of interest points. Then we will try to improve the accuracy of the results obtained.

Before applying these algorithms we will explain how we obtained a simple approximation of the bucket. Because both algorithms are very sensitive to irregular edges this should simplify our task.

To decide between both techniques we will only consider the accuracy obtained in the results.

So the steps (based on the different algorithms) will be:

1. Smooth the bucket
2. Find the corners of the smoothed bucket to have an approximation
3. Find the corners (candidates) of the original bucket
4. Apply a distance filter to choose between all the candidates using the corners of the smoothed bucket.

3.3.1 Smoothing the bucket

As we said before, applying the algorithms directly has very confusing results: a lot of “false” corners appear along the edges of the bucket. That is because the sides of the bucket are not perfectly straight lines and some rocks may be partially outside of the bucket.

In order to solve this problem we had to find a way to obtain the elementary shape of the bucket to simplify the work for the corner detection algorithms. This step will be independent of the algorithm finally chosen. Thinking about it we realized two important factors:

1. The vehicle is always moving forward: it means that the rotation of the bucket image is always small, so we may use a shape not invariant to rotation to try to apply as a morphological operator.
2. After the resize of the image, the dimensions of the main body of the bucket were always similar (120 * 70 pixels).

Attending to this factor we design a filter system consisting of:

1. Convert the image of the bucket into binary mode (previously we filtered the NaN data of the image).
2. Build a 10% size element of and ideal bucket. It will be the next matrix:

```
[0 0 0 0 0 0 0 0 0;
 0 0 0 1 1 1 0 0 0;
 0 1 1 1 1 1 1 1 0;
 0 1 1 1 1 1 1 1 0;
 0 1 1 1 1 1 1 1 0;
 0 1 1 1 1 1 1 1 0;
 0 1 1 1 1 1 1 1 0;
 0 1 1 1 1 1 1 1 0;
 0 1 1 1 1 1 1 1 0;
 0 1 1 1 1 1 1 1 0;
 0 1 1 1 1 1 1 1 0;
 0 1 1 1 1 1 1 1 0;
 0 1 1 1 1 1 1 1 0;
 0 0 0 0 0 0 0 0 0];
```

3. Apply six times the erosion operation in the bucket using this element.
4. Apply six times the dilation operation in the bucket using this element.
5. Finally find the mine walls and delete them, cutting the image.

The result of the steps 3 and 4 is the same as applying the erosion with a 60% size element and then applies the dilation but faster (is an optimal way of doing it). Using a big element the main shape will be preserved and all the irregular corners will be filtered (the smoothed bucket will be smaller). The results of the algorithm are very good, giving us a very good approximation of the bucket with very regular edges that should make the corner’s detection algorithm work better.

Let’s follow the process step by step. The figure 28 shows an original bucket (of our chosen image). Then figure 29 shows the image binarized. In figure 30 shows the bucket smoothed after the erosion and dilation.

Realized that the mine walls have been removed and now we just have one figure in the image: the bucket. The problem is that in the original image they are still there. We must cut the image (see figure 31) because once we will try to find the candidates in the original image the mine walls will cause a lot of candidates to appear, increasing the time of execution and in the worst cases causing a malfunction.

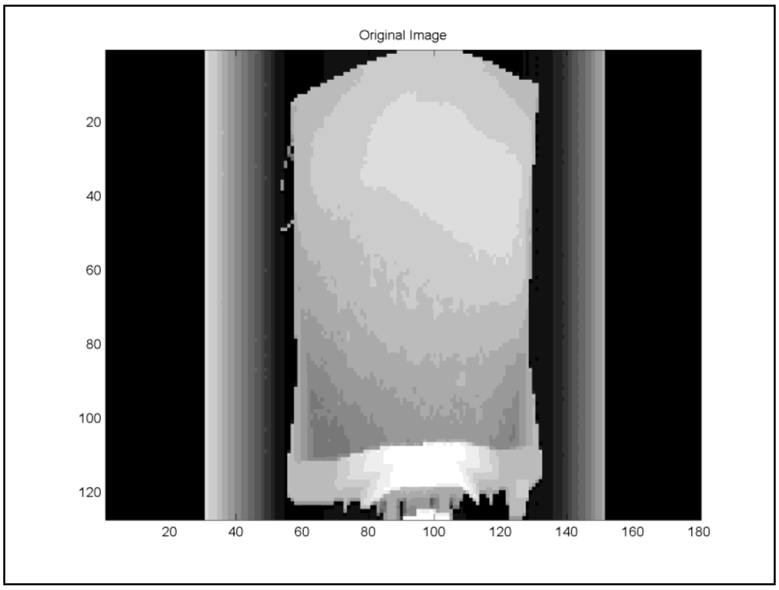


Figure 28: Original bucket image

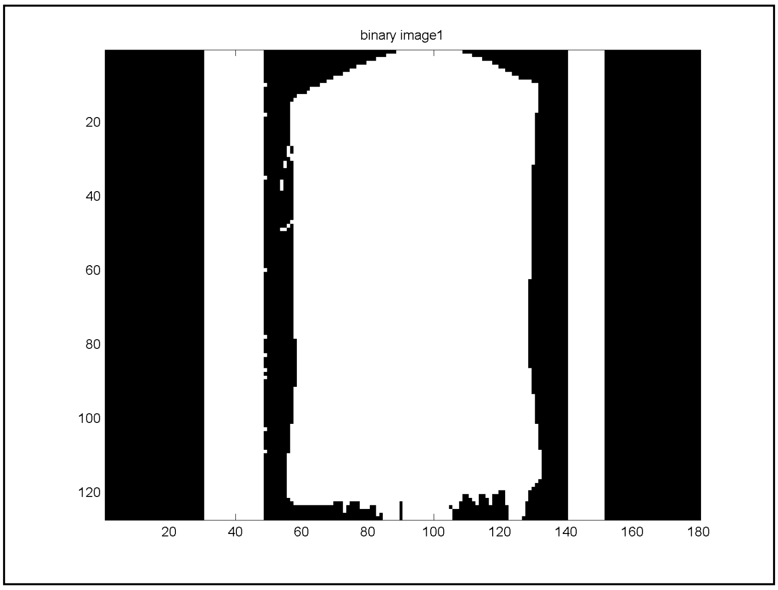


Figure 29: Binarized bucket image

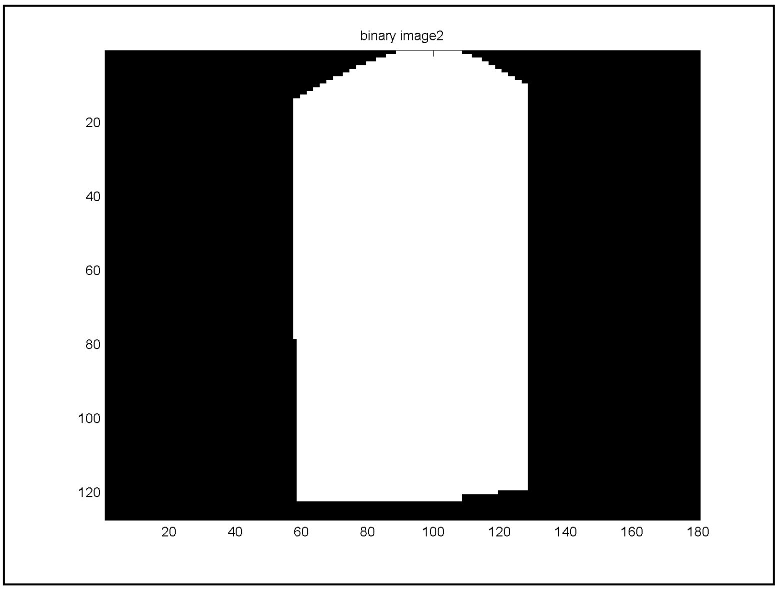


Figure 30: Smoothed bucket

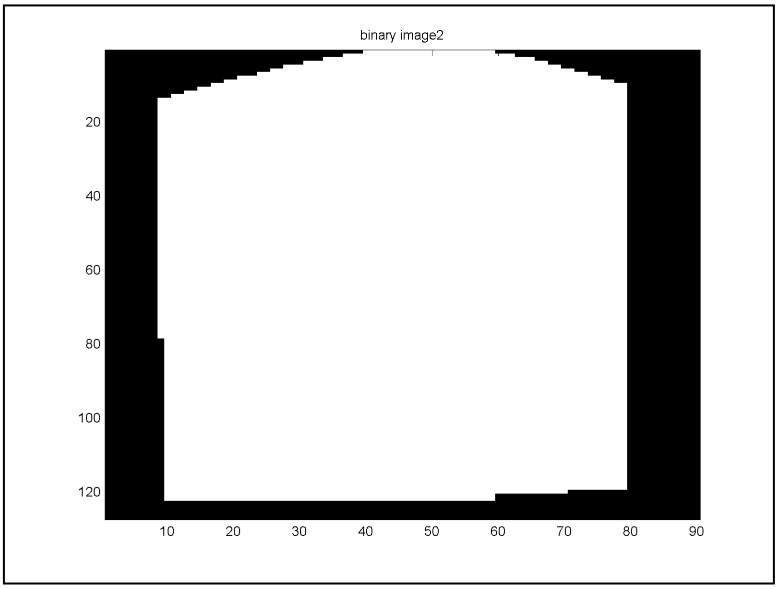


Figure 31: Smoothed bucket with mine walls cut

We have to say that the algorithm worked perfectly. In the 27 images of the test bench successfully identified and with its bucket successfully delimitated 27 images had their bucket image successfully smooth (100%).

3.3.2 Find the corners of the smoothed bucket. Choosing a method

Now it is time to choose an algorithm to detect the bucket corners. We have two choices: the first one, the Hough method, is an algorithm really good for finding edges (lines) which compose a figure in an image. Looking for the extremes of those lines we should be able to detect the corners of the figure, in our case the bucket.

A theoretical base of the algorithm is explained in appendix – B of this report so we will focus on the steps of implementation and the results obtained with it.

The second one, the Harris algorithm, is more specific. The goal of it is to find corners in an image so it should be ideal for our task. It is implemented in Matlab too so it is obvious that it will be a candidate to test. As it directly finds the corners of the figure it is supposed that not too much post process work would be needed.

As we said before for Hough, the theoretical base of Harris is explained in appendix – C of this report so we will focus on the steps of implementation and the results obtained with it.

3.3.2.1 Hough method

In order to detect the bucket corners we will apply these steps:

1. Obtain the bucket edges: For this step we will obtain the inner boundary, which is the result of subtract to the original image the eroded image with a square 3x3 pixels (8-connected). See Equation 1.

$$\text{Inner boundary} = \text{Im} - \text{erode (Im, square)}$$

Equation 1: Inner boundary

2. Apply the Hough algorithm implemented in Matlab. This is a sequence of different Matlab functions with the following control parameters that have been chosen attending to one goal: find the two vertical lines at the sides of the bucket.
    - ‘RhoResolution’ = 1 (To build the accumulation matrix of the method the jump between two consecutive Rho will be 1 pixel)
    - ‘Theta’ = -10:1:10 (To build the accumulation matrix of the method the Theta parameter will go from -10º to 10 º and will be increased 1º in each step. This means that we will look for vertical lines)
    - ‘Scaling’ = 15 pixels (Parameter that depending on its value will delete the detected lines which are too close)
    - ‘MaxPeaks’ = 2 (Parameter that determine how many lines will be shown. We will take the 2 more representatives)
    - ‘Min Length’ = 50 pixels (Lines obtained smaller than this parameter will be deleted or will not be considered)
    - ‘Fill Gap’ = 100 pixels (Lines with the same orientation and less distance between them than Fill Gap will be merged)

3. The results of these operations will be a data structure called ‘line’ with all the information needed (starting and ending points of the line, etc.). These points should determine the corners of the smoothed bucket.

Let’s follow now the procedure with images. In figure 32 we can see the binary and cut bucket image. In figure 33 we can see the inner boundary of the bucket. In figure 34 can see the results of the ‘line’ data structure. In green we can see the edges detected and in red and blue the starting and ending points of each edge.

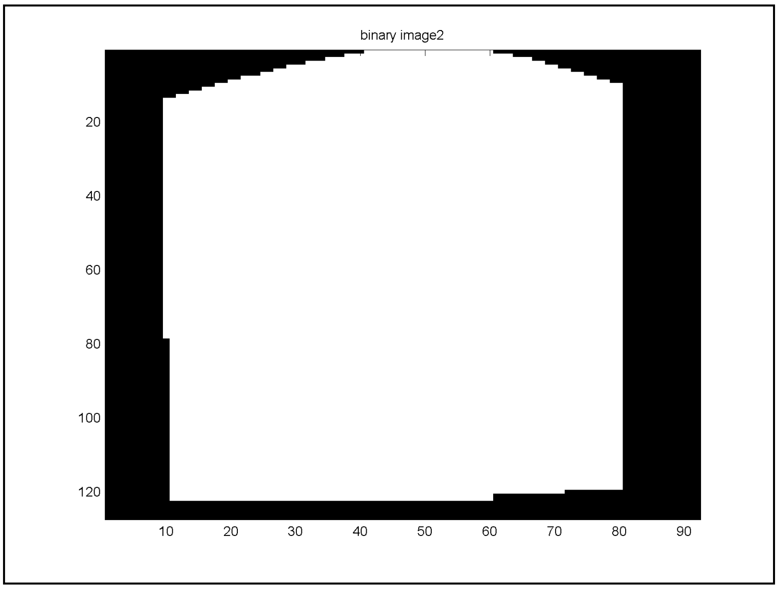


Figure 32: Binary and cut bucket

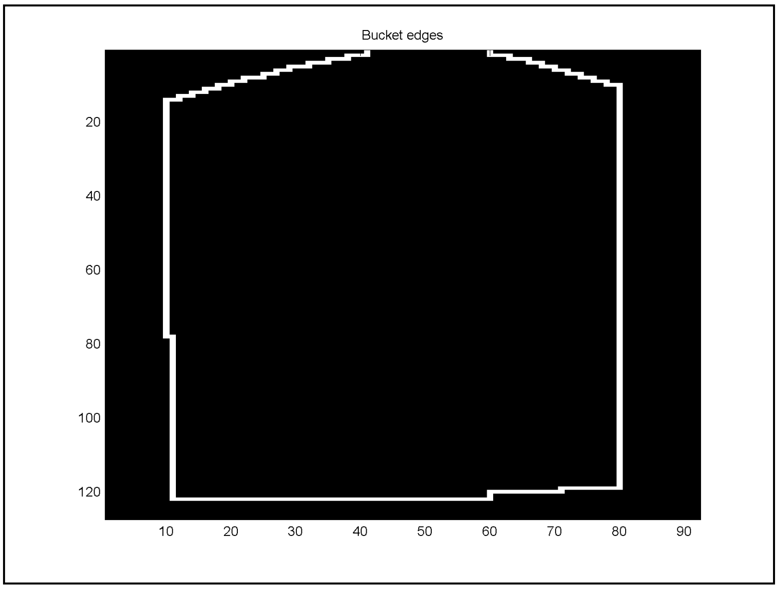


Figure 33: Inner boundary of the bucket

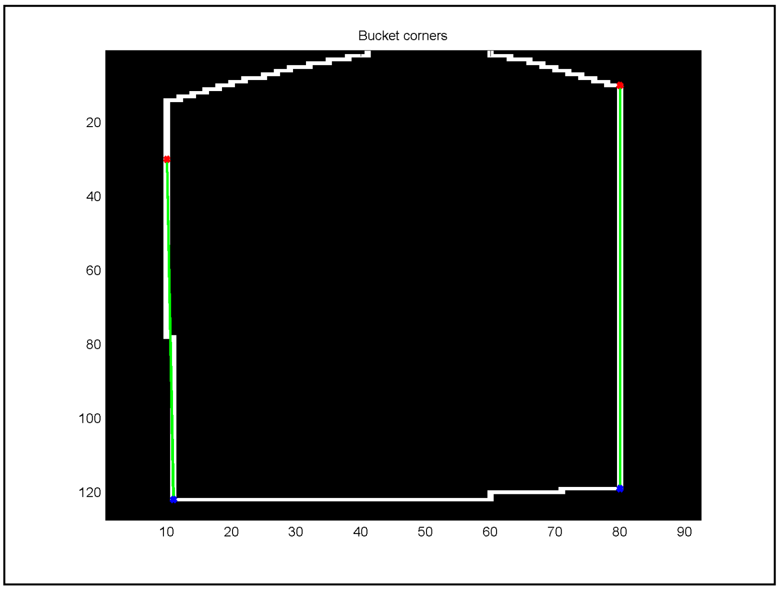


Figure 34: Hough algorithm results

3.3.2.1.2 Results

The results obtained were not as good as expected and figure 34 resumes very well the behaviour of the algorithm in the test bench. As we can see in the left edge, the red point does not match with the bucket corner.

After testing the Hough algorithm in our test bench of 30 images we obtained that just 8 of 27 images (29.63 %) have their corners successfully found. 19 images like figure 34 had one or more corners that need some kind of correction. See figure 35.

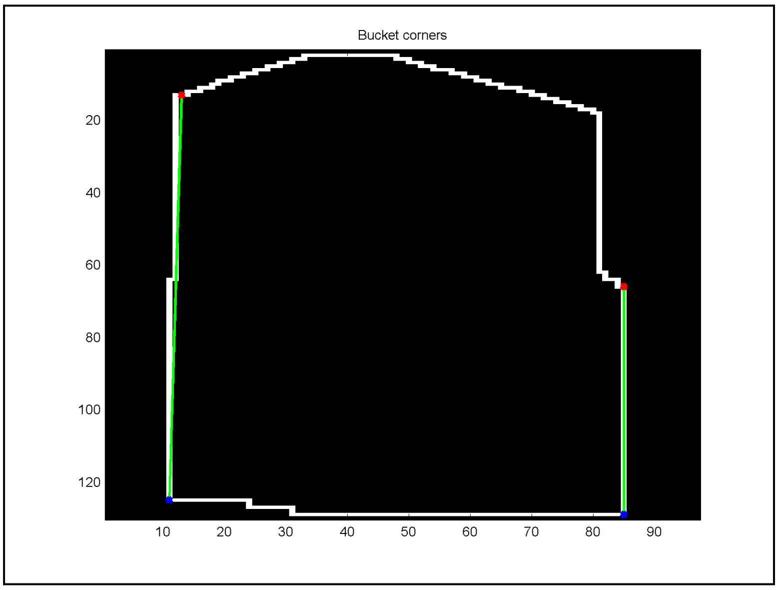


Figure 35: Hough detection error

This is probably because the sensitivity of the algorithm: the Hough method is very good when you need to find regular figures with straight edges like squares or triangles but is not so good to find circles or rotated figures. Because of that in our first tests of the algorithm where we tried to find all the edges of the bucket and we tried with many different combinations of the control parameters so a lot of small lines appear in the upper part of the bucket (see figure 36). This made it difficult to find the upper corners.

The other point is that a small pick in a straight edge results in the split of a line which delimitate the edge. This produced two new candidates to be corners.

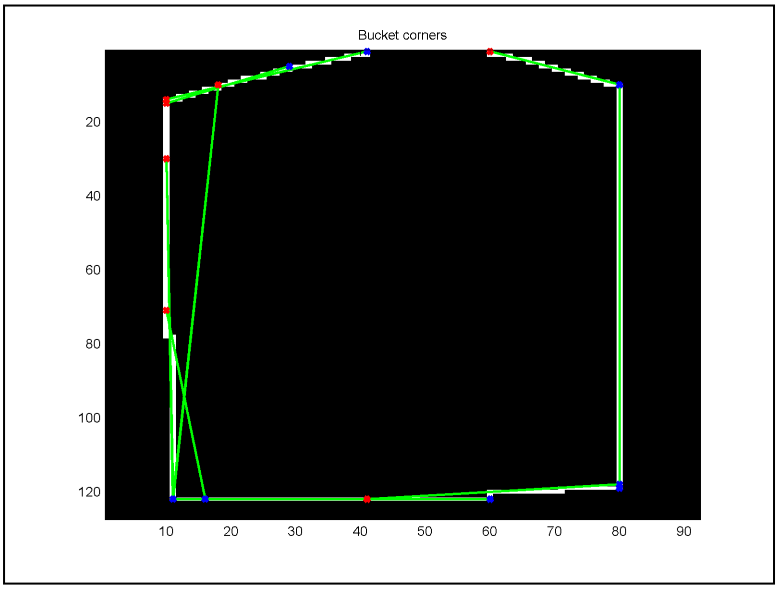


Figure 36: Hough method results allowing shorter edges and more peaks. Note that is the same bucket as figure 34.

Because of the bad results of the Hough method in the "easy" task of find the corners of the smoothed bucket we decided to early discard this method as a candidate of the full algorithm and we will continue all the procedure of finding the bucket corners using the Harris method which results more efficient as we will forward see.

Maybe if the figure had been something simpler the Hough method will have worked better so my early conclusion is that depending on what we are looking for in the image the Hough method will be a candidate to be used. This will be discussed later in chapter 5.

3.3.2.2 Harris method

In order to detect the bucket corners we will apply these steps:

1. Find the candidate corners using the Matlab function 'cornermetric'. This function has a sensitive parameter, K, that after some test we have finally set to K = 0.16
2. Select from the candidates the 4 final corners. The filter divides the image in four quadrants. For each quadrant we will select just one candidate, according to how far is from an imaginary 'Y' axis that divides the image in two parts

In order to understand this process better we can have a look to some figures: in figure 37 we can see the smoothed binary bucket. In figure 38 we can see all the candidates as red spots that the function 'cornermetric' has found. In figure 39 we can see the final corners of the figure after applying the filter and in figure 40 we can see where this corners match in the original bucket image.

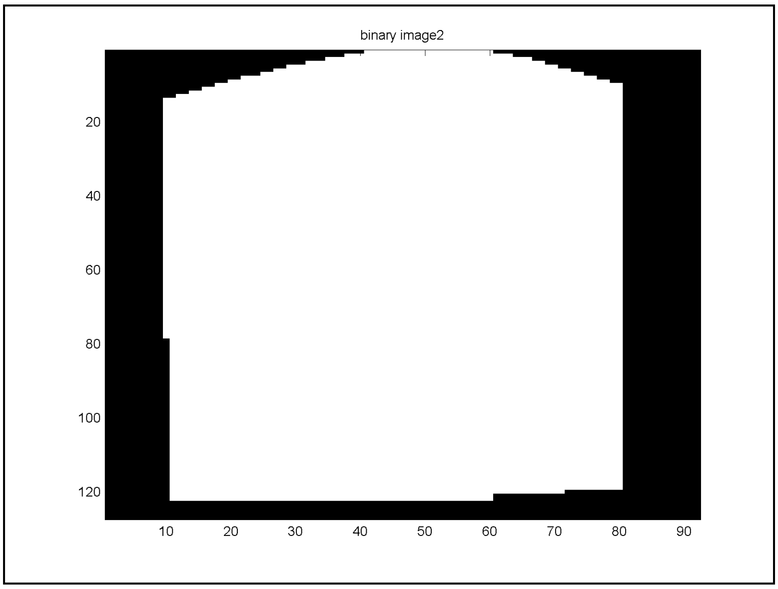


Figure 37: Smoothed binary bucket

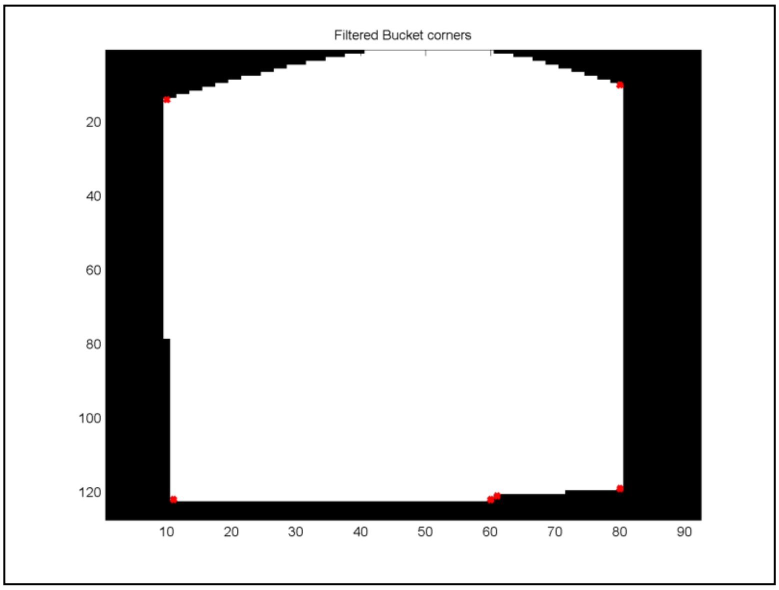


Figure 38: Corner candidates

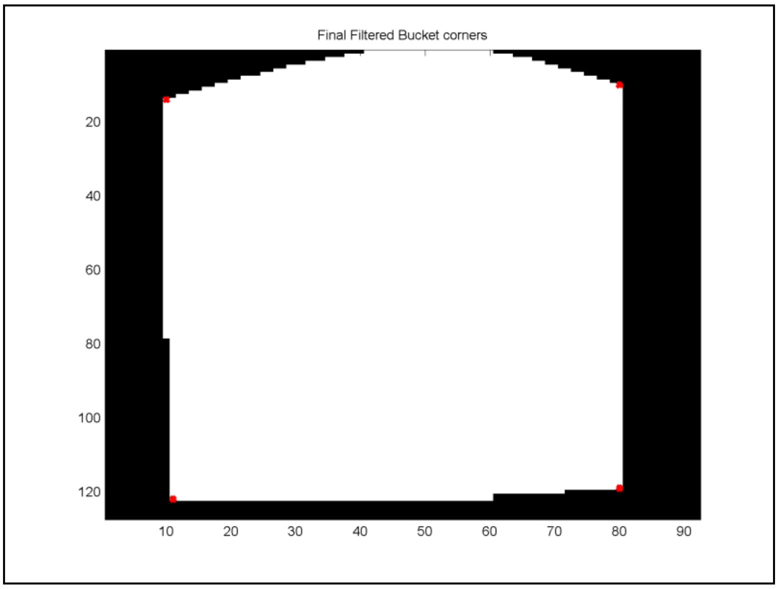


Figure 39: Final smoothed bucket corners

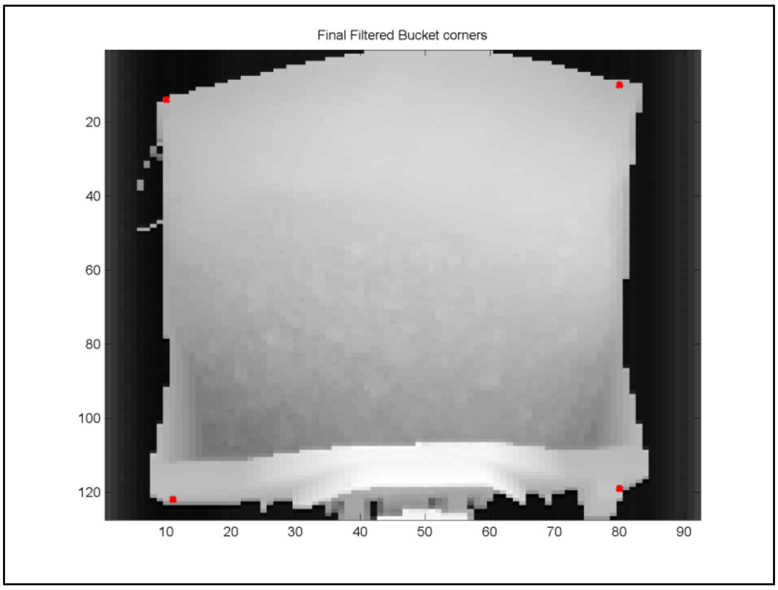


Figure 40: Corners of the smoothed bucket matched to the original bucket image

3.3.2.2.2 Results

After the testing of the testing of the Harris method in our test bench of 30 images we obtained the next results:

- 16 of 27 images (59.26%) have their corners perfectly found
- 11 of 27 images (40.74%) have had a small deviation in the final position of one or more detected corners. See figure 41

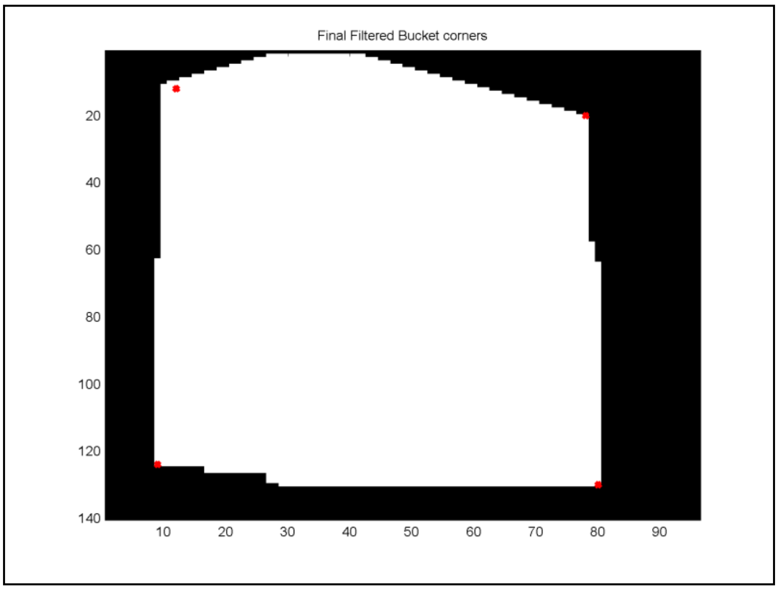


Figure 41: Smoothed bucket with a small deviation in its top left corner

As we can see in figure 41 the deviation is not too big so these images which the corners could not been exactly found can still be used for our porpoise: what we are looking for is a good approximation of where the final corners will be and give us a reference, a region to start the search.

We have to say that during the implementation of this method, before we got the great idea of applying the filter to the bucket image the Harris method worked worse than the Hough method, showing a great number of candidates. See figure 42, an image from the early tests.

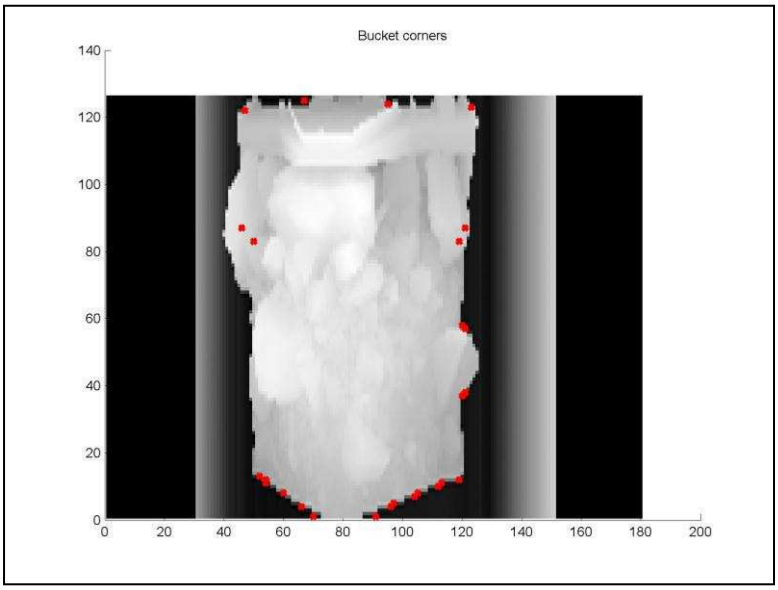


Figure 42: Result of the early tests of Harris method

The presence of NaN data and the mine walls greatly disturbed the results of the Harris method too till the point of consider any element outside the bucket as a candidate. See figure 43.

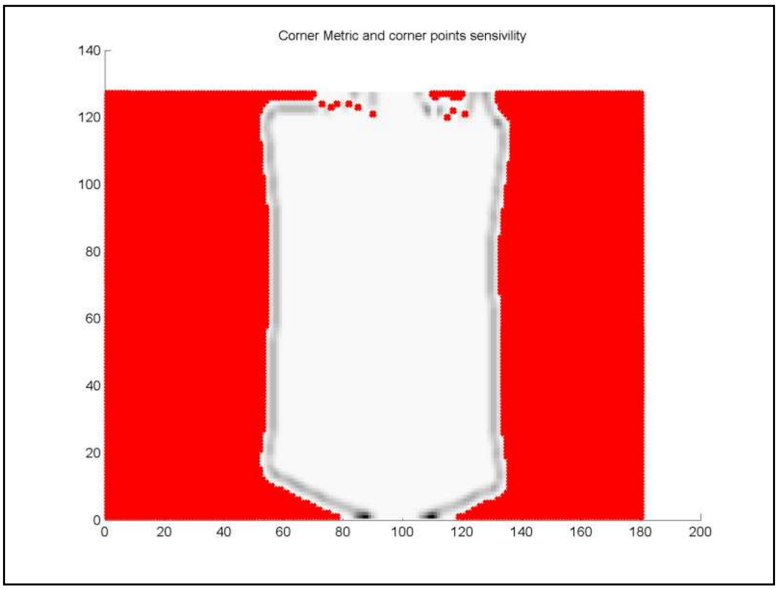


Figure 43: Result of the early tests of Harris method. Every red spot is a candidate to be a corner!

Because of this distortion we started thinking in a way to reduce this effect and try to find a first approximation before applying the Harris method directly in the original bucket image and the results of this early filter seems to be really good.

3.3.2.3 Results

As we have said before we have chosen the Harris method to implement all the corner detection system as its accuracy its better than the Hough method.

We will talk about the time of execution at the end of this section (3.3). We still consider it important but we cannot stop remembering the reader how important is the accuracy at this point. Every small approximation we do introduce an error in the final result but the error introduce because of a small deviation finding the corners will have a big impact in the final result.

3.3.3 Find the corners of the original bucket

As we have seen in section 3.3.2.2 we have used Harris method and 'cornermetric' function in order to find all the possible candidates for bucket corners.

We have previously cut the image so we have just to call 'cornermetric' with k=0.05 to the binarized bucket and result in figure 44.

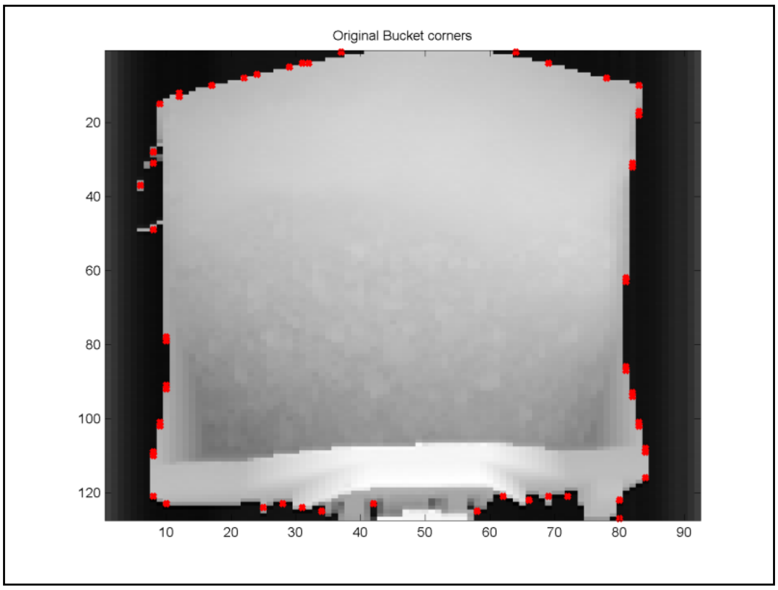


Figure 44: Harris to the original bucket image with K = 0.05

Note that with a smaller value of K we detect a bigger number of candidates. As we can see in the theory of appendix - C a smaller value of K sensitivity parameter allows us to choose between candidates that have less probability to be real corners. That is good for us because attending to the irregular shape of the bucket the real corners will not have always the biggest probability.

To demonstrate this we can have a look to the figure 45, which shows the same bucket candidates for a K = 0.16.

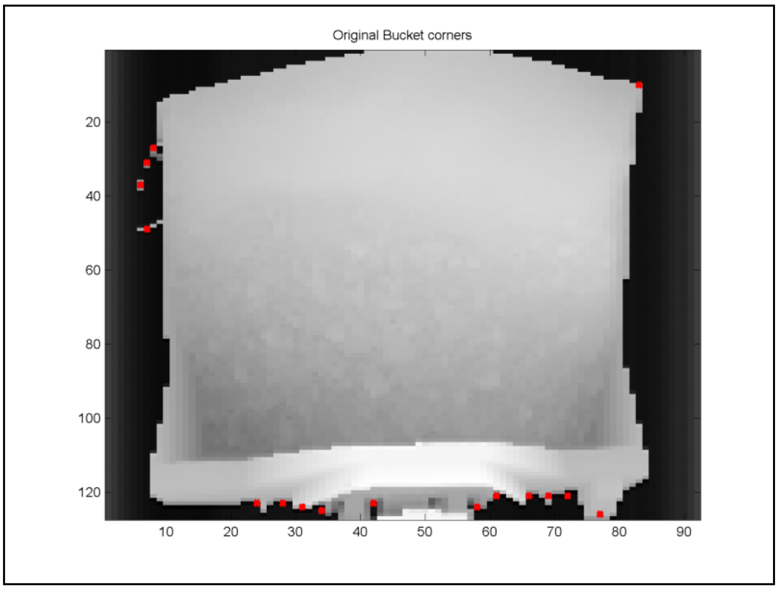


Figure 45: Harris to the original bucket image with K = 0.16

3.3.4 Apply the distance filter

Now we have a good approximation of where the bucket corners should be (the corners of the smoothed bucket) and a group of candidates to be corners of the original bucket. The idea we have had is simple: we are going to apply a distance filter. We will choose the candidate for each corner which is nearest to the smoothed bucket corners.

The problem of this method is that because of the smoothing of the bucket, the approximated corners may be nearest to corners that are not real corners. If we have a look to some images (figures 46 and 47) we realize that the rows of the approximated corners are quite good but they should be nearest to the lateral sides of the bucket.

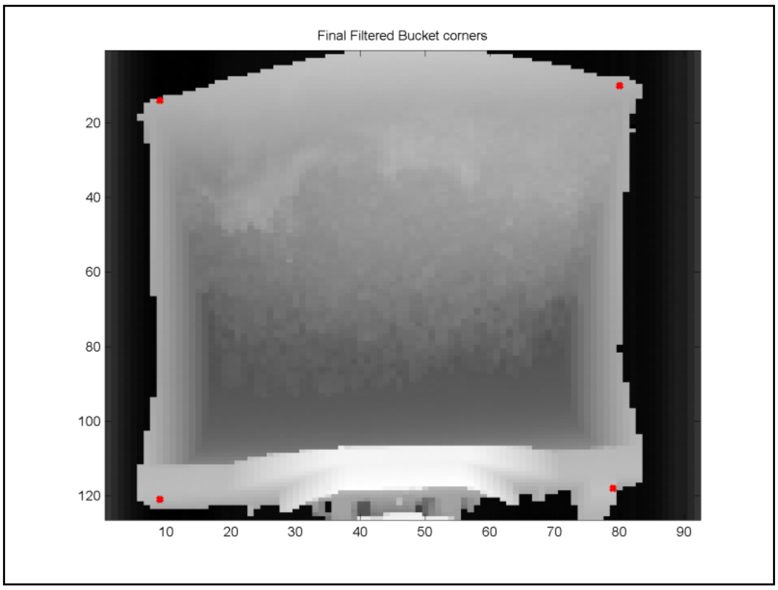


Figure 46: Example of smoothed corners over original bucket image

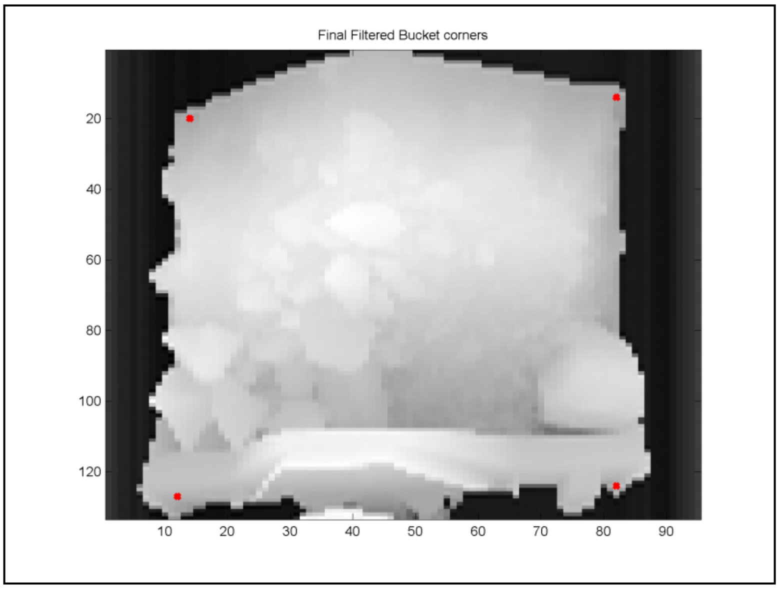


Figure 47: Example of smoothed corners over original bucket image

To solve this small problem before applying our distance filter we will move the approximated corners to the sides, decreasing it 'X' coordinate five pixels for the left corners and increasing it in five pixels for the right ones. With this small correction the chances of select the proper candidates will be increased.

The steps of the final filter will be:

1. Move the corners of the smoothed bucket to the sides 5 pixels along the 'X' axis.
2. Calculate the distance from each candidate to the approximated corners.
3. Select the candidate which is closer to an approximated corner. If there is not a candidate close enough to an approximated corner (if every distance is bigger than 10 pixels) the final corner will be the approximated corner. We have included this as security: it is better to have an approximated corner than a candidate too far.

In order to see the improvement after applying the filter let's compare two figures: figure 48 shows the final corners of the smoothed bucket over the original bucket, figure 49 shows all the candidates to be final corners detected by Harris algorithm and figure 50 shows the final corners selected by the distance filter.

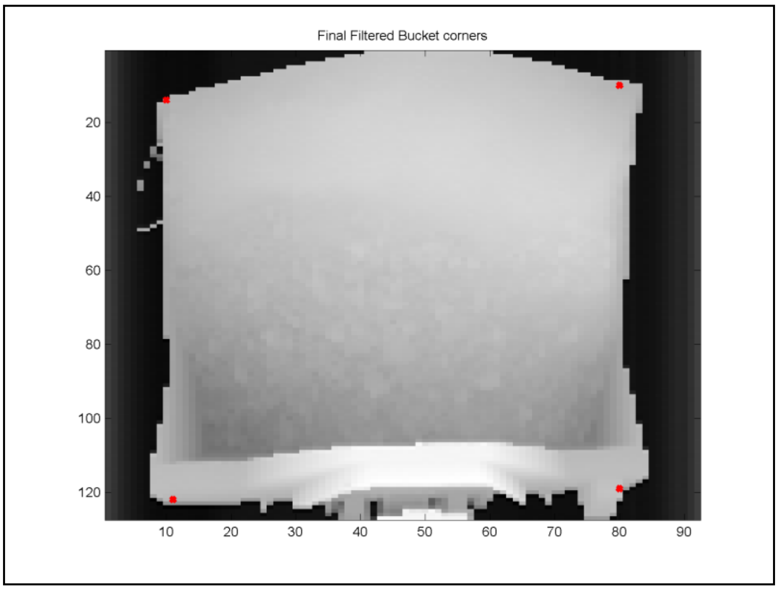


Figure 48: Final corners of the smoothed figure over the original bucket

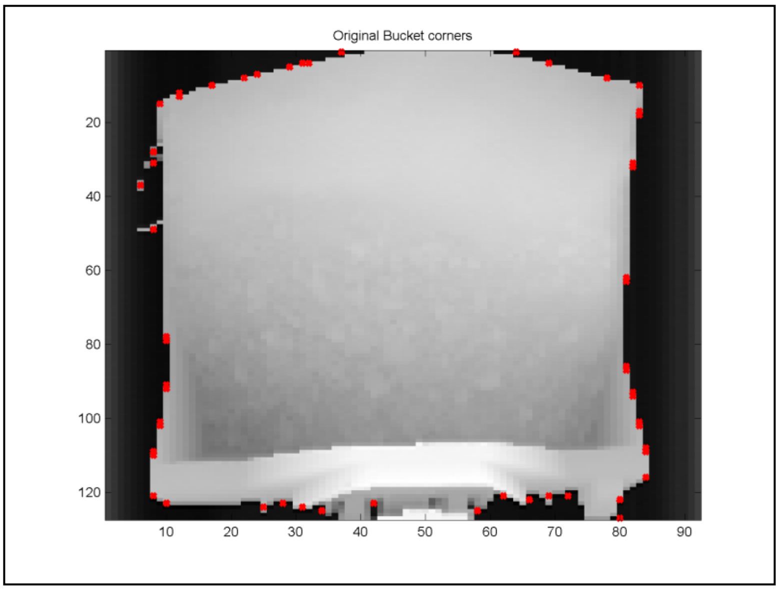


Figure 49: Candidates detected by Harris algorithm

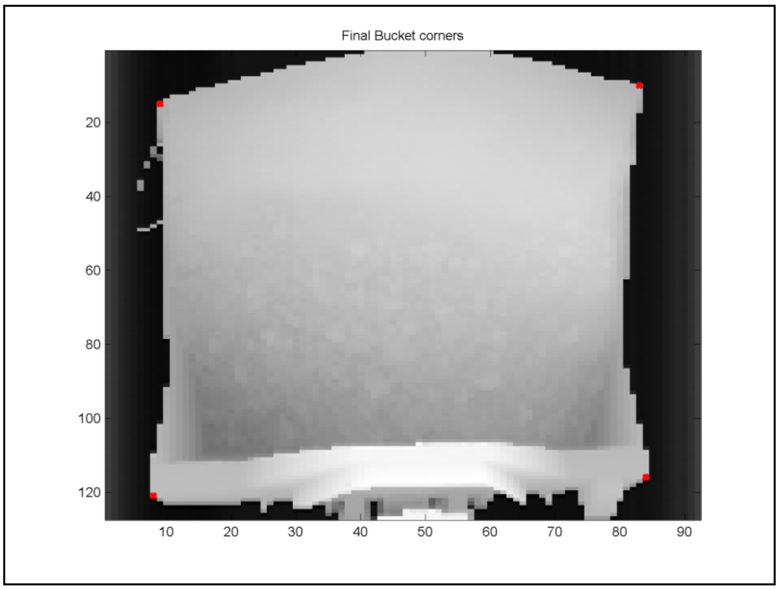


Figure 50: Final corners after applying the distance filter

The results obtained by the application of the distance filter are very good. We have obtained an increasing of accuracy in all the images of the test bench but it still can be better. This is point that can be investigated and improve in an early future in order to reduce the final error of the volume measurement. We will discuss this later in chapter 6.

3.3.6 Results

To sum up the results of chapter 3.3 we have to start saying that all the detection system has been based on Harris algorithm for corner detection. It has seemed to work better than the other candidate, the Hough algorithm for our porpoise.

The early test demonstrate that the Harris algorithm could be used but need some adds to help in the filtering of all the information and because of that we design two specific extras for the algorithm:

1. Calculation of an early approximation of where the bucket corners should be.
2. Application of a distance filter using the information produced by the first extra to select from all the candidates the most suitable ones.

The final accuracy obtained by this procedure can be considered acceptable but it could be improved in an early future.

The time of execution for the 30 images of the test bench has been 887.877 seconds (29.596 seconds per image). Considering that this should be one of the "heavier" tasks of the full procedure the time of execution obtained can be considered a good one.

3.4 The geometric transform and volume measurement

Applying the geometric transform (Zorin [4]) between two images should be as easy as solve an equation system (see appendix – D). Unfortunately it is not true, there are many aspects to consider but the most important ones are the size of the images and the different errors of out of bound that appear: we refer to all the information that is discarded during the match of the images and the consequent error produced.

The procedure we are going to follow is this:

1. Compare the image we want measure with an image of an empty bucket in order to known which one is bigger.
2. Apply the geometric transform from the biggest one onto the smallest one.
3. Calculate the absolute value of the subtraction of the image generated with the image of the empty bucket. As a result we should have just the rocks.
4. The error made should look like a salt and pepper noise. Because of that we will apply a median filter to reduce the possible error made during the transformation. The size of the filter will be 9*9 pixels.
5. Calculate the volume, knowing the area of a pixels and the height. In order to do that we will consider that the size of the bucket is 3900mm*3092mm so we will be able to know the approximate area of a pixel.

The problem we had is that we do not know the exactly volume of rocks for each image (except the images of empty buckets) so we must design a test to measure the final error made along the process.

The test we finally designed consisted on:

1. We select one image; we do not care which one, and calculate its bucket corners.
2. We rotate this image a small amount of degrees (3 degrees will be enough) and calculate its new corners.
3. Apply the geometric transform following the procedure we have commented before.
4. The volume finally obtained will be the error made because in an ideal case the images should exactly match.

For a better understanding of this process we will follow each step with figures. In figure 51 we can see the bucket image with its corners found. In figure 52 we can see the bucket image rotated and its new corners. In figure 53 we can see the applying of the geometric transform to the rotated image. Observe that the rotation has been corrected. In figure 54 we can observe the difference of the images (the error made) and in figure 55 we can observe the final error after applying the median filter.

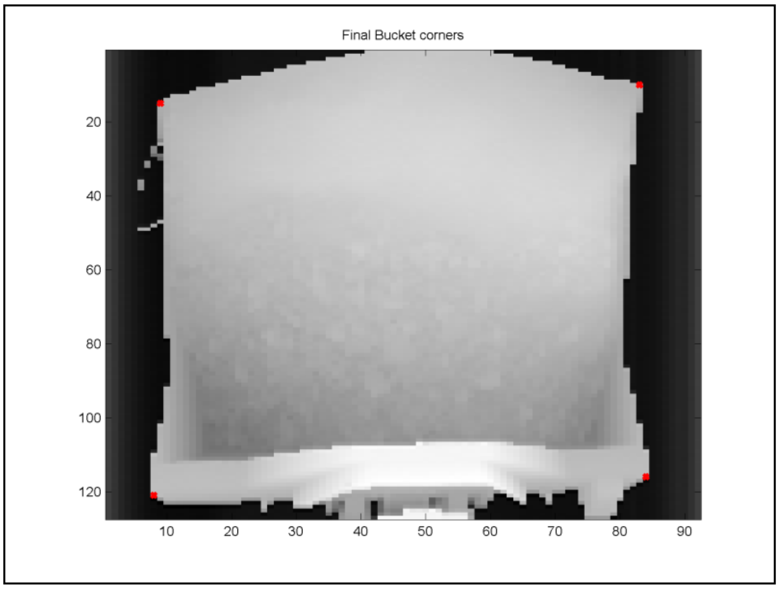


Figure 51: Original bucket image

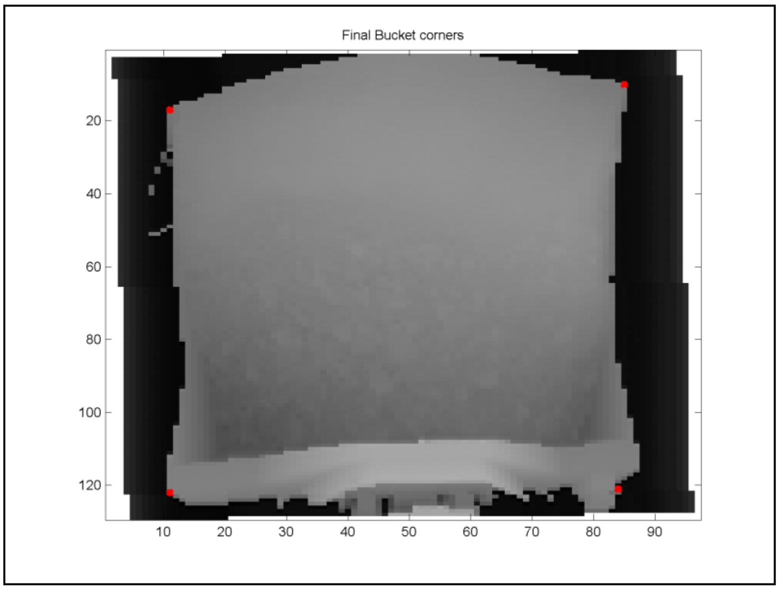


Figure 52: Rotated image

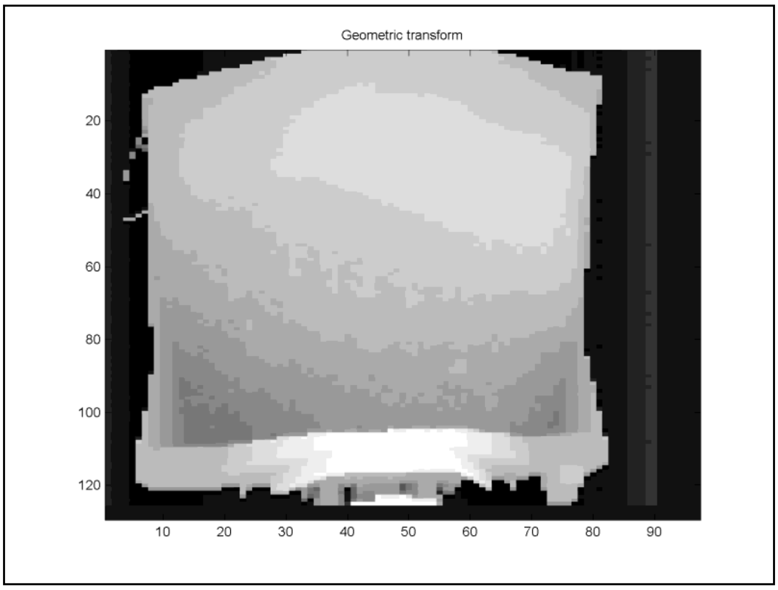


Figure 54: Result of the geometric transform to the rotated image

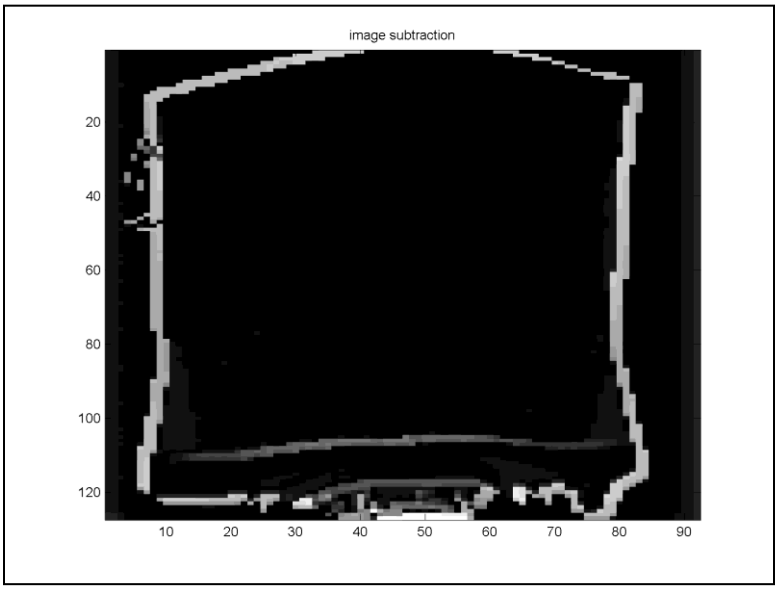


Figure 55: Result of the image subtraction (error)

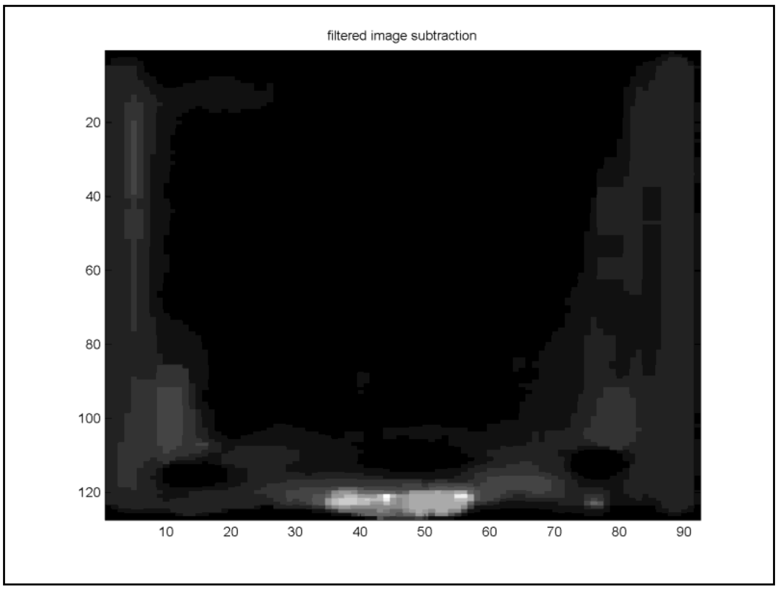


Figure 56: Filtered error. Final volume result

3.4.2 Results

As we can see in figure 54 the geometric transform does not work exactly: one image is shifted over the other and that is why the result of the subtraction looks like a perimeter. This error is made because the supposed corners do not fall exactly in the real corners of the bucket. If we just observe figure 54 we can see that a big source of error is the lower part of the bucket. In an early future we should filter this part before the volume measurement.

Thanks to the test method previously commented we have been able to measure that for the image in figure 51, a 3º rotational variation can produce a 0.84427 cubical meters difference in volume.

For the rest of images of the test bench, no one had a difference bigger than one cubical meter. The question that the reader is probably asking is: is this a big difference or not?

Well, we do not have the exactly capacity of the bucket but we can estimate it on about 8-12 cubical meters from the schematic on appendix – E. This should correspond to a variation of 10% approximately.

The problem is that we do not know the impact of this variation in the final calculation of the density. We have to consider too that all these volume measurements are estimations and without precise data these conclusions are just orientative.

Finally the time of execution for the entire test bench 1556.672987 seconds (51.89 seconds per image approximately) which is inside the parameters requested in the original design.

4. Results

At this point we will sum up the results of the full method we have designed and implemented along chapter 3 and we will follow it with images. This will give us an idea of a normal execution of the algorithm and will make it easy to understand the full process of the image.

4.1 Vehicle identification

At this step we will identify the vehicle in the image and we will interrupt the image processing if the vehicle does not correspond to a Toro2500E LHD.

Finding the edges of the vehicle (figure 56) we can calculate its width graph and using morphological operators we will be able to obtained the closed gradient of the width (figure 57). Looking for maximum peaks in the closed gradient of the width (figure 58) we will be able to identify the vehicle as we have explained in chapter 3.1.2.1.

This results in the successful identification of 29 of 30 images of the test bench.

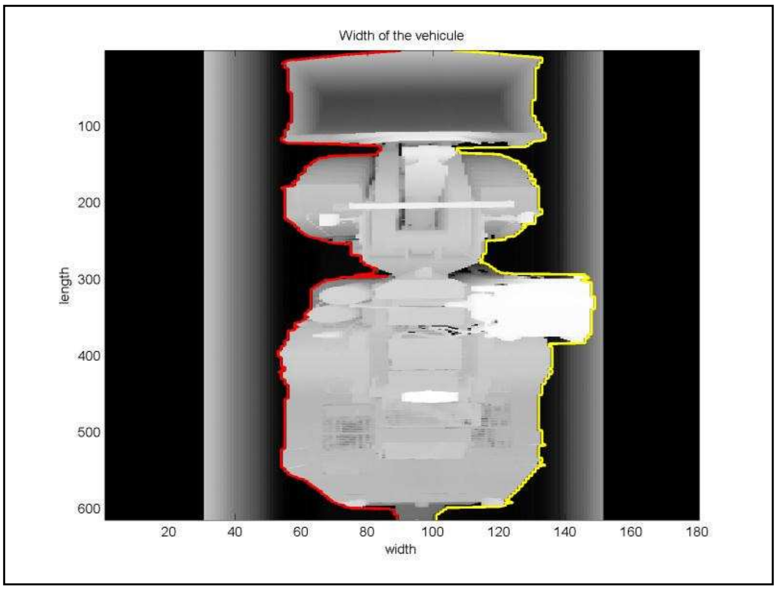


Figure 57: Vehicle's edges

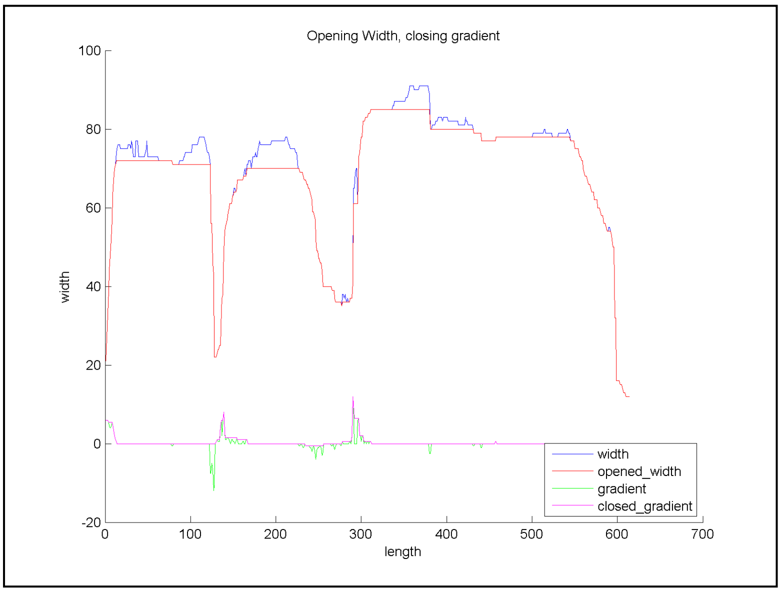


Figure 58: Width analysis

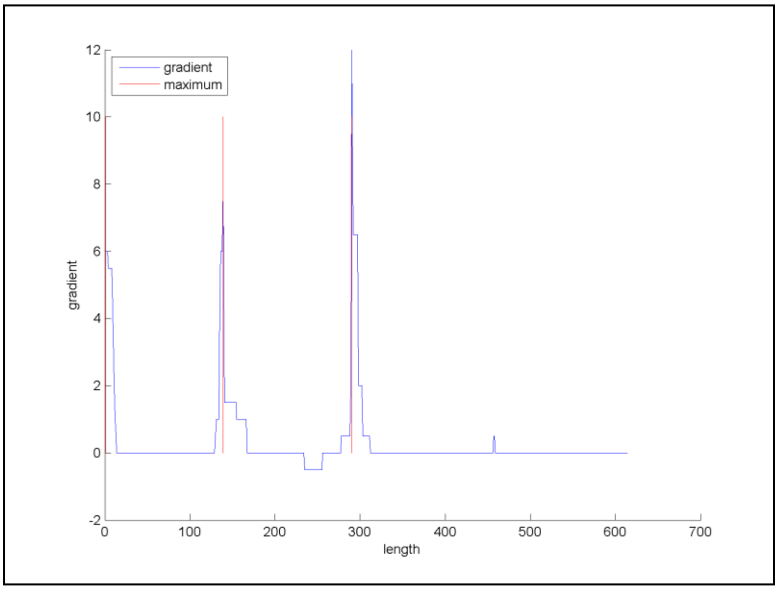


Figure 59: Closed gradient max detection

4.2 Filter the bucket

Continue with the procedure of 4.1 we will continue using the width graph and the morphological operators to identify where the bucket ends and cut it from the rest of the image.

Looking in the opened gradient of the width we will find the minimum peak that will correspond to the end of the bucket (figure 59 and 60).

This results in the successful bucket identification of 27 of 27 images of the test bench.

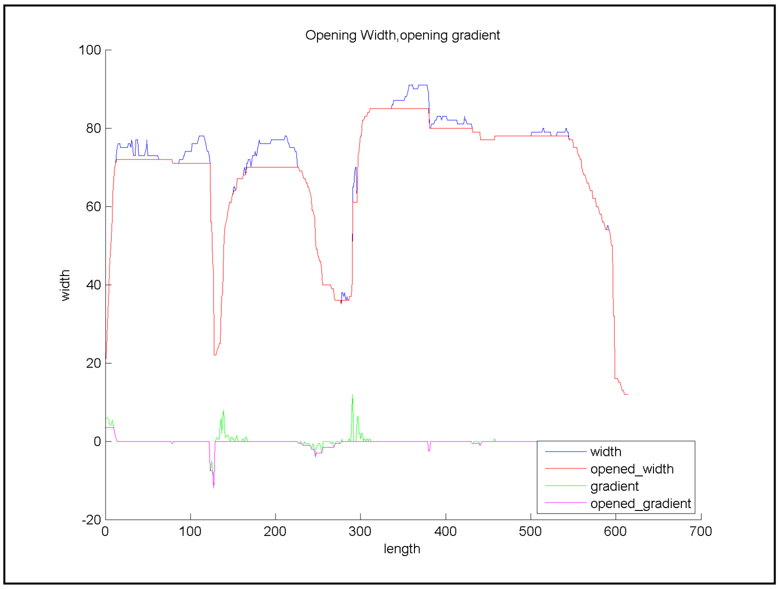


Figure 60: Width analysis, opened gradient

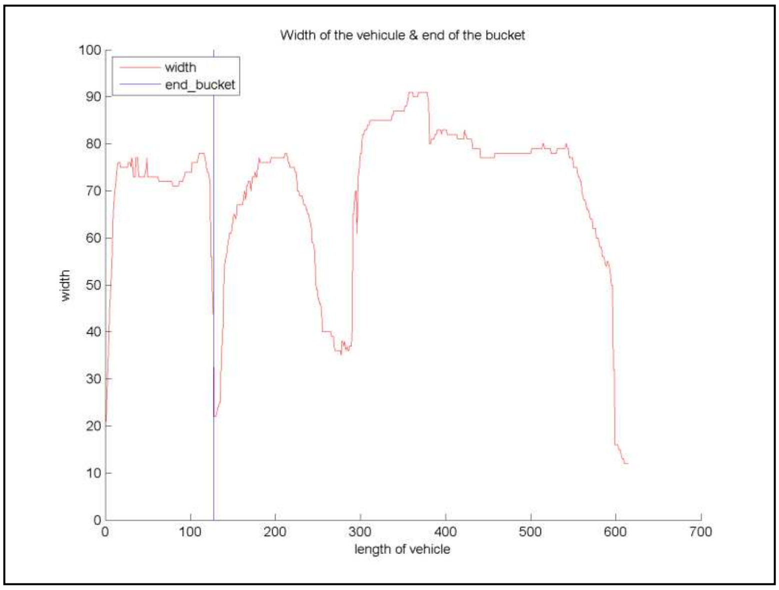


Figure 61: Width graph and end of the bucket

4.3 Corner detection

This has been the most complicated step of the full method. The first step consists on finding the corners of the smoothed bucket. From the cut image of the bucket (figure 61) we first have to binarize it (figure 62) and apply an erosion followed by a dilation in order to obtain a simplified bucket (figure 63). After cut it to remove the mine walls we apply the Harris method to this figure to obtain an approximation of the bucket corners (figure 64).

Now applying Harris to the original binarized image we will obtained the candidates to be real corners (figure 65) that will be filtered using the information from figure 64 in order to obtain the final bucket corners (figure 66).

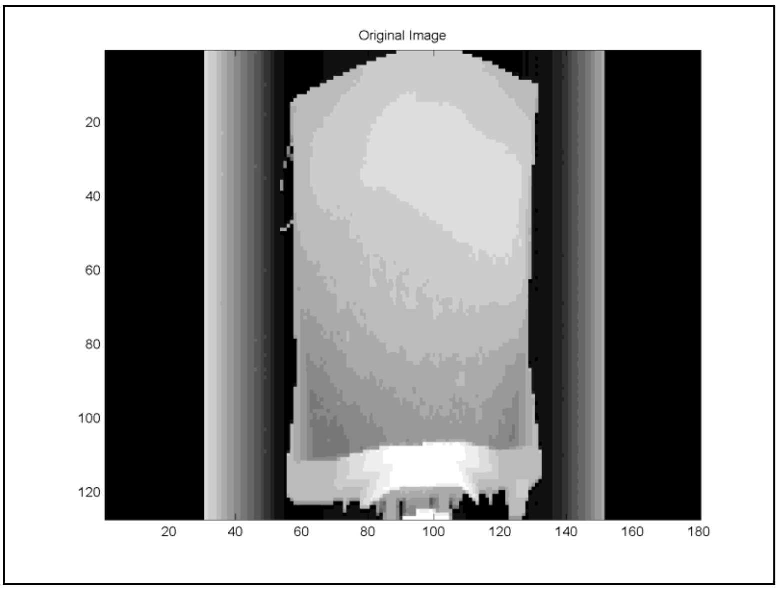


Figure 62: Bucket image

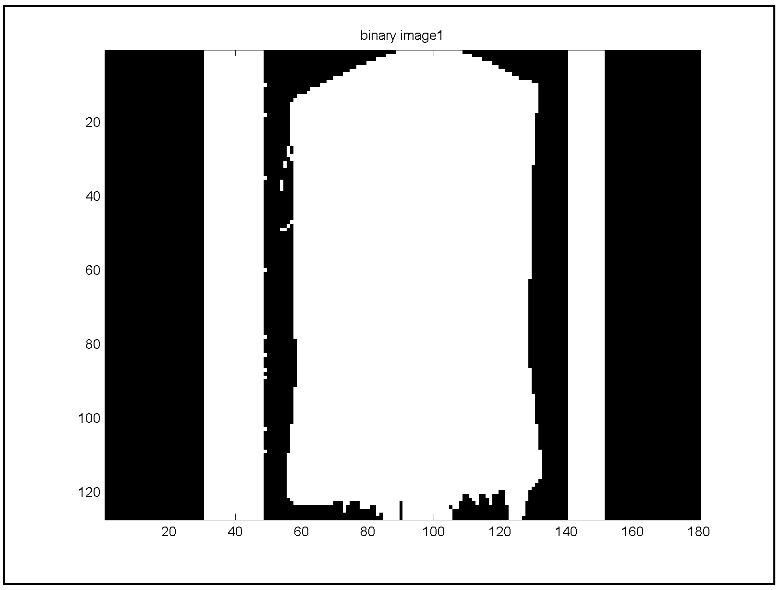


Figure 63: Binarized bucket image

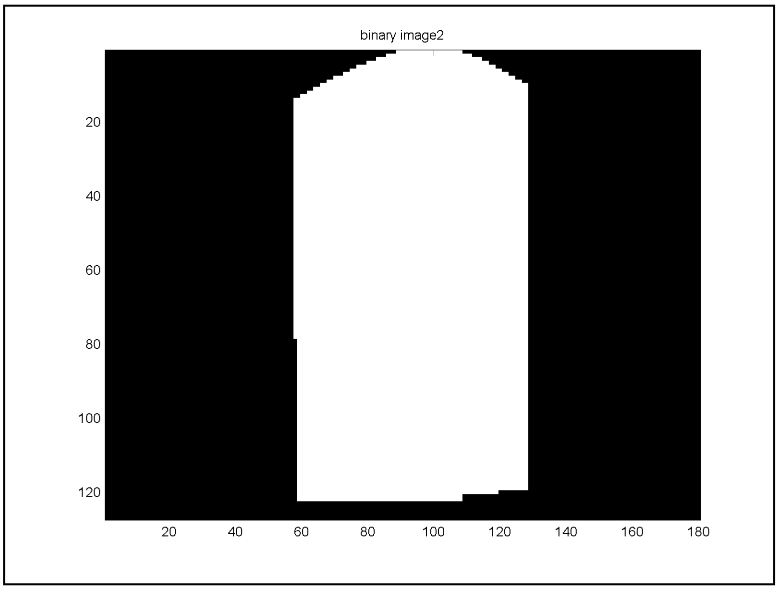


Figure 64: Smoothed bucket

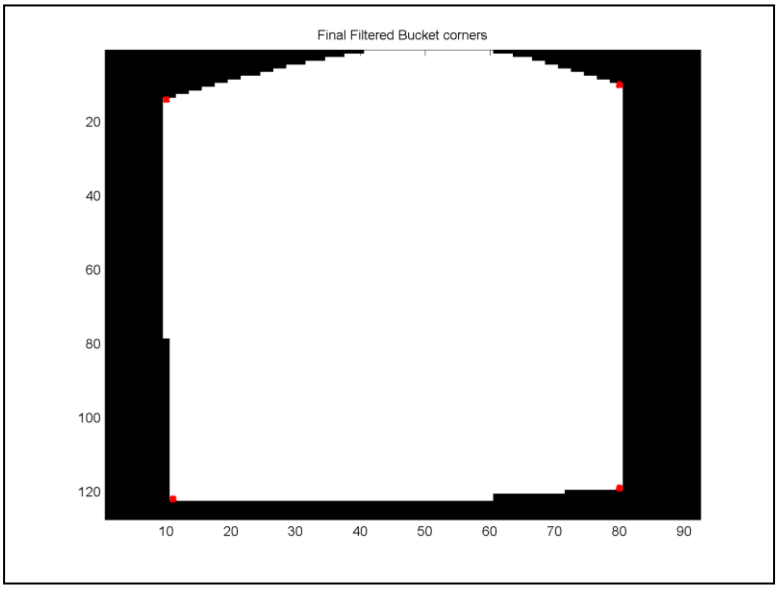


Figure 65: Corners of the smoothed bucket

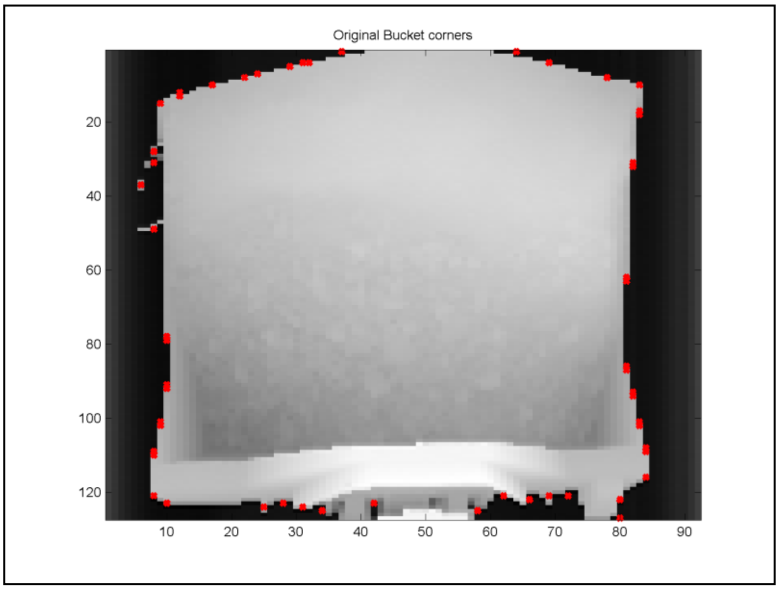


Figure 66: Candidates obtained by Harris

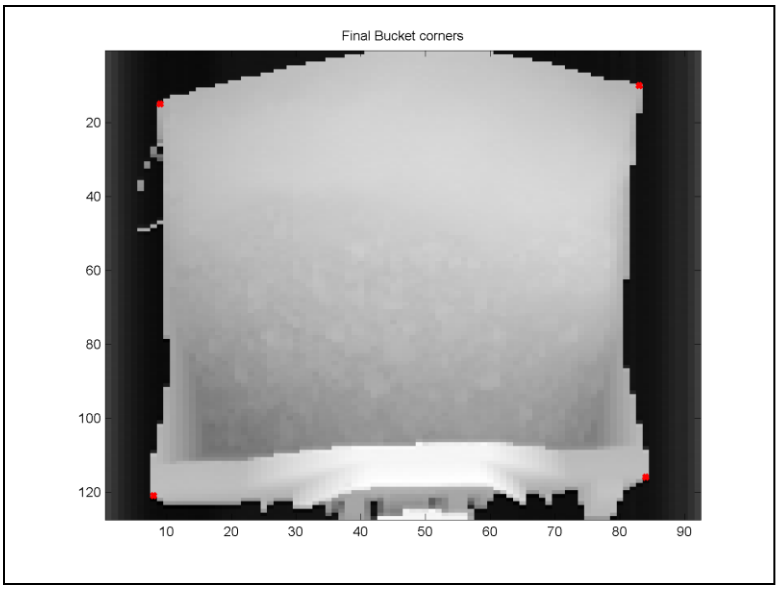


Figure 67: Corners obtained after the distance filter

4.4 The geometric transform and volume measurement

The final goal of applying the geometric transform is matching two images to make possible the final volume measurement but as we do not have enough data to test the final accuracy of the system (we should have the exactly volume of rocks carried by the excavator in each image) we have designed our own test.

This test consists on measure the difference in volume introduced by a rotation translation in an image. In order to do it, we have started from an image from a bucket. After finding its corners (figure 67) we apply a rotation translation of 3º to the image and we found the corners of the new image (figure 68). After that we have applied the geometric transform to correct this rotation (figure 69). The difference in volume should be the absolute value of the difference of these two images (figure 70). Because of this error looks like salt and pepper noise we have applied a median filter to reduce this error (figure 71).

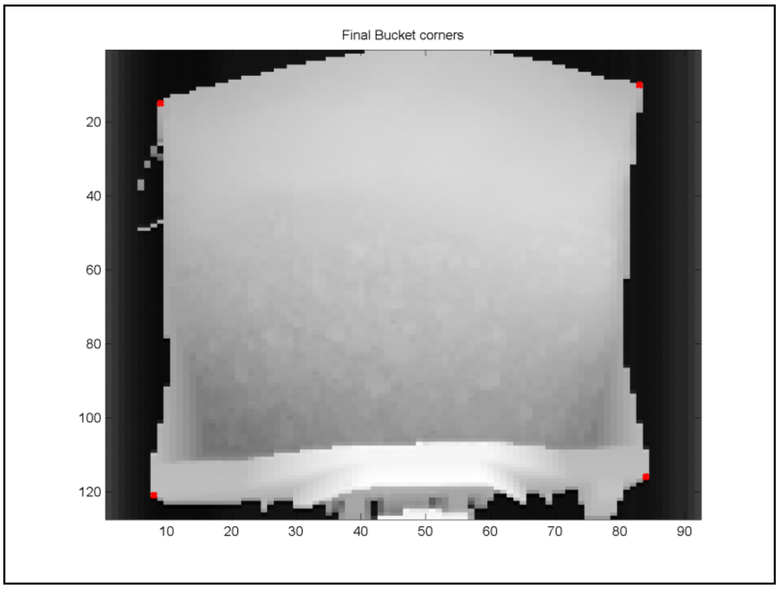


Figure 68: Original bucket image and corners

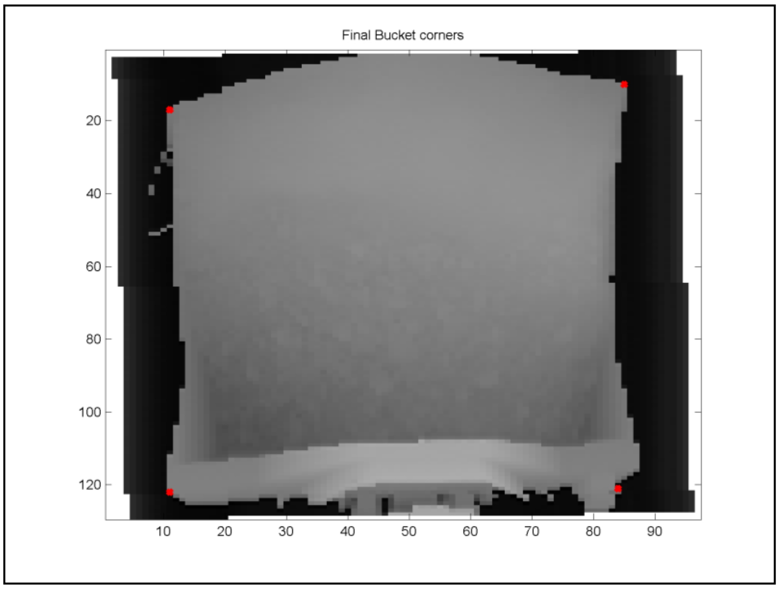


Figure 69: 3º rotated bucket and corners

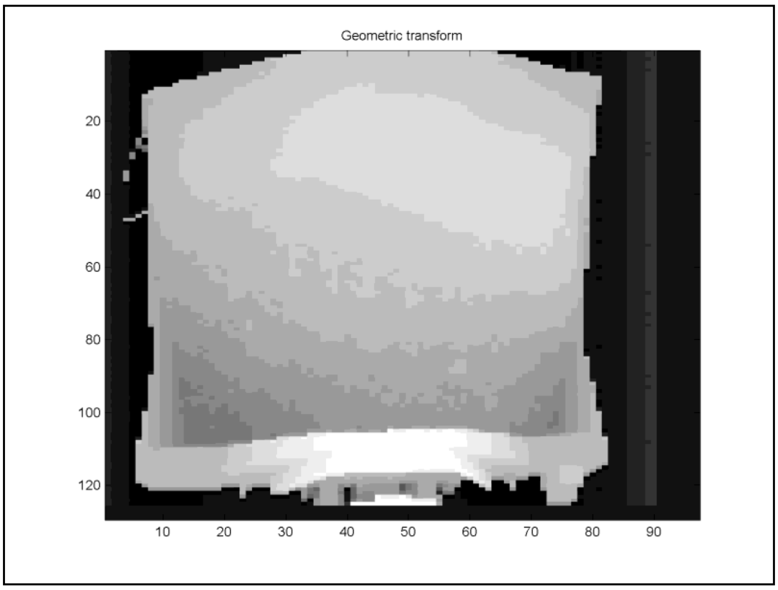


Figure 70: Geometric transform of the rotated bucket

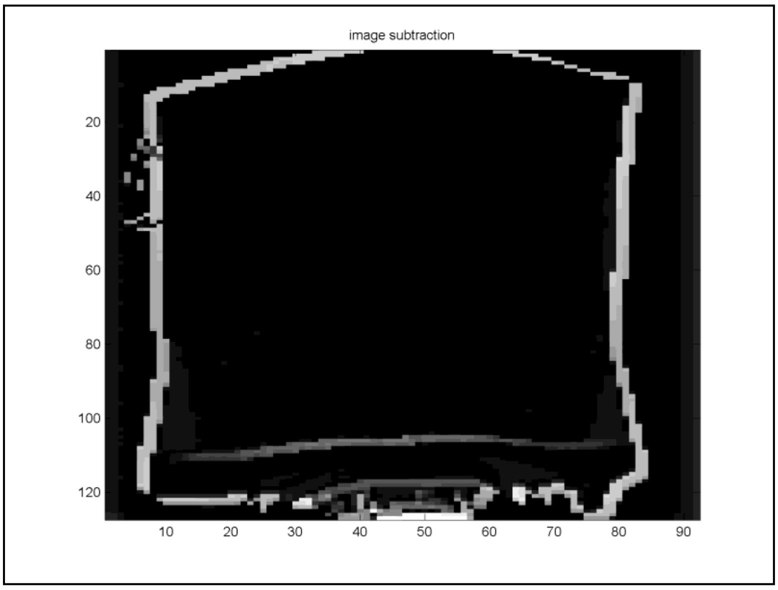


Figure 71: Image subtraction (error)

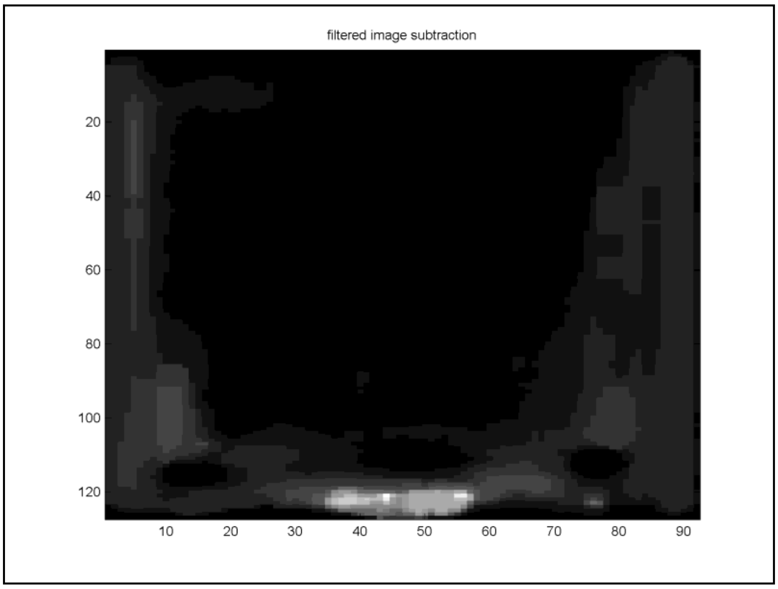


Figure 72: Filtered error (final volume error)

For the example showed along this chapter we can measure the final error consequence of a 3º rotation translation as 0.84 cubic meters and the time of execution as 55.665452 seconds running in the computer before mentioned.

5. Analysis, discussion and conclusions

At this chapter is time to answer many questions and be critic with the final algorithm. We will remark too the success we have reach and to sum up limitations and benefits of our system.

The first thing we have to analyse to determine if the system works well are the results. To analyse the final results we have two parameters we wanted to look for:

1. Time of execution
2. Accuracy (finding the bucket, finding the corners and finally during the geometric transform)

We have to be satisfied with the first one. We are supposed to have two minutes per image to make the full analysis and we have reached our goal in less than one minute. This was supposed to be easy and this big difference allows the user of the system to use a worst physical dispositive or display more information. It will allow too to add some extras to the system accomplishing the time of execution.

The accuracy is something more difficult to analyse. Looking to the final result we can say that the error is relative big and maybe we should be able to reduce it with some improvements. But what we want to discuss here is what works well or where this error is introduced.

The first step is the vehicle identification. The accuracy of the identification system is not perfect but it is really high and we are satisfied with its behaviour. With a fast view over the system an observer could determine that this step is not relevant for the accuracy of the final result but this is not true. In this step we make the first approximation that introduces an error in the final calculation of volume: the rescale of the image. If we reduce the image we will delete rows from the bucket with relevant information and if we add rows these will include information obtained from the rows close to them so it will not be relevant and just an approximation.

This is very important: the first source of error in the real volume measurement is introduced in the rescale of the image. The problem we have is that it is very hard to determine the influence of the rescale and if it is really relevant in the final volume calculation.

We must say that the identification system has a limitation: if the vehicle image is not all at the same scale it will not work. To explain this better the reader must imagine that if the vehicle is moving at any speed when it is passing over the scanner and then it speeds up or decelerates the last half of the image will be compressed or stretched so the proportions of the vehicle will not be the same and the vehicle identification system will determine that is another kind of vehicle and the image will be discard.

The second step is to find the bucket. We have to say that this step works really well: it simplify our work as we do not have to performance the corner detection in the full image. Its accuracy is really good and has worked well along all test bench.

The third step is absolutely the hardest and the step that has taken more time to design, test and improve. As we have said before the accuracy in this step has been a full priority as a small deviation would have a big impact in the final result. A big effort has been done designing small extras to the Harris method in order to have the best possible accuracy.

The idea of using a smoothed bucket to have an approximation of the final corners works really well. The shift we do later maybe could be improved somehow to get better results but it is necessary: without it the results in every case we have considered were worst.

The Harris method over the original image could be improved too. As we have used a Matlab function we cannot modify it in order to improve its accuracy changing the acceptable range of values of K for example.

But as final result the corners found at this step are really closed to the bucket real corners and the extras we have added seems to work quite good and may be used as a base in future developments.

The last step is probably the weakest of all the process. As we solve the equation system the values obtained had to be truncated to get integer index values to generate the new image and this introduces an error.

The common case of apply the geometric transform from a big image into a small one generates an error too: some pixels of the bigger image must be discarded in order to match both images.

Because of these two big limitations the geometric transform is the point where the biggest error is introduced and joined to the rescale of the image we have identified the biggest inconvenients of this method.

Finally we have observed too that after the subtraction of the images the bottom part of the bucket is a main source of error. We cannot remove it before the application of the geometric transform because it is useful for find the bucket corners but once we have subtract the images if it has not been precise it represent a big percentage of the final error and it should be corrected.

6. Future work

At this point there are two things we have to talk about:

1. How can we continue the development of the system
2. Other applications this system could have

The first one can be orientated again in two directions: the first one is the improvement of accuracy. The second is the robust property of the algorithm and its generalization for other kind of vehicles.

About the improvement of accuracy we have talked many times along this report. The different sources of error have been commented: the rescale of the image, the truncation of values during the geometric transform or the necessity of delete the bottom part of the bucket after the image subtraction. The corner detection system could be improved too improving the heuristic that choose between all the candidates from Harris method.

In order to improve the robust property and make it more general the first thing we should do is modify the algorithm to choose the structure elements for the morphological operators basing on characteristics of the vehicle (for example its length). This will give the final algorithm tolerance to future changes in the vehicles or will allow to include new kind of vehicles in the future so the identification system would be more versatile. Continue in this line we could improve the identification system to not discard vehicles that are too close to the mine walls. Maybe if the gradient pick cannot be found we could check the height matrix and if the starting value is quite high it should mean that the vehicle is really close to the wall. This is an idea that could work.

Talking about the second one we just can say that the possibilities of image processing in industry are huge. The corner detection and volume measurement can be used to identify different tools along a transport belt and make a robot to do different actions depending of the element it has in front of him. Other fields where volume measurement and density calculation is useful could be the recycling industry where separate different kind of materials is essential for the process.

Only the imagination is the limit and in an early future the computer vision and the processing image techniques will be part of the industry at all levels. As soon as we can make this kind of systems as precise as the human eye this kind of programs will be part of everyday life, helping us at home, driving to work or just in our relaxing favourite activities.

References

[1] Thurley, M. Fragmentation Size Measurement using 3D Surface Imaging, Lulea University of Technology, Lulea, Sweden

[2] Hough, P.V.C. Method and means for recognizing complex patterns, U.S. Patent 3,069,654, Dec. 18, 1962.

[3] Harris, C. & Stephens, M. A combined corner and edge detector, Plessey Research Roke Manor, United Kingdom, The Plessey Company pic. 1988

[4] Zorin, Denis & Barr, Alan H. Correction of geometric perceptual distortions in pictures, SIGGRAPH '95 Proceedings of the 22nd annual conference on Computer graphics and interactive techniques. ACM New York, NY, USA 1995

[5] R. Dougherty, E. & A. Lotufo, R. Hands-on Morphological Image Processing, Tutorial Texts in Optical Engineering Volume TT59, SPIE PRESS, Bellingham, Washington, USA

APPENDIX – A
Morphological Operators

All this notes have been taken from Dougherty and Lotufo [2]. For a deepest lecture consult the reference.

Chapter 1

- Erosion
  $A \ominus B = \{z \in E | B_z \subseteq A\}$
  $B_z = \{b + z | b \in B\}$ translation of b from vector z
  $A \ominus B = \bigcap_{b \in B} A_{-b}$

- Dilation
  $A \oplus B = \bigcup_{b \in B} A_b$
  Note that dilation is conmutative
  $A \oplus B = B \oplus A = \bigcup_{a \in A} B_a$
  $A \oplus B = \{z \in E | (B^s)_z \cap A \neq \varnothing\}$
  $B^s = \{x \in E | -x \in B\}$. simetry of B

- Algebraic Properties
  - They are associative $(A \oplus B) \oplus C = A \oplus (B \oplus C)$
  - $(A \ominus B) \ominus C = A \ominus (B \oplus C)$
  - They satisfy duality property $A \oplus B = (A^c \ominus B^s)^c$.

Chapeter 2

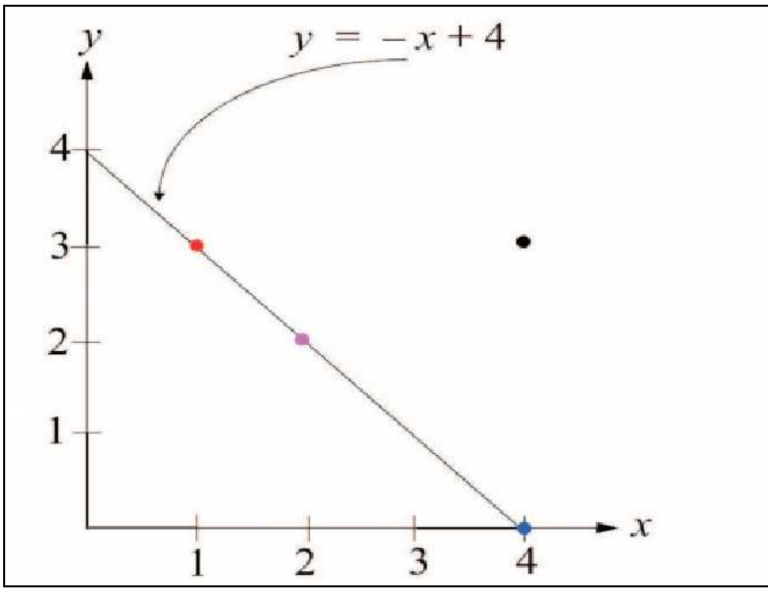
- Opening
  $A \circ B = (A \ominus B) \oplus B$.
  $A \circ B = \bigcup_{B_x \subseteq A} B_x$
- Closing
  $A \bullet B = (A \oplus B) \ominus B$
  $A \bullet B = (A^c \circ B^s)^c$
  $X^c = \{x \in E | x \notin X\}$ complement of X respect to E
- Algebraic Properties
  - $A \bullet B = (A^c \circ B^s)^c$.
  - $A \circ B \subseteq A$,
  - $A \subseteq A \bullet B$.

The chapter 5 about morphological operator in gray scale images is highly recommend if the lecture wants to get deeper into morrphological analysis.

APPENDIX – B Hough Method

Let's say we have an edge detected image. How do we put this into some practical use, for example automatically detecting shapes and object sizes?

The result of applying the Hough transform will be a set of parameterized lines. But how do we produce these lines? Let's start considering an easy example with three points in a line and a fourth that does not (figure 72).



y	x
3	1
2	2
0	4
3	4

Figure 73: Parameterized lines

We know the line parameterization: y = k*x + m and let's consider x and y as constants and k and m as variables. That can be considered as a transformation from (x,y) space to (k,m) space. Our points plotted in a (k,m) space will make an intersection in our line (figure 73).

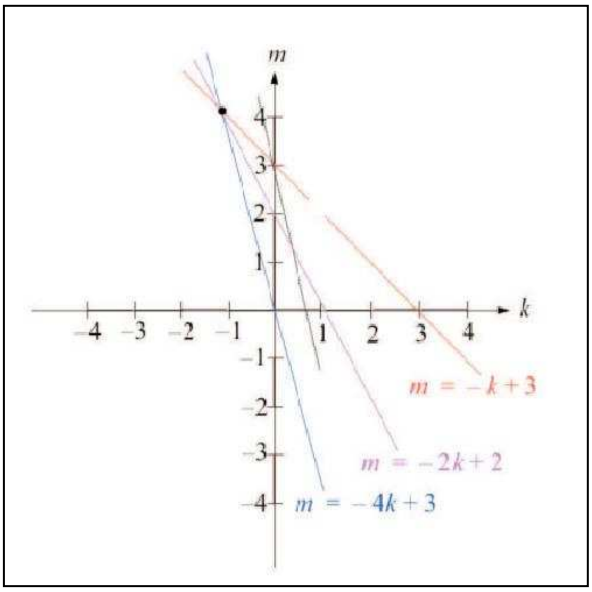


Figure 74: Points in (k,m) space

The problem is that the (k,m) space cannot be used because vertical lines have k = ∞ so we will use a polar system of coordinates. Now a line will be represented as a point (see figure 74).

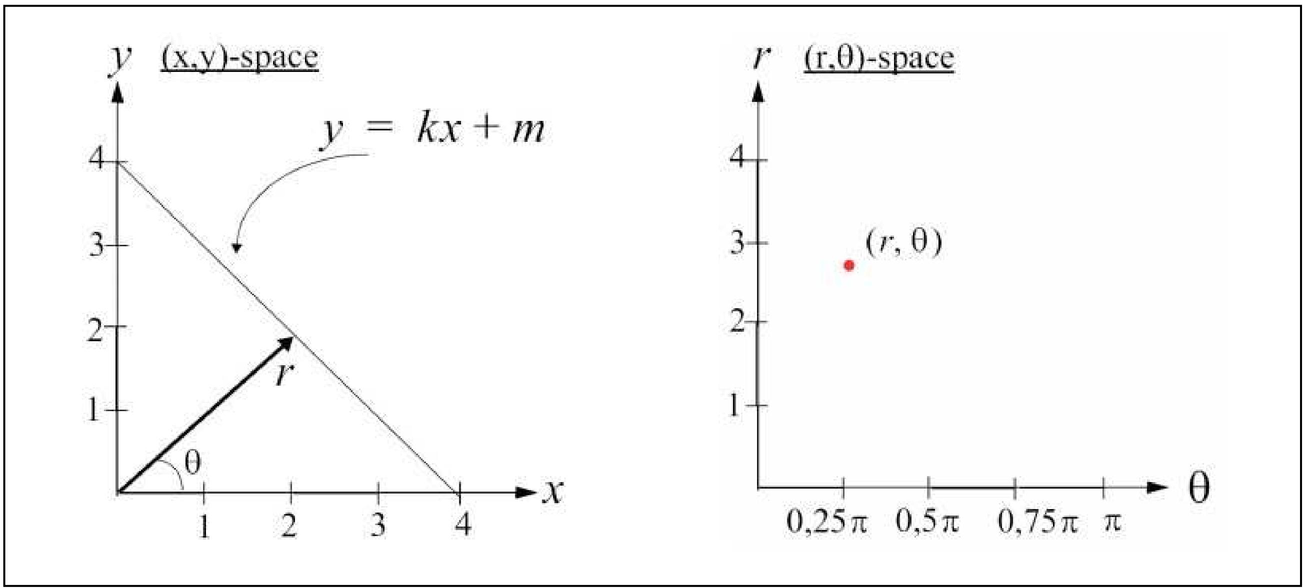


Figure 75: Transformation to a polar system of coordinates

Now we will see how we obtained a line using the Hough transform by the intersection of the sinusoidal forms of two points that belong to that line and their accumulation matrix (figure 75)

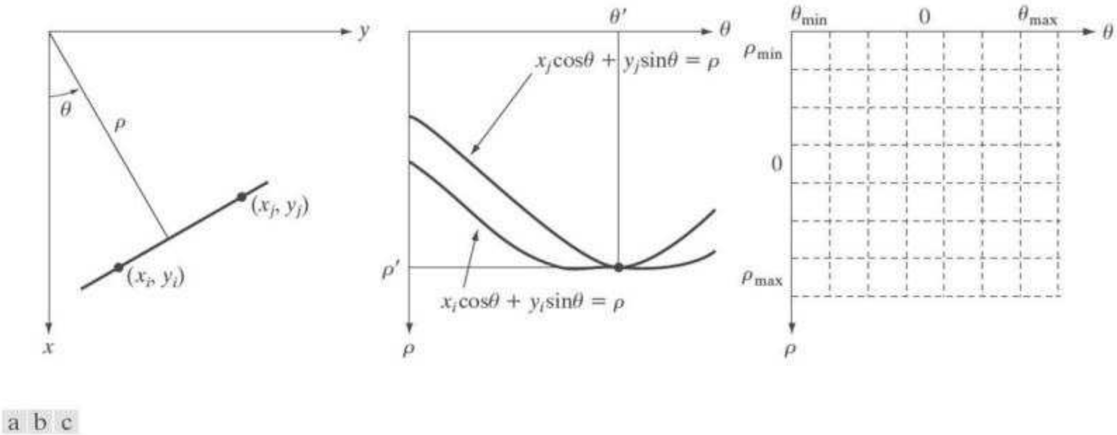


FIGURE 10.32 (a) (ρ, θ) parameterization of line in the xy-plane. (b) Sinusoidal curves in the $\rho\theta$-plane; the point of intersection (ρ', θ') corresponds to the line passing through points (x_i, y_i) and (x_j, y_j) in the xy-plane. (c) Division of the $\rho\theta$-plane into accumulator cells.

Figure 76: Parameterization of line, representation of points in polar coordinates and accumulation matrix

Let's explain this deeper: all the points in a line have the same (r,θ) parameters and for each point we have infinite lines that pass through it. Because of this we will discretize the (r, θ) space and for each point in the image we will compute a finite set of angles $\theta = \theta 1, \theta 2,\ldots,\theta n$ and for each θi we will calculate:

$$ri = x \cos \theta i + y \sin \theta i$$

The accumulator matrix is a matrix where each column corresponds to an angle θin and each rows corresponds to bins or intervals of the resulting distances r. Then for each point in the image we will compute its (r,θ) values and increase the correspond values in the accumulator matrix. In the end the higher values in the accumulator matrix will correspond to lines in the image.

The Hough transform can be generalized to find other shapes, e.g. Circles, but the computational complexity increases drastically.

APPENDIX – C Harris Method

Harris and Stephens improved upon Moravec's corner detector by considering the differential of the corner score with respect to direction directly, instead of using shifted patches. (This corner score is often referred to as autocorrelation, since the term is used in the paper in which this detector is described. However, the mathematics in the paper clearly indicates that the sum of squared differences is used.)

Without loss of generality, we will assume a grey scale 2-dimensional image is used. Let this image be given by I. Consider taking an image patch over the area (u,v) and shifting it by (x,y). The weighted sum of squared differences (SSD) between these two patches, denoted S, is given by:

$$S(x,y) = \sum_u \sum_v w(u,v) \; \left(I(u+x,v+y) - I(u,v)\right)^2$$

I(u+x,v+y) can be approximated by a Taylor expansion. Let Ix and Iy be the partial derivates of I, such that

$$I(u+x,v+y) \approx I(u,v) + I_x(u,v)x + I_y(u,v)y$$

This produces the approximation

$$S(x,y) \approx \sum_u \sum_v w(u,v) \; \left(I_x(u,v)x + I_y(u,v)y\right)^2,$$

Which can be written in matrix form:

$$S(x,y) \approx \begin{pmatrix} x & y \end{pmatrix} A \begin{pmatrix} x \\ y \end{pmatrix},$$

Where A is the structure tensor,

$$A = \sum_u \sum_v w(u,v) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} = \begin{bmatrix} \langle I_x^2 \rangle & \langle I_x I_y \rangle \\ \langle I_x I_y \rangle & \langle I_y^2 \rangle \end{bmatrix}$$

This matrix is a Harris matrix, and angle brackets denote averaging (i.e. summation over (u,v)). If a circular window (or circularly weighted window, such as a Gaussian) is used, then the response will be isotropic.

A corner (or in general an interest point) is characterized by a large variation of S in all directions of the vector (X Y). By analysing the eigenvalues of A, this characterization can be expressed in the following way: A should have two "large" eigenvalues for an interest point. Based on the magnitudes of the eigenvalues, the following inferences can be made based on this argument:

1. If $\lambda_1 \approx 0$ and $\lambda_2 \approx 0$ then this pixel (x,y) has no features of interest.
2. If $\lambda_1 \approx 0$ and λ2 has some large positive value, then an edge is found.
3. If λ1 and λ2 have large positive values, then a corner is found.

Harris and Stephens note that exact computation of the eigenvalues is computationally expensive, since it requires the computation of a square root, and instead suggest the following function Mc, where κ is a tuneable sensitivity parameter:

$$M_c = \lambda_1\lambda_2 - \kappa\,(\lambda_1 + \lambda_2)^2 = \det(A) - \kappa\,\mathrm{trace}^2(A)$$

Therefore, the algorithm does not have to actually compute the eigenvalue decomposition of the matrix A and instead it is sufficient to evaluate the determinant and trace of A to find corners, or rather interest points in general.

The value of κ has to be determined empirically, and in the literature values in the range 0.04 - 0.15 have been reported as feasible.

APPENDIX – E Geometric Transform

What we see in an image depends on the position of the camera and objects in the image may be affected by a noticeable perspective distortion. The transformation is a function that can convert an image from one shape into another.

Here is a general equation to convert an arbitrary four sided polygon to another arbitrary four sided polygon:

$$\hat{i} = a_1 ij + a_2 i + a_3 j + a_4$$

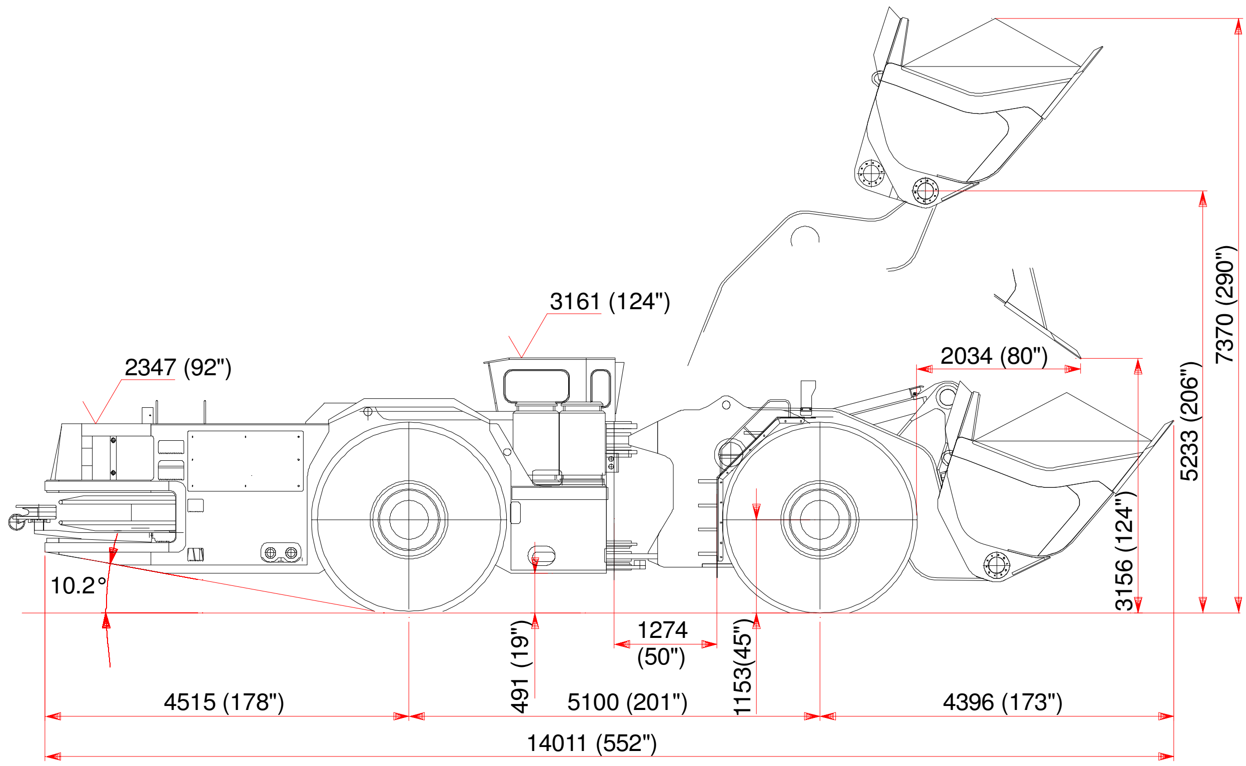
$$\hat{j} = b_1 ij + b_2 i + b_3 j + b_4$$

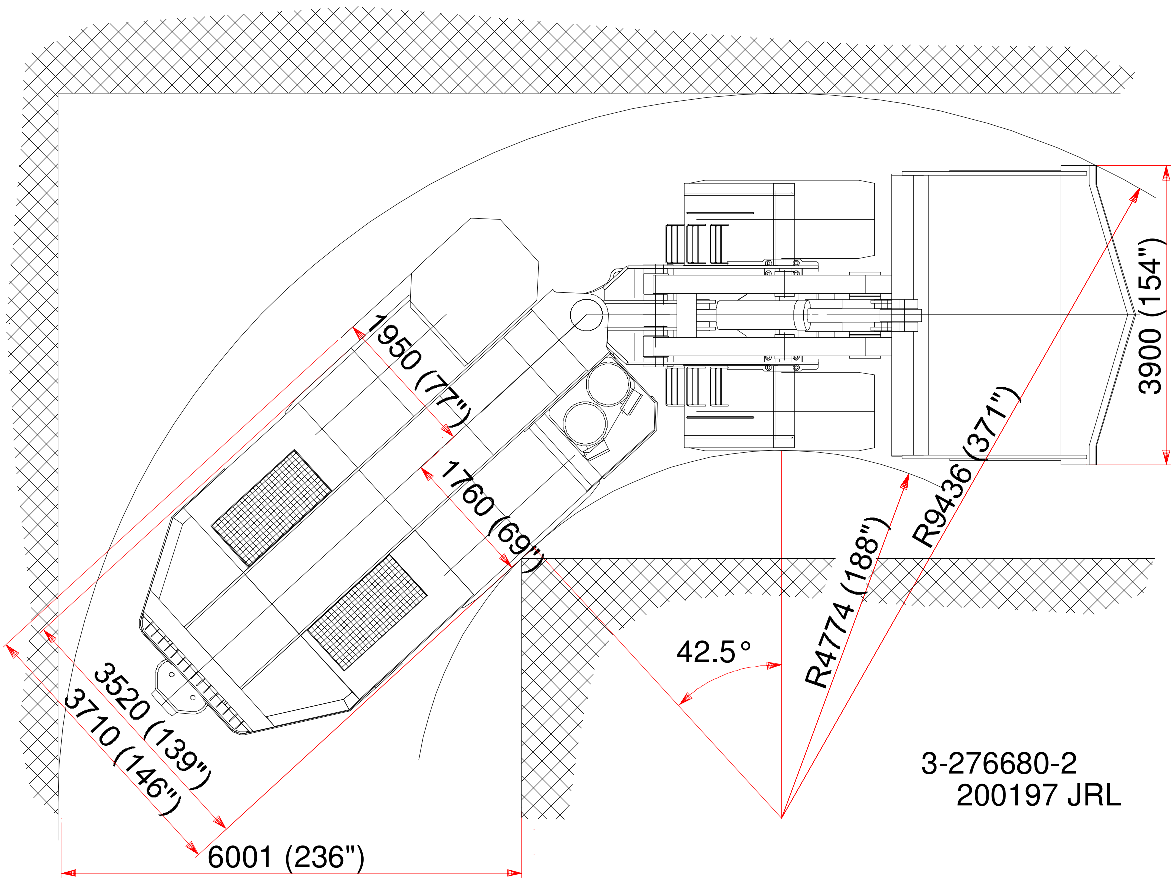
We apply this equation to our images to calculate the corresponding (î,^j) in the distorted image from any (i,j) in the undistorted image. This will be used to correct the distortion.

The general equation contains 8 unknown values which need to be define so we need to create 8 simultaneous equations where we know the values of (î,^j), (i,j). In order to solve this we define 4 points in the undistorted image and 4 matching points in the distorted image so we will have all the values we need. Substituting these values into the general equation we can get our 8 simultaneous equations.

In our case this points will be the bucket's corners of an image of the empty bucket and an image of a full bucket. Applying this transform will result in an alignment of the buckets.

APPENDIX – F Vehicle Schematic





Note: Dimensions subject to bucket options

Sandvik Mining and Construction Oy
Loaders Turku
P.O. Box 434
20101 Turku, Finland
Tel. +358 205 44 131
Fax +358 205 44 130
e-mail: toro@sandvik.com
Internet: www.smc.sandvik.com

Main dimensions

Total length	14 011 mm (552")
Width without bucket	3 710 mm (146")
Maximum width	3 900 mm (154")
Height with safety cabin	3 161 mm (124")
Height without cabin:	
rocker arm	2 845 mm (112")
frame	2 660 mm (105")
motors	2 570 mm (101")

Weights

Operating weight	77 500 kg (170 900 lb)
Total loaded weight	101 600 kg (224 000 lb)
Shipping weight	77 500 kg (170 900 lb)
Axle weights without load:	
front axle	36 650 kg (80 800 lb)
rear axle	39 950 kg (88 100 lb)
Axle weights with load:	
front axle	74 900 kg (165 100 lb)
rear axle	26 700 kg (58 900 lb)

Unit weight is dependent on the selected options

Capacities

Tramming capacity	25 000 kg (55 100 lb)
Breakout force, lift	540 kN (55 000 kg) (121 400 lb)
Breakout force, tilt	520 kN (53 000 kg) (116 900 lb)
Tipping load	57 000 kg (125 700 lb)
Bucket std.	10m³ (13 yd³), HB500/400

Bucket motion times

Raising time	9,0 sec.
Lowering time	6,2 sec.
Tipping time	2,5 sec.

Driving speeds forward and reverse

1st gear	3,6 km/h (2,2 mph)
2nd gear	6,3 km/h (3,9 mph)
3rd gear	10,5 km/h (6,5 mph)
4th gear	16,0 km/h (9,9 mph)

Frame

Rear and front frame	Welded steel construction
Material	Raex Multisteel N (St/Fe 355)
Central hinge	Adjustable upper bearing
Material	Raex Multisteel N (St/Fe 355)

Tanks are welded to frame.

Standard converter & transmission

Dana, C 16852

Weight	470 kg (1036 lb)
Gear shifter	Electrical with push buttons and interlock protection

Standard axles

Front axle	Dana, 25D 8860, fixed
Rear axle	Dana, 25D 8860, rear axle oscillating ± 8°
Dry weight (approx.)	5 500 kg (12 125 lb)

Standard tyres

Tyre size	40/65-39 D-LUG L5 56 ply (brand and type of the tyres subject to availability)
Weight (with rim)	2 500 kg (5 512 lb)
Air pressure, front	730 kPa (7.3 bar) (105 psi)
Air pressure, rear	500 kPa (5.0 bar) (73 psi)

Other type of tyres available to user's choice. In certain applications the productive capabilities of the loader may exceed the TKPH value given by tyre manufacturer.
Sandvik Mining and Construction Oy recommends that the user consult their tyre supplier to evaluate conditions and to find the best solution for application.

Standard motor

Drive motor:	Siemens 1 LA8 317, Squirrel cage motor
Power	315 kW
Voltage	1000 V
Frequency	50 Hz
Speed of rotation	1500 rpm
Isolation klass	F
Isolation	IP 55
Weight	1500 kg
Pump motors:	2pcs, Siemens 1 LA6 280, Squirrel cage motor
Power	75 kW
Voltage	1000 V
Frequency	50 Hz
Speed of rotation	1500 rpm
Isolation class	F
Isolation	IP 55
Weight	610 kg
Fan motors	3pcs, VEM KPEP100L4, Squirrel cage motor
Power	2,2 kW
Voltage	400 V
Frequency	50 Hz
Speed of rotation	1500 rpm
Isolation class	F
Isolation	IP 55

Cabin

Protection rating	ROPS & FOPS
Seat	Swinging
Air conditioning	

Sandvik Mining and Construction Oy
Loaders Turku
P.O. Box 434
20101 Turku, Finland
Tel. +358 205 44 131
Fax +358 205 44 130
e-mail: toro@sandvik.com
Internet: www.smc.sandvik.com

Steering hydraulics

Electrically controlled hydraulic, centre-point articulation, power steering with two double acting cylinders. Steering controlled by stick. Interlock protection. Emergency steering is optional.

Turning angle	±42,5°
Turning radius (with std bucket):	
Right	inner 4774 mm (188") outer 9436 mm (371")
Left	inner 4760 mm (187") outer 9293 mm (366")
Main components in steering system:	
Main valve	Rexroth
Servo control valve	Rexroth
Steering cylinders	Tamrock
Steering lever	Gessmann
Steering and servo hydraulic pumps	piston type, Rexroth
Pressure accumulators for emergency steering (optional)	3pcs, Bosch
Pressure settings:	
Steering hydraulics,main relief valve	25,0 MPa(250bar)
Shock load valves	12,5 MPa (125 bar)
Pump	12,0 MPa (120 bar)
Pre charge pressure for pressure accumulators	6,0 MPa (60 bar)

Bucket hydraulics

The bucket hydraulics has two pumps. One is for the servo circuit and other delivers oil to the bucket hydraulic main valve. The oil flow from steering hydraulic pump is directed to bucket hydraulics when steering is not used. The servo system is electrically controlled. Adjustable-displacement piston pump. Boom suspension system (optional) with two pressure accumulators.

Main components:	
Boom system	z-link
Lift cylinder	2pcs, ø250 mm, Tamrock
Tilt cylinder	1pcs, ø320 mm, Tamrock
Servo control valve	
Pump	piston type, Rexroth
Servo pump	gear type, Commercial
Control lever	Gessmann
Main valve	Rexroth
Return filter	FinnFilter
Pressure filter	Fairey
Fittings	ORFS
Pressure setting for:	
Servo circuit	3,5 MPa (35 bar)
Bucket hydraulics	25,0 MPa (250 bar)
Shock load valves	25,0 MPa (250 bar)
Pump	24,0 MPa (240 bar)
Pre-charge pressure for pressure accumulators	6,0 MPa (60 bar)
Hydraulic oil tank capacity	appr. 800 l (211 gal)

Standard brakes

SERVICE BRAKES

Service brakes are hydraulically operated and liquid cooled (LCB) multi-disc brakes on all wheels, two separate circuits for the front and rear axle.

PARKING BRAKE

Spring applied multi-disc liquid cooled brake on the input shaft of the front axle. Brake is released with oil pressure from the hydraulic system. The oil flow is controlled with a lever in the operator's cabin.

The parking brake engages automatically if transmission oil pressure is too low or the electric current is cut off.

EMERGENCY BRAKE

The emergency brake uses the same brakes as the service brake and is controlled with a parking brake lever in the cabin.

Main components in the brake system:	
Pressure accumulator	Bosch
Brake pedal valve	Rexroth
Parking brake	Tamrock / Finngear
Charging valve	Rexroth
Cooling circulation oil filter	FinnFilter

Standard Lubrication system

Automatic central lubrication system

Electrical equipment

Cable as an option	
Cable reeling	Logic controlled
Driving, parking and working lights	6 pcs 24 V 70W H1 4 pcs Megalight
Electrical gauges	
Back up alarm	light and buzzer

Others

Decal language	EU-languages

Standard manuals

Instructions manual	(1pc)	
- General mandatory safety instructions		EU-languages
- Start up of a new machine		English
- Operator's manual		EU-languages
- Maintenance manual		EU-languages
Spare part manual	(1pc)	English
Workshop manual	(1pc)	English
ToolMan CD	(2pcs)	Manuals in pdf form
- General mandatory safety instructions		
- Operator's manual		
- Maintenance manual		
- Spare part manual		

Sandvik Mining and Construction Oy
Loaders Turku
P.O. Box 434
20101 Turku, Finland
Tel. +358 205 44 131
Fax +358 205 44 130
e-mail: toro@sandvik.com
Internet: www.smc.sandvik.com

Main options:

* replaces standard equipment

- Round cable, 4x120 mm, 350 m (380 yd).
- VICTOR plugs for cable.
- Cable anchor.
- Cable shock absorber.
- Power supply box.
- Reactive power compensator.
- Round cable, 4x95 mm^2, 350 m (380 yd) (can be used with rpc only).
- Cameras front and rear & monitor.
- Cassette player.
- Ride control system for boom.
- Bucket counter, electrical.
- Emergency steering (CEN).
- Fire suppression system, double ANSUL (CEN).
- Accordance with CE- norms (CEN).
- Fire extinguisher 12kg (CEN).
- TORO RRC, complete.
- RRC interface (TORO std.).
- Lockable main switch
- Disassembly needed shaft dim: TBA.
- CatBase/LinkOne spare part manuals and additional Instructions, Workshop, Spare part manuals, ToolMan CD's are available.

Sandvik Mining and Construction Oy
Loaders Turku
P.O. Box 434
20101 Turku, Finland
Tel. +358 205 44 131
Fax +358 205 44 130
e-mail: toro@sandvik.com
Internet: www.smc.sandvik.com

APPENDIX – G Tools

For the development of this application we have use:

- Dropbox 1.1.40 for the version control
- Matlab 2010 in order to develop the program. We have implemented the functions for each step in different folders and one main program who calls them. All the inputs are in one folder and all the outputs generate during the execution will be saved in other different one.
- The computer used is a laptop ACER ASPIRE 5542G.

APPENDIX – H Time diagram