Colour Dot Matrix Printer with leJOS NXT

Riley Davidson , Graham Burgsma

***Abstract*—Building a colour dot-matrix printer with leJOS without utilizing dithering is possible, but the resolution and quality will not be perfect.**

***Index Terms*—leJOS, NXT, Lego Mindstorm, Dot-matrix printer**

I. Introduction

The goal of this project is to create a colour printer with the leJOS NXT which is a java based Lego Mindstorm. Utilizing the parts given in the starts Lego Mindstorm NXT package, a dot matrix printer will be created. We will also be utilizing the leJOS software which allows for the Mindstorm to be programmed using Java. As well as being a dot matrix printer, sounds should also be played by the printer in order to prompt the user to change the marker colour.

A. Problem Definition

The problem that will be investigated is that of using the leJOS software to create a dot-matrix printers. The goal of this project is to create the printer without using a standard dithering techniques that can be found online. There are several complications with this problem, such as how to efficiently use multiple colours.

B. Dot Matrix Printing

A dot matrix printer is a device which creates an image out of many of dots, that when viewed together, forms an overall image. Each print element seen is represented by 1 bit of data which is stored in matrix form. Therefore an overall image can be drawn by simply printing dots wherever the matrix specifies [1].

C. Why leJOS?

The default programming language for the Mindstorms is created for basic use in mind, and Java is a much more common programming language. We are also very familiar with java, and decided to use a language that we were comfortable with. This project also required advanced image processing that could not be done using the Mindstorm default language.

II. Evolution of the printer

Since this was our first time working with the leJOS software as well and the Mindstorm kit, multiple iterations of the printer were created. At each step we identified the goals of the version, and then identified the issues with that version and what we had to improve on. While this process could go on forever, we had the end point in mind being a function colour dot-matrix printer which also played sounds.

For all of the versions we decided to use one picture to test how well we are improving at each version. We decided on superman because he has a good blend of colours.

Fig. 1: Source image of superman

Version 1

Goals of the First Version:

1) Follows steps in the book to create the base printer model.
2) Install the leJOS OS on the LEGO Mindstorm to allow java programs to run on the Mindstorm.
3) Create 3 separate programs in java
   - One to do edge detection and scale the image to a size that is printable on the paper. It then converted the image to a 2D matrix which indicated where the edges of the image was.
   - One to run on the PC and send the image data to the Mindstorm via USB
   - One to be loaded onto the Mindstorm and handle the moving of all the robot parts
4) Draw a basic black dot image

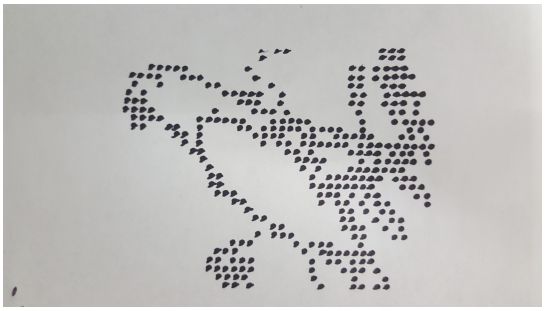

Fig. 2: First version output of a parrot sitting on a log

Problems with First Version:

1) When using the motor to move the printer arm horizontally, it wouldn't be very accurate when moving very short distances. This is because there was only one wheel

which moved the horizontal arm, which meant that for short distances, only one side of the arm would properly move forward, so it would be on an angle rather than moving forward horizontally.
2) Using 3 separate programs is not ideal

Version 2

Goals of the Second Version:

1) Combine 3 programs together for ease of use
   - This will be done by using a different library for scaling the picture size which will enable us to combine the two PC programs into one
2) Include different colours
   - This will be accomplished by having a colour palette and calculating which colour each pixel is closest to in our palette. This is explained in more detail in the details section.
3) Include a border
   - This will be accomplished by placing the pen down on the page, then moving the printer arm, followed by rolling the paper through the printer while the pen is still down.

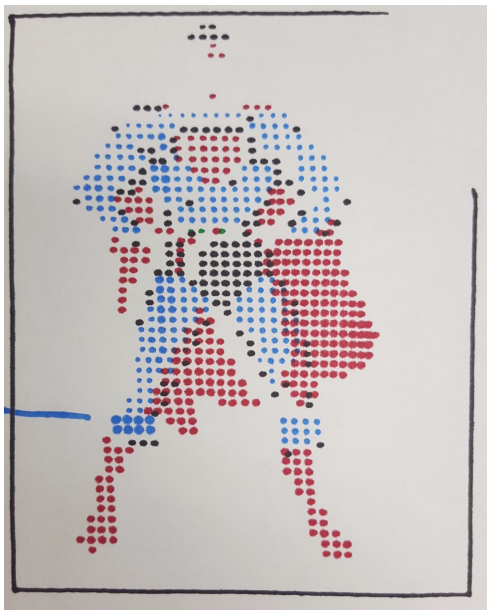

Fig. 3: Second version output of superman

Problems with Second Version:

1) Dot size is not consistent. Some dots seem bigger than others. This is also the cause for the blue line in the superman drawing.
2) The page is not calibrated properly for different colours. As seen by the black dots on superman's cape, there was some overlap of the colours. This is because the printer did not reset the paper 100% properly, so for each pass the colours didn't line up
3) Border sometimes doesn't touch the page when drawing.
4) Printer resolution is not very good

Version 3

Goals of the Third version:

1) Make Dot sizes consistent
   - This will be accomplished by calibrating how much force the pen should use when drawing on the page. The first thing the printer does is slowly move the pen arm down, and when the user presses the reset button on the printer arm, it will save the position that the pen arm is in, thus allowing the pen to use the same amount of force each time.
2) Make different colours draw in the proper spots
   - The page needs to recalibrate after every colour has finished drawing. We will start by lowering the speed of returning the paper to its starting position. Second we need to ensure that the arm is stopped at the same place each time when it resets. To do this we need to make sure the button sensor is fully pushed in when the arm resets.
3) Fix drawing of border
   - Need to find the proper amount of pressure to apply to the page so the pen stays down, but it doesn't stop the paper from moving.
4) Increase resolution
   - The problem originally with the larger resolution was that it was too memory intensive to send the whole matrix representing the image to the NXT at once. In order to draw a higher resolution matrix, we needed to send the matrix line by line to the NXT

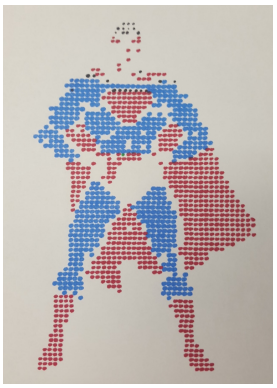

Fig. 4: Third version output of superman

Problems with Third Version:

1) The printer died when trying to complete the black colour pass. Unsure of the reason, but it seems that the batteries died during execution due to old batteries

Version 4

Goals of the Fourth Version:

1) Make it so the printer doesn't die
   - Remove unneeded motor movement (Such as resetting the pen after each dot, and instead reset the pen once per row)

2) Include sounds which prompt the user to change pen colours
   - Needed to upload each colour to the NXT brick before hand. The NXT brick also has limited space, so we had to make the sound files as small as possible.

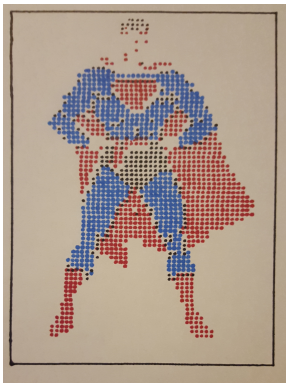

Fig. 5: Fourth version output of superman

Problems with Fourth Version:

1) Canny edge detection is not the best
2) Doesn't print Superman's belt, recognizes it as white as white is the closest colour to yellow

Version 5

Goals of the Fifth Version:

1) Change the edge detection to use Sobel instead of Canny
   - Canny was a third party library. Wrote our own Sobel edge detection.
2) Add yellow to the colour palette
   - Yellow is a distinct colour that doesn't closely match any other color in the palette, should add more detail to our images

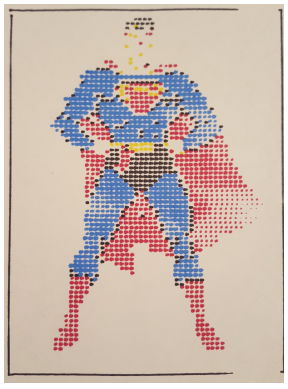

Fig. 6: Fifth version output of superman

Problems with Fifth Version:

1) Does not print Superman's face
   - This is due to the goal of the printer to simply print multiple colours, rather than focusing on the detail of the face. The closest colour to the skin is white, so it doesn't print the face.

2) Not perfectly aligned

   - For each colour used, the alignment goes off a tiny bit. This means the more colours used, the more it will be off alignment. This is simply a limitation of the hardware as we weren't able to make this any better.

For the final version, we also included a photo of something other than superman to show that our printer can print more than just superman, this is shown in Figure 8. We stuck with the superhero theme still.

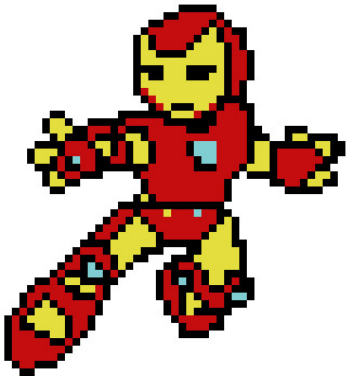

Fig. 7: Ironman source image

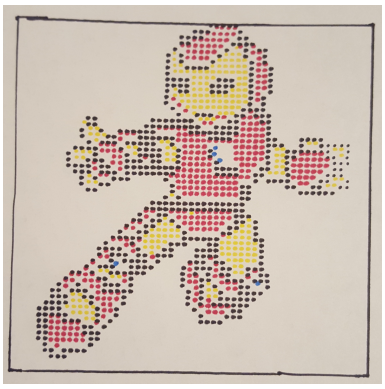

Fig. 8: Fifth generation output of ironman

III. Physical Printer

Part	Use
Motor A	Moves the printer arm horizontally to get in in the proper position on the page
Motor B	Controls moving the pen up and down
Motor C	Moves the paper up after the printer has finished drawing on the current line
Colour Sensor	Used to tell when a piece of paper has been loaded in, and the machine is ready to print.
Bump Sensor	Used to tell when the printers arm has moved back to its starting position

A. Physical Design

The basic printer was created by following a guide. There were multiple problems with this base model which we had to correct.

Keeping the paper down while drawing: We had to create a mechanism to keep the paper down whole drawing. When we first tried to draw dots, the paper would sometimes come off of the paper track or get stuck ot the marker. We created the following peice to hold the paper down.

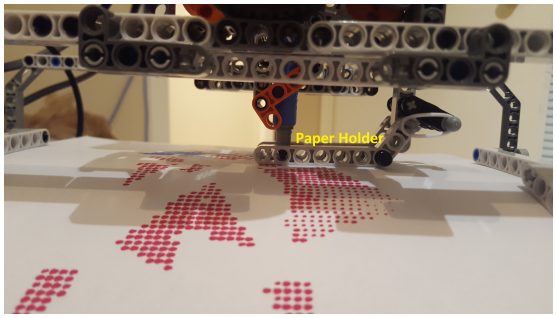

Fig. 9: Paper holder holding the paper down

In Figure 9 we can see the piece holding the paper down.

In order to allow the paper to slide into the tray, the piece needs to lift up to allow the paper to enter, as shown in Figure 10.

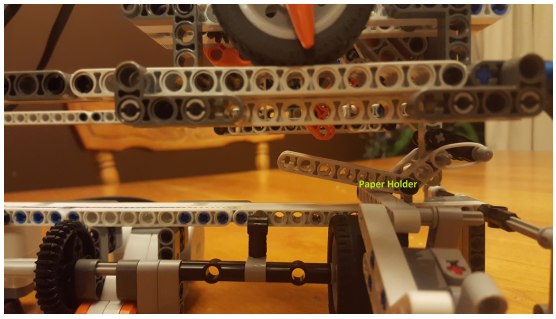

Fig. 10: Paper holder up to allow paper to feed in

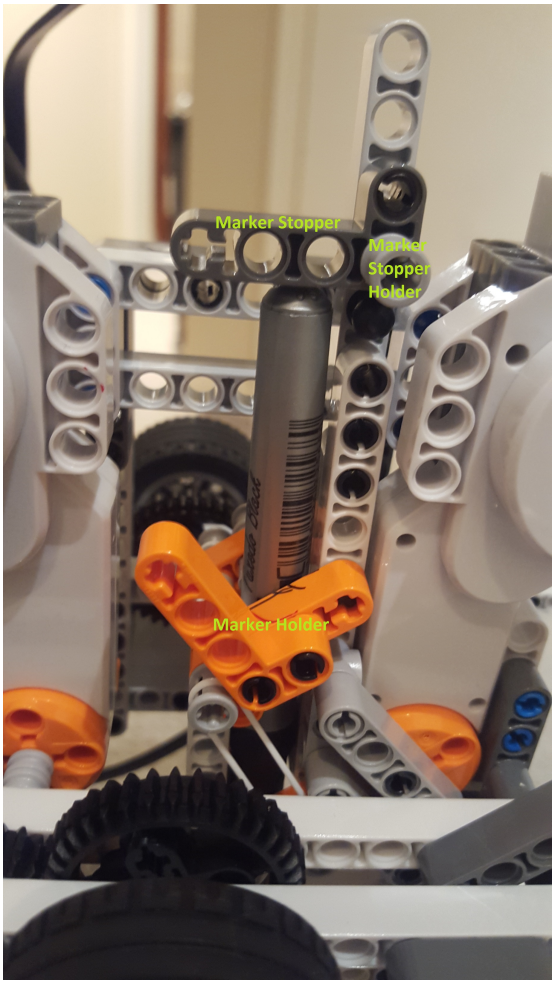

Fig. 11: The marker holding apparatus

Marker holder: In order to allow for multiple different colours, we needed to come up with a simple way to switch markers. We did this by adding the 'Marker Stopper' which held the marker in place. We made the 'Marker Stopper Holder' which when removed, allows you to remove the 'Marker Stopper' and replace the marker. Once the marker is replace, you simply put the 'Marker Stopper' back in position, and insert the 'Marker holder stopper'

Extra wheel: We extended the size of the printer to include another wheel on the opposite wheel track. The base version only had a wheel on one of the tracks, which made it not ideal for moving the printer arm short distance forward. We decided to extend the whole base of the printer in order to accommodate for the extra wheel on the other printer track. This can be seen in Figure 14.

B. Motors

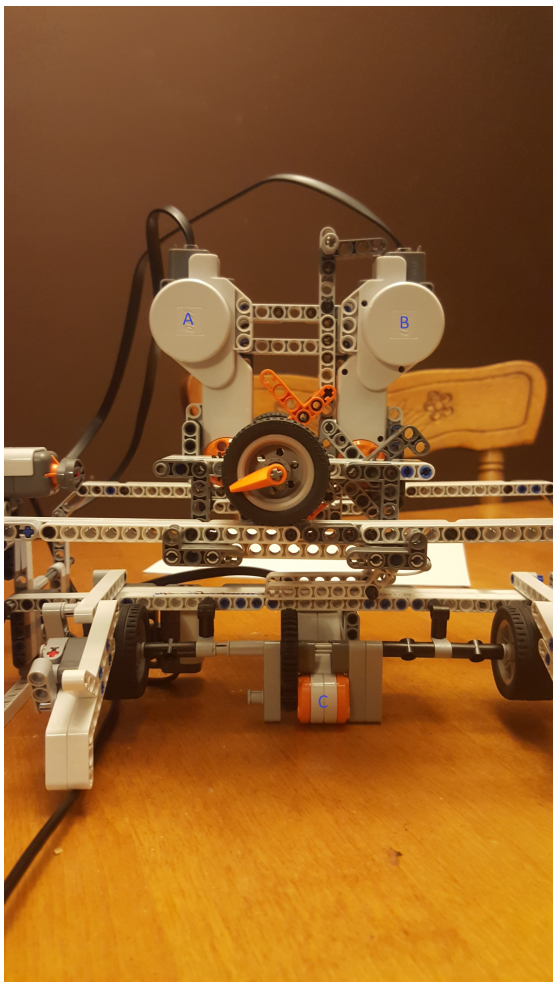

Fig. 12: Visual of the different motors

There are 3 motors that are used for this printer as shown in Figure. 12

1) This motor handles moving the printer arm along the horizontal track. It does this by turning two wheels which are attached to a track, which allows for the arm to move forward and backward
2) Handles the movement of the pen/marker up and down. This handles drawing the actual dots on the page
3) Moves the paper vertically. This allows the paper to be moved down so that the printer can print vertically on the page as well.

C. Sensors

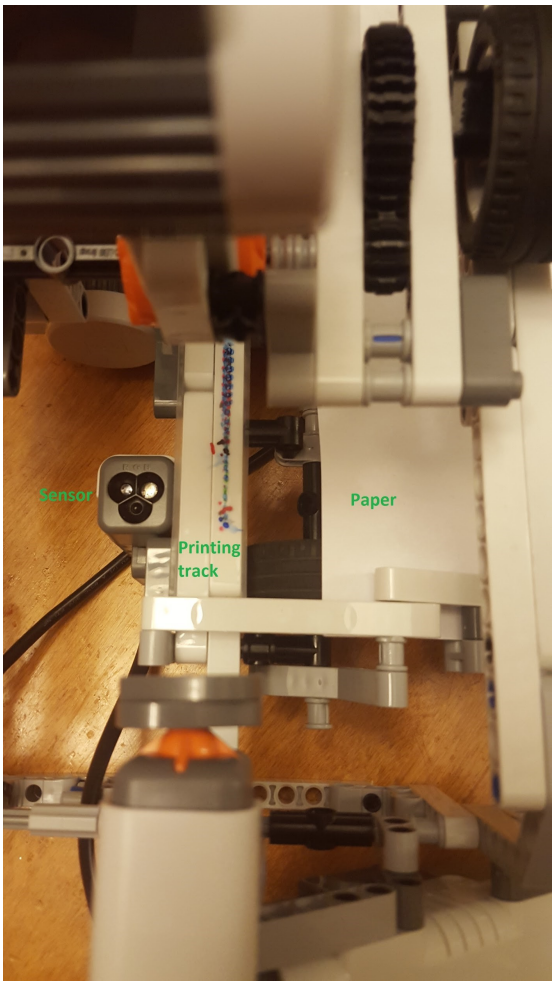

Fig. 13: Visual of the colour sensor

Colour Sensor: The following relates to Figure. 13

1) The colour sensor is used to determine the vertical positioning of the paper. This allows us to know when paper is present on the printing track. The motor which rolls the paper will keep rolling forward until it sees white, which means paper is on the printing track, and it can start printing.
2) The printing track is where the pen/marker draws dots on the page to make up the image.
3) The paper is the to be drawn on. In the image the paper hasn't been loaded onto the printing track yet, so the printer will not print.

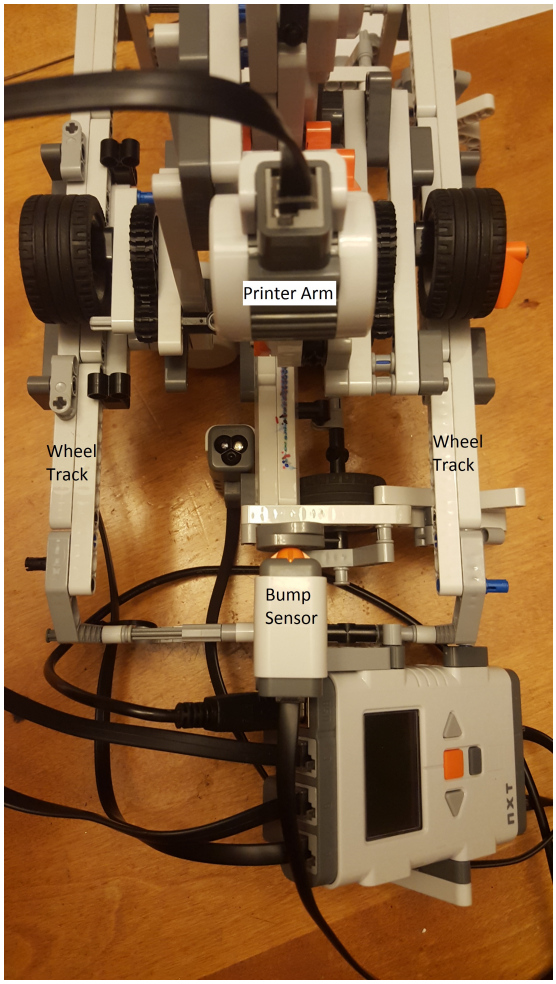

Fig. 14: Visual of the bump sensor

Bump Sensor: The following relates to Figure. 14

1) The printer arm moves back in forth in order to handle printing horizontally along the page. It travels along the wheel track using a motor which controls the wheels on the wheel tracks.
2) The bump sensor will trigger whenever the printer arm moves all the way to the left side. The bump sensor allows us to calibrate where the start of the image should be, as we know when the button is pressed that the printer arm is reset to the starting position

IV. Java programs

A. PC Program

The program that runs on the PC has two main functions: image processing and data transfer to the NXT.

The Image processing takes any image in jpg format as input. It then uses sobel edge detection to determine the edges in the image. The result is a black and white image. This image is then scaled down to the printable size, the max resolution for the width of the printable image is 54 dots. Since the image is scaled down so small, a lot of detail is lost. To counter this, scale area averaging is used which preserves detail in the image by taking the average from the surrounding pixels. Since the resulting image is black and white, the original image is duplicated and scaled down to the printable size. To create the print matrix, the edge detected image is traversed and wherever there is a colour value it gets the colour value from the original scaled down image. This colour value is then compared to the colour palette (white, black, red, green, yellow, blue) and the one with the closest euclidean distance is entered into the print matrix.

Once the print matrix has been created, it must be sent to the matrix to the NXT. Since the NXT has such limited memory it cannot store a large matrix without running out of memory. To fix this problem the print matrix is sent row by row. To achieve this, but the PC and NXT need to communicate with an input and output stream over USB. Since reading an input with a data input stream is blocking, it can be used to notify back and forth when to send the data. The PC program first waits to read a boolean, then it writes the height and width of the print matrix then waits for another boolean. Once received the program sends a row of the print matrix via output stream, flushes the stream and waits until the NXT is ready to receive another row.

B. NXT Program

The NXT program runs separately from the PC program and is responsible for reading the print matrix and then moving the motors accordingly to print the image.

1) Moves the slider to the far right so the paper can easily be added
2) Calibrates the marker to the top position.
3) Feeds in the paper
4) Opens input and output streams
5) Moves slider to the middle and asks for the first marker (audible and on display)
6) Reads print matrix dimensions
7) Reads matrix row
8) Print matrix row
9) Increment paper
10) Repeat 7 to 9 for each row in the matrix
11) Reverse paper to start
12) Repeat 5 to 11 for each colour
13) Draw border around printed image
14) Eject paper

V. Special Outputs

We attempted to make our printer as real as possible, and as such made it as annoying as possible to use. We also tried to mimic a real printers ability to waste all of your ink and your time producing an unusable image, which is where these outputs came from.

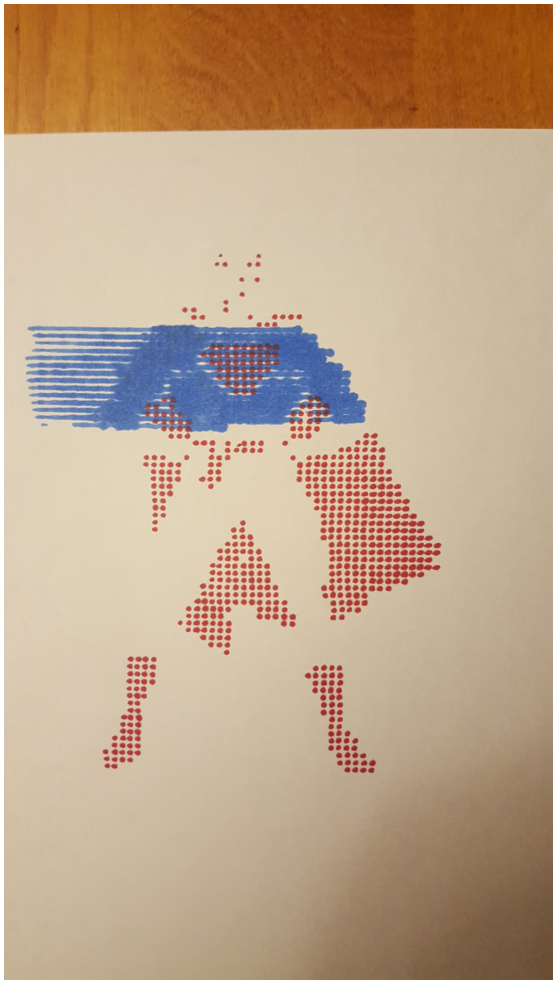

Fig. 15: Superman has a blue cape also

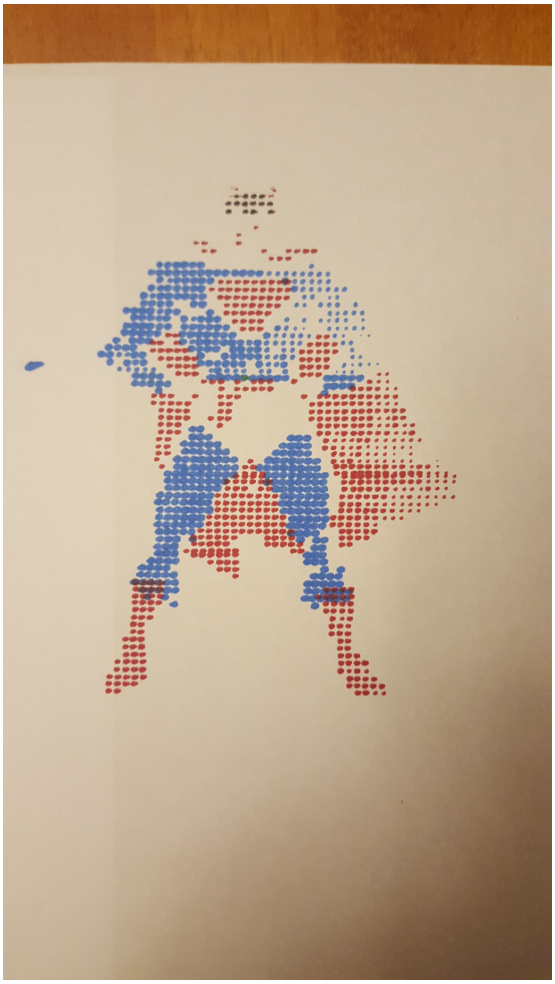

Fig. 16: Supermans right side is his bad side

As shown in Figure. 15 you can see why it is very important to ensure that when the marker is inserted into the holder, that is set at the right height. In this case, it performed properly on the red pass, but when the marker was changed to the blue colour it had trouble. The blue marker wasn't placed as high up in the holder, and as such when it tried to lift up off the page, it actually stayed down and left an interesting trail.

What happened in Figure. 16 is that the pen wasn't being reset properly after each dot was drawn. Since the printer draws left to right, we can see that the dots on the left are darker than the ones on the right. Also there are some dots that werent even drawn on the right side of the image. This was an issue for some time, as it was hard to reset the pen to the correct height, but if you reset it every time it would take a very long time to create an image. Through lots of tinkering and various changes, we were able to get the marker holder to reset to the same position quickly after each dot.

VI. Moving forward/Additional Research

A. *Dithering*

Dithering techniques are usually used when creating a dot matrix printer. This is because with dithering you can convert a regular image to one comprised of dots very easily. Dithering can convert a greyscale image to black and white, such that the density of the black dots in the created image closely resembles the average grey level of the individual. A dithered image is shown in Figure. 17.

Fig. 17: A dithered image

B. Why didn't we utilize dithering?

The goal of this project was to create a coloured dot matrix printer, which is not as easy to do with dithering techniques. While it is still achievable, we wanted to come up with our own solution to create the coloured dot matrix printer.

Instead of using an already created dithering technique (Such as Floyd-Steinberg dithering[2]), we tried to utilize edge detection and scale area averaging to preserve detail in the image.

C. Next Steps

In order to make our printer more reliable, we need to find a way to utilize smaller markers that don't bleed as much. We specifically designed our robot around the use of the 'fine tipped" markers, but found that these bled to much and could only handle 54 dots on a single row. This made it hard to make an accurate image.

We also need to have a better way of determining when the paper has returned to its original position after each colour has finished its pass. Since we could not get the rotations 100% accurate, it caused some overlap on the different colours. One way we could do this is by adding a black line or square to the underside of the paper, thus creating a marker where the starting position is.

We could also try to utilize dithering techniques as described above, but it would require additional research into multi-coloured dithering.

REFERENCES

[1] T. Kawanabe. Dot matrix printer, September 15 1998. US Patent 5,806,997.

[2] L. Steinberg R.W. Floyd. An adaptive algorithm for spatial grey scale. *Proceedings of the Society of Information Display*, 17:75–77, 1976.