Project report: Edge Detecting Image Copier

Hongrui Wu and Eric Fisher

Motivation and Overall Concept

In one of my previous classes at UCSD I, Hongrui , learned that the one of the process the brain performs is edge detection. The human mind prioritizes detecting the places in an image where the intensity or color of the image changes dramatically, the edges. When we draw pictures we prioritize drawing the lines which is another term for the edges. When viewing the lines we easily identify the subject of the image despite none of these lines appearing in the real world scene. I say that they are not there, but they are in the sense that they are the gradient of the image.

We would like to make a copying machine that, for the goals we will propose basic goals, uses one pin. Despite only being able to deposit two colors, light and dark, we want the image to be easy to interpret and visually impressive. To do that we want to draw an edge image of the image being copied.

In brief, we will build a machine that will scan an image, and determine light levels at points across it at a fixed resolution. We will write software that performs a Canny edge detection algorithm on the image. The machine will draw the edge image on a piece of paper.

Functional Definition

We will make a mechanical device that can suspend a dolly over two dimensional image, and this dolly will contain a light sensor to measure the darkness of the shades of the image and a pin. The sensor should be held just above the image and should be able to move across the breadth of the entire image. The pin should also move across the breadth of a field the same size as the image. It needs a mechanism that raises it from the paper and lowers it to the paper.

Within the software of the microcontroller, we will declare a two dimensional array in which the controller will store information about the darkness of the image. To achieve knowledge of the image a high resolution, the hardware must move the sensor small and accurate distances. We will attempt this using a belt, but if that does not work, we could try a rack and pinion . We will drive each sprocket or pinion with a stepper motor. Four hundred steps per resolution, which is consistent with some stepper motors currently in the lab. It is not necessary to have an angular resolution from the stepper motor less that the gear backlash we can achieve.

We will construct a wooden frame from common lumber sizes such as 2 by 2 or 2 by 4 that is slightly larger than an 8.5 by 11 inch piece of paper, about 9.5 by 12 inches. The top of the frame should be two to three inches above the paper. Above the frame should be a beam that spans the

frame in the shorter direction, held to the frame by two slighting interfaces. The beam will be moved across the frame via two stepper motors in tandem. The reason for using two is to prevent one end of the beam lagging or the mechanics binding. The dolly will be attached to the beam by another sliding interface and will be actuated by a stepper motor. The pin will be attached to the dolly via two links. It will be lowered and raised by one stepper motor. A servo motor could be used instead, if a stepper motor is too heavy or uses too many pins, but, although a stepper motor uses fewer pins than a stepper motor, we find stepper motors easier to use than servo motors and our control will be more exact. If we use APDS9960 sensor, we will only have enough pins for a servo motor for the pin. The sensor will also be mounted on the dolly. The arduino will rest on a corner of the frame.

The software will perform the following procedure. If one of two interrupt signals received, the hardware will be told to move the sensor over the image in a systematic pattern, and an array will be filled with the light intensity values from the image. The software will then pad the image and convolve it with a Gaussian kernel. It will then convolve it with sibel operators in both directions, and calculate the magnitude and angle of the gradient of image, which will be two separate arrays. The software will use the angle information to perform non maximum suppression on the magnitude array to thin the lines. The thinned magnitude array will then be binarized with a threshold to form the edge image.

If a second interrupt signal is put into the arduino, the hardware will then put a dot on whatever is in the frame at every edge point in a systematic way. There are two separate interrupt pins because we may want to print the same image multiple times before scanning a second one.

Sensor

Option 1 :

We would like to use a Pololu-QTR-1RC reflectance sensor to detect the shade of the image. It has three pins, 5v, ground, and data. The output is a oscillating digital signal that has a shorter pulse with, the brighter the image. Pololu claims to have developed an arduino library for the sensor.

QTR-1RC output (yellow) when 1/8" above a black line and microcontroller timing of that output (blue).

Problem to be solved: The output format of the Pololu-QTR-1RC reflectance sensor is digital I/O-compatible signal that can be read as a timed high pulse. We need to measure the time for the pulse to decay and do some math to convert the decay time to the relative color value we want. To do this, our programming need to convert digital signal to the frequency that made it up by Fourier transform and then we are able to track the decay time. However, this kind of pulse occurs at each dot on the sample photo which means we run our transform code for all individual digital signal. Although the sensor will obtain the meaningful result within 1ms for a single dot, consider our

8.5 by 11 inch sample photo which contains a huge amount dots, we can't control the amount of time it need to scan and transform all the signal to the value we want, not to mention that coding complicated Fourier transform in C is a big challenge .

Figure.1 the digital

Option 2:

We are going to use APDS9960 Proximity, Light, RGB, and Gesture Sensor. Since there are no such sensor that can detect the white and black color value directly, we would like to use a RGB sensor to detect the shape of image. The light sensor works by shining a white light at an object and then recording the reflected color. Since our sample photo is black and white, the RGB values that sensor returns are same, either (255,255,255) or (0,0,0). Therefore, we can only use one of them in the code, which is much easier than using a three element array to record and memorize the RGB value.There is also an arduino library for this sensor.

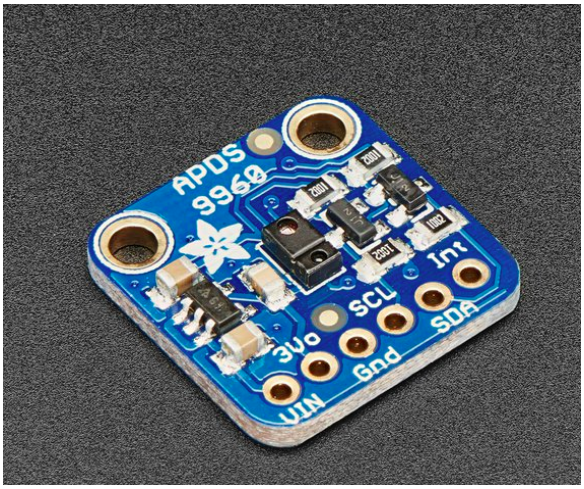


Figure.2 the RGB

Problem to be solved: As metnioned above, the light sensor shining a white light when it detect the color. Notice that the white light will diverge after shooting from the light source. The divergence light will cause more reflect light for a single dot, resulting the color value it return be more inaccurate. To solve this, we plan to use a cone shaped "tube" to reduce the light diverge

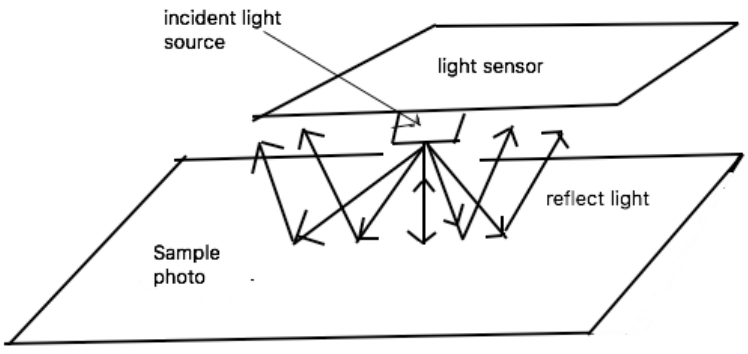


Figure.3 the

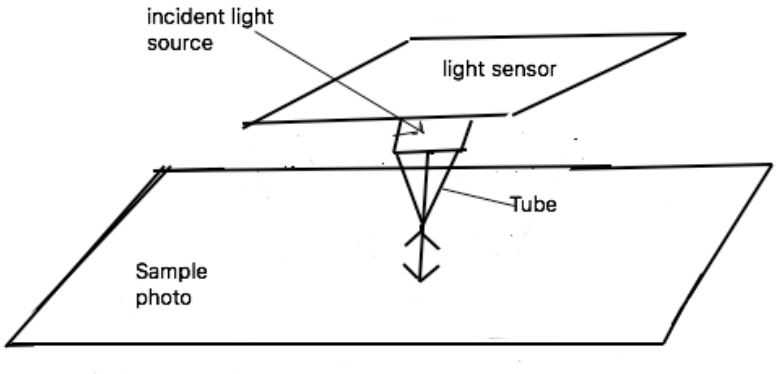


Figure.4 the reduced

Mechanical Considerations

We will build the frame out of wood, as that is an easy material to work with and correct mistakes. The wood will be held together with carpentry techniques such as using internal fasteners such as pegs with glue as well as bolts and metal fastening plates. We will try to use common lumber sizes as much as possible and avoid whittling the wood. To fasten the stepper motors to the wood, we should use metal bands and the joints for the pin actuator will be made with bolts with lock nuts.

The sliding functionality is the most important part of the hardware to work correctly. Our first choice is to use a 22 inch drawer slide and use the middle of our its travel range. This implementation will be easy if it will work. Another possibility is just to cut a slot in the wooden frame and just have wheels on the beam rest in the slots and we will fall back on that if a drawer slide does not work. To fasten the dolly to the beam with wheels, the dolly will rest mainly below the beam, but will have to wrap over the top of it to to put the rollers on top of the beam. We would like to just use more drawer slides for the dolly.

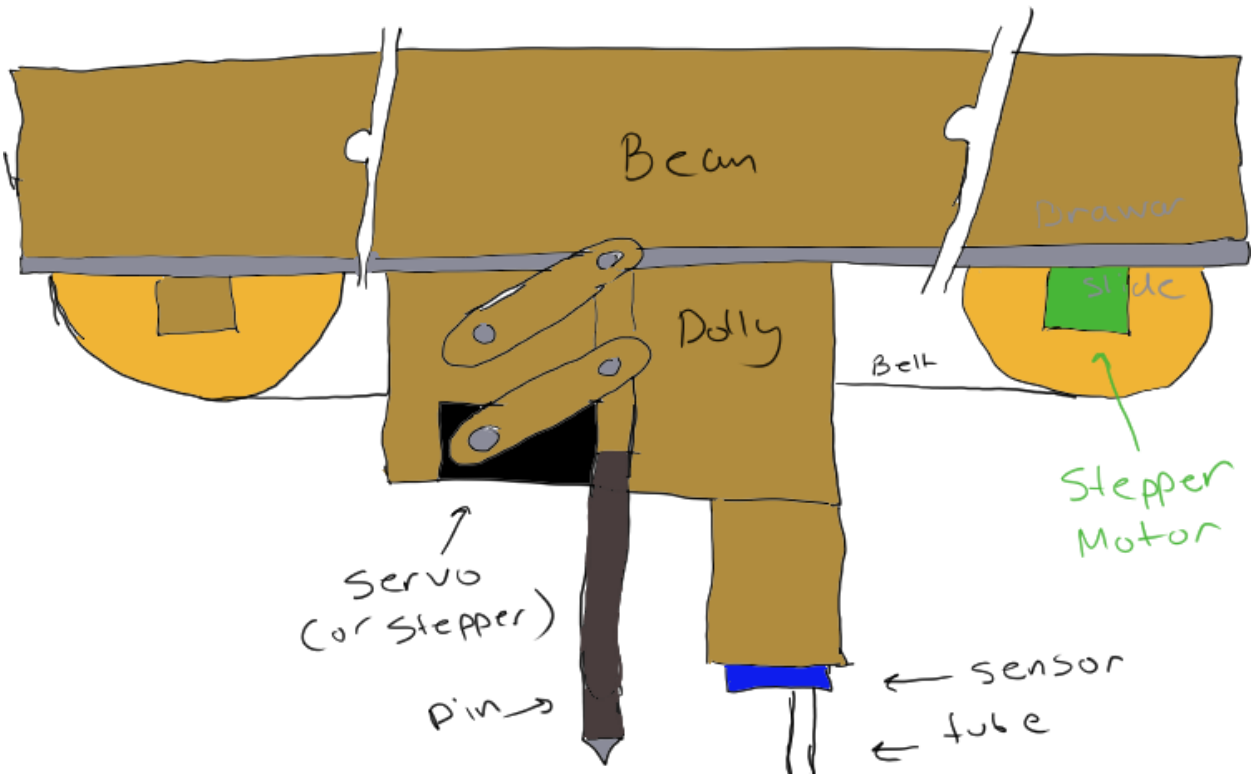
The most difficult part of the hardware to build is the motorized linear actuation of the frame and dolly. To move the beam and dolly, we imagine a stepper motor driving a timing belt attached to pulley on the two ends of the frame or the beam. The beam or dolly is attached to the belt by clamping it onto its teeth. The stepper motor moves the beam or dolly by turning one of the pulleys and moving the belt, and by measuring the belt, it should be easy to determine how much horizontal motion corresponds to one step on the stepper motor.

The pin actuator will be constructed with three wooden lengths, one driven, one idle, and one the holding the pin. The driven length should have one end attached directly to the output shaft of a stepper motor or servo motor.

We may want a way to adjust the belt tension, so we can make sure the belts are tight enough to avoid slack on the order of one pixel.

Our hardware is a bit intimidating but we believe it is achievable.

Drawings:



Stepper
Motor

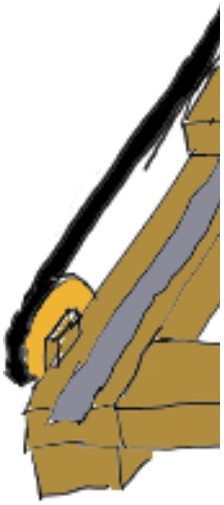


Figure.5&6 The image scanning and

Electrical Considerations

We will use four stepper motors. Two of those stepper motors are synchronous so we should be able to input the same four arduino pins into two separate h-bridges and drive the two stepper motors as if there were only one. We need four more pins and another h-bridge for each of the remaining two motors. Thus we need twelve digital outputs of a microcontroller to drive the motors. We need one analog input for the sensor, and two pins for interrupts. We will have barely enough pins available with an arduino uno but we will think we can make it work .

The total list of electrical components is four stepper motors, four h-bridges, one light sensor, one arduino, two resistors, two push buttons, an external power supply, and the wires connecting them.

Because there will be a sensor and at least one motor on the dolly, we must route at least six possibly much more, flexible wires to the dolly and make sure they do not get caught in the mechanics. We will route them by fastening them first to the beam and then to the dolly to keep them organized.

The electronics, we predict, will not be difficult to assemble.

Diagram shown on the following pages. We have two diagrams showing our two options: either using three stepper motors or using two stepper motors and one servo motor.

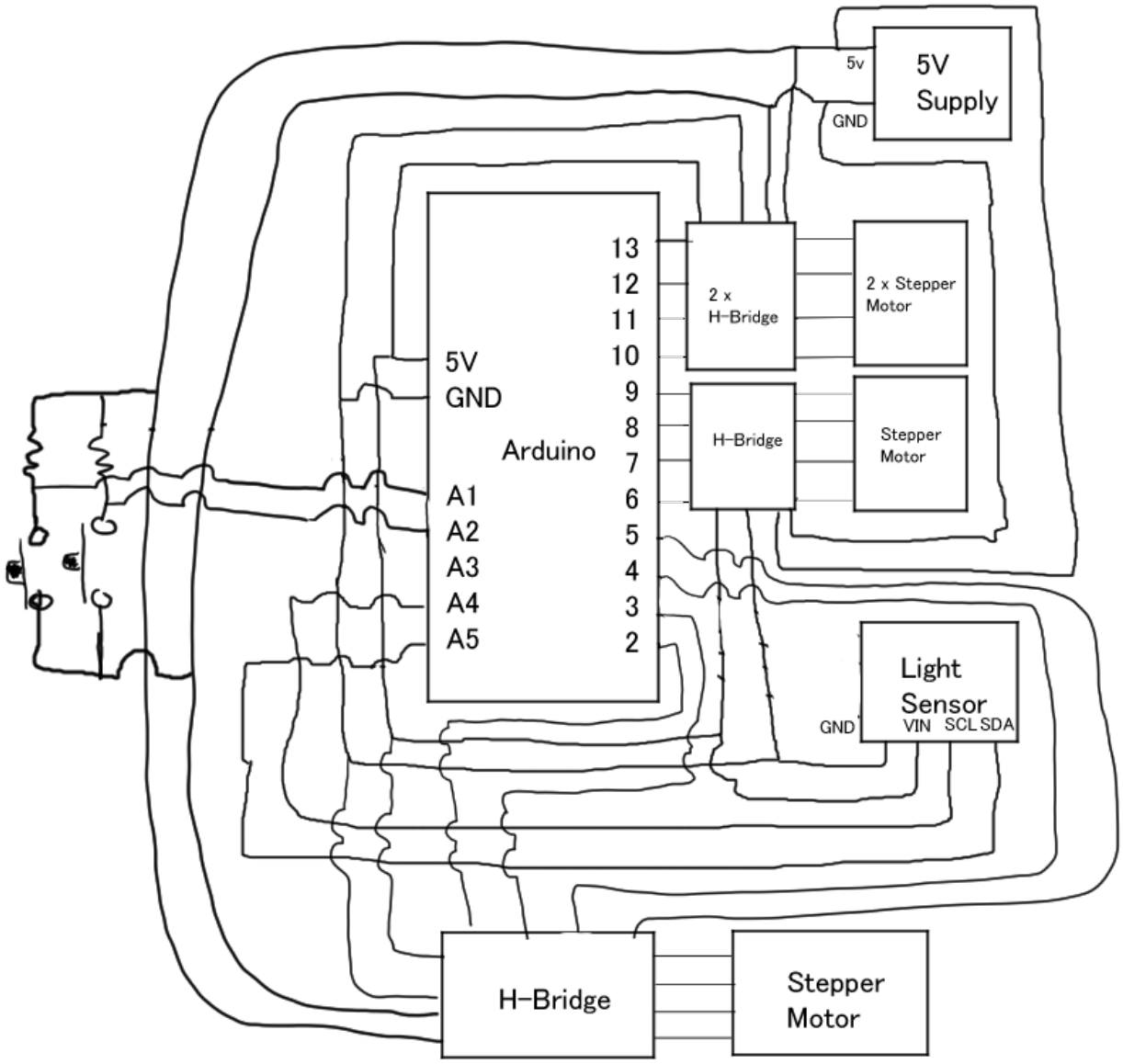


Figure.7 Block diagram for using 3x

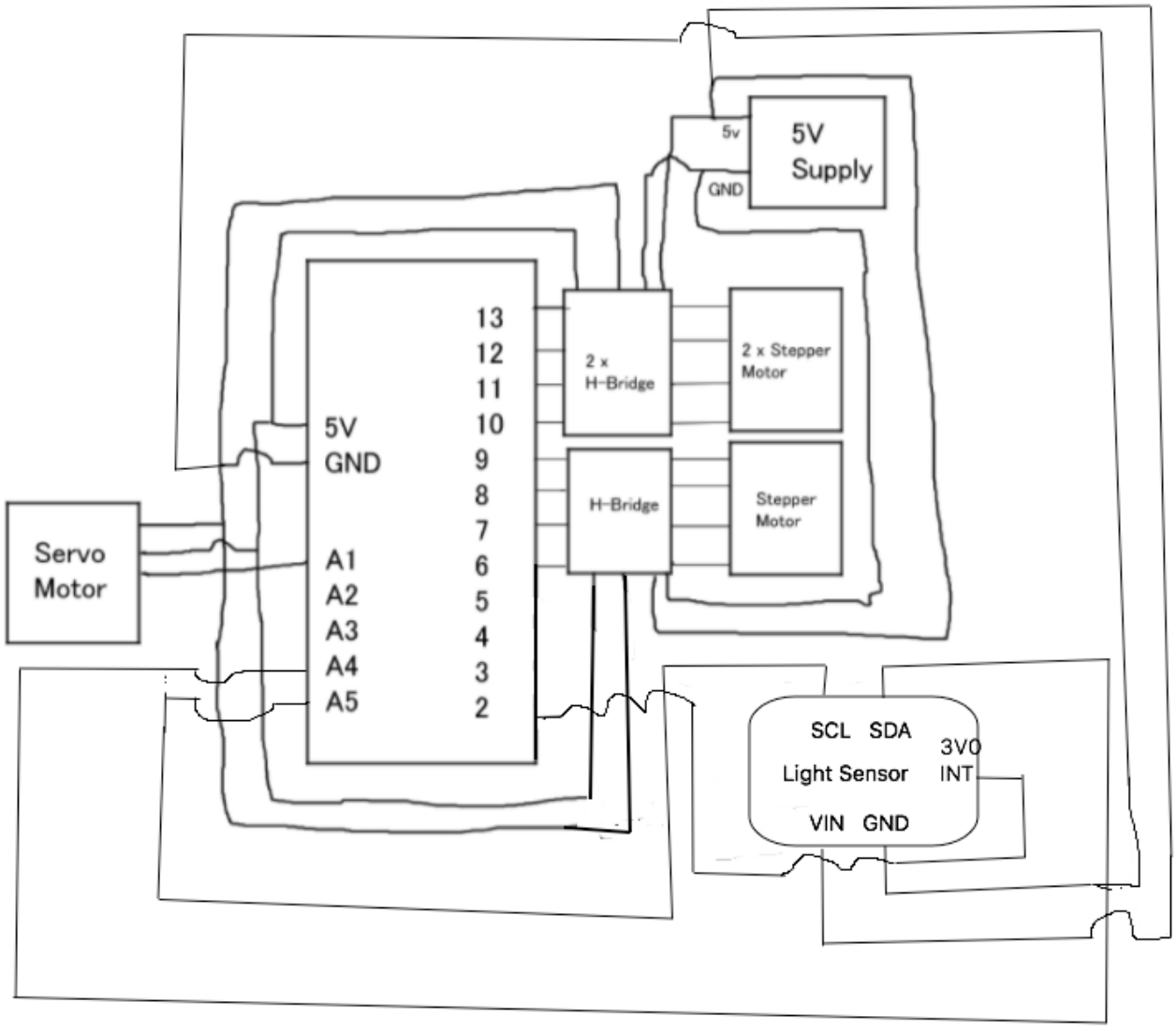


Figure.8 Block diagram for using 2x

Interface

Again we need twelve digital outputs of a microcontroller to drive the motors. We need one analog input for the sensor, and two pins for interrupts. We will have barely enough pins use with an arduino uno we will think we can make it work .

The user will interact with the device using only two push buttons We will connect two pins on the arduino to ground, and through a push button to five volts. Pressing the button will cause a rising event which will trigger an interrupt. Depending on which button is pressed, the apparatus will either scan or draw. The scanning and drawing process will be automatically done by our apparatus.

Software

The idea for the Canny edge detection came from a class that Eric took previously, and in that class, he wrote a script that performs canny edge detection using MATLAB. We need permission to reuse ideas from that class. We will not reuse any code, though, because we will write the code for the arduino in C, not MATLAB and Eric's MATLAB code took advantage of many built in MATLAB functions. Although our software seems complicated at the surface, previous experience with image processing makes it not as complicated as it seems.

The software will need to perform the many small tasks. We will need to pad an array, which we will accomplish by declaring a larger array and loop the contents of the original array into the larger array. We also need to perform convolutions which we will accomplish with nested for loops four loops deep. We need to scan in the image, and pad the image with two rows or two columns on every side. We will then convolve the image with a five by five gaussian kernel. We then convolve that array with two seibel operators, one vertical and one horizontal and store the two results separately. We then quadratic sum the contents of both of the resulting arrays pointwise, and also determine the inverse tangent of the ratio of their contents and organize those angle in four categories like rounding them to the nearest 45 degrees. We use the angles to perform non maximum suppression on the magnitude image, which is the most difficult part of the software. We then threshold the result to binarize it. We then draw a dot at every point that has the high value.

Given that atan() and sqrt() are in the math.h library with should be perform all of the mathematical operation we need.

One worry is the memory on the arduino. We will declare multiple two dimensional arrays the size of the image, and the arduino's memory could seriously limit the resolution we could achieve. The way to solve this is with dynamic memory allocation but apparently using dynamic memory allocation can cause fragmentation of the heap in an arduino.

Although the software is not particularly difficult it will be extensive and have lots of lines of code, which will make it time consuming. Eric is a bit excited about writing it, though.

We will use the stepper.h library to use the stepper motors, as well as analogRead() and interrupts(). We will also use the serial protocol to help debug the scanning functionality.

Testing

The scanning can be tested via performing the scanning very slowly over an image with high contrast, and printing the values from the sensor to to the serial monitor.

The printing functionality can be tested by hard coding a pattern for the machine to print. Base on our mechanical pin to draw, it is easy for us to see whether the arm will print the straight line,or edge, we want.

The image processing can be tested and debugged on a computer without using the hardware. Once the software and the hardware are working separately, we will combine them together. We will the the combined product using whatever pictures and drawings we have.

Safety

Although our edge detect machine doesn't have any excessive safety problem when the user operate it, there are still some potential risks we may encounter when building the hardware. In our project, we need to use several stepper motors; thus we should consider and calculate the voltage applied on each stepper motor to break or burn something. Our sensor also has a configurable interrupt that can fire when a certain proximity threshold is broken, or when a color sensor breaks a certain threshold and we need to be careful of that. We do not want wires disconnecting and dumping current into metal objects and melting insolation.

Parts and Reusability

Basic parts: light sensor, stepper motors, pulleys, belts, wood and maybe wheels.

We will need to order our light sensor, and it may be usable for future projects. We will have to solder pins or wires to the sensor, so the way to interfaces the the sensor to the arduino will be fixed by us.

The stepper motors already exist in the lab.

We will buy belts, pulleys, drawer slides or wheels may be useful for a future projects as well.

The wood, will probably supplied externally as we would like fresh wood, however we are willing to cooperate with the lab if they would like to purchase more wood.

We would like to specifically request purchase of a Pololu-QTR-1RC sensor or APDS9960 Proximity, Light, RGB, and Gesture Sensor, most likely the APDS9960 sensor.

Expansion options

For an expansion, we could use an RGB sensor to extract color information about the image. We could then do the edge detection algorithm on the red map, green map, and blue map and bitwise OR the results. The software involved will require using three element array to record and memorize the color value for edge detection. Then we would better detect edges due to changes in color.

Descope Options

The most difficult part of the software is the non-maximum suppression, so if the software is taking too much time it will be the first piece of functionality to be eliminated. That will make the drawing process less elegant, because the order in which to draw the points becomes less determined.

If the project is still taking too long, we could not edge detect at all and just binarize the image.