Name: Adam Hebert

Date: 5/3/2012

Project Final Report

Sudoku Solver

Original Concept

My original project concept was to create a Sudoku solving application for the android OS. This project would require the use of the android's camera and additional software in order to solve the puzzle. First, code would need to e created in order for the webcam to retrieve the original puzzle from a newspaper, or image. After, that part would be complete, I would develop an algorithm to solve the puzzle, and then output the solved puzzle. If I had time I wanted the solution to overlay over the video image of the puzzle. Due to time constraints, having the webcam read inputs into a matrix in MATLAB, then having the algorithm solve the puzzle would have been enough.

What could be done

I found out soon after that overlaying the solution over the video image may be too much. I then began to program the algorithm and test it using a preset 2d matrix that mimicked a Sudoku puzzle. Figure 1 will illustrate this idea.

```
M = [0 0 8 0 9 0 5 0 0;
     0 0 1 0 7 0 4 0 0;
     0 0 4 0 3 0 6 0 0;
     0 1 0 0 0 6 0 0 7;
     0 9 0 0 0 3 0 0 0;
     0 2 0 0 5 0 0 6 0;
     0 5 0 0 4 0 0 2 0;
     0 0 0 8 0 0 0 3 0;
     6 0 0 1 0 0 0 4 0];
```

Figure 1: 9x9 2d matrix

Using this method allowed me to test the code in order to see if it functioned properly. M would be used as the Sudoku puzzle. Once the algorithm was developed, I then began to find different methods of using the webcam to obtain a matrix. This proved to be a challenging task. Getting the video was not hard. Matlab can detect video recording devices that are installed on any PC. Figure 2 and 3 will illustrate this process.

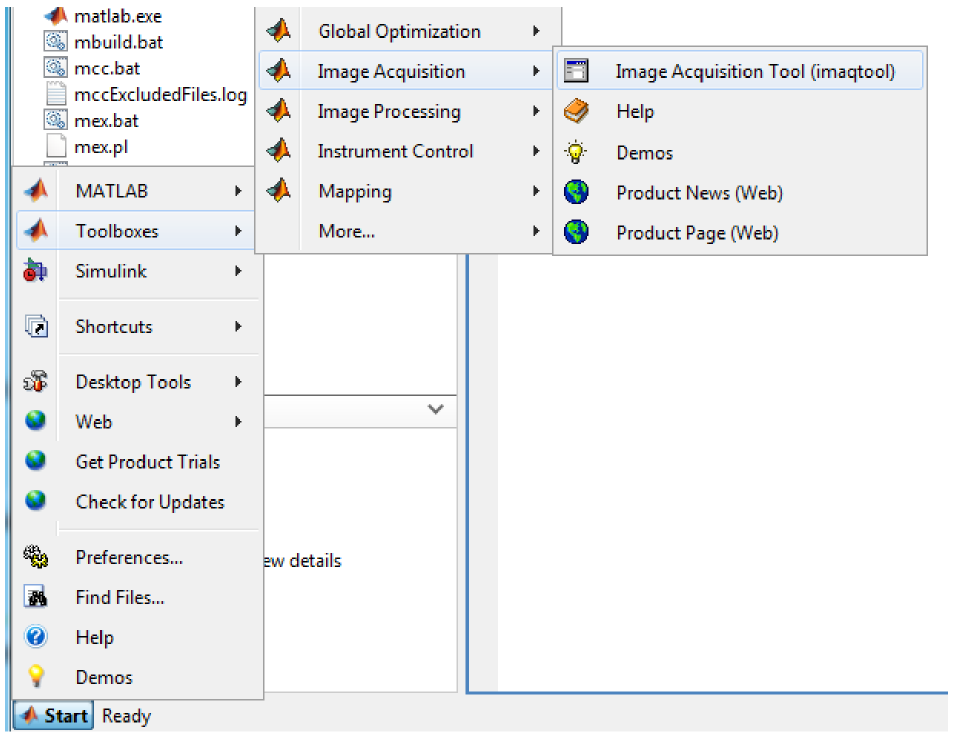

Figure 2

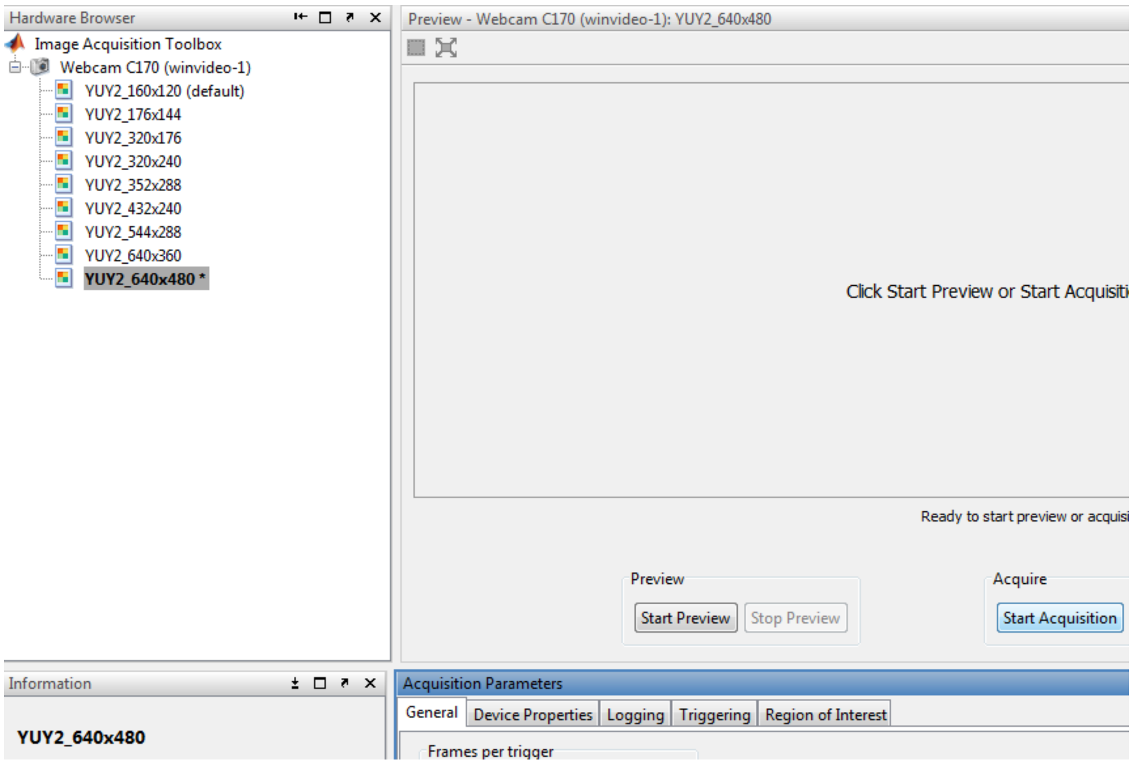

Figure 3

What needed to be done next is make sure the camera was to only pick up the Sudoku puzzle and its numbers. A template was used in order to compare the numbers on the puzzle with the test numbers. One set proved to be fairly successful. Most Sudoku puzzles use one type of font, allowing the template of numbers to be small. The following cod was used to get the camera working in MATLAB:

```
video = videoinput('winvideo',1,YUY2_640x480);
```

After the video was acquired, there needed to be some filtering and threshold operations done to ensure MATLAB could properly detect the puzzle only. Noise removal was also necessary to make sure stray objects were not seen as numbers.

When the program looks for objects and they match the template data, they will circle them in red; otherwise they will be outlines in blue. Anything outlined in blue will be ignored and not stored into the 2d array. In order to make sure the correct numbers are being found, the code establishes a boundary for where the Sudoku puzzle is located. It then only looks for objects inside that pink square. The code then overlays a grid using the set boundary, and if there is no object in the square, a 0 is put into the 2d matrix. Otherwise, the number that best matches the template data is stored into the matrix.

To ensure the image of the Sudoku puzzle is correct, it must be checked 3 times for consistency. After the 3rd check, it then sends the original Sudoku matrix to my algorithm for it to be solved. After being solved, I output the solution into a Sudoku-like grid.

The way the algorithm solves the puzzle is similar to how a person solves the Sudoku puzzle. My code uses a 3d matrix in order to store the puzzle and the possible answers for what should be in a vacant square. It checks each row, column, and block for possible numbers that could be places in each square. If it just so happens that there is only one possible number for a square, it will update the matrix. After the update, it will continue finding possibilities. If, for example, there are two possibilities the code will try each in order to see which may be correct. This method essentially is an educated brute force method.

What did not work

As stated in the original idea section I wanted to create an android app using the matlab code that worked. I knew there would be a problem initially due to android not being able to compile matlab code. I thought there may be a way to convert the matlab code into another language such as C/C++/Java. There were a few different methods I tried, and some I was unable to try due to the lack of time.

MATLAB Compiler:

This utility for MATLAB allows you to convert your MATLAB code into C. I believed this may be the tool that would allow me to get my code onto the android platform. After the conversion into C, I was unable to compile the resulting code. I received multiple error messages saying there may be libraries missing. I had not found out about this utility in time to fully explore how it functioned. I believe given more time, I would be able to use OpenCV to get the application working on the android.

ADDI:

This is an android app that allows you to input matlab commands, and see the figures created. The issue with this software is that it is very basic, and cannot run many of the toolbox commands I would need in order to solve a Sudoku puzzle. The program also allowed you to upload .m files onto your phone and open them with addi. Even with this ability, it was unable to run my code.

MLConnect:

This is another android app that allows the user to create a local server which in turn allows the android device to talk with the PC. This method is not as practical as ADDI would have been if it has worked. The program allowed the user to type MATLAB commands into their version of matlab and it would execute those commands. I was never able to get this to work. I did establish a connection, but even so it did not work as planned.

Android Cam as Webcam:

By this time in the semester I was just trying to see if there was any way to implement the android device into my project. This would allow the android to be used as a video capturing device in MATLAB. Unfortunately, the Image Acquisition Toolbox did not recognize the android device as such. This method did not work, thus, I stayed with the MATLAB and webcam method.

Conclusion:

I really enjoyed working on this project. Even post-creation I'm surprised how well it works. I've seen other Sudoku solver, but they took roughly 15+ seconds to solve. Given the correct cam positions my solver can complete in less than 8. Probably the hardest task was getting a MATLAB code to somehow work on the Android platform. I have yet to find a puzzle on the internet that my solver cannot solve.