COMP3419 Assignment Manual/Report

Gareth McFarlane, SID 430172980

I. Introduction

The following is a report on the completed program which includes a brief introduction to the field of Machine Vision, the overall design of the program, difficulties encountered during programming, my approach to coding, its strengths and shortcomings and finally a conclusion and analysis.

II. Report

A. Introduction

For humans to perform basic tasks, things like object recognition, shape detection, colour detection and so on, we hardly need to think about them. Despite how complex they are, we treat them as trivial actions. For a computer to replicate these trivial actions however, it is infinitely more difficult. The field of Machine Vision aims to improve the ways in which computers can replicate these actions. The goal of this assignment was to learn more about Machine Vision and enable a computer to see, detect and recognise colours and shapes and process these detections into further actions to create sounds, simulating a musician in a way. I believe I was successful in this aim and have learnt a great deal about the field along the way.

B. Implementation Approach

Initially, I had a fairly concrete idea of how I would build the program. I decided to choose Java due to the advantages of object oriented design and this paradigm assisted me in building the program. The basic idea was to have 4 - 5 classes, each class handling a specific part of the processing. One would read a frame from the camera, another would take this frame and return any details about squares that were detected. The remaining 2 - 3 classes were involved in MIDI sequencing and playback. I was slightly unsure of how many classes to implement here due to the complexity of MIDI instruments and sequencing. In the end I believe I created a somewhat elegant design to solve these problems.

C. Program Design

The final implementation consists of 7 classes in total. These classes are:

1) CardDrumming: This is the base class of the project. It handles construction of the Panel object and starts the program.

2) Panel: This is the GUI element of the program. It is constructed by the CardDrumming class and its responsibility is to draw the input frame, circles and squares and pass square information onto MIDIManager.

3) MIDIManager: This class handles the starting and stopping of sounds. It gets information from the Panel class about which drum to start beating and calls methods from MIDIPlayer to make these sounds.

4) MIDIPlayer: This class is the base class for sound playback. It has methods for playing various MIDI instruments.

5) SquareDetector: This class takes the input image and detects squares in the image. It returns information such as square position, size and centre of mass. The Panel class gathers this information to create sounds.

6) Drum: This class contains instances of each individual drum seen on-screen and is assigned a position, radius and instrument ID.

7) ImageConvert: This is a static class that simply takes an OpenCV Mat and converts it to a BufferedImage.

8) HSVTest: This is a calibration class used to find the HSV values of a particular object.

D. Implementation Difficulties

I initially had quite a few difficulties with this assignment, specifically with OpenCV. The first was getting it installed. The process of building and making OpenCV was not very straightforward and as a result, it took me a week to properly configure it. During this time however, I was able to begin work on the MIDI classes. The second difficulty was documentation. Java documentation for OpenCV is virtually non-existent and any documentation is usually directly copied from the C++ documentation. This usually isn't helpful as C++ OpenCV methods are quite different to Java methods so there was a lot of guesswork involved. The third was the OpenCV version. I decided to use the brand new OpenCV 3.0 RC1 build. As it is so new, and such a big jump from the popular OpenCV 2.4 build, guides were hard to come by and documentation was scarce. Through some trial and error and educated guesses, I was able to complete it however.

A somewhat difficult part was finding the correct way to detect objects. I tried numerous strategies including OpenCV's inbuilt detection methods, I tried implementing my own edge detection methods. Eventually I settled on a robust thresholding method which requires the program to know the HSV values of the object. This is slightly tedious to set up initially but works well after.

Another small issue was planning the MIDI classes. It took a long time to finalise the structure of these classes and come up with an efficient way of handling these sounds.

E. Strengths and Weaknesses

Overall, I believe my assignment performs very well with the correct configuration. If the HSV value ranges are specified accurately, the program should always detect at least part of the coloured object and hit the correct drum. The program is also expandable, allowing for potentially infinite square detectors with unique colours, and more drums added. Obviously the disadvantage of this program is the need for manual calibration of the specified HSV values to achieve full accuracy.

F. Conclusion

Overall I’m happy with the way the program turned out. I’m glad I took the time to plan everything out and come up with an efficient, elegant, object oriented design that allows for extensibility and maintenance. There is room for improvement however, as an extension I could factor in the angles of the squares to possibly alter tempo, pitch or volume of the instruments. This is fairly easy to implement however I didn’t have the time to completely finish this. Drumming is slightly cumbersome because of the need to align the beats of the drums. Despite this, the program works well and if nothing else, demonstrates how colours and shapes are detected in OpenCV.