EIE1 Project Report

Josh Brown and Ravi Woods

Progress this term

This term, we progressed from a basic dot product file to more complex convolution filters. After the dot product accelerator, we were given a Quartus project that implemented a blur filter, and a coloured filter. From this we needed to progress to an edge detection filter.

Black and White

The first step was to take the original feed and turn it into a grayscale image, as the edge detected image needed this functionality. The simplest way of achieving this was to take the mean of the three colour intensities, and output these in the three colour slots. This worked as expected. In addition, we decided to have some fun with the colours of the image, by changing Green to Red, Red to Blue, and Blue to Green (Fig 1).

Figure 1: Colour switching on a green highlighter.

Sobel Edge Detection

a) The basic algorithm

When designing the edge detecting filter, we first needed to work out a basic algorithm for it. From information on the internet, we came up with a step-by-step process:

1. Convolve the 9 different pixels with G_X (See Fig 2) – we did this with each separate colour.
2. Convolve the 9 different pixels with G_Y
3. Find $\big||G_x| + |G_Y|\big|$
4. Set the output (for Red, Green and Blue), to the mean of the three values found in (3).

This would then get us a grayscale image, with white edges and black background.

-1	0	+1
-2	0	+2
-1	0	+1

Gx

+1	+2	+1
0	0	0
-1	-2	-1

Gy

Figure 2: Sobel convolution filters.

b) Defining further functions

In the code, we needed to take an absolute value a number of times. So, we defined a function as such. This returned one of three values:
If the value is greater than 1023, it would return 1023, as 1023 is the largest intensity allowable
If the value, x, is less than 0, it would return –x , as negative intensities aren't allowed either
However, if the value, x, is between 0 and 1023, it will simply return x.

From this, we needed to declare the function. We did this in the header, .h, and defined the ac_int input as true, as a signed input is a possibility.

From this, we got an output as expected (Fig 3). As you can see, it can pick up basic shapes, and more obvious lines. However, since the resolution of the camera is low, text couldn't be seen well. This rules out any sort of text detection for the final project.

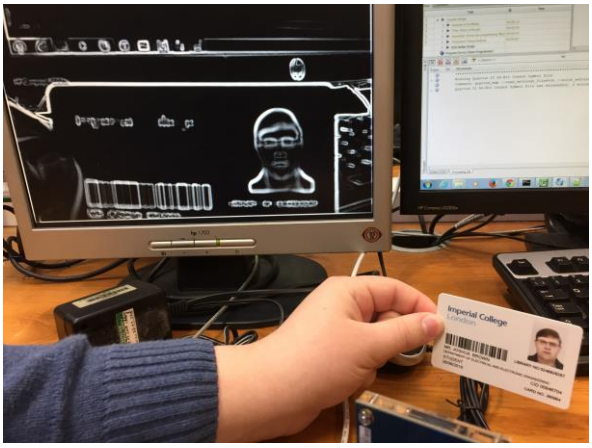


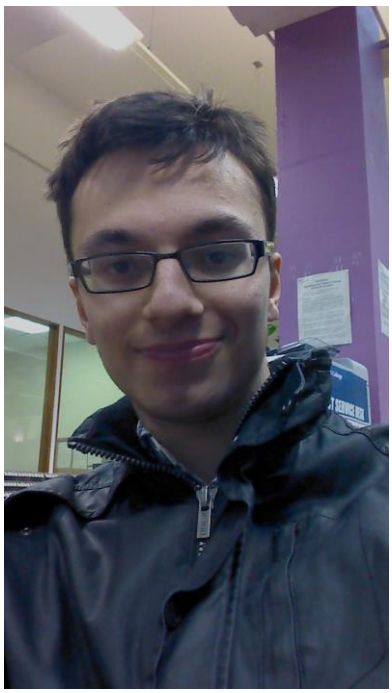
Figure 3: Sobel edge detection example.

Sharpen Filter

Following this success, we aimed to create a sharpening filter for the image. This involved convolving the 3x3 grid with this matrix:

$$\begin{pmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{pmatrix}$$

However, the output didn't work as expected (Fig 4) – it was rather psychedelic! We believe this a problem with the fact that it convolves the three separate colour separately, but we didn't have time to debug this.



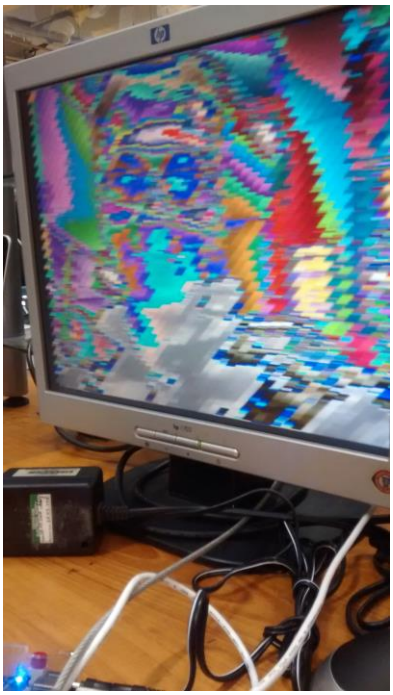


Figure 4: Sharpen Filter attempt.

Future Project Ideas

Next term, we will have the opportunity to undertake a longer project using the FPGA's image processing capabilities. Below are several ideas for projects that we believe would be achievable using the time and technology available.

Virtual green screen

The FPGA and a camera could be used to implement a virtual green screen effect. The process of replacing a specific colour from the background with a static image (or even

potentially a video) is very simple, but sampling a random background and replacing those pixels that do not change would be slightly more complex.

The system would operate by taking a still image of the background before the subject steps into the frame. The pixels from the camera could then be compared with those in the reference image and, if they are the same, replaced by the pixels in either another video feed or a static image. Depending on the difficulty of this, multiple backgrounds could be stored on an SD card and selected using the toggle switches on the DE0 board.

Hand gestures

Edge detection could be applied to an image and used to detect hand gestures waving at the screen. With some change detection, it would be possible to implement a system of gesture based controls. These would be demonstrated by navigating through a mocked up operating system or web browser. Proposed gestures are:

- Swipe left (previous screen, rewind, undo)
- Swipe right (next screen, fast forward, redo)
- Swipe down (save or select one of a series of images, perhaps representing products in an online store)
- Two hands moving apart (zoom out)
- Two hands moving together (zoom in)

As an extension, custom gestures could be added by the user. For example, a lecturer could save a hand movement that they use naturally, but which would advance the slides in a presentation.

The primary challenge involved with realising this idea is the issue of having to detect the motion over a number of different frames, and the fact that there is no defined start or end point for each gesture.

Counting fingers

A reduced version of the previous proposition, this project would have the camera taking images of a hand (if necessary with a coloured glove) and recognise the numbers being represented. This could be extended to other gestures. The numbers could be outputted using either the LEDs or the seven segment displays. This should be a fairly simple task and could be used as a proof of concept for other gesture based projects.

Collision detection

Again, using an edge detection mask, the FPGA could perform collision detection for a moving vehicle, for example, EEBug. This would involve moving at a set speed and detecting perceived size changes in the objects in the image. This would give an indication of distance and the FPGA could send a cut out (or even change direction) signal to the vehicle. This project would involve having the FPGA operate without peripheral connections. Power could be provided by a battery (the power supply in the lab is rated at 9V 1A, with the USB supply being 5V 500mA). There would be no need for a monitor unless required for debugging/demonstration purposes, since the output will be sent directly to the motor controller. The USB and mouse would also not be required.

Whilst conceptually challenging, this has the potential to be a very interesting and exciting project.

Camera Enhancements

The functionality of the camera on the DE0 board is, at best, minimal. Exposure has to be adjusted manually and there is no provision to save the images generated by the image processing effects. We could potentially implement this functionality ourselved. Exposure could be set by counting the number of white (or near white) and black (or very dark) pixels, and increasing or decreasing exposure to try to equate these numbers. Image capture could be achieved by sending the pixels from a given frame to both the monitor and an SD card for storage.

This project could be extended by adding other image enhancement functionality, such as stabilization or increased zoom. Image capture would be a nice addition to most of the other proposed projects.

Graduated Image Effects

The mouse input could be used to dictate the strength of different effects. It would be quite simple to implement sharpens, blurs, tints etc. and use the mouse left ←→ right to dictate the strength of these. For example, a blue tint would divide the red and green values of every pixel by a constant. This constant could be adjusted between 1.5 and 10 by moving the mouse. Different effects could be selected using the toggle switches on the DE0 board. This need not be a challenging project but its difficulty will scale with the number of effects implemented.