EE 443 PROJECT REPORT: POST-FILTERING PHOTOGRAPHIC TOOL

Emre Demir, Ömer Alperen Katı, İbrahim Aslan
Electrical and Electronics Engineering Department, Boğaziçi University, Turkey

a) Summary

The image processing algorithms are widely employed in the technological devices used in daily life and industry to solve various problems such as edge detection and image quality improvements. Those algorithms can be utilized in embedded systems to enhance performance while keeping the cost reasonably low. In this paper, an OV7670 Camera Module is connected to De1-SoC. This camera is used to try to send frames to the board. Then, several image processing algorithms such as Sobel/Canny Edge detection and dithering process the embedded image data. Furthermore, different states in slide switches result in different outputs of those algorithms to be displayed on the VGA screen. In addition, previously determined numbers that correspond to each filtering mode are monitored on 7-segment displays.

b) Introduction

In this project, the main aim is to utilize the OV7670 Camera Module to take a photo and display it on the VGA display using different signal processing algorithms. For this purpose, the VGA monitor is connected to the De1-SoC board from the VGA out port, and the camera is connected to the De1-SoC board from 2x7 LTC Expansion Header and General Purpose I/O (GPIO) pins.

In literature, there are lots of system-on-chip projects done only with the help of VGA and C programming or other external components such as a mouse, keyboard etc. In this paper, the primary motivation of the authors is to include a rarely used component to the project, which is OV7670 Camera Module, and to bring innovation to EE443, which is only included in the specialization options of electronics and control, with the image processing algorithms.

The main challenge will be to use GPIO pins and LTC pins to manipulate the camera to the authors' knowledge. Those pins are far from the ARM Cortex A9, and vulnerable to external and internal noise. In addition, the intelligent way of communication between the board and the camera has to be programmed manually, and this part is not going to be a straightforward task to figure out, which is the main focus of the project.

c) Broader impact

By employing image processing tasks in embedded systems, the need for cloud, server or desktop will not be valid anymore. For some cases, in which the channel capacity is limited under challenging conditions, the image data with low resolution can be compensated with image quality improvement methods such as dithering. At the same time, those algorithms require comparatively more resources from time to time and can be optimized to make do with fewer resources or accelerate the process [5].

d) Task Description

The tasks can be divided into two parts as follows: camera, image processing, and complementary parts.

1. OV7670 Camera Module

A) Hardware Specifications

The hardware connections of the camera can be seen in Fig. 1. The pins from D0 to D7 are selected to be used to get the data coming from the camera's pin of identical names, and the pins from D8-D10 are connected to VSYNC, HREF and PCLK pins of the camera, respectively. Those 2x7 LTC Header and GPIO pin connections can be seen in Fig. 1.

Also, our camera has a pin named XCLK, which enables the camera to operate. The connection of XCLK can be seen in Fig. 2. It is driven to digital pin number 11 of Arduino Uno via a resistor of 4.7kOhm, and it is grounded via another resistor of the same value. The reason for using Arduino Uno for the clock generation is that assignment of a GPIO pin for a clock signal can result in some delay, and this signal can interfere with the other used pins.

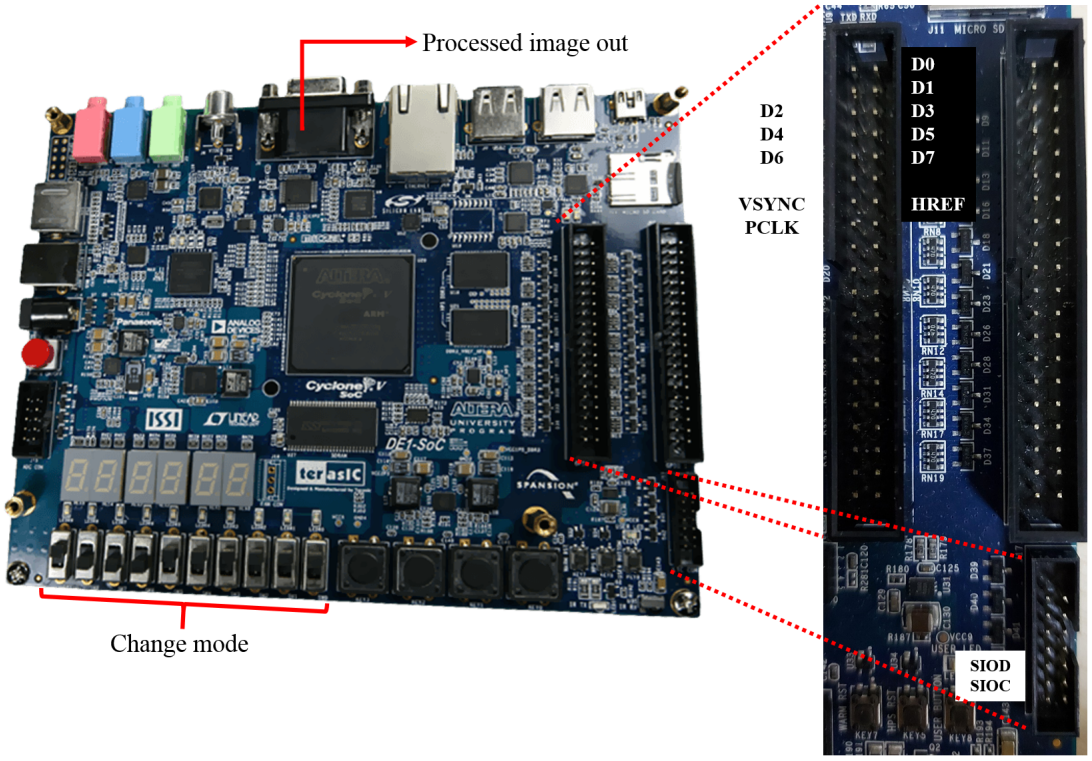


Fig: 1: The GPIO and 2x7 LTC header pin connections.

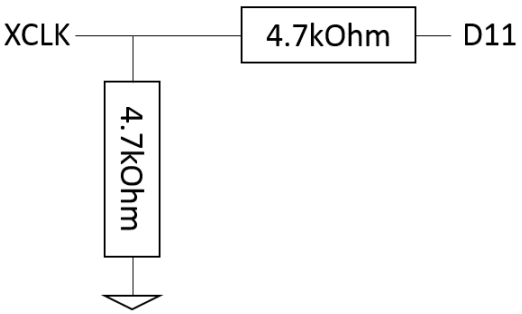


Fig. 2: The connections of the XLCK pin to the D11 (clock) pin of Arduino Uno.

B) Software Specifications

In the software, the camera should be initialized such that previously determined GPIO pins are configured to the input mode. After that, camera configuration is done via I2C protocol. The camera is configured to output QVGA- RGB565 frames.

After configuration, the ARM processor is ready to capture the frames and process them. Capturing is done via the blocking method. Firstly, VSYNC signal is waited to fall (high to low). It indicates the start of the frame. Then, HREF signal is waited to be high. It indicates the start of the line. Then, PCLK is waited to be low and high. Data pins are ready to be read at the rising edges of the clock.

2. Signal Processing Modules

In the signal processing part, the implemented image processing methods are mainly dithering and edge detection algorithms.

A) Dithering

Two types of techniques, namely Random and Error Diffusion (Floyd-Steinberg) dithering, are used in dithering.

i) Random Dithering

Basically, in random dithering, RGB565 pixel value is separated into colour values. Depending on the summation of the distances of each color value to their lower and upper limit, we randomly but in a weighted manner assign 0 or 255 to the corresponding pixel. The dithering mechanism is fundamental and as follows:

rand()%255-(255-current_pixel) < 0 ? 0 : 255;

ii) Floyd-Steinberg Dithering

In this type of dithering, possible errors are calculated for each color value, and the pixel is

rounded to the minimum of these error summations. Then, the exact error is determined, and this error is distributed to its surrounding pixels using the suggested fractions [1], as can be seen in Fig. 3.

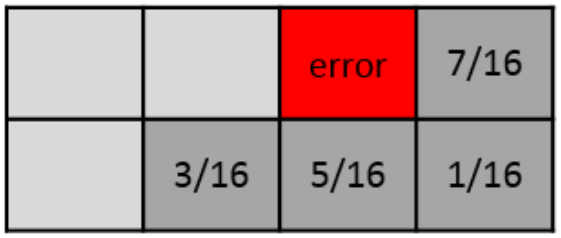


Fig 3: The pixels subject to the distribution of the error with the suggested fractions.

B) Edge Detection

i) Sobel Edge detection

In this method, the centre of the filter is comparatively more emphasized. The noise level is commonly reduced with the help of this method [2]. The steps of the algorithm are presented in Fig. 4.

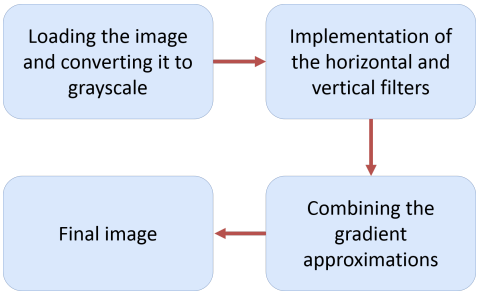


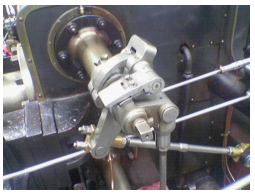
Fig. 4: Steps of the Sobel Edge detection algorithm.

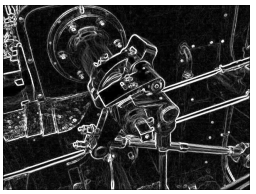
To implement the method, the gray-scale version of the images are needed. The horizontal and vertical filters are different since this method emphasizes the center of the filter.

$$G_x = \begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix} * A$$

$$G_y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * A$$

$$G = \sqrt{G_x^2 + G_y^2}$$





(a) Original image (b) Sobel method output

Fig. 5

ii) Canny Edge detection

In terms of both being common and practical, this is the leading edge detection method. It is a multistage algorithm. Therefore, it is computationally heavy[3][4]. The flowchart of the Canny Edge detection algorithm can be seen in Fig. 6.

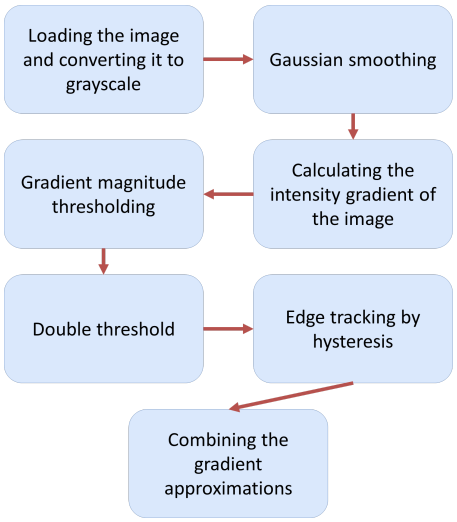


Fig. 6

Firstly, the input image is converted to grayscale. Then, since this method is noise-sensitive, noise is filtered out. The kernel element of dimension (2k+1)x(2k+1) is calculated using the formula below.

$$H_{ij} = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{(i-(k+1))^2 + (j-(k+1))^2}{2\sigma^2}\right)$$

$$1 \le i, j \le (2k+1)$$

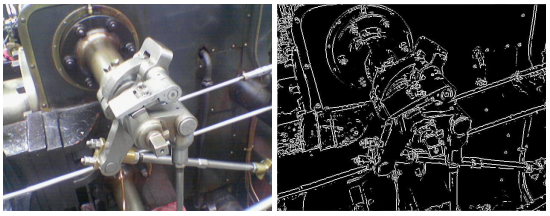
The first derivative in both horizontal and vertical directions is taken to calculate the intensity gradient of the image. The edge direction is determined by evaluating atan2 of Gx and Gy as follows.

$$G = \sqrt{G_x^2 + G_y^2} \quad \Theta = atan2(G_x, G_y)$$

Gradient magnitude thresholding is done by comparing consecutive pixels in the direction of the gradient. Provided that the current pixel is the highest, this value is recorded. Otherwise, it is suppressed.

The double threshold part suppresses the variations in colour and noise originating from the rest of the edge pixels. The pixels are divided into three groups using higher and lower threshold values, the edge pixel is marked as strong, weak and suppressed pixels, respectively.

Since the source of the weak pixels is not identified, it could be either variations due to noise or colour or true edge pixels. Also, it is more probable to detect a weak true edge pixel close to a strong pixel. Therefore, if this is observed, the weak signal is preserved. An example of an input-output pair can be seen in Fig. 5.



(a) Original image (b) Canny method output

Fig. 5: Upper and lower thresholds are taken 50 and 20 for the Canny method, respectively.

e) Design Implementation (1 pages)

1) Signal processing

The design of signal processing algorithms is planned to be relatively straightforward. OV7670 will provide the board with a 2D array of size 240x320 pixels. This array is passed to the image processing functions, and by iterating over those arrays pixel by pixel by using for loops, the pixel data is converted into the desired form. The output arrays are also passed into those functions to store the processed data.

2) Complementary parts

As it is previously explained, it is planned to employ a mechanism to control the mode of the displayed image. This mechanism is to be realized by inserting all displaying states into a while loop with if statements checking the switch positions.

A feature is added to the Canny Edge detection algorithm in order for users to update the upper and lower threshold values. This is done using push buttons. As the button is pressed, the next lower and upper threshold pair is selected from a catalogue, and displayed on a 7-segment display.

For the implementation of the project, authors find it obviously more suitable to write in C since high performance is not needed, and ease of implementation is of interest. It has a satisfactorily high speed since it does all the image processing first, and then it only switches between the filtering modes to display the image.

Because the camera gives a relatively low resolution on average, processing algorithms don't take too much to execute. Overall, our board is a product of technology whose level is way higher than the requirements of our project. For this reason, a more user-friendly prototype that requires fewer resources can be designed to get into the market.

REFERENCES

[1] https://shihn.ca/posts/2020/dithering/

[2] G. M. H. Amer and A. M. Abushaala, "Edge detection methods," 2015 2nd World Symposium on Web Applications and Networking (WSWAN), 2015, pp. 1-7.

[3] Wikipedia contributors. (2021, December 28). Canny edge detector. In *Wikipedia, The Free Encyclopedia*. Retrieved 21:30, January 18, 2022.

[4] Wikipedia contributors. (2020, June 19). Deriche edge detector. In *Wikipedia, The Free Encyclopedia*. Retrieved 21:31, January 18, 2022.

[5] Bailey, D.G. Design for Embedded Image Processing on FPGAs; John Wiley & Sons: Singapore, Singapore, 2011.