EE296A Embedded SoC Design Spring 2017

San Jose State University
Department of Electrical Engineering
Laboratory Assignment #4

In this lab you will use the OpenCV installed in Part 1. You will implement and accelerate a Sobel filter using Openmp pragmas. An OpenCV template will be provided in Canvas which captures images from a webcam attached to the Zybo board.

You can compile the code on Zybo using the following command g++ `pkg-config opencv --cflags`my_code.cpp -o my_code `pkg-config opencv --libs`

This will only work if you have successfully completed lab part 1 and have a working installation of OpenCV on the Zybo board

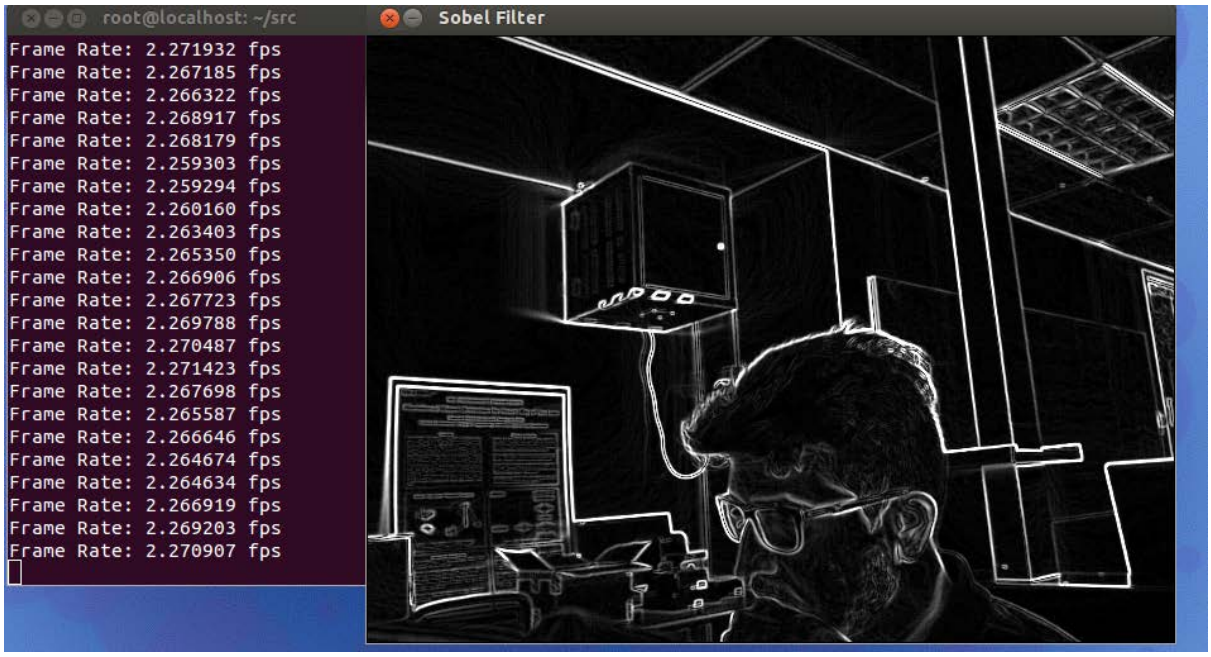
- **Sobel Filter Code**

```
#include <iostream>
#include <cmath>
#include <stdio.h>
#include <omp.h>
#include "opencv2/imgproc/imgproc.hpp"
#include "opencv2/highgui/highgui.hpp"

using namespace std;
using namespace cv;

int main()
{
        double fps;
        double sum_fps;
       double avg;
        double t = 0;
        double counter=1;
        char str[5];
        Mat src, dst;
        int gx, gy, sum, x, y;
        CvCapture* capture = 0;
        capture = cvCaptureFromCAM( 0 );

while(1) {

        IplImage* image0=cvQueryFrame(capture);
```

```
	Mat m = cv::cvarrToMat(image0);
	cvtColor(m, src, CV_BGR2GRAY);
	dst = src.clone();
	t = (double)getTickCount();
	if( !src.data )
	{
		return -1;
	}

//using dynamic scheduling to improve the pragma by providing one iteration to each thread//
//we can only use chunk with dynamic scheduling and chunk provide number of iteration to each thread//
//making variable private to use them in one thread only//
#pragma omp parallel for private (x, y)
	for(y = 0; y < src.rows; y++) //starting the operation from pixel 0 row and column//
	{
		for(x = 0; x < src.cols; x++)
		{

		dst.at<uchar>(y,x) = 0; //giving the value 0 to pixels of dst with unsigned char because of 0 to 255 limit//
		}
	}

#pragma omp parallel for private (gx, gy, sum, x, y) schedule (dynamic, 480) num_threads (2)
	for(y = 1; y < src.rows - 1; y++)	//starting the operation from pixel 1 row and column//
	{
		for(x = 1; x < src.cols - 1; x++)
		{

//using gradient method creating mask for gx and gy//
		gx = src.at<uchar>(y - 1, x - 1) +
		2 * src.at<uchar>(y, x - 1) +
		src.at<uchar>(y + 1, x - 1) -
		src.at<uchar>(y - 1, x + 1) -
		2 * src.at<uchar>(y, x + 1) -
		src.at<uchar>(y + 1, x + 1);   //gx will become horizontal mask//
```

```
                gy = src.at<uchar>(y - 1, x - 1) +
                2 * src.at<uchar>(y - 1, x) +
                src.at<uchar>(y - 1, x + 1) -
                src.at<uchar>(y + 1, x - 1) -
                2 * src.at<uchar>(y + 1, x) -
                src.at<uchar>(y + 1, x + 1);   //gy will become vertical mask//

                sum = abs(gx) + abs(gy);      //adding absolute value of gx
mask and gy mask to sum//
                sum = sum > 255 ? 255 : sum;       //setting the white color
intensities//
                sum = sum < 0 ? 0 : sum;      //settin the black color
intensities//
                dst.at<uchar>(y,x) = sum;     //giving the value of sum to
pixels of dst with unsigned char because of 0 to 255 limit//

                }
        }

        t = ((double)getTickCount() - t) / getTickFrequency();
        fps = 1.0 / t;
       sum_fps+=fps;
       avg=sum_fps/counter;
        CvFont font;
       double hScale=1.0;
       double vScale=1.0;
       int    lineWidth=1;
       sprintf(str, "fps: %5f", avg);
        printf("Frame Rate: %f fps\n", avg);
       counter++;
        namedWindow("Sobel Filter");
        imshow("Sobel Filter", dst);

        char key = (char) waitKey(20);
        if(key == 27) break;
 }

        return 0;
}
```

- **Output Without Acceleration**



- **Output With Acceleration**