Department of Electrical & Electronic Engineering

University of Nottingham Ningbo China

EEE1039 Course Work Report:

Image Processing with OpenCV

Shun YAO

20321234

INTRODUCTION:

This course work aims to recognize and target the black line in a video taken by raspberryPi camera. The black track was set up in lab. Combine raspberryPi camera and the vehicle together, drive vehicle goes through the black track directly to record the original video which means vehicle was no need to track the black line as figure1 shows.

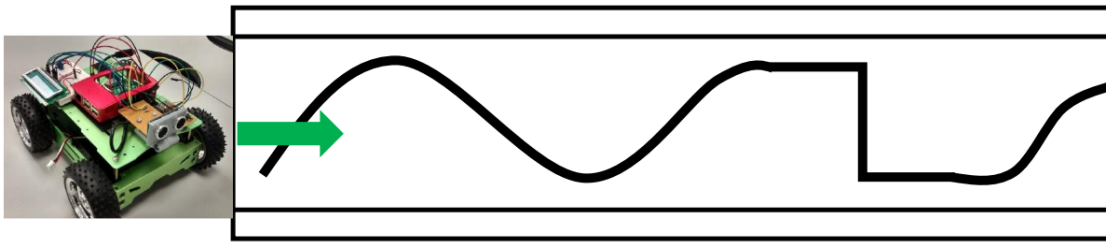

Figure1. The track for original video recording

Set up the time duration of original video recording beforehand, make sure the recorded video could be processed properly.

Using codeblocks on personal computer or on the provided raspberryPi and OpenCV to process the video find the horizontal center of the black track and indicate its location in frame (figure2). Indicate the black center of each frame in original video and display coordinate of the black center as processed frame. Save all the processed frame as a new processed video. Programme must be C++.

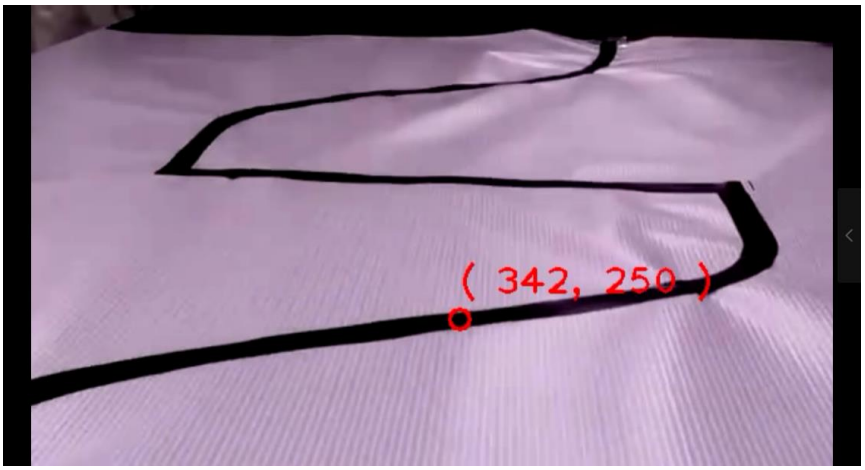

Figure2. Sample image for processing video

METHOD:

The raspberryPi camera was stick to the top of vehicle for the original video recording. Programme the vehicle to run forward in speed of 15, record video from the top of the running vehicle. The function “system” was used to set up video recording:

system("raspivid -vf -o labvid3.h264 -t 15000 &");

This instruction will command raspberryPi start recording a video named “labvid3.h264” for 15000 milliseconds in background and the vehicle was driving forward simultaneously by another function:

serialPrintf(robot, "#Baffff 015 015 017 017");

Original video was finished after several experiments, and it shared to every member in same group. Duration and resolution of it could be reset into any number that can fulfil all requirements.

Video is the combination of many frames, process every frame of original video could form a new processed video. In that case, processing of a static image should start first. Run OpenCV on personal computer needs to download opencv document and set up configurator environment which will take plenty of time. So, it was decided to process the video on the provided raspberryPi OS.

It is required to recognize black line on white background so the binarization of image should be able to clearly distinguish the track and the background. Following function was used to gain the binarization of an image and one processed sample shows in figure3.

```
Mat gray;
cv::cvtColor(im, gray, CV_BGR2GRAY);
Mat gray4;
cv::threshold(gray, gray4, 125, 255, cv::THRESH_BINARY|cv::THRESH_OTSU);
```

Figure3. A sample of processed image

Read and record all the pixel gray value by function below. All gray value were recorded to a txt document (figure4).

grayValue = (int)gray4.at<uchar>(x, y);

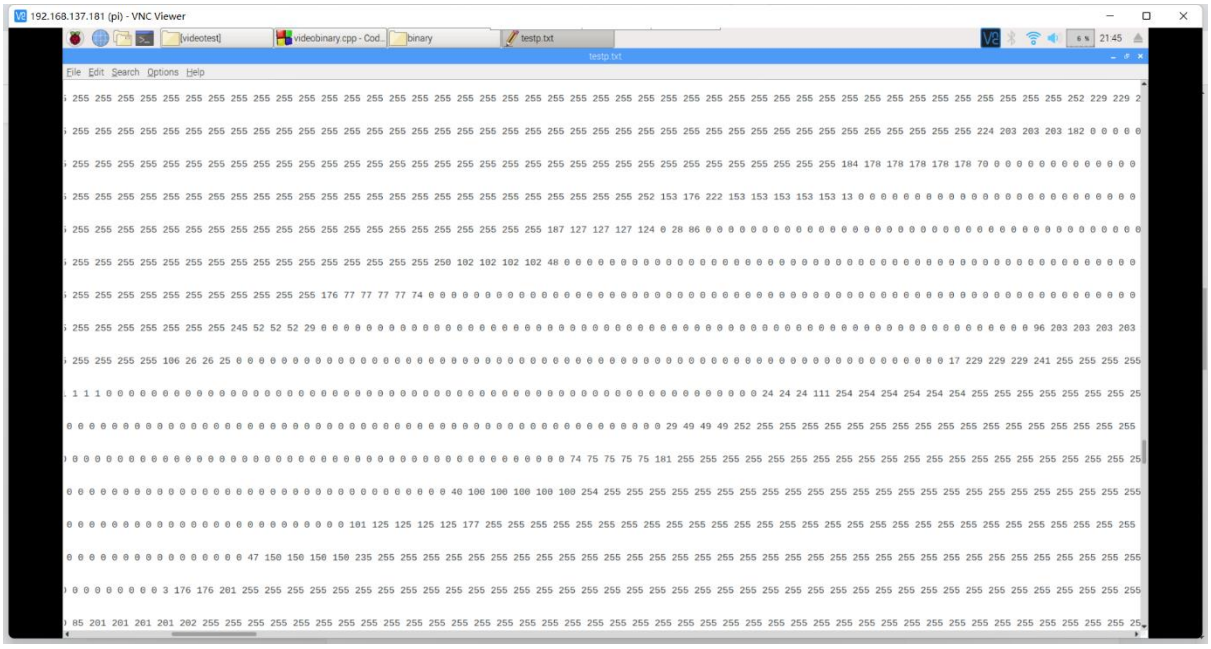

Figure4: A view of gray value of the binary image

Gray value of black track is zero as figure 4 shows. To find the horizontal center of the black track on one row, it is possible to take average of all column number which satisfy gray value equal to zero. In the programme, row number 250 was chosen as reference row. The gray value of every pixel on that row were readied by a “for” loop.

If one column on the row has gray value of zero, the column number will be added to a sum and a counter adds one as there are one more column number satisfy gary value equal to zero.

The horizontal center column number of the black track could be calculated using the value of sum divided by the value of counter.

Following functions could draw a circle and put text at a specified point. Those were used to display coordinate on original image.

```
cv::circle(im, Point(x, 250), 8, cv::Scalar(0, 0, 255), 2);
cv::putText(im, sstr.str(), cv::Point(x, 230), CV_FONT_HERSHEY_PLAIN, 2, cv::Scalar(0, 0, 255), 2);
```

One static image is fully processed after the coordinate properly displayed. Then, put those procedures into a loop to make every frame of the video processed.

Processed video needs to be exported out as a new video, so every new frame should be saved in a new video document. The new video should have the same frequency and size as the original video. Following functions could be used to get video frequency per stitch and their size.

```
double fps = cap.get(CAP_PROP_FPS);//Get video frequency per stitch
Size size = Size(cap.get(CAP_PROP_FRAME_WIDTH), cap.get(CAP_PROP_FRAME_HEIGHT));
```

And specify file location, encoder, frame rate, width and height of the Processed video by function below.

```
VideoWriter writer;
writer.open("Processed video.avi", VideoWriter::fourcc('M','J','P','G'), fps, size);
```

RESULT:

The original video has a duration of 15 second and a resolution of 1080 ×720 which is too much calculation for raspberryPi. Transform it into resolution of 580 ×330, make calculation simpler and faster. Upload this video to the processing programme.

Provided raspberryPi OS has downloaded opencv document and set up the configurator environment. So, the programme could work properly on raspberryPi.

For static image processing, one screen shot from the original video was used as the sample (figure5). Running a programme to gain the binarization of this image and read gray value of all the pixels on this binary image. Record those gray value into a text document successfully.

Figure5. Sample of static image processing

Based on that the gray value on black track is zero, the “for” loop below was written on the programme to read all pixels on one specific row.

Reference row was chosen as row 250. This “for” loop will run for number of columns on that row times.

Variable “c” stand for number of columns; “x” stands for the sum of all number of column which satisfy gray value equal to zero; “range” stand for the number of column that have gary value equal to zero.

```
for ( c = 0; c < gray4.cols; c++ )
    {
      grayValue = (int)gray4.at<uchar>( 250, c );
      printf("grayvalue = %d    c = %d\n", grayValue, c);
      if ( grayValue == 0 ){
        x += c;//gain the sum of c
        range ++;//calculate the number of pixel that satisfy grayvalue = 0
      }
    }
```

When the loop ended, the average value of column number which has gray value of zero could be calculated using “x” divided by “range”. However, when there is no pixel on reference row have the gray value of zero, “range” will be zero. To avoid zero appears in denominator, if “range” equal to zero after “for” loop ended, skip the calculation, and set up “x” equal to zero directly. Meanwhile, when no black track detected on the row, display “out of view” on the frame.

Function “putText” could only display string on the image while “x” is variable. To display coordinate of horizontal center of the black track, it is required to put variable “x” into a string using a “stringstream” shows below.

```
sstr << "( " << x <<", 250 )";
```

The position of the text were related to point (x, 250). In that case, when the point reaches the edge of video, displayed text might out of view. So, if the value of “x” was greater than 400, minus the horizontal ordinate for 200 to move back the text within the view of video.

Using a “while” loop to repeat all the image processing procedure. In that case, the video could be processed frame by frame to indicate the coordinate as figure6 shows.

Read the frequency and size of the original video to form a new processed video 1 : 1. Within the processing “while” loop use instruction “writer << im” to write every processed frame into the specified new video.

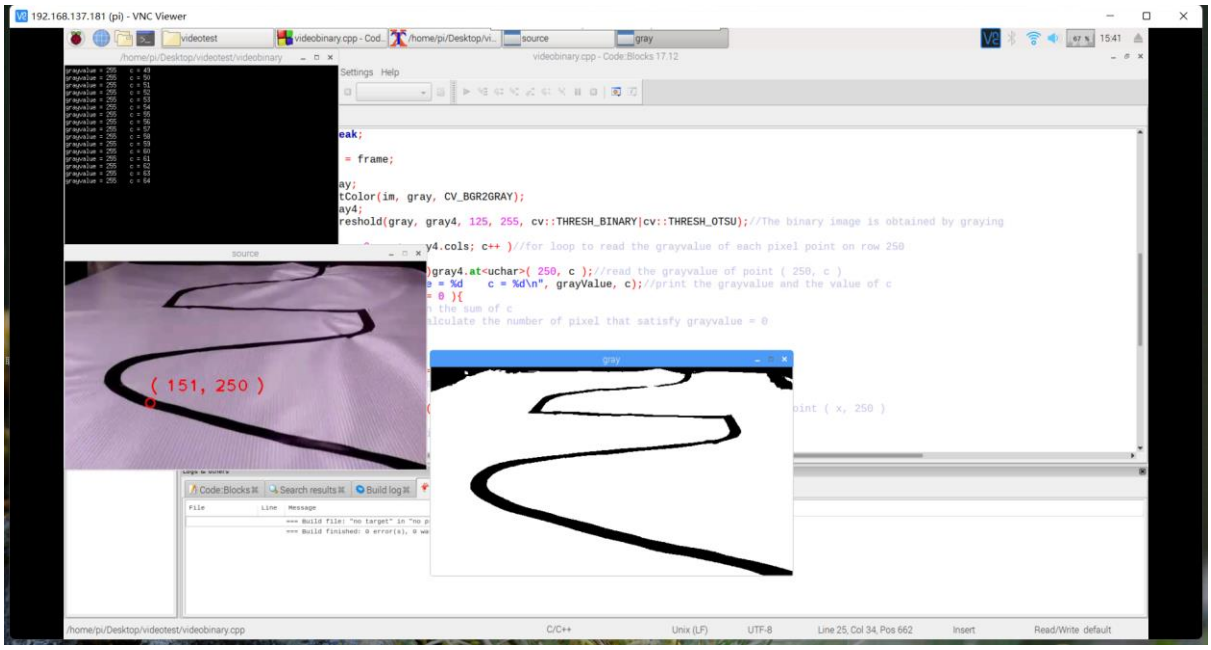

Figure6. Sample of processed video playing in programme

Run the programme once, after the video playing completed, the new processed video was stored into document “Processed video.avi” under the same folder as the programme.

DISCUSION:

Image process task was completed on raspberryPi OS and the processed video has uploaded to OneDrive. For the whole procedure, there are some future improvements could be made.

This experiment didn’t set up suitable resolution for original video recording beforehand but to resize the video after this video token. The resize procedure is not necessary, it possible to skip this procedure if the resolution has been properly set up for raspberryPi camera. This will reduce the calculation of programme and make it run faster. Following function could set the raspberryPi camera recording the video has with the resolution of 580 × 330.

system("raspivid -vf -o labvid3.h264 -t 15000 -w 580 -h 330 &");

Original video recorded by raspberryPi camera is in the format of “h264”. The processed video in this experiment was in format of “mp4” which requires extra transform. Future study of processing video in format “h264” is highly recommended.

The whole processing programme was written in C++ with several library access to opencv document. It will be helpful to study every line of the programme and get familiar with it instead of simply put everything together. Knowing the function of each sentence will helps the programme more effective and flexible.

Appendix I:

Processed video.avi

https://nottinghamedu1-my.sharepoint.com/:v:/g/personal/ssysy4_nottingham_edu_cn/EX4zTnlTRy1Pkpbaq_8kq_gBDG8pMcNNAoShWJjZ9vIfcg?e=5n8VUk

Appendix II:

```
#include <opencv2/opencv.hpp>
#include "opencv2/highgui.hpp"
#include <iostream>
#include <string.h>

using namespace cv;
using namespace std;

int main(int argc, char const *argv[])
{
   cv::VideoCapture cap;
   cap.open("labvid.mp4");//open the video

   if (!cap.isOpened())//in case video open failed
      return 0;

   cv::Mat frame;

   int grayValue, c, x, range;//initialize variables
   double fps = cap.get(CAP_PROP_FPS);//Get video frequency per stitch
  Size size = Size(cap.get(CAP_PROP_FRAME_WIDTH), cap.get(CAP_PROP_FRAME_HEIGHT));//obtain the video width and video height of the frame

  VideoWriter writer;
  writer.open("Processed video2.avi", VideoWriter::fourcc('M','J','P','G'), fps, size);//specify file location, encoder, frame rate, width and height

   while(1) {

      cap >> frame;
      if (frame.empty())//break the while loop when video all readied
         break;

      Mat im = frame;

      Mat gray;
      cv::cvtColor(im, gray, CV_BGR2GRAY);
      Mat gray4;
      cv::threshold(gray, gray4, 125, 255, cv::THRESH_BINARY|cv::THRESH_OTSU);//The binary image is obtained by graying

      for ( c = 0; c < gray4.cols; c++ )//for loop to read the grayvalue of each pixel point on row 250
      {
         grayValue = (int)gray4.at<uchar>( 250, c );//read the grayvalue of point ( 250, c )
         printf("grayvalue = %d    c = %d\n", grayValue, c);//print the grayvalue and the value of c
         if ( grayValue == 0 ){
            x += c;//gain the sum of c
            range ++;//calculate the number of pixel that satisfy grayvalue = 0
         }
      }
```

```
            if ( range == 0 ) x = 0;//if no pixel on that row has grayvalue = 0 let x = 0
            else x = x/range;//gain the average of c where satisfy grayvalue = 0

            cv::circle(im, Point(x, 250), 8, cv::Scalar(0, 0, 255), 2);//show a circle on point ( x, 250 )

            stringstream sstr;//initialize the display message

            if ( x > 400 ) x = x - 200;
            else if ( x == 0 ) sstr << "out of view";

            sstr << "( " << x <<", 250 )";

            cv::putText(im, sstr.str(), cv::Point(x, 230), CV_FONT_HERSHEY_PLAIN, 2, cv::Scalar(0, 0,
255), 2);//put the message on point ( x, 230 )

            cv::imshow("source", frame);
            cv::imshow("gray", gray4);//show the two processed frame

            writer << im;//write every processed frame into the specified location

            x = 0, range = 0;//reinitialize the value of x and range for the next loop

            cv::waitKey(50);

        }//end while

        writer.release();
        cap.release();

        return 0;
    }
```