



Advanced Computer Vision

Final Project

Lab : VPILab

Advisor : Cheng-Ming Huang(黃正民)

Student : Yu Cho(卓諭)

Student ID : 106318025

Data : 2018.01.15

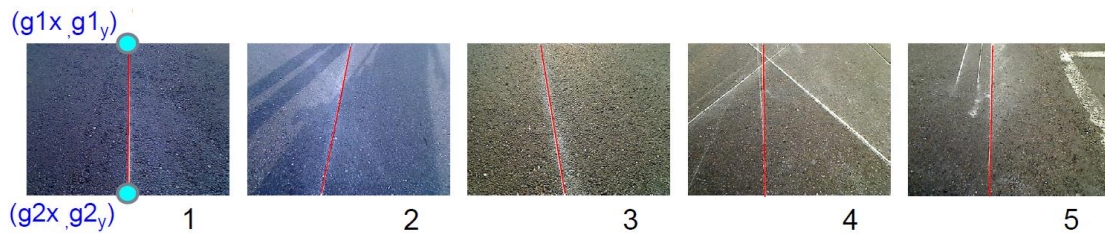
I. Problem

Design one algorithm to identify the specific lines in the image sequences.

1. Initialize by mouse selection or known position and size.
2. Use the same program and parameters for all videos.

Evaluation

1. Error = $\frac{1}{2} \left(\sqrt{(p1_x - g1_x)^2 + (p1_y - g1_y)^2} + \sqrt{(p2_x - g2_x)^2 + (p2_y - g2_y)^2} \right)$
2. Show the error trajectory (each frame) and average error (over all frames) of each video.
3. (Average) Computational time of one frame



II. Method

1. Environment

IDE: **Microsoft Visual Studio 2013**

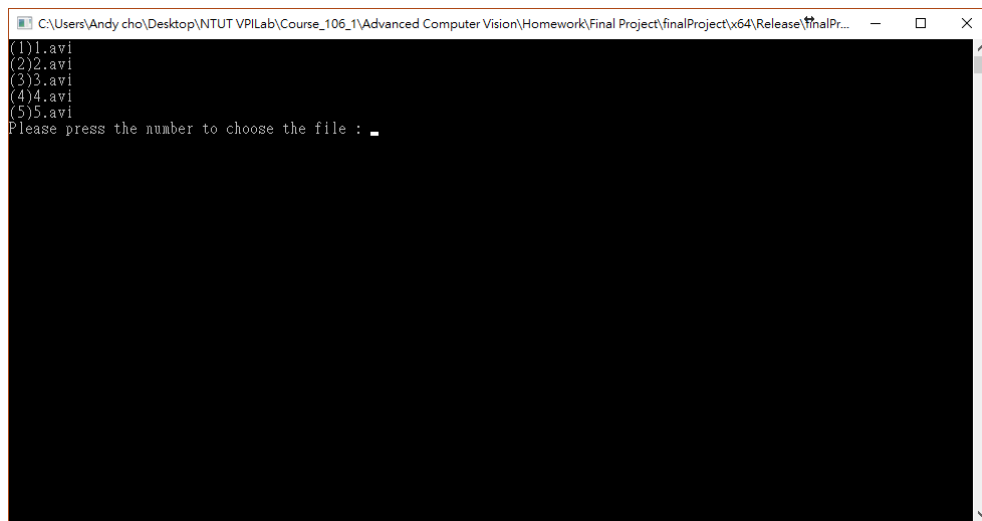
Library: **OpenCV 3.3**

2. Program interface

Input: **1.avi, 2.avi, 3.avi, 4.avi, 5.avi**

Output: **output.avi, average execution time, error per frame**

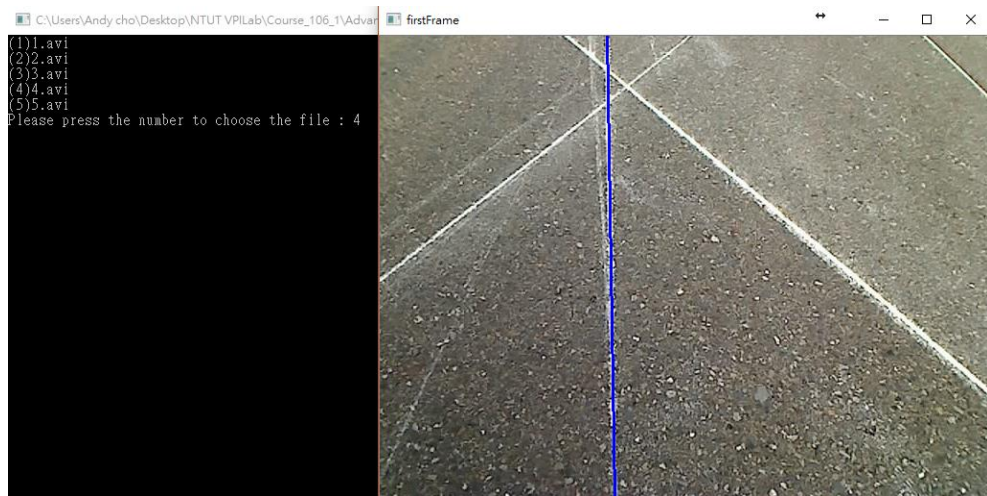
Step1. 選取追線影片介面



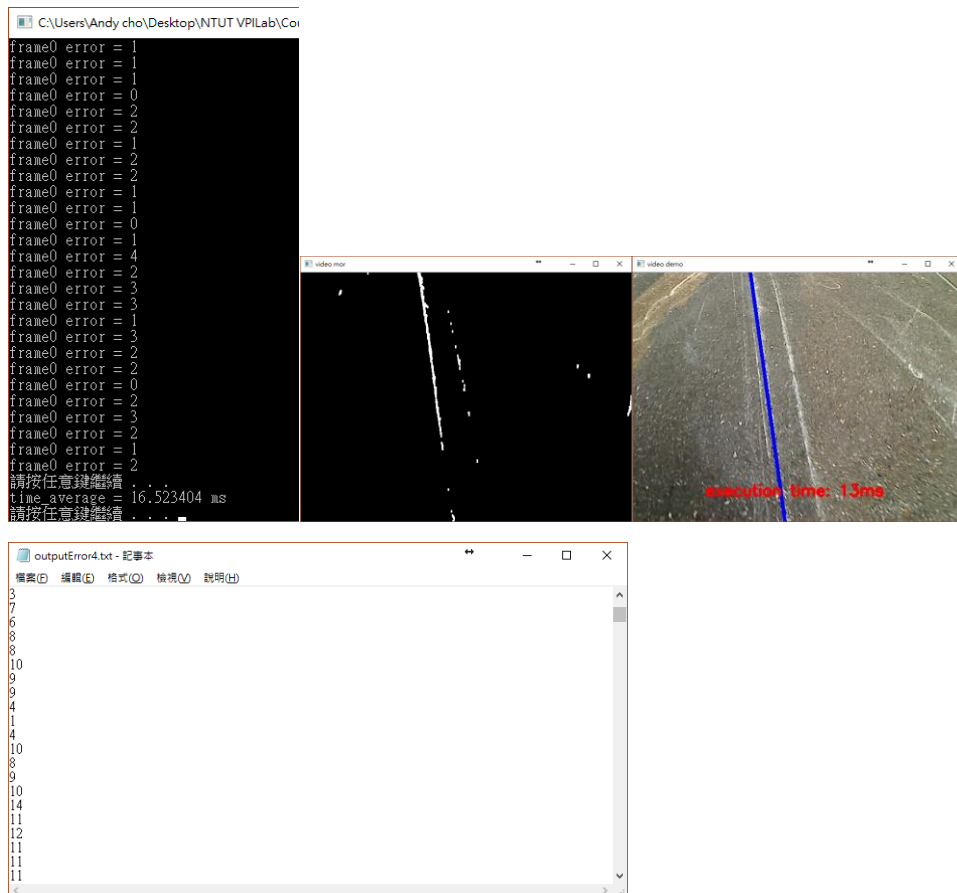
Step2. 輸入數字以選取欲追蹤的影片



Step3. 使用滑鼠右鍵選擇欲追蹤線段的第一個點，按住不放直到欲追蹤線段的第二個點時放開右鍵，並按下 ESC，系統即儲存初始線段兩頂點座標。

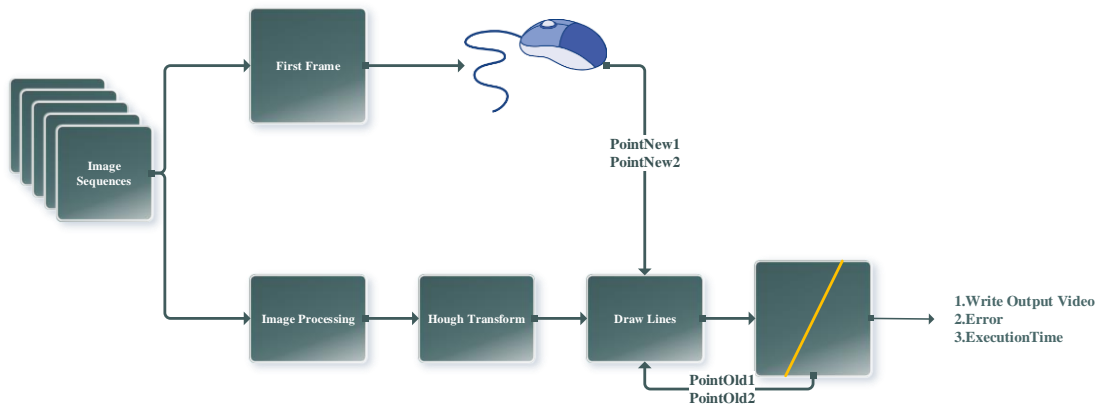


Step4. 開始追線，過程中會計算運行時間以及誤差量最終輸出追蹤影片、誤差量文件檔以及平均運行時間。



3. Architecture

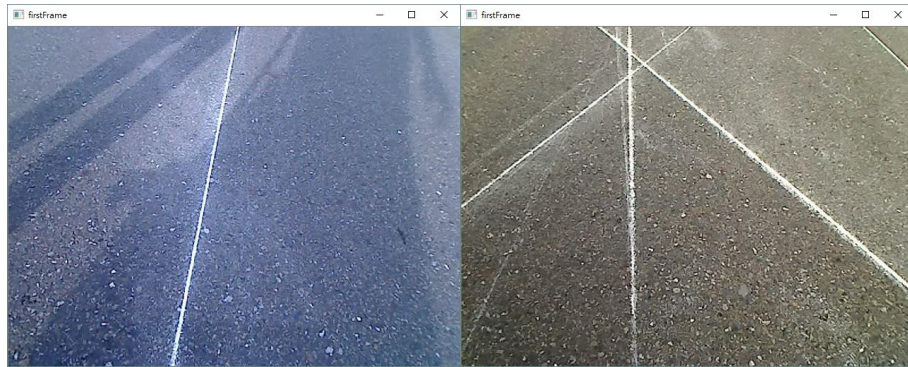
影片第一個 frame 進來後，使用滑鼠士檢決定初始的直線，之後系統會記錄該座標點，每一個 frame 進來後所找到的座標點都是 PointNew，而所有的 PointNew 會經由與 PointOld 相比後的最小平方誤差選出最小誤差的點，當作要畫出的直線，而判斷完後決定的最後一條要畫出的直線後，該直線所對應的座標點將成為下一個 frame 要比較的 PointOld，持續循環直到影片結束。



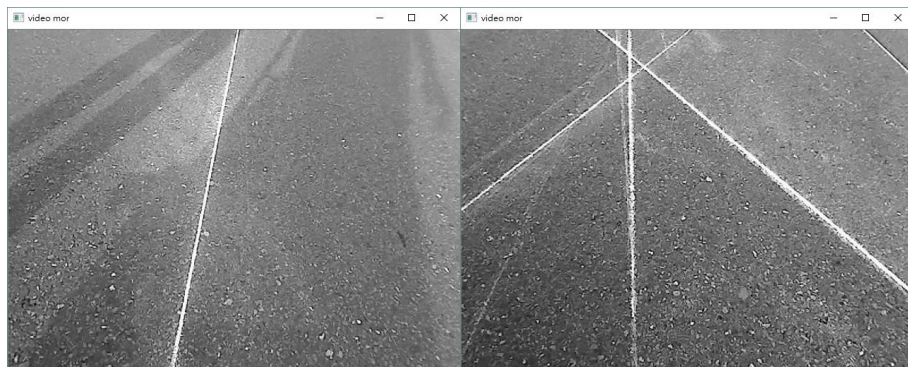
(圖一、系統架構圖)

4. Image processing

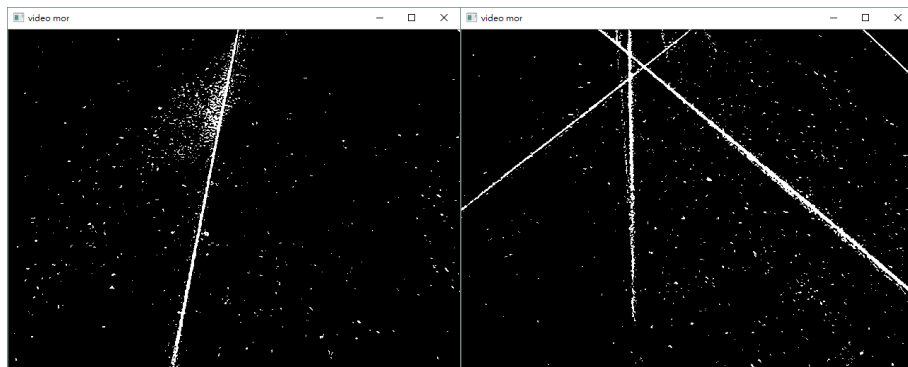
Original



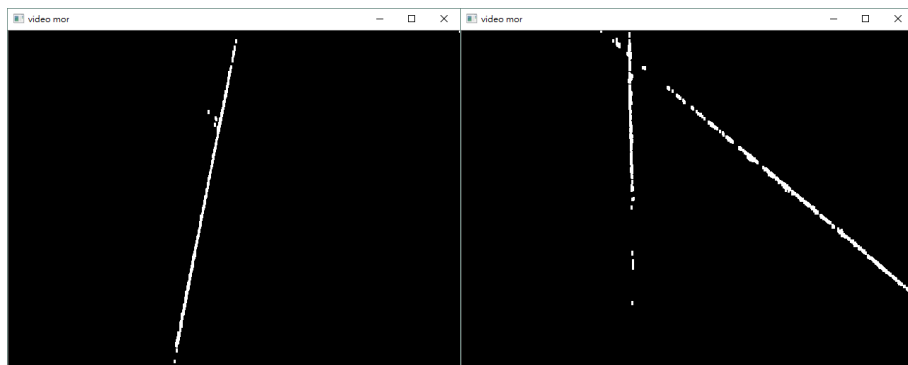
Step1. Grayscale(解決光影變化)



Step2. Binary(減少影像資訊)



Step3. Opening(去除影像雜訊)



5. Hough transform

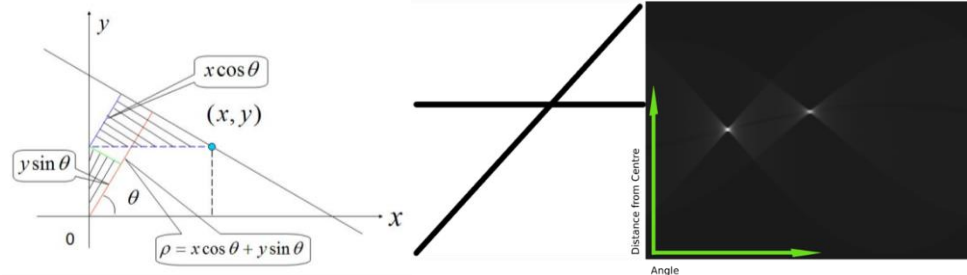
霍夫變換在電腦視覺中是一個特徵檢測的技術，此演算法最初的目的是找出物件的直線特徵，變化後的霍夫轉換不僅能識別直線，也能識別任何形狀，核心演算法概念為把圖像中影像的點集合(空間域)映射至參數坐標系(參數域)上，例如，空間域中一條直線映射至參數域中為一個點，空間域中一個點映射至參數域中則為一個弦波。

以下以極座標表示空間域的方程式(避面奇異點產生例如垂直線段造成的無限大效應):

$$\rho = x \cos \theta + y \sin \theta$$

其中兩參數的意義如下:

以下為霍夫轉換坐標系示意圖:



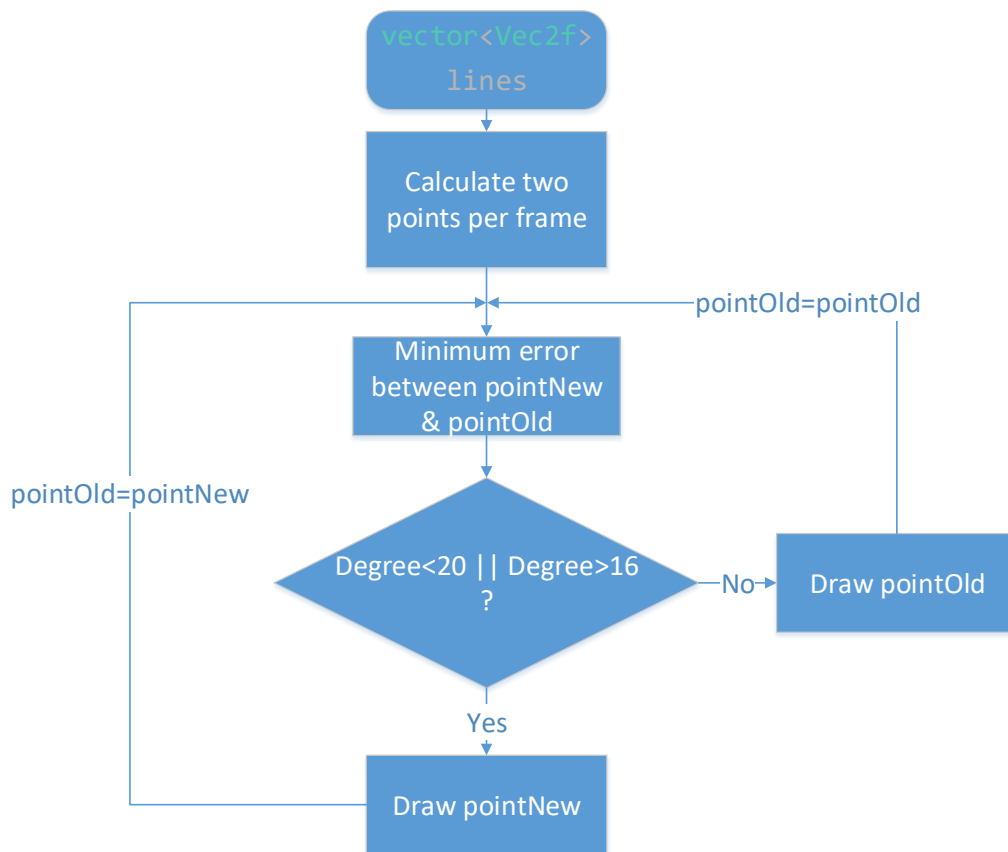
程式中使用 OpenCV 的函式實現霍夫轉換:

```
void HoughLines(InputArray image, OutputArray lines, double rho, double theta, int threshold, double srn = 0, double stn = 0)
```

- image: 輸入圖，8 位元單通道二值化圖。
- lines: 將所有線的資料存在 `vector< Vec2f >`，`Vec2f` 為每個線的資料，分別有 ρ 、 θ 這兩個參數， ρ 表示和左上角(0,0)的距離， θ 是線的旋轉角度，單位弧度，垂直線的 θ 為 0，水平線的 θ 為 $\pi/2$ 。
- rho: 距離解析度，越小表示定位要求越準確，但也較易造成應該是同條線的點判為不同線。
- theta: 角度解析度，越小表示角度要求越準確，但也較易造成應該是同條線的點判為不同線。
- threshold: 累積個數閾值，超過此值的線才會存在 lines 這個容器內。
- srn: 距離除數。
- stn: 角度除數。

6. How to draw lines


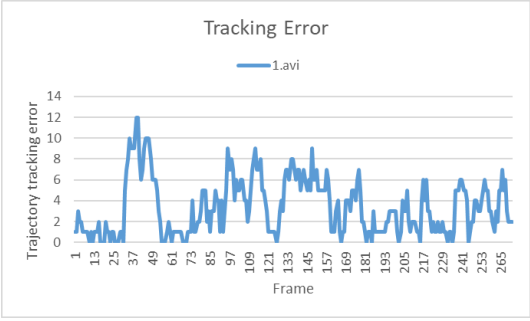

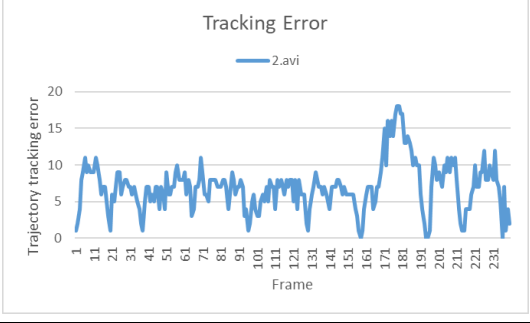
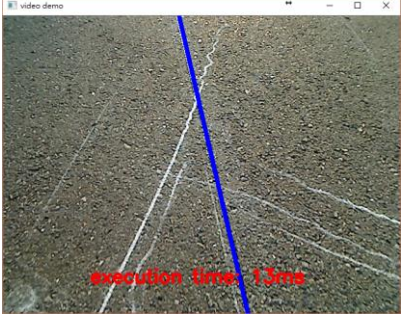
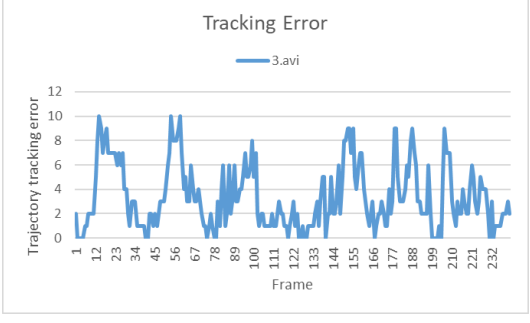

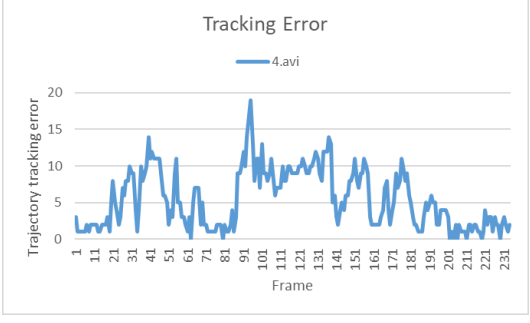

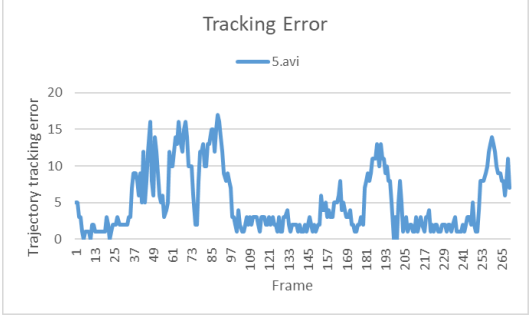
霍夫轉換函式會將每一個 frame 有的直線全部以 (ρ, θ) 方式儲存成向量，之後我會將其換算成兩點座標，並且找出所有的座標對(兩點連成一線)，其中與 pointOld 最小平方差之和差距最小的那一對座標為主要畫出的座標，再進入一個判斷式，若欲畫出的唯一直線的角度沒有小於 20 度或者大於 160 度，則依然保留前一 frame 畫出的直線不再畫新線，並且將最後畫在此 frame 上的直線對應的座標點儲存到 pointOld，下一個 frame 中再繼續與新座標做最小平方差找出離 pointOld 誤差最小的座標對。



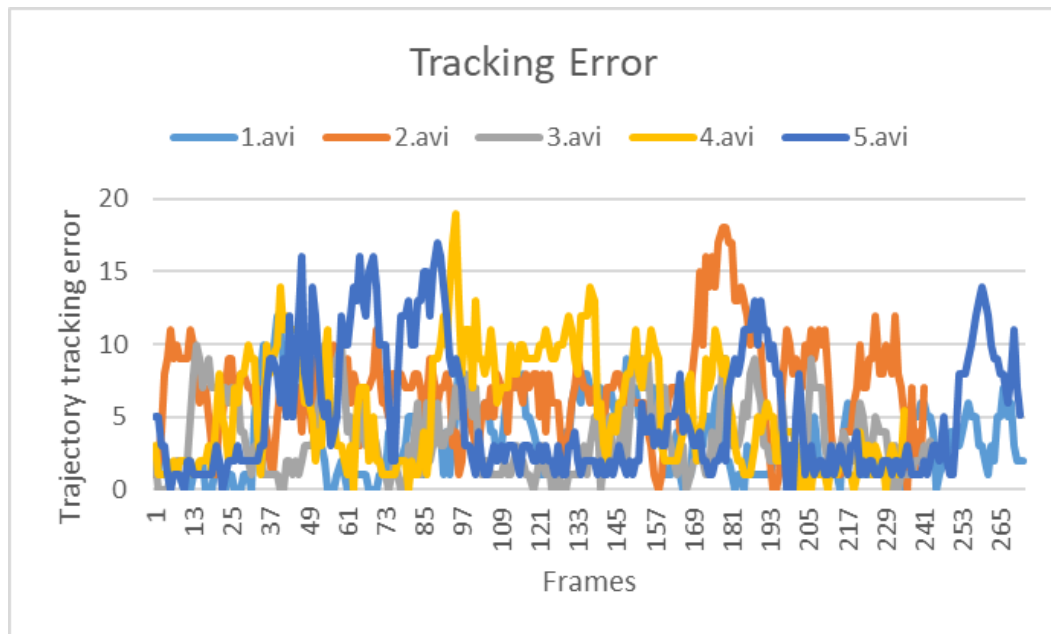
(圖二、Draw Lines 流程圖)

III. Result

(表一、輸出結果及誤差曲線)

	Video	Tracking Error
1.avi		
2.avi		
3.avi		
4.avi		
5.avi		

(圖三、總追蹤誤差曲線)



(表二、影片平均誤差及執行時間)

	1.avi	2.avi	3.avi	4.avi	5.avi
平均誤差	3.4	6.9	3.36	5.466	5.1
平均執行時間(ms)	14.135	14.327	14.72	16.468	17.646

結果影片連結:

output1: <https://youtu.be/ZGVv7Gzzbzg>

output2: <https://youtu.be/GRU4yhlLLjY>

output3: <https://youtu.be/MDWr7mR3GTk>

output4: https://youtu.be/aY_r86gWpjY

output5: <https://youtu.be/LkcARMiAWJ8>

IV. Discussion

一開始我先單純使用 OpenCV 的函式去找直線，去了解該函式的用法以及輸出結果，會發現霍夫轉換會找出整張影像所有可能的直線，使整張影像很雜亂，因此我是先將每張 frame 傳入我寫的一個影像前處理副程式中，裡面先將影像灰階再使用中值濾波器濾除路面柏油造成的雜點，之後將影像做閉合的形態學處理，最後使用 Canny() 找邊緣，此方法可以追蹤到直線，但是發現運算時間過長，因為這樣下來電腦掃了 3 次整張影像，十分沒有效率，最後我使用的流程為，**灰階化>>二值化>>形態學 Opening 處理**，灰階化主要的目的為使影像的資訊減少只使用單通道去分析，而二值化除了可以將影像資訊減少外，也可以避免光影變化所帶來的干擾，最後的 Opening 是為了將直線的輪廓更加連續，此方法依然可以將影像雜訊濾除而且比我最初使用的流程快上 3 倍左右。

經過處理之後會發現，影像中錯誤的直線追蹤減少很多，但會發現在正確的直線上，疊合了許多條霍夫轉換後的直線，為了要刪除這些直線只顯示一條，我設了條件式將所有找到的直線做比較找到與前一刻直線最小平方差最小的一條霍夫轉換直線，這樣可以正確的在影像中只輸出一條追蹤線，但會發現直線會產生過大的偏差，因此我增加了限制條件去輸出直線，避免此問題發生。

在最後的簡報中我的 error 在 4.avi 中最高達到了 46 個 pixels 但是**發現我使用的誤差公式與題目定義不同，在最初的誤差少除以 2，因此我全部追蹤影片的平均誤差可以直接少一半**。在 2.avi 影片中，會發現平均誤差最大，是因為在影像的前處理時，沒有將影像上方的雜訊處理完整，導致使用霍夫轉換時產生過大的誤差，而最大誤差量在 4.avi 中產生，達到了 18 個 pixels，原因是在該影片中，有交叉的直線干擾以及影像的前處理無法完整的濾除該影片的雜點，此兩項因素造成此結果。除此之外，**我的追蹤效果在平均誤差來看都在 5 個 pixels 左右，最好的達到了 3.3 個 pixels**。

未來若要將演算法做改善，我想要嘗試將 OpenCV 使用的霍夫轉換做改善，因為要掃描整張影像太耗費時間了，可以先設定影像的條件在去從通過條件的區域做霍夫轉換即可，而在影像前處理中我會嘗試使用型態學的 skeleton 去取出影像物件中的骨幹特徵，比較差別逐步改進處理流程。

V. Reference

1. <http://monkeycoding.com/?p=656>
2. <http://monkeycoding.com/?p=653>
3. <http://monkeycoding.com/?p=577>
4. https://en.wikipedia.org/wiki/Hough_transform
5. http://www2.lssh.tp.edu.tw/~hlf/class-1/lang-c/c_file.htm