111 學年度第 1 學期

機器手臂智慧視覺

Homework_2

基於道路車道線檢測與車輛辨識之自動輔助駕駛系統

姓名：　柯佐勳

學號：　611415055

系所：　電機所

教師：　賴文能老師

日期：　2023.1.15

一、簡介

這次的作業是要以影像的辨識與處理，來完成車子前方的車道線偵測，然而隨著車輛自動駕駛輔助系統的蓬勃發展，駕駛者希望能夠在最最安全且最舒適的方式，使車輛可安穩地在高速公路上行駛在車道中線以及與前車保有安全距離。由於在現實應用中會因為環境因素，如光線影響車道辨識的能見度降低，因此希望藉由本次的作業程式實作以及上課時吸收到的理論知識做活用，並研究了有關車道線檢測的相關論文[1]，設計一高準確率、穩定且多功能(辨識車輛種類、車道偏左偏右警示)的視覺車道線檢測自動駕駛輔助系統。

我將在以下章節分別介紹，本次作業實驗中所使用的理論、系統演算法、實驗的影像與結果分析，並對實驗過程中遇到的問題進行探討與分析。

二、方法

本章節將介紹在我在本次作業中所使用的方法，並對方法之理論進行解析，Fig. 1 為我設計基於影像做車道線檢測之流程圖，流程由三個階段所組成: (1)影像預處理，(2)設置自適應的 ROI 區域，(3)車道線檢測與實時追蹤。

一開始先將輸入之影像做影像之預處理(Grayscale、Gaussian Blur 減少影像中的雜訊)，再透過 Canny 演算法對影像進行全域的邊緣檢測，緊接著設置四點之局部 ROI 區域，只對車道之區域做偵測左右之車道線，此用意是為了移除非車道線的特徵，達到穩定的偵測結果，最後以 OpenCV 中的累計概率霍夫轉換(Progressive Probability Hough Transform; PPHT)函式對經過 ROI 設置的局部邊緣影像做車道線之匹配偵側，而 PPHT 算法為標準霍夫轉換算法的優化，透過三個閥值分別為累加器中的投票像素數量、最小之線長度與左右兩條線之間距，以隨機採樣邊緣點並提取直線的兩個端點，得到最佳的車道線檢測，不會去顯示過多的直線在影像中，基於縮小化採樣點數量之數據集，來減少搜索點的計算量，進而縮短推算之時間。

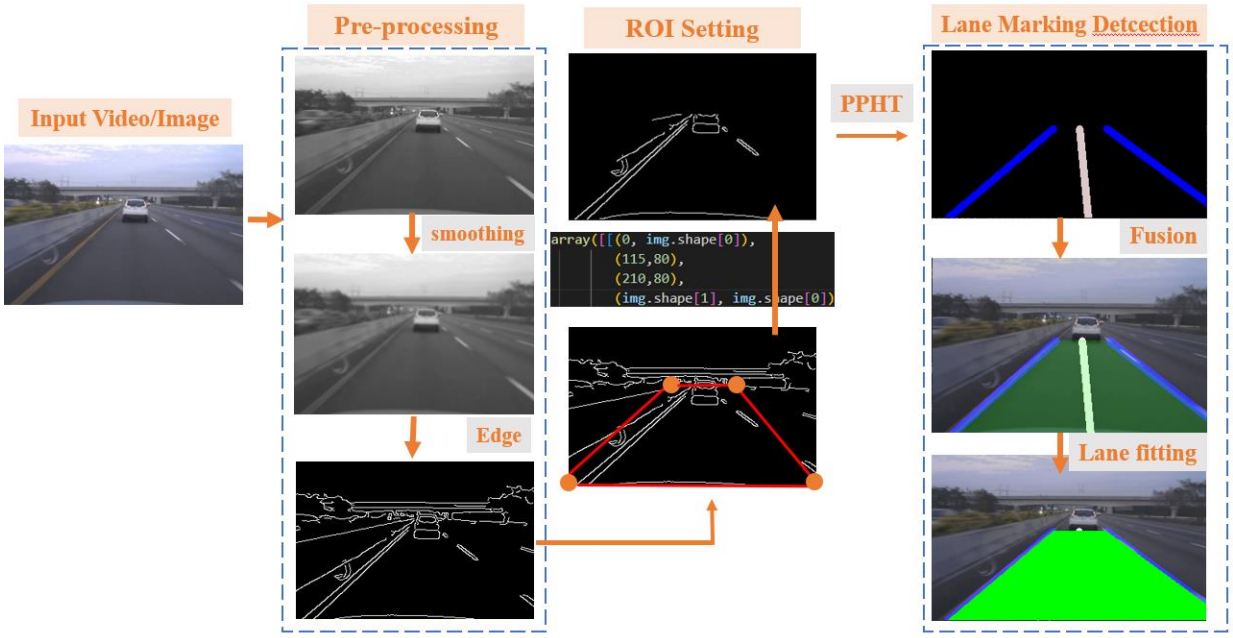

Fig. 1 車道線檢測之影像流程圖

2.1 Canny edge detector

Canny 邊緣檢測器是一種邊緣檢測演算法，可以顯著減少圖像中檢測到的錯誤邊緣的數量，透過雜訊的處理與邊緣定位有著良好的表現，而 Canny 的輸入為 Sobel 濾波的效果，首先經過影像灰階化與 Gaussian filter 消除影像之雜訊處理，這時透過 Sobel operator 公式(2-1)，求出影像之水平軸(x-axis)與垂直軸(y-axis)上之一階導數梯度值。

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad , \quad G_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \tag{2-1}$$

求得G_x與G_y梯度後，利用公式(2-2)做平方相加並開根號可得到邊緣的梯度強度以及利用公式(2-3)之反三角函數 arctan 求得邊緣的梯度方向角度。

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

$$\theta = \arctan(\frac{G_y}{G_x}) \quad (2\text{-}3)$$

Canny edge 方法最大的特點在於非極大值抑制(Non-maximum Suppression; NMS)以及雙閥值之設計，而透過公式(2-1))到(2-3)之計算後，通常邊緣會出現在梯度較高的地方，而雙閥值的設計是給予上、下兩個閥值來判斷該 pixel 是否為邊緣，因此此方法在於影像的邊緣檢測處理上，擁有較佳的效果。

2.2 Hough transform

Hough Transform (HT)為一種特徵提取技術， 其可用於隔離圖像中特定形狀的特徵的技術，應用在圖像分析、計算機視覺和數字圖像處理領域。目的是通過投票機制在特定類型的形狀內找到對象的不完美實例。而此投票機制是在一個參數空間中(也可稱之為累加器空間)進行的，在這個參數空間中，候選對像被當作所謂的累加器空間中的局部最大值來獲得，所述累加器空間由用於計算霍夫變換的算法明確地構建，完整轉換過程如 Fig. 2 所示。

霍夫轉換主要優點是能容忍特徵邊界描述中的間隙，並且不會受到影像雜訊的影響。而本次作業用此技術來檢測直線，而直線的方程式可由斜率與截距作表示，如公式(2-4)之斜截式。

$$y = mx + b \quad (2\text{-}4)$$

考量到垂直線之斜率不存在，因此為了避免發生斜率無限大之問題，使用另一種直線表示式，Hesse 法線式做為直線之描述，如公式(2-5)。

$$\rho = x\cos\theta + y\sin\theta \quad (2\text{-}5)$$

Hough 轉換就是一個找尋(ρ, θ)座標交點的演算法，ρ 為原點到直線上最近點之距離，θ 為 x 軸與連接原點與最近點直線之間的夾角角度。當找到此(ρ, θ)交點後，轉換為(x, y)座標後，就會在影像空間中形成一條直線了。

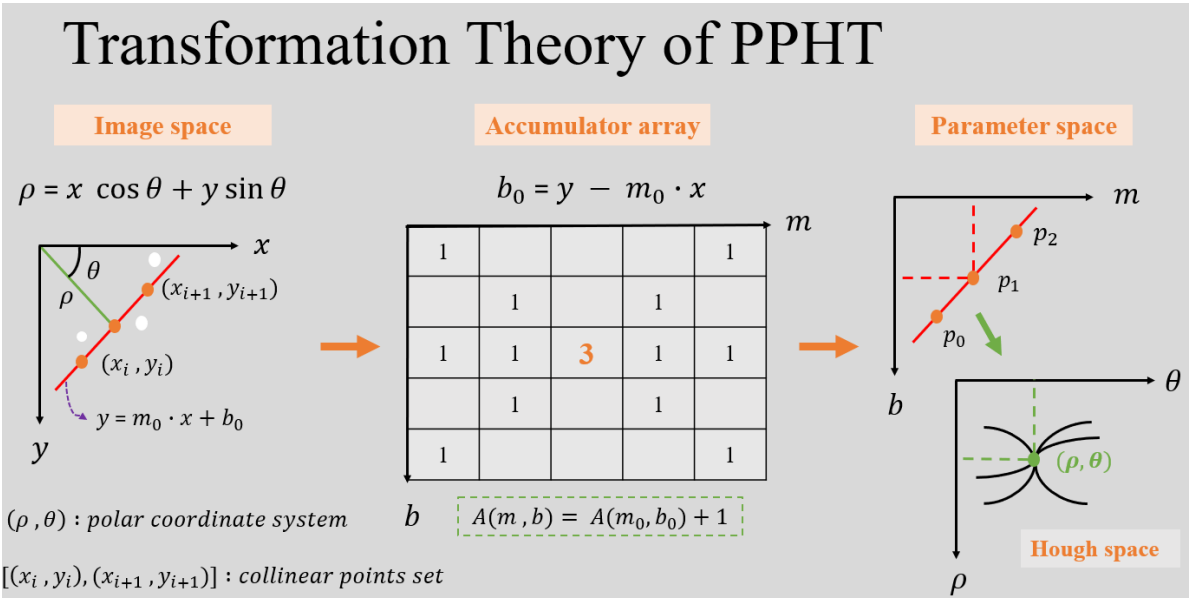

Fig. 2 霍夫轉換概念示意圖

2.3 YOLOv5

YOLOv5 是一 one-stage 的物件偵測方法，顧名思義就是只須對圖像做一次 CNN 架構找出符合的物件，判斷出圖像內哪個區域有相符合的物件，且機率最高，並對應該區域以方框做標註與顯示物件類別名稱。

而 YOLO 的實際做法是先將圖像平均成 S×S 的網格(grid cell)，當被偵測物件的中心位於某個網格內時，則該網格就要負責去偵測該物件，而每個網格需要負責預測數個 bounding box 以及屬於各個類別之機率，其中每個 bounding box 會帶有五個預測值分別為(x , y , w , h 與 confidence)，x , y 為某個物件在該網格之中心點座標，該物件相對應的寬與高為 w , h，confidence 是用於表是該物件是否為一個物件的信心程度(confidence score)，而 confidence score 在 YOLO 論文中被定義為預測框(bounding box)與 Ground Truth 框的 IOU (重疊率)值。

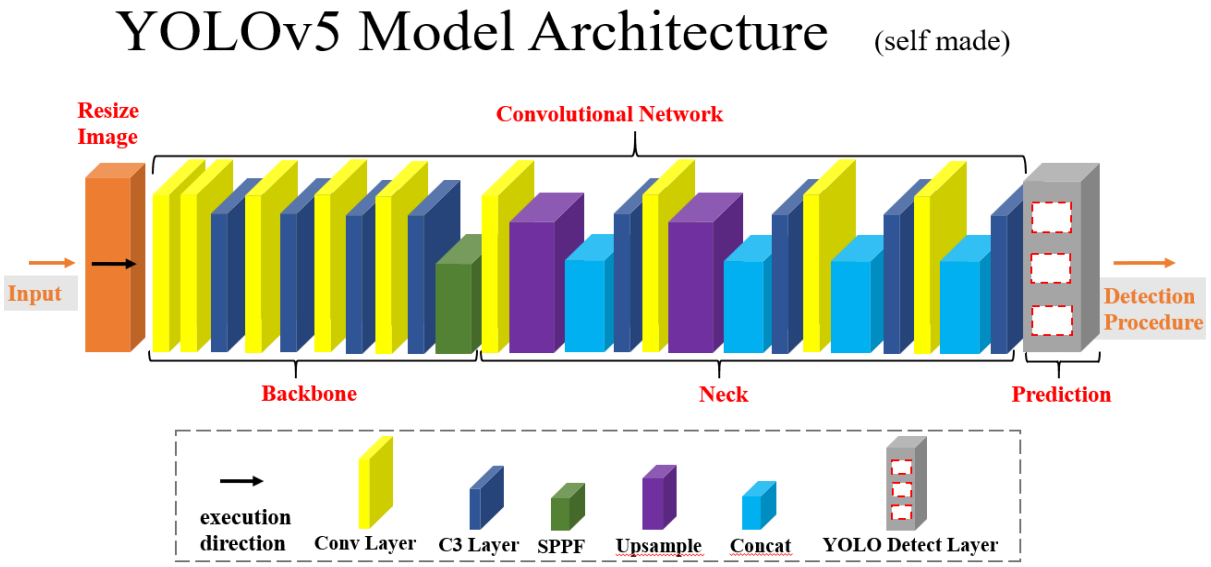

Fig. 3　YOLOv5 之網路模型

在本次作業中我使用了 YOLOv5 之網路模型，如 Fig. 3 所示，將自己建立的標籤數據集(car.yaml:內有 car 與 track 兩種類別標籤)以及圖片做為輸入，並採用 YOLOv5 提供之 YOLOv5s 模型，來進行模型之訓練並得到訓練結果，如 Fig. 4 所示。而 YOLOv5 之網路模型中，CNN 網路可分為三個階段或稱為層:

(1) Backbone layer :內含有卷積層、C3 層與 SPPF 網路，以提取輸入圖像中的特徵，類似於物件檢測器之功能。

(2) Neck layer : 用於提取 SPPF 網路中的得到的圖像特徵，內含有上採樣層(Upsampling layer)作為改善模型中較微弱、不明顯特徵之傳播，以及蒐集 Feature map。

(3) Head layer : 呈現最後的預測結果，內含 bounding box 座標向量等資訊。

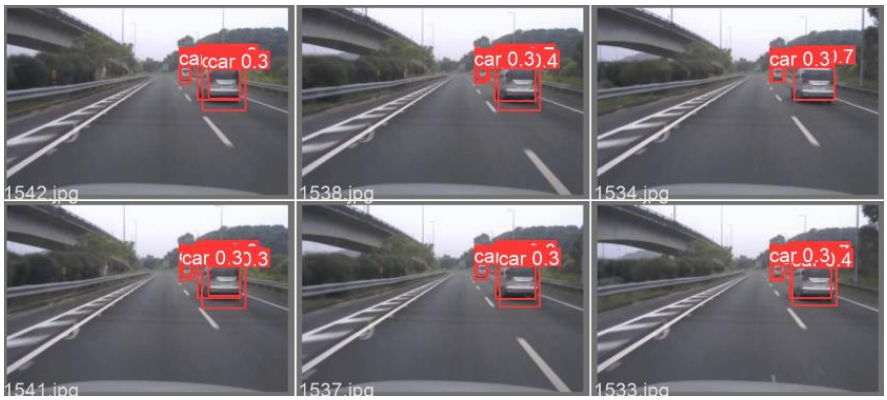

Fig. 4　YOLOv5 之訓練結果示意圖

三、實驗結果與討論

在本次作業實驗中，使用了三組不同高速公路之車道影像作為實驗對象，如 Fig. 5 所示，其中一組影像中擁有汽車以及卡車，可作為車輛種類辨識之測試，有一組影像之車道較為特殊，在直行車道線偵測的過程中，會因交流道的網格線導致檢測錯誤，如 Fig. 13 所示。

Fig. 5 本次作業實驗所使用的三組影像測試集

3.1 實驗環境設置

本次作業實驗中所使用之硬體設備與實驗環境，如 Table. 1 所示。

Table. 1 實驗環境設置

Operating System	Windows 10 64 bits
CPU	Intel® Core™ i5-8265 @ 1.60 GHz
GPU	NVIDIA GeForce MX250 2GB
IDE	Visual Studio Code
Open source libraries	PyTorch 1.13.1、opencv-python 4.5.5、 yolov5 6.1.0、tensorflow 2.8.0、numpy 1.20.0
Programming Language	Python 3.9.7

Table. 2 為使用 YOLOv5s 之網路模型做影像與數據訓練所設置的超參數，由於 Fig. 5 中中間的影像序列集內有兩種不同款式之車種，因此設置的標籤總共有兩種['car','track']，在 IOU(Intersection over Union)閥值的設置則會去影響檢測框的精確度，因此為了避免在框取車輛時，有過多的不必要的眶出現，因此從原先設置的 0.45 提升至 0.5，使推測出的物體區域越準確。

Table. 2　YOLOv5s 模型訓練之超參數設置

Parameters	Values
Training epoches	50
Batch size	5
Learning rate	5
IOU threshold	0.5

而網路模型則選用 YOLOv5.pt，因為其網路深度、尺寸與其他模型相比之下較小且為檢測速度最快之模型，可藉由 Fig. 7 來驗證，因此採用此模型來執行本次作業中的車輛檢測功能與標記。

Backbone	Pytorch(ms)	TensorRT_FP16(ms)
yolov5n-0.5	7.7	2.1
yolov5n-face	7.7	2.4
yolov5s-face	5.6	2.2
yolov5m-face	9.9	3.3
yolov5l-face	15.9	4.5

Fig. 7 比較 YOLOv5 所提供的不同模型之推理速度

來源: https://github.com/deepcam-cn/yolov5-face

3.2 Canny edge detection

下圖 Fig. 8 (a)、(b)為本次作業實驗中進行的第一階段:影像預處理，影像在經過灰階與高斯濾波去雜訊後，使用 Canny 演算法做影像之邊緣檢測，如 Fig. 8 (b)所示，可以清晰地透過影像觀察出車道線以及車子的邊緣。

Fig. 8 (a) 原始影像

Fig. 8 (b) Canny edge 檢測結果圖

3.3 Hough transform

下圖 Fig. 9 (a)、(b)為本次作業實驗中進行的第二與第三階段: 設置自適應的 ROI 區域以及車道線檢測與實時追蹤，對 Fig. 8 (b) Canny 邊緣檢測結果之影像，透過設置四個端點之 ROI 區域，做四邊形遮罩(也可為多邊形)與融合處理後，可得到自適應設置之 ROI 區域，如 Fig. 9 (a)所示，此目的是為了將重點放置在檢測車子所行駛的車道中，能盡可能地減少其他不重要區域之干擾，進而提升 Hough transform 做檢測的穩定性。而 Hough transform 經過搜索、累計投票與運算後，在影像中將得到的最佳直線做呈現，如 Fig. 9 (b)所示，準確地將車道線與車道區域做顯示，可看出此方法的檢測效果是非常好的。而我為了增加系統的靈活性，在車道中間顯示一條白線，作為偵測車子目前的方位是靠左或靠右的情況，進而提醒駕駛者該靠左或靠右做穩定地駕駛在自己的車道中。但由於老師所提供之影像，得出的白線斜率值並沒有太大的差異，因此我將以自己在網路上所截錄的影片做偏左偏右之警示結果，如 Fig. 10 (a)、(b)所示。

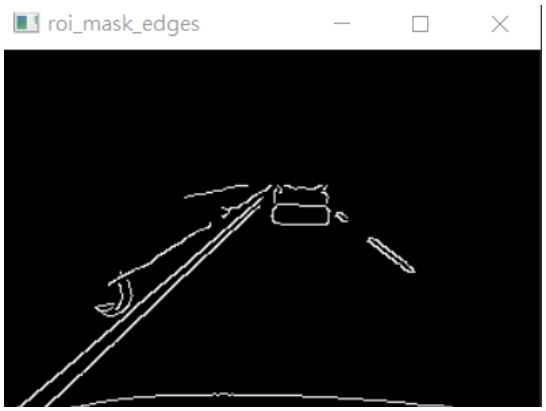

Fig. 9 (a) 自適應設置 ROI

Fig. 9 (b) 車道線與車道之顯示結果圖

Fig. 10 (a) 當車輛太靠車道之右邊時，
告知駕駛者需靠左行修正行駛之警示通知

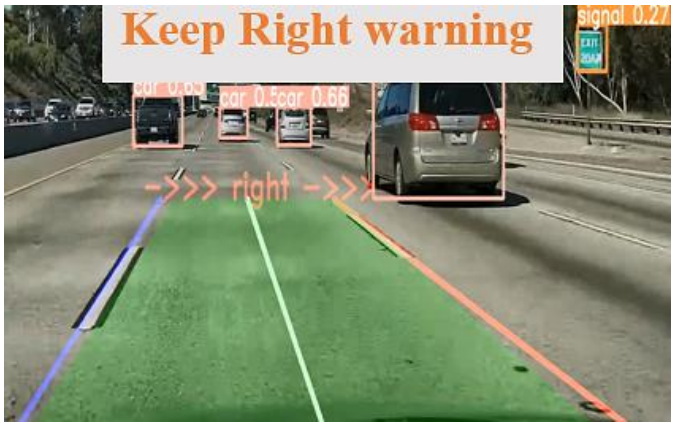

Fig. 10 (b) 當車輛太靠車道之左邊時，
告知駕駛者需靠右修正行駛之警示通知

3.4 YOLOv5

Fig. 11 為經過 YOLOv5 網路模型之訓練與檢測之影像流程圖，可發現在訓練完後，會先以 One-Hot encoding 方式初步對資料集中的標籤類別名稱以數字做表示，如標籤 ‘car’為 0，‘track’為 1，以利後續在檢測時的檢測框顯示以及標籤類別名稱之顯示，如 Fig. 12 所示。

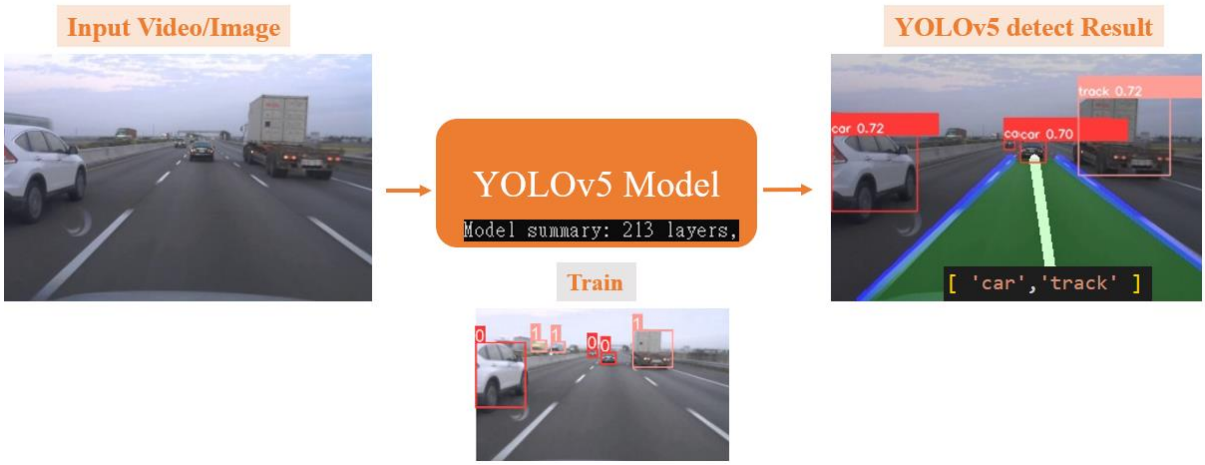

Fig. 11　YOLOv5　模型之訓練與檢測影像結果示意圖

Fig. 12　YOLO v5 偵測車輛與辨識車輛種類，並以方框對物件框取顯示

四、問題陳述與分析

在這次的作業實驗中，我參考了論文[1]的方法步驟，先對影像預處理(灰階、Gaussian 去雜訊、Canny edge detector)，設置自適應的 ROI 區域，以利執行霍夫轉換的執行，提升檢測線的精確度以及消除不必要的區域，最後顯示車道線檢測與實時追蹤之結果。

雖然我已有對影像做許多的處理，以及調整 Canny edge 之上下閥值甚至測試了多組累計概率霍夫轉換之三項參數，還是沒辦法使每一幀擁有最佳的車道線檢測之結果，因此我統計了在偵測時發生錯誤的機率次數統計，如 Table. 3 所示，而線檢測錯誤含括為未檢測到線以及檢測到非該車道線之情形，如 Fig. 13 所示。

Table. 3　車道線檢測誤判率

影像測試集序列	偵測正確數	偵測錯誤	誤判率
01	40	8	16.6%
02	38	10	21%
03	16	10	**38.4%**

Fig. 13　車道線偵測錯誤示意圖(影像測試集序列:03)

4.1　改善車道線誤判之狀況

在 Table. 3 中可發現影像序列為 03 之測試集，線檢測之錯誤率相較於其他兩個影像測試集高了 17.4~21.8%，而我發現不只會受到左側之網格線影響直線之形成，如 Fig. 13 左圖所示，也會因為有某些未經處理之雜訊點顯示在影像中，導致直線檢測到別的車道了，如 Fig. 13 右圖所示。因此我將增加影像處理之功能，採用透視轉換(Perspective Transform)將自適應之 ROI 車道區域轉換為平面影像，如 Fig. 14 所示。

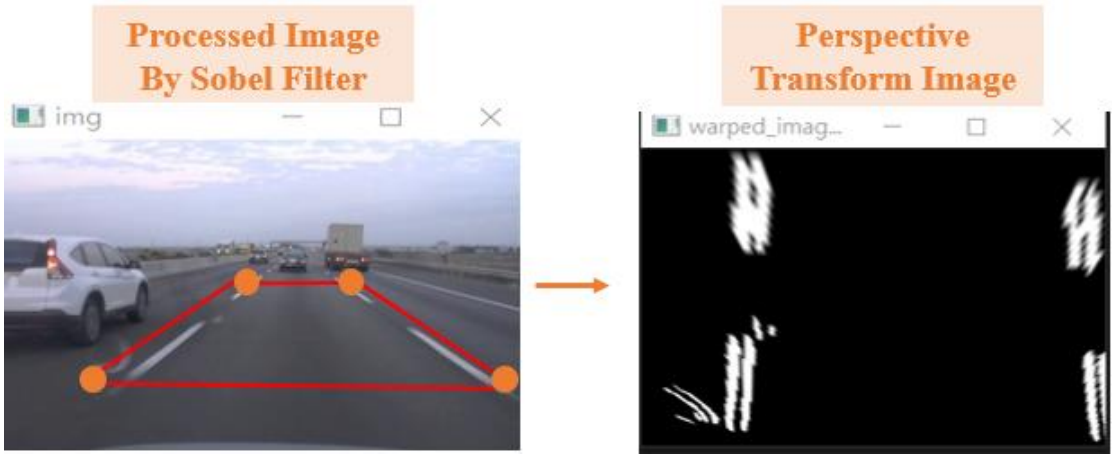

Fig. 14　增加透視轉換使直線偵測區域專注在車道上

為了增加車道線檢測之穩定性與降低誤判率，因此我參考了其他應用在車道線檢測之相關論文[2]的方法，將我的作業實驗以新的演算法機制做一個效果優化，而另一種線檢測之演算法影像流程圖如 Fig. 15 所示，第一階段一樣是做影像之預處理，但邊緣檢測方法則改用 Sobel operation，並且以透視轉換將該車子所行駛之車道做'Bird-eye view"(俯瞰圖)之變換；而下一階段則是做車道線之偵測，透過連續的 sliding window 對車道線做擬合匹配， sliding window 之數量是可調的(本實驗設置 8 個滑動視窗)，若設置的數量越多，越能準確且嚴謹地追蹤車道線的實時變化。最後再將我們的車道線偵測結果透過反透視矩陣，將偵測車道之區域結果擬合至原影像中做觀測。Table. 4 中為使用新的演算法做車道線偵測之錯誤率統計，可發現錯誤率均壓縮至 10%以下，雖然我沒將車道線繪製在影像中，但我想也可透過偵測車道之區域，來得知車輛位置與車道中心位置的偏差，當偏差過多時，就必須提醒駕駛者須將車輛穩定地行使在車道中線，以達到安全駕駛之目的。

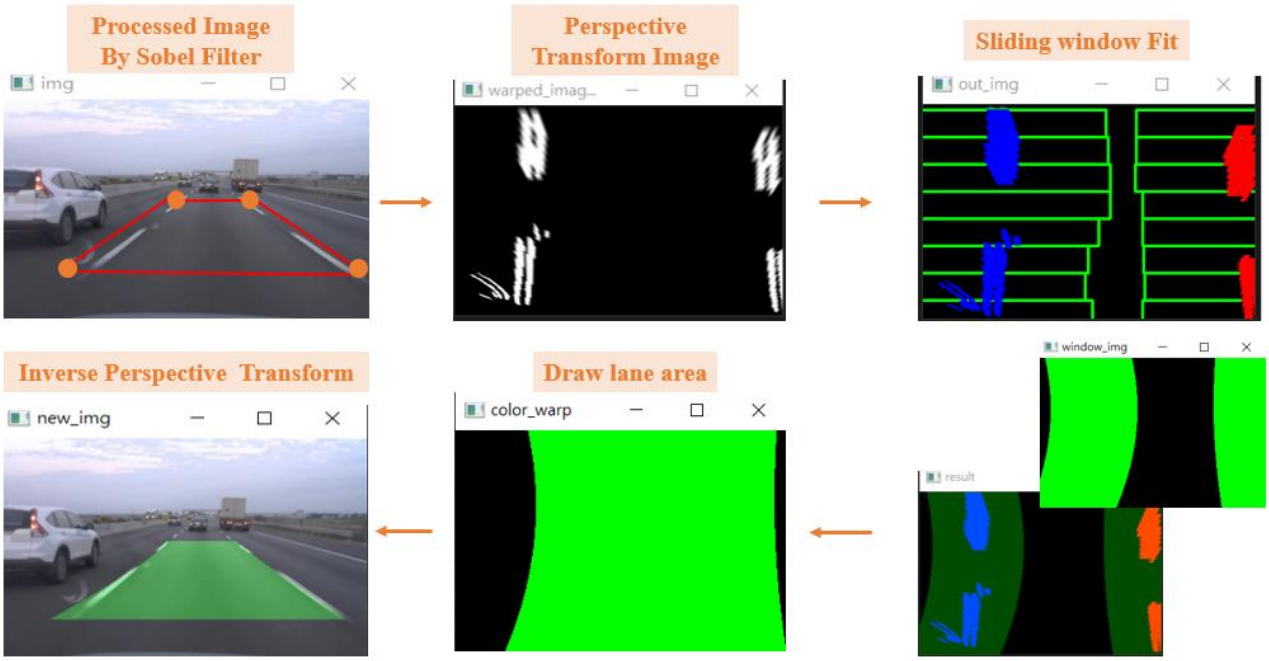

Fig. 15　以另一種直線檢測方法(Sliding Window fitting)做車道線追蹤

Table. 4　使用 Sliding Window fitting 做車道線檢測之誤判率

影像測試集序列	偵測正確數	偵測錯誤	誤判率
01	47	1	2 %
02	46	2	4.1%
03	24	2	7.7%

參考文獻

[1] M. Marzougui, A. Alasiry, Y. Kortli, and J. Baili. A Lane Tracking Method Based on Progressive Probabilistic Hough Transform, In Proceedings of the IEEE Access,2020.

[2] H. Zhu. An Efficient Lane Line Detection Method Based on Computer Vision, Journal of Physics: Conference Series ,2021.