# 國立中正大學

# 電機工程研究所

# 機器手臂智慧視覺

# 作業(二)

影像車道辨識與車距偵測實作

Lane detection and car distance measurement implementation

指導教授: 賴文能 博士

研究生: 黃士懷、黃柏涵

民 國 一 一 三 年 一 月 十 六 日

# 目錄


1. Introduction ..... 3
2. Method ..... 3
   - 2.1. 車道線偵測 ..... 4
     - 2.1.1. Canny edge operator ..... 4
     - 2.1.2. Hough transform ..... 5
   - 2.2. 車輛偵測 ..... 6
   - 2.3. 車距估測 ..... 8
     - 2.3.1. Look-up table ..... 8
     - 2.3.2. 焦距推算估測 ..... 9
3. Experiment & Result ..... 10
   - 3.1. 實驗環境 & 資料集 ..... 10
   - 3.2. 實驗結果 ..... 10
4. Conclusion ..... 13
5. Reference ..... 14


# 圖目錄


*圖表 1、實驗流程圖 ..... 4*

*圖表 2、Canny 處理後的影像 ..... 5*

*圖表 3、Hough transform 座標轉換示意圖 ..... 5*

*圖表 4、Hough transform 處理後的影像 ..... 6*

*圖表 5、車道線示意圖 ..... 6*

*圖表 6、YOLO v7 與其他模型的比較 ..... 7*

*圖表 7、YOLO 車輛偵測示意圖 ..... 7*

*圖表 8、取樣影像與實際距離標記 ..... 8*

*圖表 9、Look-up table ..... 8*

*圖表 10、針孔成像示意圖 ..... 9*

*圖表 11、推算焦距用影像 ..... 9*

*圖表 12、圖表 11 放大圖 ..... 9*

*圖表 13、無前車結果圖 ..... 11*

*圖表 14、Look-up table 估測影像 ..... 11*

*圖表 15、焦距推算估測影像 ..... 12*

# 1. Introduction

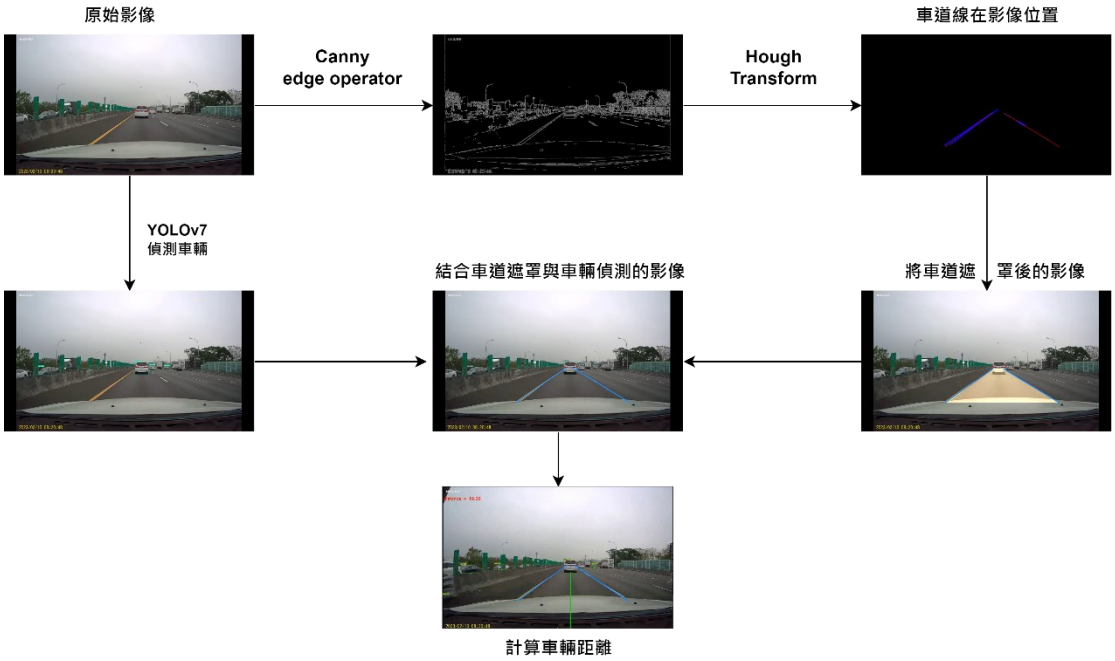
隨著科技不斷的進步，在這個社會當中，交通的便捷性已成為一個人們不斷追求的目標，其中，自動駕駛這個課題更是備受矚目，各大廠牌都耗費大量人力資源和成本在發展這個技術，而目前自動駕駛這項技術多以雷達、光達、GPS 以及電腦影像訊號對周遭環境進行感測和運算，判斷合適的導航路線亦或是判斷障礙物等，並將運算出的控制結果轉換為對車體的控制訊號並加以輔助或是操控。

在本次的作業實作當中，我們將使用上述所提到的電腦影像訊號作為輸入，並使用電腦視覺方法進行處理，將行駛途中的行車紀錄器影像中的車道和前車進行標示，並透過標示計算出與前車的實際距離，藉此實作了解到部分自動駕駛技術所使用到的影像視覺方法的實作以及應用。這是我們的程式架構：https://github.com/shih-huai/Car-distance-estimation.git。

# 2. Method

本次實作的目標是標記行駛中行車紀錄器所錄製的影片中的車道與車輛，並根據所偵測出的前車在影像中的位置，計算實際上與前車的距離。可以將實驗分為三個步驟: 車道線偵測、車輛偵測以及前車距離估算，演算法流程如下表所示。

| **Algorithm 1 : Lane detection** |
|---|
| **Input** : one frame from video |
| **Output** : distance of front car |
| 1. start |
| 2. Use Gaussian filter blur image |
| 3. Use Canny edge operator catch edge |
| 4. Hough transform get all lines |
| 5. Choose the specify slope |
| 6. Draw the car lane and front range |
| 7. Detect car by YOLO v7 |
| 8. If Ensure car is in the front range :<br>estimate the distance of front car |
| 9. end |




圖表 1、實驗流程圖

## 2.1. 車道線偵測

首先我們對輸入的影像使用 Gaussian Filter 去除影像中的雜訊後，使用 Canny edge operator[1]檢測出影像中的邊緣。，再將我們所感興趣的範圍進行遮罩(也就是車道的位置)，之後再使用 Hough Transform[2]來找出影像中的所有直線的兩端點座標，並對其進行篩選，選出斜率合適且位置相符的斜線位置作為車道的左右邊線，再透過這兩條邊線的位置和設定好的前後範圍生成車道的範圍。

### 2.1.1. Canny edge operator

Canny 邊緣檢測[1]是一種常用於圖像處理的多階段算法，其內容包含高斯平滑、梯度計算、非極大值抑制、雙閾值檢測和邊緣跟蹤等步驟，透過這些步驟可以達到保留真實邊緣並降低雜訊的影響。在使用 Gaussian filter 將影像去除雜訊的之後，利用 Sobel operator (式 2-1) 得到水平以及垂直方向的梯度。

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

而實際的梯度值就是將水平和垂直梯度進行平方相加，最後將其開根號就可以得到梯度的強度，而梯度的方向則須用 arctan 函數 (式 2-2) 來求出。

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

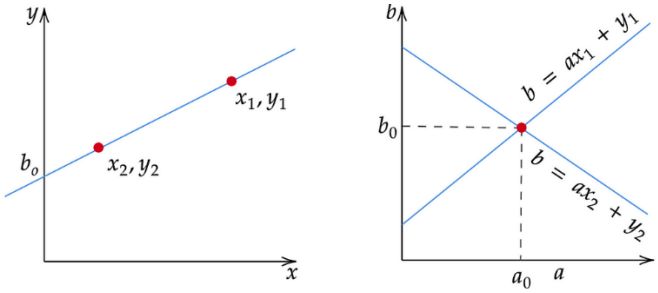
邊界在影像中通常代表著有大幅度變化，因此邊緣的梯度強度也會很高，因此透過設定閾值的方式即可將非邊緣的部分排除掉，只留下影像的邊緣。

圖表 2、Canny 處理後的影像

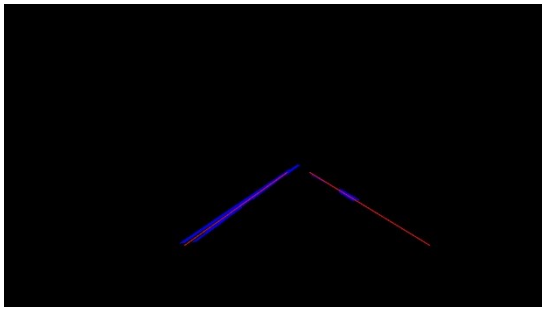
## 2.1.2. Hough transform

Hough Transform [2]是一種用於檢測圖像中的直線、圓形等形狀的圖像處理技術，其基本概念在於將圖像中的點映射到參數空間如圖 2，在這種映射下，影像的點在參數空間會被映射成一條線，而影像中的線在參數空間就是影像中線上每個點被映射到參數空間的線的交會點。然後透過觀察經過交會點上的線的數量，當線的數量越多，代表在影像中這條線存在的機率越高，而透過設定好的閾值判斷在影像中是否有這條線。

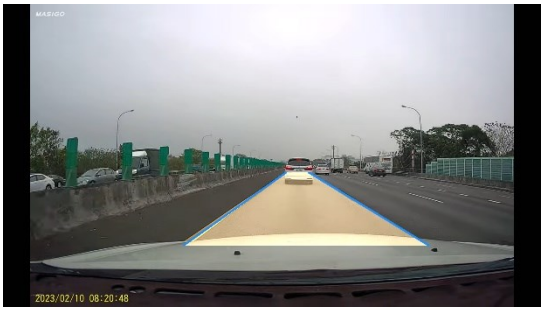
由下圖可以看到，藍色線段為 Hough transform 計算出的所有線段，這些線段被限定在一定的斜率內，而紅色線段為所有藍色線段的平均成果。圖 4 為顯示在原圖的成果。




圖表 3、Hough transform 座標轉換示意圖



圖表 4、Hough transform 處理後的影像

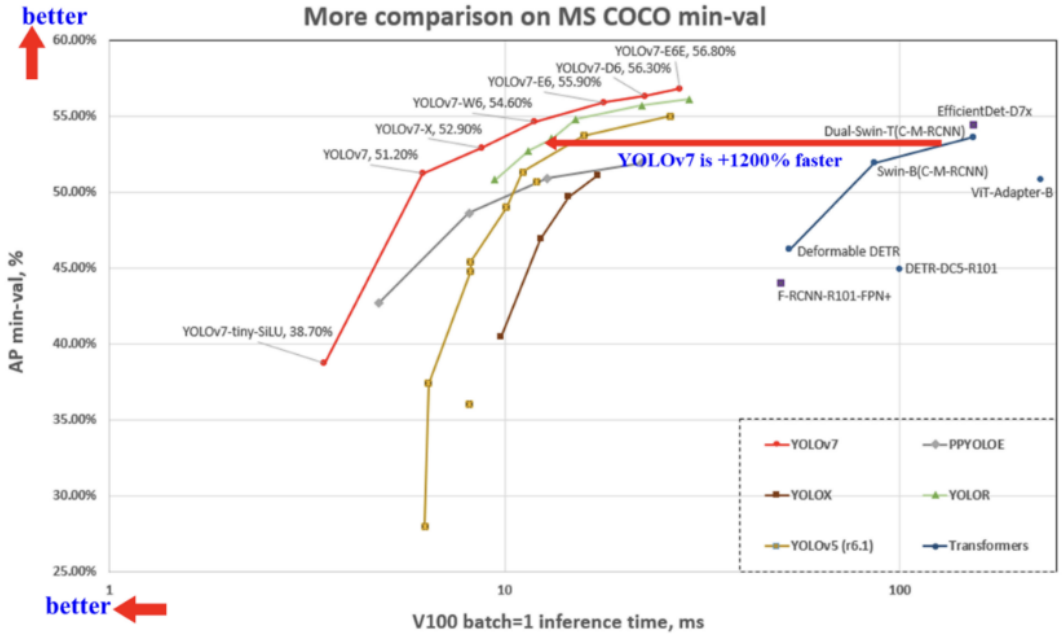



圖表 5、車道線示意圖

## 2.2. 車輛偵測

使用 Yolo v7 架構[3]偵測出影像中的每一台車，並取得每一台車於影像座標中的位置，使用此座標位置對影像中的車輛進行框選，最後再將車道遮罩與車輛偵測結合得到最終結果圖。

YOLOv7（You Only Look Once version 7）[3]是 YOLO（You Only Look Once）物體檢測系列模型的最新版本，此模型是 2022 年 7 月由中研院王建堯博士推出的。YOLO 是一種快速、實時的物體檢測算法，它能夠在單一前向傳播中檢測並定位圖像中的多個物體。而 YOLOv7 又是其中最優化的版本，在其原論文中有提到此模型在 5 FPS~160 FPS 範圍中的速度以及準確率都超過了其他已發表的物體偵測模型，比較結果如圖 2.，而此模型能夠超過其他模型的原因是在它針對模型架構優化和訓練過程優化，減少了目前最先進方法約 40%的參數量和 50%的運算量。




圖表 6、YOLO v7 與其他模型的比較


圖表 7、YOLO 車輛偵測示意圖

## 2.3. 車距估測

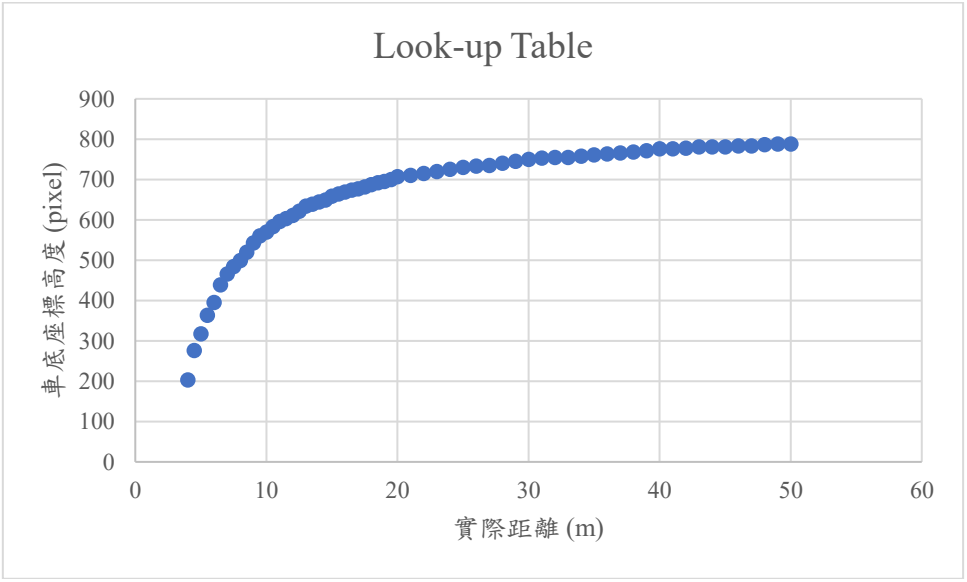
### 2.3.1. Look-up table

使用預先在實際到路上拍攝的距離作為樣本，4 到 20 公尺，每間隔 0.5 公尺拍攝一張影像，21 到 50 公尺，每間隔 1 公尺拍攝一張影像，總計有 63 張影像。我們做成 look-up table 進行查表，因為實際 pixel 並不會與樣本相同，所以中間值使用內插法估算而得，如以下公式 (式 2-3)，其中 y 是影像車底與影像底部的像素距離，minPixel 和 maxPixel 分別是從 look-up table 獲取比 y 小和比 y 大的像素距離，minDist 和 maxDist 分別是 minPixel 和 maxPixel 對應的實際車距。

由於車距與實際距離由下圖可知並非呈現線性關係，我們無法使用比例推算。另外此樣本影像與實驗用影像並不相同，每台相機的內參與外參各不相同，這也會導致實驗上結果產生誤差。

$$Distance = \left(\frac{y - minPixel}{maxPixel - minPixel}\right) \times (maxDist - minDist) + minDist \quad (2\text{-}3)$$

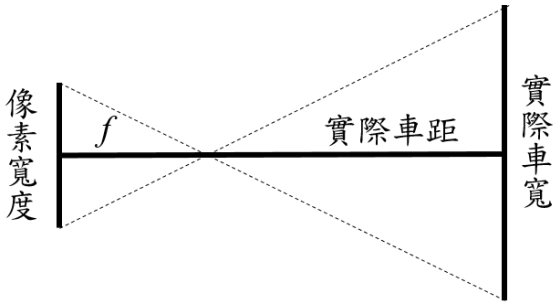

圖表 8、取樣影像與實際距離標記




圖表 9、Look-up table

### 2.3.2. 焦距推算估測

由學長的論文和全國法規資料庫可知，車道寬約為 3.65 公尺，車道線線段長約 4 公尺，間距為 6 公尺，而且每輛車使用的相機參數並不相同，所以上述的方法 (Look-up table) 有較大的誤差，圖 10 是我們使用焦距推算估測的方法。




圖表 10、針孔成像示意圖

我們假定車輛的相機為針孔成像，使用上述的資料以及實驗用影像推算焦距 $f$，下圖是推算用影像，得到的資訊如下，上面的橘線 (長度 268 pixel)，下面的橘線 (長度 363 pixel)，車道寬 (長度 3.65 公尺)，兩橘線差距 4 公尺，由針孔成像公式(式 2-4)可以推得 $f = 1121.82$ 。

$$\frac{f}{像素寬度} = \frac{實際車距}{實際車寬} \tag{2-4}$$

圖表 11、推算焦距用影像

圖表 12、圖表 11 放大圖

$$\frac{上橘線長}{f} = \frac{車道寬}{相機道上橘線車距} \Rightarrow \frac{268\,pixel}{f} = \frac{3.65m}{相機道上橘線車距}$$

$$\frac{下橘線長}{f} = \frac{車道寬}{相機到下橘線車距} \Rightarrow \frac{363\,pixel}{f} = \frac{3.65m}{相機到下橘線車距}$$

$$上橘線車距 - 下橘線車距 = 4m$$
$$\rightarrow f = 1121.82$$

有了焦距 $f$ 即可推算實際車距，有了影像的像素長度、焦距$f$和車體實際寬度，我們就可以推得與前車的實際車距為$20.89m$。

$$\frac{焦距}{影像車道寬} = \frac{實際車距}{實際車道寬} \Rightarrow \frac{1121.82}{196\ pixel} = \frac{實際車距}{3.65m}$$
$$\rightarrow 實際車距 = 20.89m$$

# 3. Experiment & Result

## 3.1. 實驗環境 & 資料集

實驗環境如下表，資料集則是使用 youtube 上提供的行車紀錄器畫面。

| | |
|---|---|
| OS | Ubuntu 22.04 |
| GPU | RTX 3080 Ti<br>(8G Memory) |
| Cuda | 11.0 |
| Pytorch | 1.7.1 |
| Python | 3.8.16 |

表格 1、環境配置

## 3.2. 實驗結果

下圖為實驗結果，圖 13 是前方區域沒有車輛的情況 (車輛太遠或不在前方範圍內)；圖 14 是利用 **look-up table** 估測出的車距圖；圖 15 是利用**焦距推算**估測出的車距圖。

由實驗我們可以觀察到，車道線並不能一直準確被辨識到，會有誤差偏離。另外經由 **look-up table** 估測出的車距，經過車道線實際距離的對照 (參考 2.3.2.)，

可以發現落差很大。經由**焦距推算**的方法的數值比照車道線，其結果較為合理。


圖表 13、無前車結果圖


圖表 14、Look-up table 估測影像


圖表 15、焦距推算估測影像

# 4. Conclusion

在這次的實驗中，我們實作了幾個傳統的影像處理辦法來對辨識車道並結合了較新的物件偵測網路來檢測車輛，也透過針孔攝影機的焦距計算辦法校正了由不同攝影機拍攝的測距影像所提供的像素距離對應的實際距離。

在辨識車道辨識的時候，我們使用了 Canny edge operator 和 Hough transform 兩種傳統影像處理方法，而這兩種方法都需要透過調整閾值來抓取最適合的結果，因此這個閾值的設計很大程度的決定了辨識結果，而在做 Hough transform 的時候，由於車道線是虛線，線與線之間並未相連，因此如何將這些線相連來辨別車道就是一大目標，在我們的實驗中，是通過限制斜率來準確的得到車道左右兩條線的位置，由此方法我們成功讓程式能夠自動判斷車道線位置。

在車輛辨識當中，由於我們使用目前效能最為優秀的 YOLOv7，因此可以準確且高速的抓取車輛位置，而且透過模型回傳的 bounding box 我們可以直接得到前車車體的尾端，方便進行後續與前車距離的計算。

在這次的作業當中有提供汽車測距校正用影像，透過這些影像，我們可以得到此影像的像素距離與實際距離的對應關係，然而，由於我們實際用來實驗的影片是使用別種行車紀錄器，由於各個相機的內部參數與外部參數接不盡相同，若是我們直接使用透過校正用影像得到的像素距離與實際距離對應關係來計算實驗影片內的前車距離只會得到錯誤的答案。為了避免此種情況，我們必須透過針孔攝影機的焦距推算找到拍攝實驗影片的行車紀錄器的焦距，以此焦距來直接對影像中的前車距離進行估算，因為此方法皆是由固定的相機參數所計算，因此此方法相較於 look-up table，估算出來的前車距離會更加貼近我們所使用的實驗影像中的情景。

在這次的實驗當中，我們實作了許多上課所學到的傳統影像處理辦法，也了解到了這些辦法在理論上與實作上的差異，在理論上我們都覺得非常直觀，也非常簡單，然而到了實作當中，光是將車道線的判斷維持在同一條線上就花費了大量時間，這就是理論與實作上的差距，也是為甚麼我們需要透過實作來理解這些理論，因為在實作當中，我們才會遇到許多光是讀書無法想像到的問題，不過也透過解決這些問題，我們才是真正的理解了這些影像處理辦法的優點以及缺點。

# 5. Reference


[1] John Canny.” A Computational Approach to Edge Detection”. IEEE Transactions on Pattern Analysis and Machine Intelligence (1986) DOI: 10.1109/TPAMI.1986.4767851

[2] Lines Detection with Hough Transform

[3] CW. Wang, B. Alex, and HY. Liao, “YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors,” arXiv.2207.02696v1 2022 Jul 6.

[4] 陳政德 ,基於單攝影機影像處理之前車距離估測, Distance Estimation of Front Vehicle Based on Single Image [碩士論文, CCU]

[5] 全國法規資料庫

[6] S538D 行車紀錄器 2K 白天畫面/高速公路

[7] https://github.com/WongKinYiu/yolov7 (Source code)

[8] 方昱勛學長 和 賴俊佑學長