**道路標線偵測的研究**

1. **研究動機：交通事故常造成人員的傷亡。而交通事故發生的原因，「駕駛行為不當」又居首位。因此，若能開發無人自動駕駛車，且讓無人駕駛車主導大部分的交通，勢必可以減少交通事故的發生，進而降低無謂的人員傷亡。**
2. **研究範圍：為了實現無人駕駛車，需要許多技術整合。例如：自動控制、號誌辨識、路徑規劃、定位等不勝枚舉。而我認為其中一項重要技術，就是如何讓車子認得道路的範圍，並安全地行駛在道路上。這其中的首要工作，就是如何讓車子認得道路的標線。**

**在此研究中，將限縮討論的範圍如下：**

* **車子行走於高速公路，車前方之行車記錄器記錄下之道路狀況做為研究對象。**
* **目的是利用較簡單的影像處理技術，找出道路邊緣的標線。**
* **先僅限於筆直道路的部分。至於彎曲轉彎的道路，將留待後續研究。**
* **找出道路標線之後，將用紅色線標示其邊線，以利後續自動控制時使用。**

1. **研究方法：針對使用的程式語言、資訊技術等說明如下：**

* **程式語言的選擇： 在此研究中，我使用Python程式語言。由於Python是一種互動式的程式語言，且提供完善的函式庫，是很親和的程式語言。另外，此研究的目的主要在於瞭解資訊技術解決問題的程序，並不是產品開發，因此不注重軟體效能，Python語言已足夠。**
* **動態影像轉靜態影像： 行車記錄器的錄像，可分解成多張靜態影像。先從分析靜態影像開始，決定使用的方法，測試，等一切滿意之後，再將這些方法應用於動態的錄像檔。**
* **彩色轉灰階： 錄像為彩色。初步認為色彩的資訊暫時可不考慮，因此將彩色影像轉換成灰階影像。**
* **影像裁切： 由於行車記錄器的鏡頭是廣角鏡，車道左右兩條標線將匯聚於遠方。因此在匯聚點之上的影像基本上已不重要。影像裁切有助於減少多餘資訊，有利於後續處理。**
* **邊緣偵測：道路標線通常與柏油或水泥路面的顏色有強烈對比。利用此特性，可以很容易找出標線的邊緣。一般常用的邊緣偵測演算法，例如：Canny edge detection，在Python的opencv函式庫中都有提供。**
* **Hough transform: 在「邊緣偵測」所找到的結果，視覺上看起來即使是一條線，但是在資訊表示上仍然是點陣圖。因此，為了進一步獲得「直線」的資訊，要進一步進行Hough transform。Hough transform的結果，會給我們一堆直線段的訊息，每條直線段以兩個端點來表示。有了這些端點，即表示已經找到了標線。**

1. **研究步驟：**

**首先我們看一下原始從行車記錄器獲得的影像：**

****

**現在車子是走在最外側車道，可以看得出來車道相當筆直。**

**將彩色影像轉換成灰階影像：**

****

**為了適當地去除雜訊，我們可以將灰階影像做模糊化，得到下列影像：**

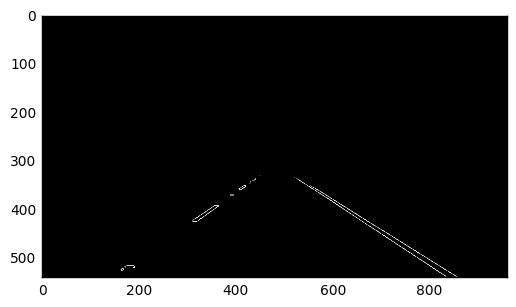
****

**模糊化之後的結果，用肉眼看起來差異並不大。**

**接下來，我們要開始進行邊緣偵測了，用的方法是opencv[1]裡面的Canny edge detection[2]。**

****

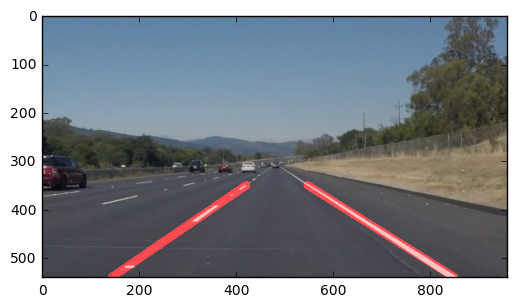
**從起結果看得出來，道路的確被偵測到了。但是後面的風景造成我們的困擾。如果能適當地裁切，會有利於後續工作。裁切的方法是這樣的：觀察到影像大小寬x高為960x540。切割一個梯形，將可涵蓋車道兩邊標線。根據影像大小，設計一梯形，其四個端點為（96,540）, (460,324), (500,324), (912,540)。這邊特別說明一下，在影像處理的領域中，座標的定義與解析幾何不同。座標原點定在影像的左上角，x座標往右越大，y座標往下越大。裁切結果如下圖。**



**從這張圖很明顯可以發現，背後雜訊都不見了。剩下的只是車道線而已。**

**然而，我們的工作還沒完成。目前看到圖中的線，其實仍然是由一堆點所構成的。這樣的資訊對於後續使用仍然不足。因此需要進一步做Line detection。在影像處理及電腦視覺的領域中，提到其中一個方法叫做Hough Transform[3]，而我們即是要用此方法。**

**經過Hough Transform之後，得到一堆線段。線段由兩個端點來表示。若將端點連成一線畫在原圖上，會得到下圖。**

****

**至此，已成功地偵測到道路兩邊的標線了。**

1. **結論與研究心得：**

初步的步驟已試驗過，結果令人滿意。未來我們將進行更有挑戰性的工作，例如：光線條件不好（昏暗）的時候、道路上有陰影、部分路段柏油重新鋪設以致標線不連續、標線老舊等等。希望在以上這些惡劣環境取得的資料，都能有一定水準的正確性。

**參考文獻：**

**[1] Open CV:** [**http://opencv.org/**](http://opencv.org/)

**[2] Wikipedia, canny edge detection,** [**https://en.wikipedia.org/wiki/Canny\_edge\_detector**](https://en.wikipedia.org/wiki/Canny_edge_detector)

**[3] Wikipedia, Hough Transform, https://en.wikipedia.org/wiki/Hough\_transform**