國立雲林科技大學

電子工程所

National Yunlin University of Science and Technology

Graduate School of Electronic Engineering

報告題目 :汽車駕駛者監控系統設計與實現

報告日期(Date)： 2022/12/23

姓名(Name)： 謝昕樺(碩一生)

指導教授(Professor)： 蘇慶龍副教授

E-mail： M11113020 yuntech.edu.tw

學號 (Student ID)： M11113020

本週進度(Progress of this week)

- 準備 Paper 報告

12/10	Sat	14:00~20:00	(6.0 個小時)
12/11	Sun	14:00~19:00	(5.0 個小時)
12/11	Sun	21:00~02:00	(4.0 個小時)
12/12	Mon	13:00~15:00	(2.0 個小時)
12/12	Mon	17:00~20:00	(3.0 個小時)
12/12	Mon	23:00~02:00	(3.0 個小時)
12/13	Tue	16:30~23:00	(6.5 個小時)
12/14	Wed	13:00~21:00	(8.0 個小時)
12/15	Thu	12:30~14:00	(2.5 個小時)
12/15	Thu	16:30~18:30	(2.0 個小時)
12/16	Fri	16:30~22:00	(5.5 個小時)
12/17	Sat	14:00~16:30	(1.5 個小時)
12/17	Sat	19:00~21:00	(2.0 個小時)
12/17	Sat	00:00~04:30	(4.0 個小時)
12/18	Sun	18:00~20:00	(2.0 個小時)
12/18	Sun	22:00~02:30	(4.5 個小時)
12/19	Mon	14:00~02:00	(12.0 個小時)
12/20	Tue	14:00~15:00	(1.0 個小時)
12/20	Tue	16:00~17:30	(1.5 個小時)
12/20	Tue	23:00~01:30	(2.5 個小時)

12/9 進度	12/21 進度
1. 看學長 code 架構 2. Yolov5/Yolov7 架構	1. 準備 Paper 報告 2. 查看車道線論文
待辦事項	

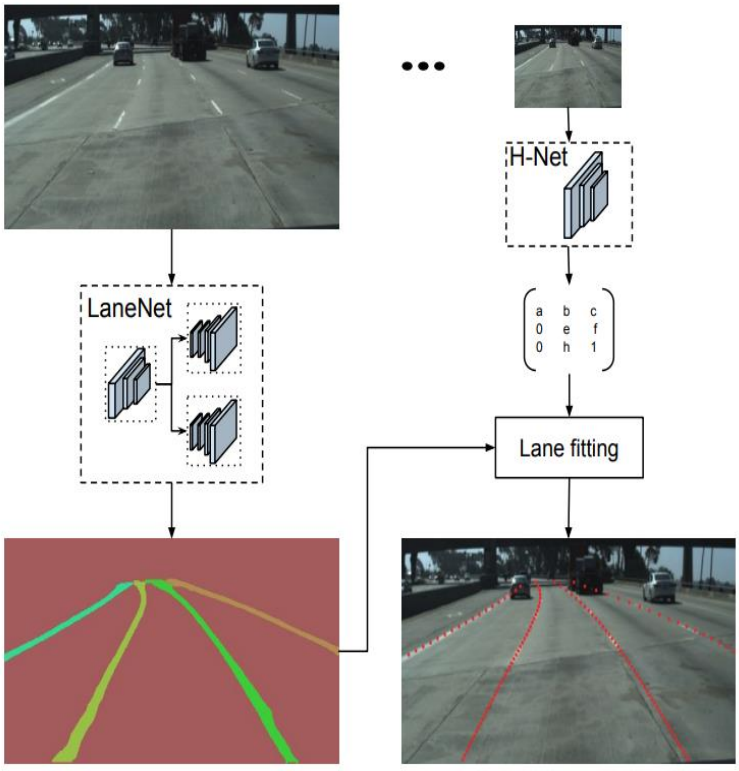
閱讀文獻

Towards End-to-End Lane Detection: an Instance Segmentation Approach

摘要

論文作者提出一種多分支網路架構，將車道線以實例分割， 讓每條車道線都有自己的實例去進行端到端訓練，接著在運用 H-Net 網路中的 H 矩陣對屬於同一點車道線的像素做回歸

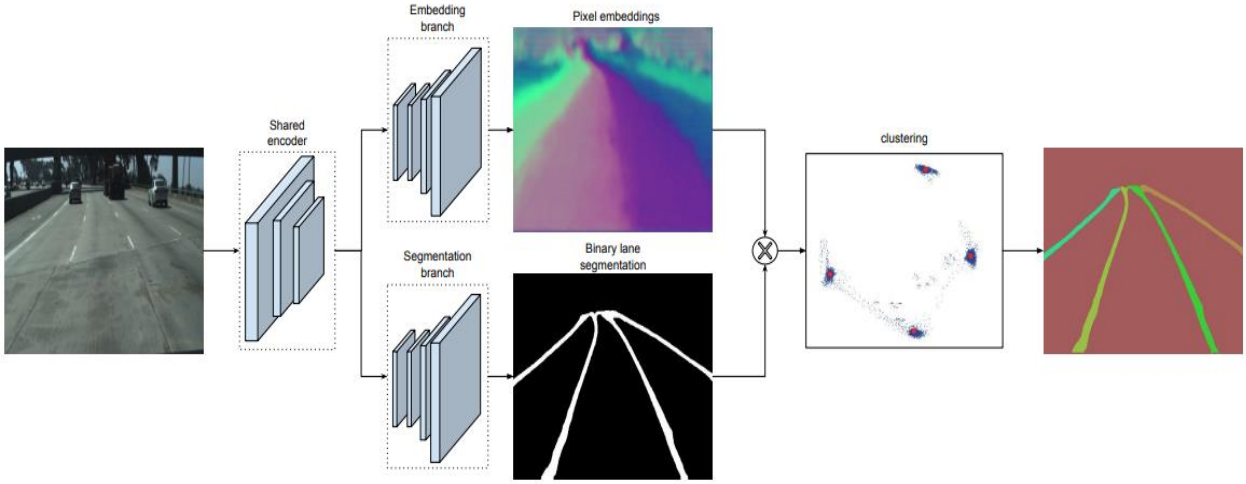
重點整理:



本論文使用了兩種網路架構分別為 LaneNet 和 H-Net，LaneNet 是輸入用 Encoder 的方式，輸出 Decoder 分為兩個分支，分別為語義分割和實例分割，最後將兩張圖合併，而 H-NET 是利用 Conv 和全連接層計算 H 矩陣內 6 個參數

- LaneNet

LaneNet 是利用 Encoder 和 Decoder 的方式，其中 Encoder 共用，Decoder 分成了兩個分支，分別為語義分割(semantic segmentation)和實例分割(instance segmentation)

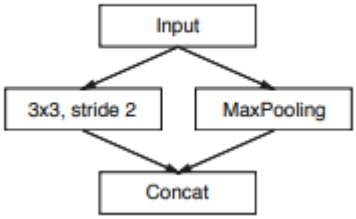


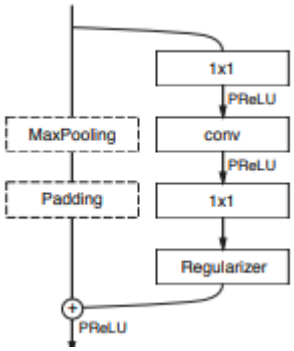
- 語義分割(semantic segmentation)

將相同類別的像素運用相同顏色做分類，而這篇論文是將每個像素點給一個 Label 去判定是車道線還是背影，目的是將車道線和背景做分割

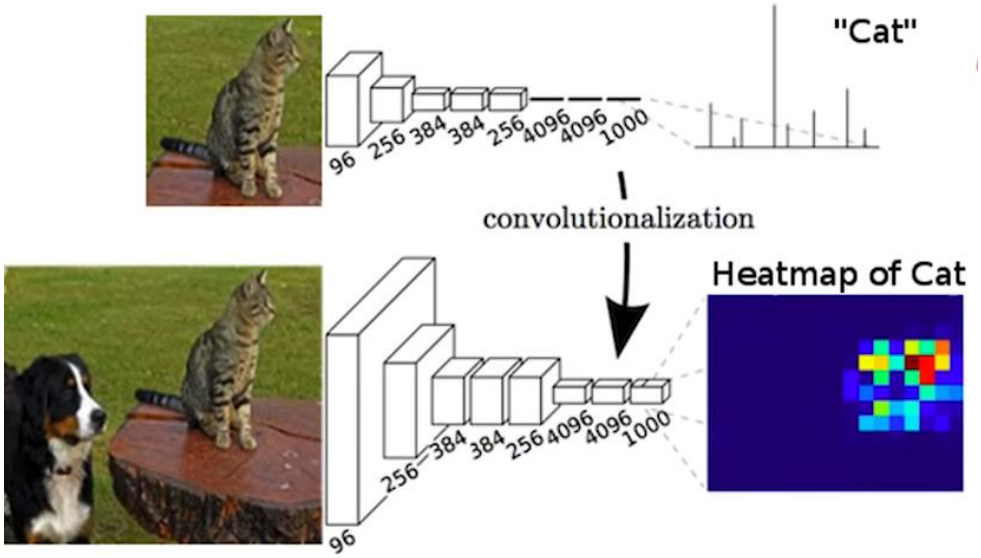
本篇論文是參考 E-Net 論文中使用的 VGG16 架構，並將 VGG16 架構中 Fully Connect 改成 Fully Convolution，Stage 1~3 為 Encoder，其中 Stage 1 和 Stage2 都使用下採樣，Stage 3 沒有下採樣，Satge 4~5 為 Decoder，將照片還原為原本大小並對其像素分類

Name	Type	Output size
initial		$16 \times 256 \times 256$
bottleneck1.0	downsampling	$64 \times 128 \times 128$
4× bottleneck1.x		$64 \times 128 \times 128$
bottleneck2.0	downsampling	$128 \times 64 \times 64$
bottleneck2.1		$128 \times 64 \times 64$
bottleneck2.2	dilated 2	$128 \times 64 \times 64$
bottleneck2.3	asymmetric 5	$128 \times 64 \times 64$
bottleneck2.4	dilated 4	$128 \times 64 \times 64$
bottleneck2.5		$128 \times 64 \times 64$
bottleneck2.6	dilated 8	$128 \times 64 \times 64$
bottleneck2.7	asymmetric 5	$128 \times 64 \times 64$
bottleneck2.8	dilated 16	$128 \times 64 \times 64$
Repeat section 2, without bottleneck2.0		
bottleneck4.0	upsampling	$64 \times 128 \times 128$
bottleneck4.1		$64 \times 128 \times 128$
bottleneck4.2		$64 \times 128 \times 128$
bottleneck5.0	upsampling	$16 \times 256 \times 256$
bottleneck5.1		$16 \times 256 \times 256$
fullconv		$C \times 512 \times 512$

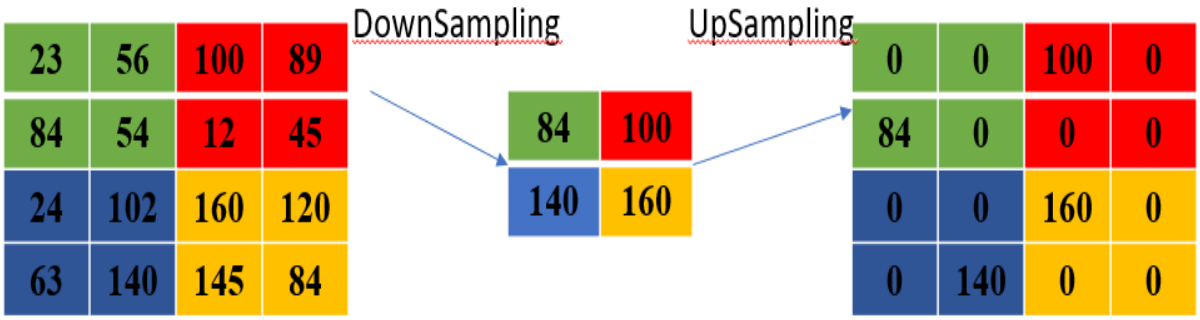




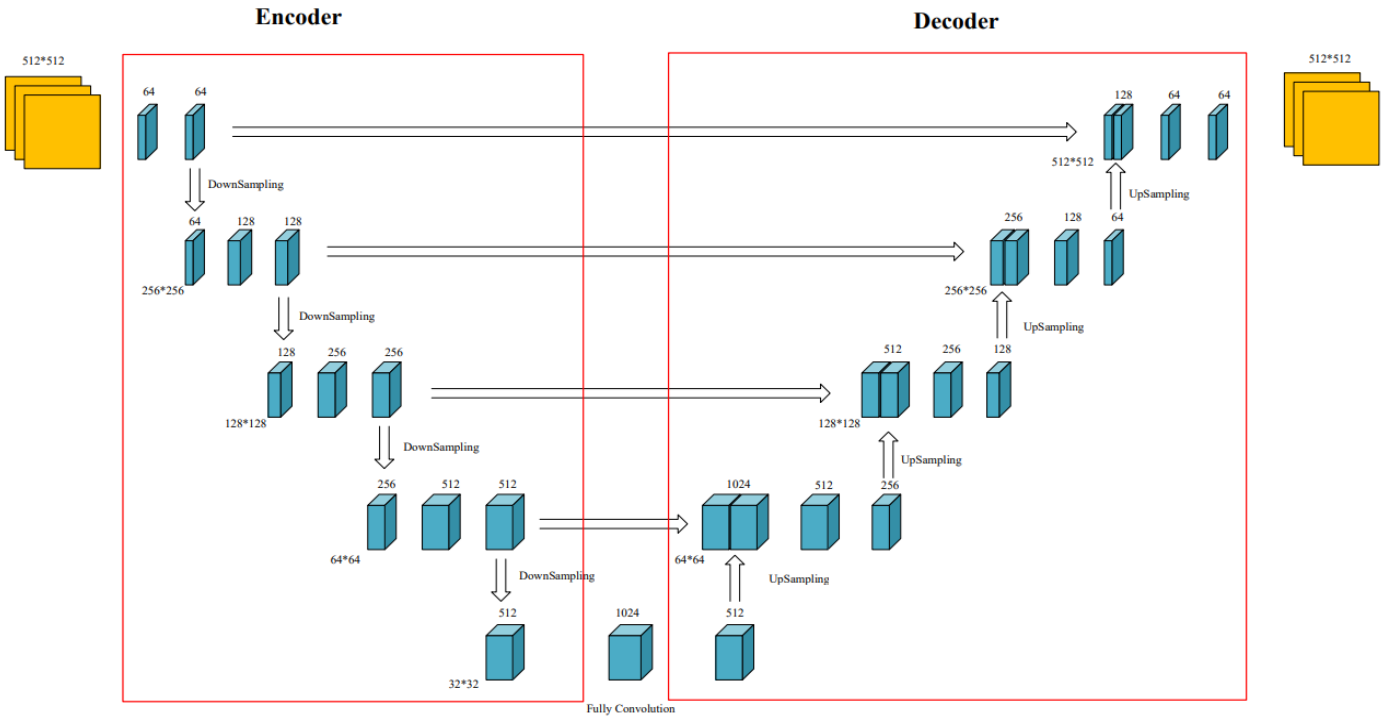
首先是 Fully Convolution 部分，傳統 Conv 是將最後一層特徵壓縮至一個 1*1*n 維的 Vector 後進全連接利用 Softmax 進行分類，這樣做會讓 Image 丟失很多空間訊息，沒有辦法利用 Decoder 還原每個像素點的類別，若是將 Fully Connect 改成 Fully Conv 方式輸出到最後可以有 N 維的 Feature Map，這樣就可以對像素進行分類



接下來是 Upsampling 部分，當今天要 Decoder 回原本 Feature Map Size 的時候，是會先去尋找 Encoder 相同層的 Feature Map Size，將 Maxpooling 後的位置記錄下來，將其位置複製給 Decoder 的 Feature Map Size，其餘空洞地方補 0



接下來將 Encoder 經過幾層 Conv，Decoder 就做幾層 Conv，並且和 Encoder 的 Feature Map 做 Concat

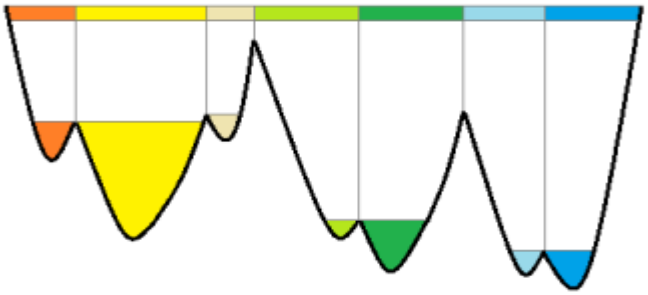


- 實例分割(instance segmentation)

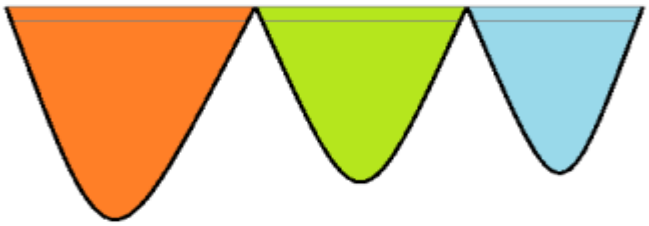
除了分類相同類別的像素外，還要知道有幾組相同類別的像素，每組都以不同顏色做標記，而這篇論文是將標記出來的車道線再進行分組

依據 Input 輸入有幾組車道線就分幾組

車道線分組是參考 Deep Watershed Transform for Instance Segmentation 這篇論文，傳統的 Watershed Transform 會因為噪聲太多而產生許多分水嶺，因此利用卷積網路去學習特徵取代傳統梯度值



(a) Traditional Watershed Energy



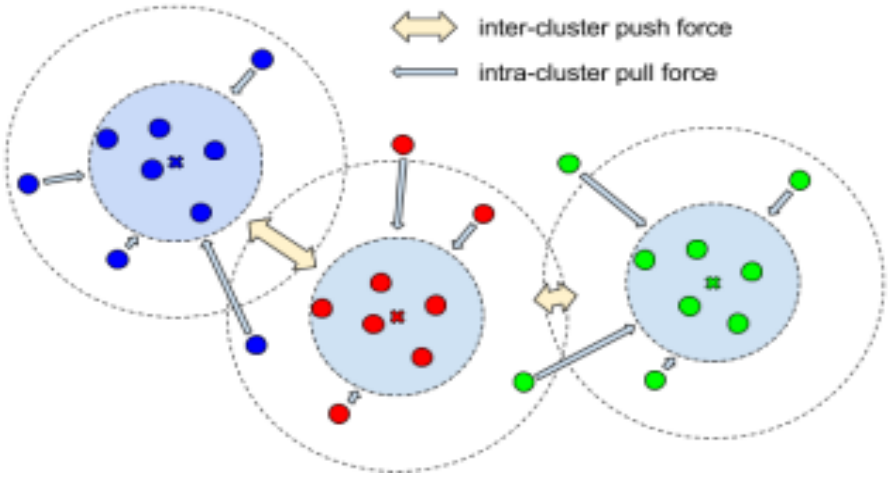
(b) Our learned energy

Watershed Transform 作法會找出局部最小值，將他們分為 N 類，接下來由這 N 類向外延伸找尋，以下圖為例目前是分成兩類，將 0 放入 Priority Queue 內，此時局部最小值為 1，並搜尋 0 周圍 8 個像素點看有沒有像素點 1，若有與最近的 0 歸為同一類，若像素點 1 是在 2 組 8 個像素點之外，則自己歸類為一類以此類推

2	2	2	3	2
4	0	4	3	4
7	5	7	5	7
7	7	8	9	7
1	7	2	1	3
2	3	0	3	3
4	5	1	7	4
3	3	2	3	3

2B	2B	2B	3B	3B
4B	0B	4B	3B	4B
7	5B	7	5B	7
7	7	8	9	7
1C	7	2A	1A	3A
2C	3	0A	3A	3A
4C	5	1A	7A	4A
3C	3	2A	3A	3A

而相同 Label 的 Pixel 應該要更靠近彼此，不同 Label 的需要遠離彼此，因此 Loss 為式 1-1 和式 1-2，Loss Total 為式 1-3



$$L_{var} = \frac{1}{C}\sum_{C=1}^{C}\frac{1}{Nc}\sum_{i=1}^{Nc}\left[\left|\left|\mu c - X_i\right|\right| - \delta_V\right]_+^2 \qquad \text{式 1-1}$$

$$L_{dist} = \frac{1}{C(C-1)}\sum_{CA=1}^{C}\sum_{CB=1,CA\neq CB}^{C}\left[\delta d - ||\mu_{CA} - \mu_{CB}||\right]_+^2 \qquad \text{式 1-2}$$

$$L_{Total} = L_{var} + L_{dist} \qquad \text{式(1-3)}$$

參數	作用
C	總共有幾條車道線
NC	每條車道線中實例像素數量
μc	每條車道線實例像素平均
X_i	當前車道線像素
δ_V	自訂義參數，用於調整μc和X_i距離
δd	自訂義參數，用於調整μ_{CA}和μ_{CB}距離
L_{var}	同一車道線像素向量X_i和車道線像素均值μc相減距離若大於δd，模型更新讓X_i更靠近μc
L_{dist}	不同車道線像素均值μ_{CA}和μ_{CB}若小於δd，模型更新讓 2 者遠離

上圖中像素點 Loss 是以 Mean Shift 為基礎，以半徑大小$2\delta_V$和$\delta_d > 6\delta_V$的圓做出發去進行分類，若在半徑大小內相同組別像素者，計算其像素點平均值，更新圓的中心點，再進行比較直到中心點不再移動為止

- H-Net

LaneNet 的輸出是每條車道線的像素點集合，需要利用多項式回歸將車道線擬合出來，因此論文利用了一個可以轉置矩陣 H(式 1-4)的 H-Net 神經網路訓練

$$H = \begin{bmatrix} a & b & c \\ 0 & d & e \\ 0 & f & 1 \end{bmatrix} \qquad \text{式 1-4}$$

轉置矩陣 H 內部有 6 個參數，因此 H-Net 輸出是一個 6 維 Vector，神經網路訓練如下

Type	Filters	Size/Stride	Output
Conv+BN+ReLU	16	3x3	128x64
Conv+BN+ReLU	16	3x3	128x64
Maxpool		2x2/2	64x32
Conv+BN+ReLU	32	3x3	64x32
Conv+BN+ReLU	32	3x3	64x32
Maxpool		2x2/2	32x16
Conv+BN+ReLU	64	3x3	32x16
Conv+BN+ReLU	64	3x3	32x16
Maxpool		2x2/2	16x8
Linear+BN+ReLU		1x1	1024
Linear		1x1	6

TABLE I

H-NET NETWORK ARCHITECTURE.

接下來要計算其損失函數，計算方法是以 y 座標去預測 x 座標，首先假設地面有 N 個車道線真實像素點$P_i = [x_i, y_i, 1]^T \in P$，接下來利用 H-Net 計算出來的輸出 H 進行座標轉換如式 1-5

$$P' = HP \qquad \text{式 1-5}$$

接下來用最小平方法進行多項式參數$\mathrm{W} = [\alpha, \beta, \gamma]^T$預測如式 1-6

利用真實座標點減去預測座標點

$$\mathrm{Loss} = \frac{1}{2}(W^T Y - X)^2$$

$$\frac{\partial Loss(W)}{\partial W} => \frac{\partial Loss(u)}{\partial u}\frac{\partial u}{\partial W} = 0$$

$$Y^T(W^T \mathrm{Y} - \mathrm{X}) = 0$$

$$\mathrm{Y}W^T Y^T - XY^T = 0$$

$$\frac{XY^T}{\mathrm{Y}W^T Y^T} = 1$$

$$\mathrm{W} = (XY^T)(\mathrm{Y}Y^T)^{-1} \qquad \text{式(1-6)}$$

接下來利用預測出結果的多項式參數$\mathrm{W} = [\alpha, \beta, \gamma]^T$預測出$x_i'$如式(1-7)

$$x_i' = \alpha y'^2 + \beta y' + \gamma \qquad \text{式(1-7)}$$

將x_i'投射回真實座標如式(1-8)

$$P_i^* = H^{-1}P_i' \qquad \text{式(1-8)}$$

最後將利用 MSE 預測x_i'和x_i的 Loss Function 如式(1-9)

$$Loss = \frac{1}{N}\sum_{i=1}^{N}(x_i' - x_i)^2 \qquad \text{式(1-9)}$$

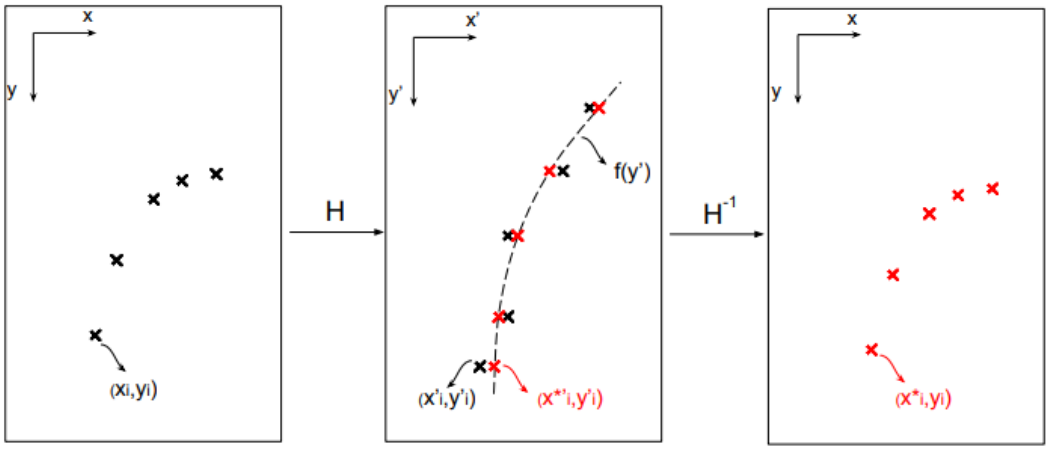


Fig. 3. Curve fitting. *Left:* The lane points are transformed using the matrix H generated by H-Net. *Mid:* A line is fitted through the transformed points and the curve is evaluated at different heights (red points). *Right:* The evaluated points are transformed back to the original image space.

- 比較結果

論文最後有利用 2 次式和 3 次式去擬合車道線，以及比較有無使用 H 轉置矩陣如下圖

	2th ordr (MSE)	3rd ordr (MSE)	Avg. miss/lane
no transform	53.91	17.23	0
fixed transform	48.09	9.42	0.105
cond. transform	33.82	**5.99**	0

並且輸入大小 512*256 圖像，得出結果為 19ms，每秒最多處理 52 偵

		time (ms)	fps
LaneNet	Forward pass	12	62.5
	Clustering	4.6	
H-Net	Forward pass	0.4	416.6
	Lane Fitting	2	
	Total	**19**	**52.6**

TABLE IV

SPEED OF THE DIFFERENT COMPONENTS FOR AN IMAGE SIZE OF 512X256 MEASURED ON A NVIDIA 1080 TI. IN TOTAL, LANE DETECTION CAN RUN AT 52 FPS.