ICG Final Project

3D reconstruction from 2D images

B04504116　電機四 鄭人豪
B04901069　電機四 林志皓

Introduction

三維重建是個越來越熱門且重要的問題，該議題探討如何從平面影像，推論出其立體結構，在虛擬實境(virtual reality)、擴增實境(Augmented reality)、自駕車領域，都能找到這項技術能貢獻，創造價值的地方。而這樣的題目結合了電腦視覺(computer vision)和計算機圖形(computer graphics)，創造出令人驚奇的視覺效果，因此我們挑選這個題目作為我們的期末專題，探討並比較當今的技術，也實做出自己的構想。

Objective

我們希望能在一水平線上左右位移拍攝同一景象中的物體，獲得兩張不同視角的 RGB 照片，判斷出相片中物體的立體模型，並顯示於螢幕上讓使用者觀賞。

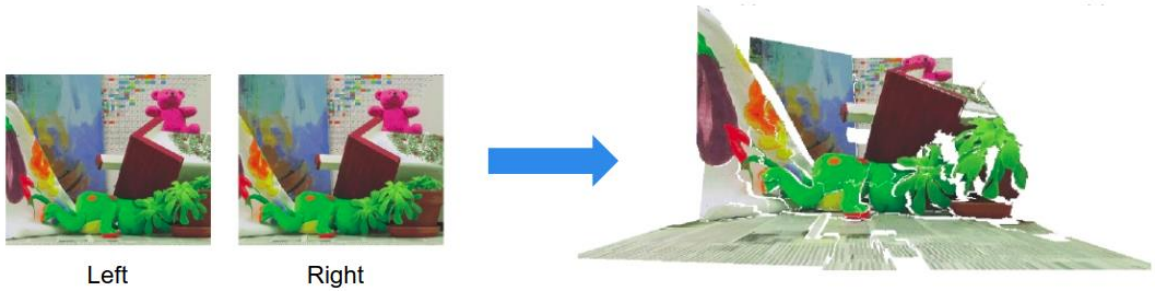


Fig 1. 目標示意圖。

Pipeline

為了達到我們三圍重建的目的，我們依照下列順序一步步完成：

1. 獲得兩張分別於同一水平面有左右位移所拍攝出的照片(如何獲得不在本次 Project 的討論範圍)。
2. 藉由"Stereo matching"技術推論出影像中物體的深度。
3. 由深度資訊和物體在影像中的位置，計算出每個物件在立體坐標系中的位置，重建出 RGBD 資訊。
4. 根據 RGBD 資訊，利用 WebGL 等將模型顯示於網頁上供使用者觀賞。

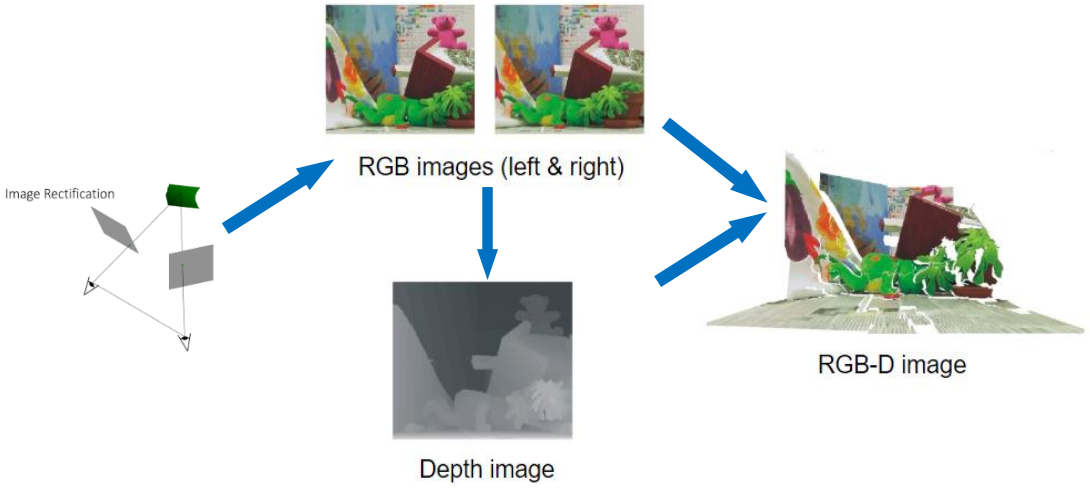


Fig2. Pipeline 示意圖。

Stereo matching

在本次專題中，Stereo matching (中譯:立體視覺匹配)扮演至為關鍵的角色。此技術能根據左右兩張照片，判斷出相片中物理的相對關係，進而推斷出深度資訊，與三維重建的效果高度相關。我們將在以下篇幅，對於傳統方法與機器學習方式的優劣及其效果進行比較。

Traditional method

在這部分我們採用傳統的 pipeline，分為以下四部分：

1. Cost computation

   在這部分，我們採用了 reference [2] 所提到 cost computation 的方法。

   初始的 cost volume 用以下方法產生（註：cost volume 是長寬為影像的長與寬，深度為最大可能 disparity 值的三維陣列）：

   由兩項標準產生：difference of color intensity 和 census cost

$$C(\mathbf{p},d) = \rho(C_{census}(\mathbf{p},d),\lambda_{census}) + \rho(C_{AD}(\mathbf{p},d),\lambda_{AD})$$

$$\rho(c,\lambda) = 1 - \exp(-\frac{c}{\lambda})$$

$$C_{AD}(\mathbf{p},d) = \frac{1}{3}\sum_{i=R,G,B} |I_i^{Left}(\mathbf{p}) - I_i^{Right}(\mathbf{pd})|$$

   使用此種 cost 計算方式的優勢：

   將兩種不同 cost 的算方式 mapping 到值域[0, 1]，如此 total cost per pixel 最大為一常數，而且不會被任一方法所 bias。

2. Cost aggregation

此步驟我們採用 cross-based aggregation region 被提出於 reference [1]。

Step1: 對於每個在左圖的 pixel，找到它直角的十字，細節如下：

1. $D_c(\mathbf{p_1}, \mathbf{p}) < \tau_1$ and $D_c(\mathbf{p_1}, \mathbf{p_1} + (1,0)) < \tau_1$
2. $D_s(\mathbf{p_1}, \mathbf{p}) < L_1$
3. $D_c(\mathbf{p_1}, \mathbf{p}) < \tau_2$, if $L_2 < D_s(\mathbf{p_1}, \mathbf{p}) < L_1$.

p1 是每個 pixel 十字上四個端點的任一端

Dc(p1,p) < τ, Dc(p1,p) 是 p1 和 p 的 color difference, τ 是一個臨界值。

color difference 定義為：

Dc (pl , p) = max$i$=R,G,B |Ii (pl) − Ii (p)|

Ds(p1,p) < L, Ds(p1,p) 是 p1 和 p 的 spatial distance, L是目前最長的軸長度（pixel 數量）.

spatial distance 定義為：Ds(p1, p) = |p1 − p|

Step2. 對右圖做 Step1.

Step3. 將初始的 cost volume 先沿著橫軸從左累積(accumulate)，再從上沿著縱軸累積。

Step.4 對每個 pixel 點對其十字所延伸出的區域用 OII (Orthogonal Integration Image) 方式累積，細節如下：

$$\begin{cases} H(p) &= \{(x, y) \mid x \in [x_p - h_p^-, x_p + h_p^+], y = y_p\} \\ V(p) &= \{(x, y) \mid x = x_p, y \in [y_p - v_p^-, y_p + v_p^+]\}. \end{cases}$$

H(p)是 horizontal arm, 而 x 是被 cross' left- 和 right-arm endpoints 所 bounded 住的 x。

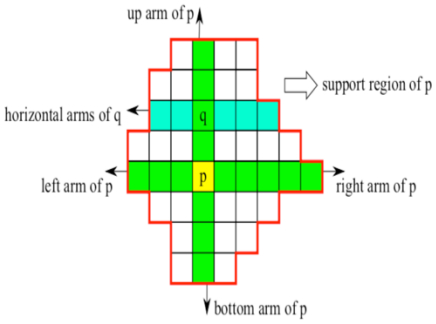
V(p)是 vertical arm, 而 y 是被它的 cross' top-和 bottom-arm endpoints 所 bounded 住的 y。

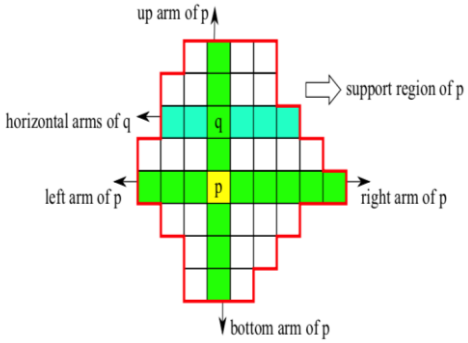
$$\begin{cases} H_d(p) &= \{(x, y_p) \mid (x, y_p) \in H(p), (x-d, y_p) \in H'(p')\} \\ V_d(p) &= \{(x_p, y) \mid (x_p, y) \in V(p), (x_p-d, y) \in V'(p')\}. \end{cases}$$

$$U_d(p) = \bigcup_{q \in V_d(p)} H_d(q)$$

U_d(p)是左圖產生出 pixel's aggregated cross region (H(p), V(p)) 和右圖產生出 pixel's aggregated cross region (H'(p), V'(p))的交集.

由下圖所示：



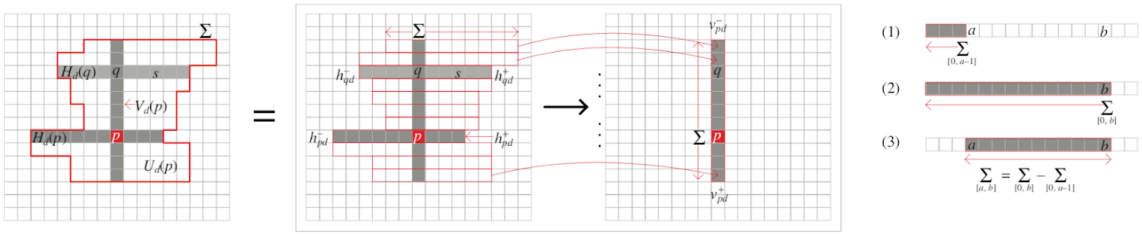


cross region per pixel from left image ∩ cross region per pixel from left image

決定完 cross aggregated region 後，我們可以用出現於 reference[1] orthogonal integral image (OII) 方法來快速算出 aggregated costs，如下：

$$E_d(p) = \sum_{s \in U_d(p)} e_d(s) = \sum_{q \in V_d(p)} \left(\sum_{s \in H_d(q)} e_d(s) \right) = \sum_{q \in V_d(p)} E_d^H(q) \tag{10}$$

OII 方法:



(a) (b) (c) (d)

首先沿著橫軸累積，再沿著縱軸累積。
得到上述計算結果後，計算每個 pixel 上由十字所圍出區域累積的 cost，此過程都可以在四次的加減法內完成。

3. Disparity optimization

此步驟我們採用 scanline optimization 和 winner-take-all (WTA)方法

Scanline Optimization:

給定 scanline direction r 和 path cost 和在 pixel p 的Cr (p, d) disparity d 依照下面方式更新優化 cost：

$C_{\mathbf{r}}(\mathbf{p}, d)$ at pixel $\mathbf{p}$ and disparity d is updated as follows:

$$C_{\mathbf{r}}(\mathbf{p}, d) = C_1(\mathbf{p}, d) + \min(C_{\mathbf{r}}(\mathbf{p} - \mathbf{r}, d), \\ C_{\mathbf{r}}(\mathbf{p} - \mathbf{r}, d \pm 1) + P_1, \\ \min_k C_{\mathbf{r}}(\mathbf{p} - \mathbf{r}, k) + P_2) - \min_k C_{\mathbf{r}}(\mathbf{p} - \mathbf{r}, k)$$

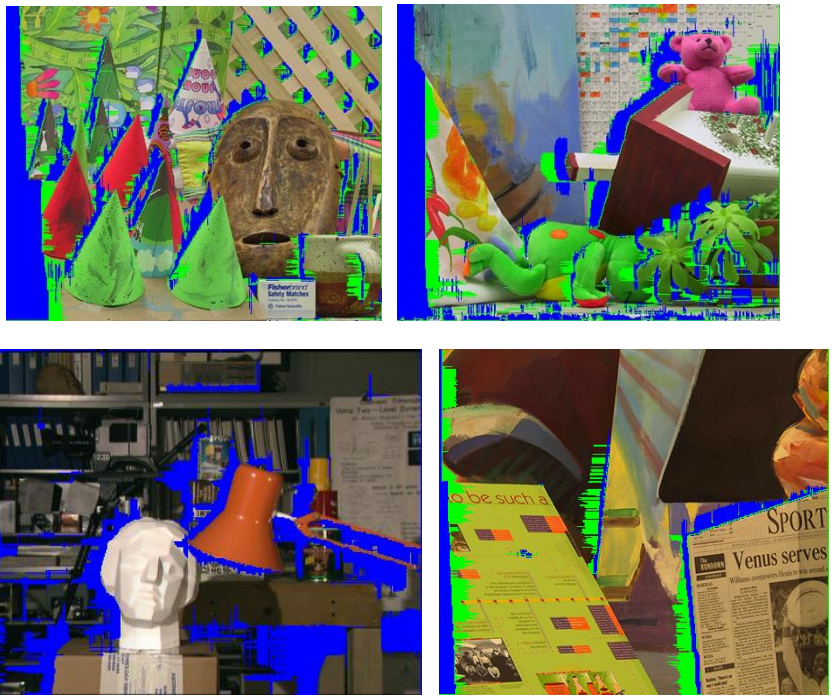
p − r 是沿著同個方向前面的 pixel 而$P1$, $P2$ ($P1 \leq P2$) 是用來逞罰鄰近區域 disparity 變化的參數。

1. $P_1 = \Pi_1, P_2 = \Pi_2$, if $D_1 < \tau_{SO}, D_2 < \tau_{SO}$.
2. $P_1 = \Pi_1/4, P_2 = \Pi_2/4$, if $D_1 < \tau_{SO}, D_2 > \tau_{SO}$.
3. $P_1 = \Pi_1/4, P_2 = \Pi_2/4$, if $D_1 > \tau_{SO}, D_2 < \tau_{SO}$.
4. $P_1 = \Pi_1/10, P_2 = \Pi_1/10$, if $D_1 > \tau_{SO}, D_2 > \tau_{SO}$.

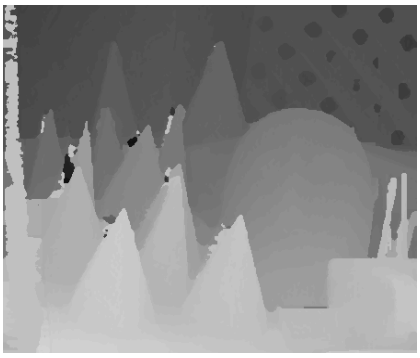
最終，不同 disparity 在每個 pixel 對應到的 cost 如下（也就是最終 cost volume 每個單元的值）：

$$C_2(\mathbf{p}, d) = \frac{1}{4} \sum_{\mathbf{r}} C_{\mathbf{r}}(\mathbf{p}, d)$$

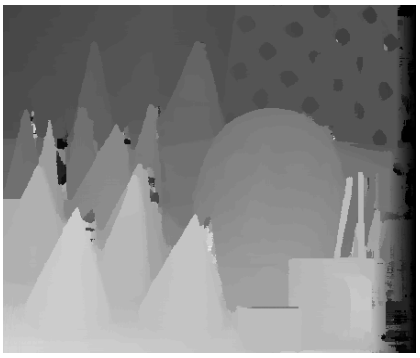
下圖為未經 refinement 的結果



藉由 left-right consistency 方法偵測出有 outliers 的區域（藍色與綠色分別代表橫軸端與縱軸端包圍區域所生成的 outliers ）



結果



來自右圖的 disparity

Outliers

4. Disparity refinement

   在此步驟我們用了四個步驟來修補一些區域（部分步驟於 reference[2]被提出）

   4 個步驟：

   1. Interpolation
   2. Depth Discontinuity Adjustment
   3. Sub-pixel Enhancement
   4. 3x3 Median Filter

Machine learning method

相較於 traditional method 的缺點，Machine learning 的方法似乎更為適合處理這類問題。在該領域中有 ”Convolution Neural Network” ，具有 space invariance，能夠有效萃取圖片中的特徵(feature)，結合底層的特徵找出更加大領域的特徵，使的圖片進一步被電腦理解，因此經常應用於電腦視覺(computer vision)領域的應用，諸如分類(classification)、語意切割(semantic segmentation)等任務。而在 stereo matching 的任務上，也能迅速有效地找出兩張圖片各個部分的相對應關係，因此我們也在這次 project 中進行嘗試，並與傳統方法進行比較。

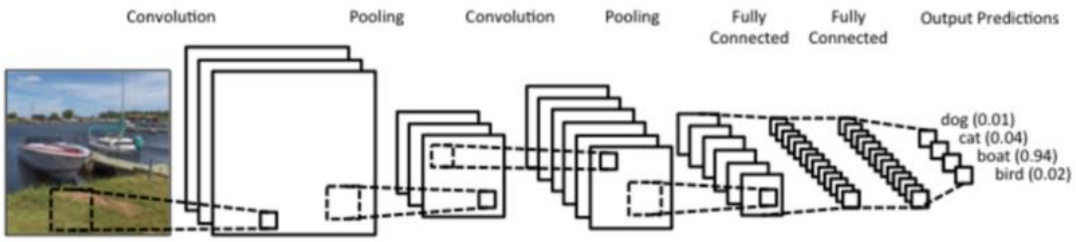


Fig 3. Convolution Neural Network 示意圖。

在這次的 project 中，我們參考一篇 2018 年的論文 “Pyramid Stereo matching network”
以下將對其進行說明。

Pyramid Stereo Matching network

Basic idea:
為了能有效學習左右兩張圖片相對應的關係，這篇論文使用 Convolution Neural Network 分別對兩張進行特徵萃取，然後在結合兩張圖片的特徵，利用同個模型(model)學習兩者之間的關係，進一步預測深度資訊。

Model Architecture:
在 model 的前半段，是對於 2 維平面萃取特徵的架構，左右兩張圖片是經過同一個 model 的，(因為萃取特徵沒必要針對左右兩張分開訓練)，而在架構中的一個巧思，便是 Spatial Pyramid Pooling Module (可見 Fig 4.)。SPP 由四種 kernel 不同大小的子模組構成，因此能夠在 multi-scale resolution 都得到資訊，對於不同深度造成的不同 disparity 都能順利蒐集到資訊，傳給下一部份進一步處理。

從兩張圖片所獲得的資訊，疊成一塊”cost volume”，接下來 3D CNN 進行處理，也就是預測深度的部分。其中的架構是三組連續的”hour glass”，也是 encoder + decoder，連續萃取特徵，並參考經典 Computer vision CNN 的 work – ResNet –的架構，有很多 skip connection，使不同解析度(resolution)的特徵能交互作用，達到更好的預測效果。
在 2018 年 3 月之前，這個 model 架構在 KITTI data set 上達到 state-of-the-art，速度也算快。

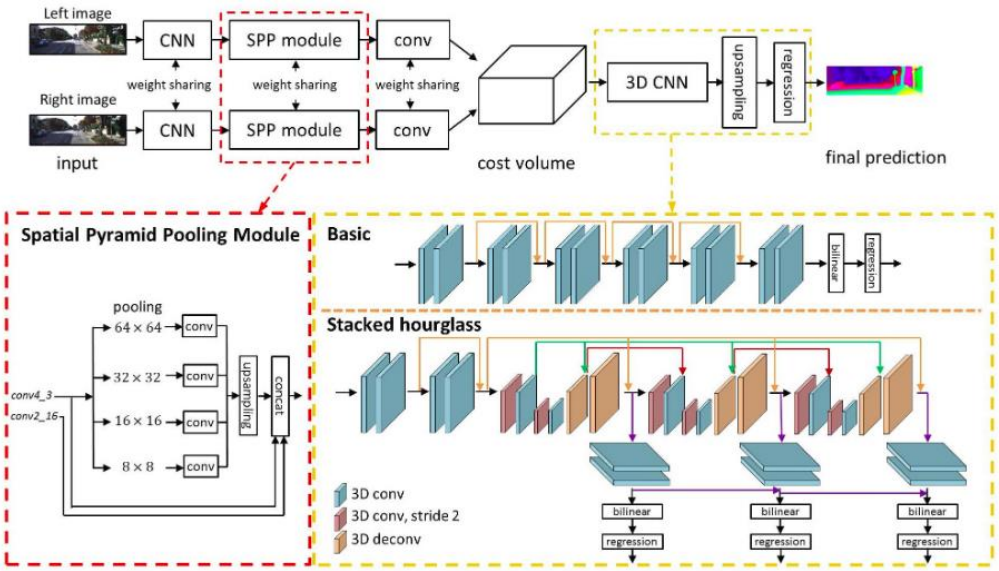


Fig 4. Model Architecture (擷取自原著論文)

Result:

訓練 model 使用的是 KITTI 2015 的資料，訓練所用 GPU 與過程分別為 Nvidia Tesla P100、15 hr, 700 epochs。訓練集是車子上左右兩台不同照相機所蒐集到的照片，許多 stereo matching 相關的文獻都會以這個資料及當中的表現作為衡量標準。以下是我們的實驗結果(測資是 model 沒看過的相片)

Fig 5. Left image

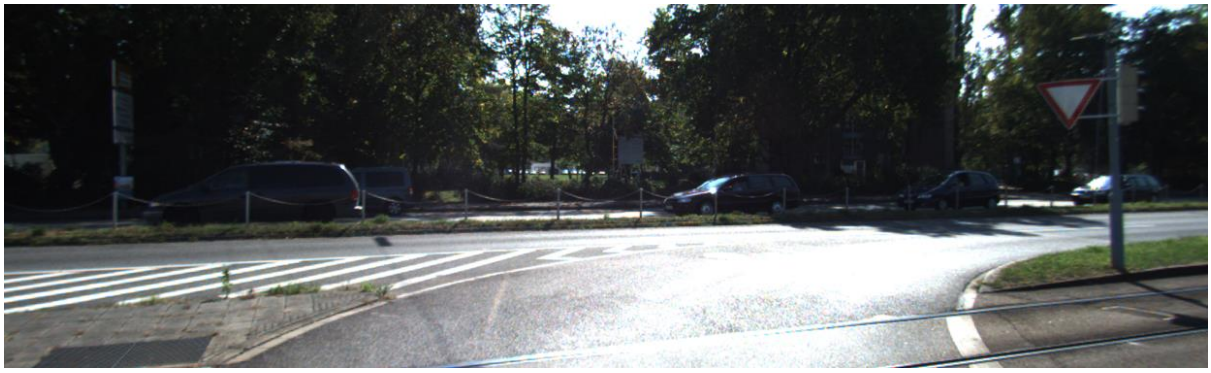


Fig 6. Right image

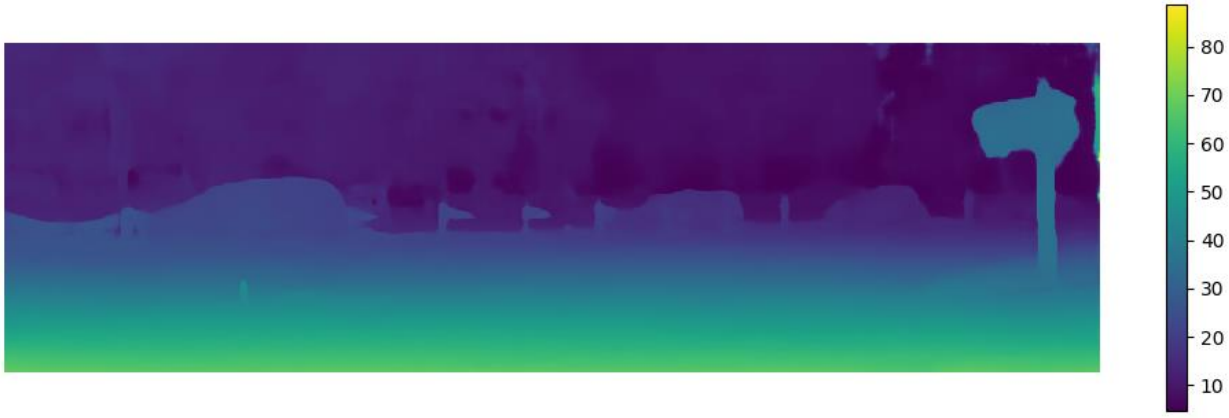


Fig 7. Depth image

在預測結果可以看得出物件輪廓明顯，可以清楚辨認出交通號誌和經過的車輛，而在預測的時間上也滿快(如果是兩組不一樣的照片時間差大約一秒) 在接下來 3D reconstruction 的部分，我們採用這個 model 所預測出的結果。

Conclusion：

在實作傳統與 Machine-learning based 產生 Disparity Map 的方法後，我們決定以 Machine-learning based 的方法作為產生圖片中深度資訊的演算法。原因如下：

1. Running Time

   傳統方法對於每一張圖片運算時間特別長，以下為我們實做的 running time：

```
Tsukuba
* Elapsed time (cost computation): 2.240678 sec.
* Elapsed time (cost aggregation): 381.830109 sec.
* Elapsed time (cost computation): 2.282026 sec.
* Elapsed time (cost aggregation): 391.524820 sec.
* Elapsed time (disparity optimization): 0.015572 sec.
* Elapsed time (disparity refinement): 1.173584 sec.
Venus
* Elapsed time (cost computation): 4.027865 sec.
* Elapsed time (cost aggregation): 717.365762 sec.
* Elapsed time (cost computation): 3.965656 sec.
* Elapsed time (cost aggregation): 734.562450 sec.
* Elapsed time (disparity optimization): 0.028704 sec.
* Elapsed time (disparity refinement): 1.574592 sec.
Teddy
* Elapsed time (cost computation): 9.067297 sec.
* Elapsed time (cost aggregation): 1488.150797 sec.
* Elapsed time (cost computation): 8.334749 sec.
* Elapsed time (cost aggregation): 1452.823378 sec.
* Elapsed time (disparity optimization): 0.121340 sec.
* Elapsed time (disparity refinement): 2.727065 sec.
Cones
* Elapsed time (cost computation): 8.328705 sec.
* Elapsed time (cost aggregation): 1016.414072 sec.
* Elapsed time (cost computation): 8.331107 sec.
* Elapsed time (cost aggregation): 1004.053331 sec.
* Elapsed time (disparity optimization): 0.109782 sec.
* Elapsed time (disparity refinement): 2.274131 sec.
```

   最大的測資（450X375, max disparity: 60）需要花 50 分鐘得出結果。
   反觀 Machine-learning based 對測資（1242X375）僅需不到一秒的時間便可預測出深度，運算時間不受圖片大小與 max disparity 所限制，而傳統方法卻因圖片大小與 max disparity 大幅增加了其運算時間。

2. Accuracy of Prediction

   下表為用傳統方法預測的 error：

Item	Disparity map	Bad-pixel ratio
Tsukuba	fig.1	2.65%
Venus	fig.2	1.75%

Teddy	fig.3	12.38%
Cones	fig.4	11.96%

平均為: 7.18%

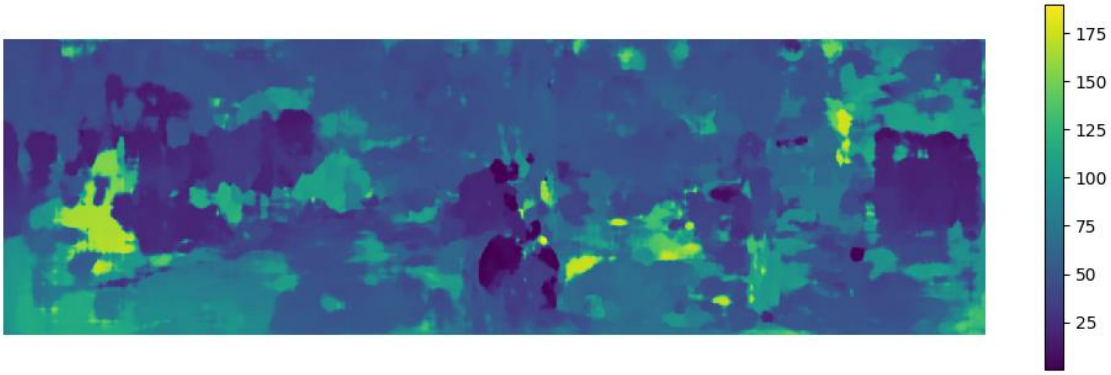
反觀 Machine-learning based 方法，其所訓練出的 model 對與訓練集相似的圖片預測平均 error 為：4.3 %
雖然 Machine-learning based 方法有一限制是它僅能對相似於訓練集的圖片做高精度深度預測，但是實際上我們可以針對我們的需求搜集訓練集訓練 model。

以下為我們自己在台大搜集部分的測資，右邊的數線代表 disparity 大小。disparity 與深度成反比，因此 disparity 越小代表離相機越遠。

Set1

左圖

右圖

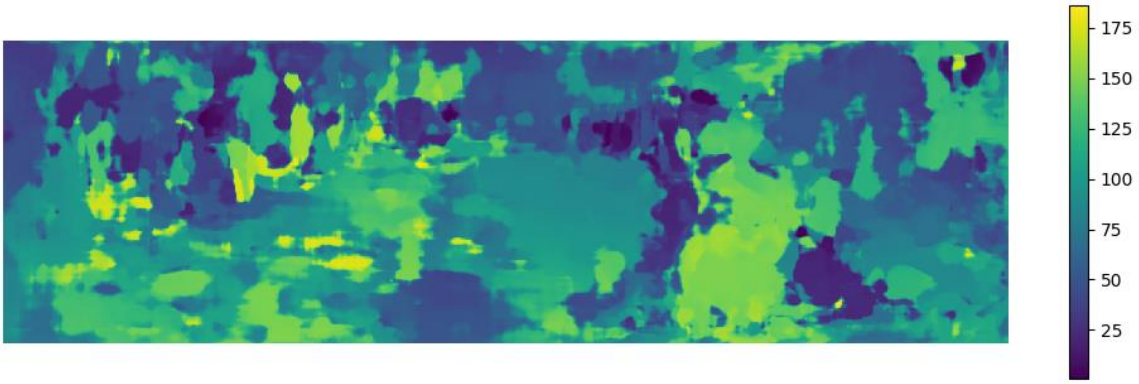


disparity

Set2

左圖

右圖



disparity

觀察上面的圖，可發現預測出的效果並不如 Fig 7.好，有幾種可能原因。其一，我們在採樣時的相機與官方測資使用的相機不同（左右間距、焦距等等）。其二，圖片中的場景與官方測資種類不同。其三，我們對於相片沒有做前處理（通常要先經過 rectify 後才能用，亦即將左右兩圖用 homography 的方式校準成只有在橫軸上有 disparity。）其四是我們採樣的場境有強光源與大面積的陰影，在諸多文獻當中這樣會影響 disparity 的判斷。不過如同前面提到的，只要將訓練集根據需求選好並做好測資的前處理，應該可以增加 model 對真實測資的預測精準度。

綜上所述，我們決定採納 Machine-learning based 方法作為我們 3d 重建的 pipeline。

3D Reconstruction

1. Object Model Generation

經過 stereo matching，我們已經獲得了每個 pixel 的深度資料，因此我們根據每個 pixel 在相片中的位置，搭配相對應的深度，得出相對於相機的三維空間座標，再搭配原圖中的 RGB 資訊，得到了有顏色的 point cloud data (RGB-D)!! 而在產生座標的時候，我們沿著各個方向軸進行 normalization，使的極端值也能限制在一定的範圍內，方便 webGL 讀檔顯示。

而在顯示方面，我們使用三種不一樣的工具:

(1) Open3D

Open3D 是 python 的套件，支援許多立體相關的 API，能讓使用者快速處理 point cloud data 或是顯示於螢幕，然而經過實驗，我們發覺效果很差，套件能直接將深度突擊 RGB 圖直接產生 point cloud 並顯示，然而卻只有黑白，深度的資訊也很差，因此我們另尋解決方案。

(2) pygame

pygame 是另一個 python 語言的套件，提供 openGL 的接口，我們採用這項工具，將每個點 render 在螢幕上，並利用上課所學的，自行實作旋轉矩陣，使 3D model 能在螢幕上旋轉，效果優良。然而由於這款套件並不如 webGL 可以將所有 vertices 存進 buffer 讓程式能去優化 render 的過程，是一個個點輪流畫在螢幕上，因此效率不高，旋轉並不連續(兩個 frame 之間需要時間計算)

Fig 8. RGBD 圖 (RGB + Depth from Fig.6)

由 Fig 8 和 Fig.9 可看到 Fig.6 的照片結合預測出的深度，處理後的 RGBD 資料，可以看出前景的交通號誌，很明顯的突出平面，路面也因遠近不同而產生凹陷曲面，效果相當不錯。

Fig 9. RGBD 圖 (another point of view)

(3)webGL

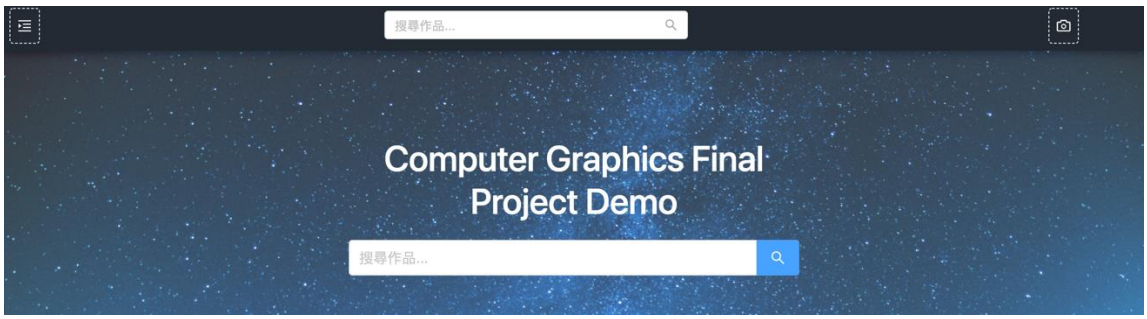
延伸作業一，將 Object 以 point 方式畫在 canvas 上，並同樣具備旋轉、平移、放大縮小和剪力功能。因為 webGL 在 render 的時候可以一次對所有 points buffer 進行處理，因此可以獲得非常流暢轉動的效果。以下為旋轉中的截圖：

Fig. webGL 結果

Application

最後，我們將以上 pipeline 結合成一個網站。在網站的實作上，我們是採用 React JS 的前端框架，由於動態式與組件化網站不是此次重點，因此不贅述，詳細可見下方此 project Github repo 中的網站實作部分。由於鄭人豪同學在本學期修區塊鏈課程，並將所學運用到此 ICG 專案當中。他將使用者上傳的圖片存至 IPFS (星際檔案系統，區塊鏈相關的技術之一)，並將圖片資訊（獨特 hash 值）存至區塊鏈中，達到網站去中心化的特色。網站 host object model 的後端由我們用 Node.js 實作。

以下為網站截圖：


搜尋作品...
Computer Graphics Final
Project Demo
搜尋作品...

主頁


搜尋作品...
上傳圖片
上傳你的Stereo圖至星際檔案系統 (IPFS)，我們將幫你製作3D模型
標題 *
ICG 2019 final demo !!!
關於本作品
A++++++++++++++++
左圖 *
Choose File l1_new.jpg
右圖 *
Choose File r1_new.jpg
* 表示必填
取消
發布

上傳區


搜尋作品...
左圖
右圖
濾鏡： 灰階
快門
濾鏡模式

相機

有濾鏡功能的相機

```
[nodemon] 1.18.4
[nodemon] to restart at any time, enter `rs`
[nodemon] watching: *.*
[nodemon] starting `node server/server.js`
listening to port 8080
```

host object model 的 server

```
System version: amd64/darwin
Golang version: go1.11.1
Swarm listening on /ip4/127.0.0.1/tcp/4001
Swarm listening on /ip4/192.168.43.173/tcp/4001
Swarm listening on /ip6/::1/tcp/4001
Swarm listening on /p2p-circuit
Swarm announcing /ip4/127.0.0.1/tcp/4001
Swarm announcing /ip4/192.168.43.173/tcp/4001
Swarm announcing /ip6/::1/tcp/4001
API server listening on /ip4/127.0.0.1/tcp/5001
Gateway (readonly) server listening on /ip4/127.0.0.1/tcp/8080
```

IPFS server

Reference

[1] *Cross-Based Local Stereo Matching Using Orthogonal Integral Images Ke Zhang et.al 2009*

[2] *On Building an Accurate Stereo Matching System on Graphics Hardware Xing Mei et.al 2011*

[3] *Pyramid Stereo Matching Network Jia-Ren Chang et.al 2018*

Repo of this project: ***https://github.com/Andy-Cheng/Computer-Graphics-Final-Project.git***