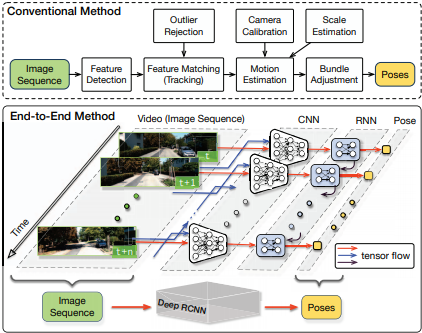
**Final Project: DeepVo**

R07921001 李尚倫、R07522625 李佳蓮、R07522653 陳健倫

**Project description**

在這次的Final，我們參考這篇Paper: DeepVO: Towards End-to-End Visual Odometry with Deep Recurrent Convolutional Neural Networks，同時參考的範例code為：<https://github.com/ChiWeiHsiao/DeepVO-pytorch>，主要是想藉由實作這篇Paper來了解SfM等相關知識。



**Implementation**

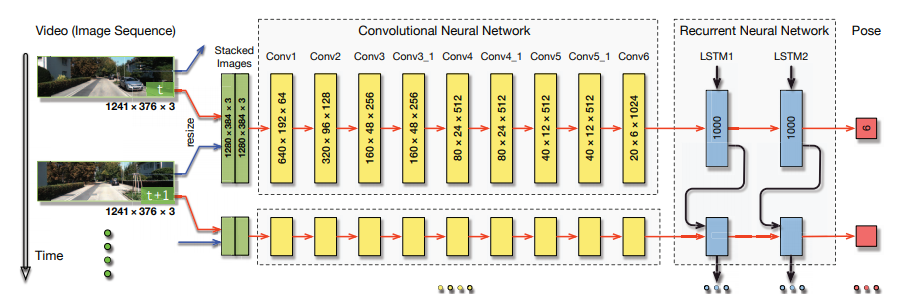
1. 實作DeepVo的 Model，用自己的DataLoader來做資料處理並訓練。
2. 將預測結果用三維圖像的方式匯出。
3. 自行實際錄影幾支影片用Blender切割作為Dataset，並丟入Model做Predict。

**How to Run the Program**

1. 執行網路別人給的getdataset.sh獲得KITTI的資料。
2. 執行Preprocess來對資料作前處理。
3. 執行myMain.py 來做訓練。
4. 執行myTest.py來做預測。
5. 執行myVisualize.py來做2D預測的繪圖。

**DeepVo Model**

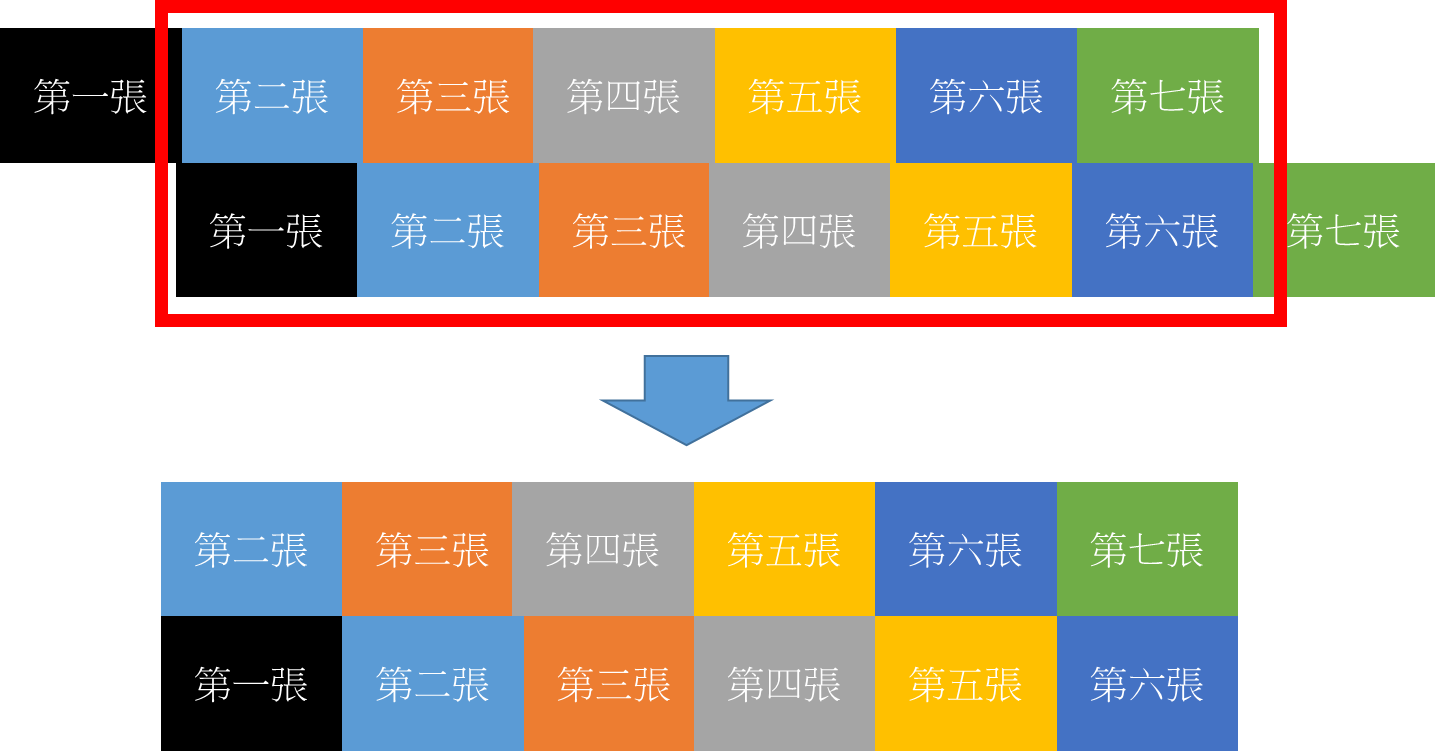
下圖是DeepVo這個架構，是用CNN+RNN作為RCNN的一個Model來做Training，CNN的部分使用網路現有的Flownet來作為Pretrain Model，RNN的部分則是使用pytorch內建的LSTM。



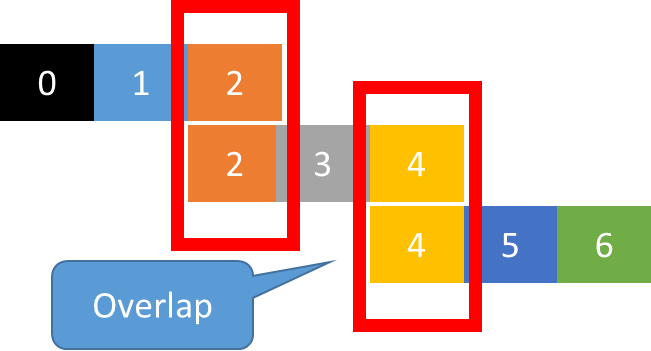
**DataLoader**

本Model在預測時的Pose，會有坐標系的問題，所以在Data load進來的時候，必須在ground truth的部分做微處理。假設我們現在一個batch有7張frame，就是1(batch)\*7(frame)\*3(RGB)\*img\_w\*img\_h，在這樣的狀況下，我們把第一個pose視為原點，就是沒有旋轉，同時translate為0，而後六張frame必須減去第一張frame的位置，變成相對坐標系的數值，才能拿來做訓練，而這就是此Model比較需要注意的小細節。

然後解決Pose的問題，再來就是照片的部分，實際做法我們是把[1:]跟[:-1]這樣的照片concate再一起，也就是將2~7張與1~6張疊在一起(如下面示意圖)，作為這個model的input ，dimension 為 1(batch)\*6(frame)\*6(RGB+RGB)\*img\_w\*img\_h，可以看到改變的地方是frame從7變成6，因為去了頭尾的張數，而RGB的通道變成6，原因就是多了一組照片，所以疊的地方是RGB的通道。如此一來 input 為 batch\*6(frame)\*6\*img\_w\*img\_h 、output與label為batch \* 6(frame) \* 6(pose)，這樣就能拿來做訓練。



而在照片擷取的部分還有的細節就是，我們可以用overlap的技巧來增加data量，假如說資料有1-15張，overlap=1，每個batch所要拿的frame數為3，那麼就是 1-3、3-5、4-6，這樣子拿下去，就能讓Model有更多的Data來訓練。



**What do we implement by ourselves**

1. 訓練時的Script都是自行撰寫，過程有一些部分參考範例Code。
2. 意即 training時的Dataloader、Model的建構皆是自行撰寫。
3. 三維度的Visualization是自行撰寫，需要讀取Predict的txt file。
4. 自行錄製影片並且用Blender切割成dataset
5. 用自製的dataset去做預測。

**What do we not implement by ourselves**

1. downloader.sh、myVisualize.py、helper.py、test.py是直接複製網路上的並做一點微調而已。
2. 獲得dataset部分，就是直接使用downloader.sh。
3. Helper.py內部有一些轉換的函式，像是Euler angle轉換我們是直接使用。
4. myVisualize.py是他的二維繪圖，我們直接使用方便與三維的做比較。
5. test.py使用的原因在於predict的部分，由於會有坐標系和Dataloader的問題，所以我們是使用自己訓練好的Model搭配他的dataloader以及對Predict的後處理函式來撰寫真正Predict的結果。

**XXXX**

在渲染時，。

**作業心得**

本Project實作了DeepVo這個Model，而我認為最困難的點就是在於座標轉換的地方，然後還有rnn的sequence to sequence training。至於model的撰寫以及dataloader的部分implement不難，但是也因為在predict的時候，貌似需要batch>1才能做predict，所以我的loader沒辦法做predict，不得已只好在最後直接使用網路的loader來做讀檔，但是model至少還是自己train的，所以也可以說對DeepVo要怎麼train有一點基礎上的理解。

**Artifact 1**

**Artifact 2**

攝影地點：

**Artifact3**

攝影地點：

Reference：