Shot Change Detection using Python and OpenCV

404410030 資工三 鄭光宇

攝影機從開啟鏡頭、關閉鏡頭，到再開啟鏡頭的瞬間，稱為一個 shot change，而這隻程式的用途就是用來偵測 shot change。

這份文件會說明：

- 關於此程式
- 環境需求
- 如何安裝所需套件
- 如何執行程式
- Shot Change Detection 實作
- Key Frame Selection 實作
- 結果
- 效能評估
- 結論
- 附錄/其他

> 以下偵測的 shot change 都是 hard cut

關於此程式

- 使用 Python + OpenCV 實作 shot change detection， 實作了多種方法偵測 shot change。主要分成兩大類：
  1. 基於 Edge Change Ratio
  2. 基於 Color Histogram

- 使用多執行緒加速程式運行
- 考量記憶體使用，漸進地讀入畫格、處理畫格
- 內建 Benchmark

環境需求

- Python : `Python 2.7`
- pip packages:
  - `numpy`

- `sklearn`
- `matplotlib==1.3.1`
- `seaborn==0.8.1`
- `opencv-python==3.3.0.10`

> pip package 的列表在同目錄的 requirements.txt

如何安裝所需套件

For Mac :

- Install Python : `brew install python`
- Install pip packages : `pip install -r requirements.txt`

如何執行程式

```
python main.py --help
```

會顯示程式執行方式

```
python main.py YOUR_VIDEO_FILE
```

會在畫面上輸出 shot change 所對應的 frame 編號 (0-based)

```
python main.py YOUR_VIDEO_FILE --keyframe_out FOO
```

會在畫面上輸出 shot change 所對應的 frame 編號 (0-based)，並且偵測影片中的 key frames，將 key frames 輸出到 FOO 的路徑底下。

(Optional)

```
python main.py YOUR_VIDEO_FILE --threshold YOUR_VALUE
```

指定 shot change detection 所使用的 threshold

```
python main.py YOUR_VIDEO_FILE --keyframe_out FOO --keyframe_threshold YOUR_VALUE
```

指定抓取 key frames 時所使用的 threshold

```
python main.py YOUR_VIDEO_FILE --scale YOUR_RATIO
```

計算前，將原始畫格降取樣到 `YOUR_RATIO` 倍大小，節省計算資源。

Shot Change Detection 實作

基於 Edge Change Ratio

這個方法的實作位於 `detector.py` 裡的 `EdgeBased` Class。

1. 先將影像前處理，使用 cv2 內建的 Canny，做邊緣偵測，並呼叫 dilate() 函數加粗邊緣。
2. 使用每張 `frame[t]` ，與它的前一張 `frame[t-1]` 的 edge feature `edge[t], edge[t-1]` 。`diff = edge[t]-edge[t-1]` 得到兩者的差值，則：
    - `sum(diff>0) / sum(edge[t]>0)` 也就是 `number of entering edges / edge[t]的邊緣像素總和` 會得到 Entering Edges Ratio
    - `sum(diff<0) / sum(edge[t-1]>0)` 也就是 `number of exiting edges / edge[t-1]的邊緣像素總和` 會得到 Exiting Edges Ratio
    - 比較 Entering Edge Ratio 與 Exiting Edges Ratio，取較大的那一個，當作 Edge Change Ratio
3. 對於所有 Edge Change Ratio >= Threshold 的 frames，判斷這些 frames 有 shot change。
4. **(optional)** 輸出結果時，可以限制一個 shot 的最小長度，使結果更為合理。

基於 Color Histogram

底下提到的方法，用 `detector.py` 裡的 `ContentBased` , `HSV1` , `HSV2` , `RGBBased` , `RGB1` , `RGB2` Class 實現。

1. 先將影像前處理，轉到 HSV color space (`ContentBased` , `HSV1` , `HSV2`)，或是直接使用 RGB color space (`RGBBased` , `RGB1` , `RGB2`)
2. 指定每個 channel 在 Color Histogram 中，要使用的 bin 數量。對於 HSV color sapce 的部分，設定 H channel 8 個 bins、S channel 4 個、V channel 4 個；對於 RGB 的部分，設定 R,G,B 的 bins 分別為 7,8,4。呼叫 numpy 內建的 histogramdd ，完成 Color Histogram 的計算，會得到一個三維的 Color Histogram，除上影像的像素數量，將 Histogram 標準化。最後我們將這個 Histogram 展開、壓平成一維向量 `H(x_t)` 。
3. 計算每一幀之間， Histogram 的距離，若距離 >= Threshold ，判斷這一幀有 shot change 。距離的計算方式有：
    - L1-Distance(x,y): `mean(abs(H(x)-H(y)))`
    - L2-Distance(x,y): `mean(square(H(x)-H(y)))`
    - 1 - Histogram intersection: `1 - sum(element_wise_min(H(x), H(y)))`

      （這裏跟 PPT 上的公式不同，因為 PPT 上的公式是找 < Threshold 的值判斷有 shot change，而我的判斷方式是 >= Threshold 判斷有 shot change，還有因為計算距離前， Histogram 有經過標準化，Histogram 的總和會是 1 ，所以才用 1 去減 Histogram Intersection，會得到介於 `[0,1]` 的值）

    - `HSV1` 、 `RGB1` 使用 L1-Distance

- `HSV2` 、 `RGB2` 使用 L2-Distance
- `ContentBased` 、 `RGBBased` 使用 `1 - Histogram intersection`

Key Frame Selection 實作

對於每個 shot ，先將時間上最中間的 frame 當作 key frame，放到 key frame 集合。之後從 shot 的開頭到結束，看看這個 frame 跟 key frame 集合裡面距離最小的距離（PPT 上是找距離最大的、判斷 < Threshold，而這裡使用 >= Threshold 判斷，所以應該找最小）是否 >= Threshold，如果是則判斷該 frame 為 key frame，將這個 frame 加入 key frame 集合。最後回傳 key frame 集合，輸出 key frames 圖片到指定的資料夾。

結果

對於 `friends.mpg` 資料，我們得到以下的 shot change：

id	frame_n
0	68
1	82
2	99
3	126
4	147
5	179
6	209
7	243
8	279
9	306
10	336
11	375
12	398
13	427
14	462

15	494
16	528
17	564
18	596
19	625
20	659
21	692
22	725
23	755
24	792
25	820
26	840
27	864
28	921

節錄 Key frames 結果如下：


F·R·I·E·N·D·S


JENNIFER ANISTON


MATTHEW PERRY

對於 `news.mpg` 資料，我們得到以下的 shot change：

id	frame_n
0	73
1	235
2	301
3	370
4	452
5	861
6	1281

節錄 Key frames 結果如下：

對於 `soccer.mpg` 資料，我們得到以下的 shot change：

id	frame_n
0	382
1	569
2	645

節錄 Key frames 結果如下：


MAN UTD
3-0
FULHAM
62:42
SPORTS 2
LIVE

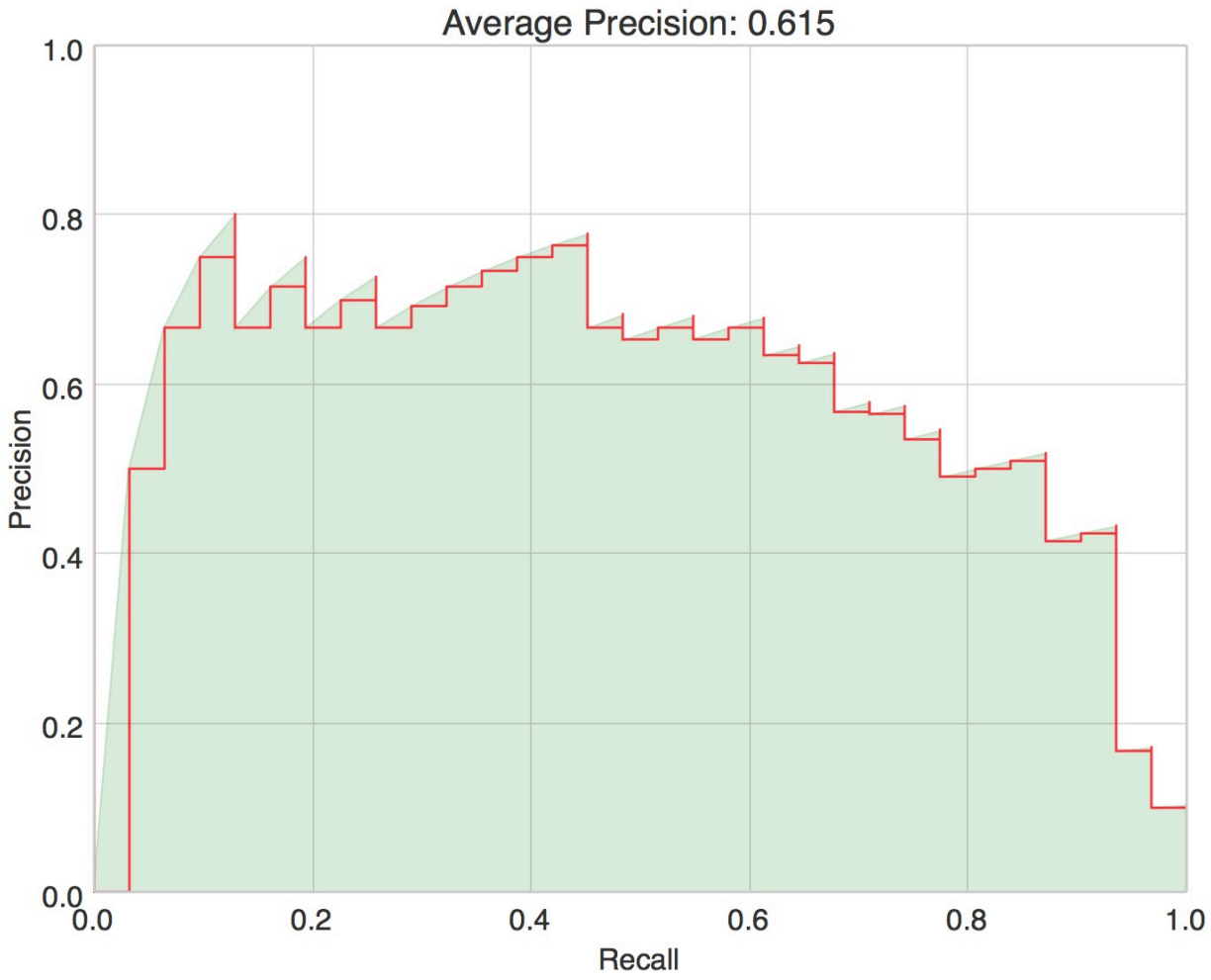

MAN UTD
3-0
FULHAM
62:55
SPORTS 2
LIVE


MAN UTD
3-0
FULHAM
63:03
SPORTS 2
LIVE

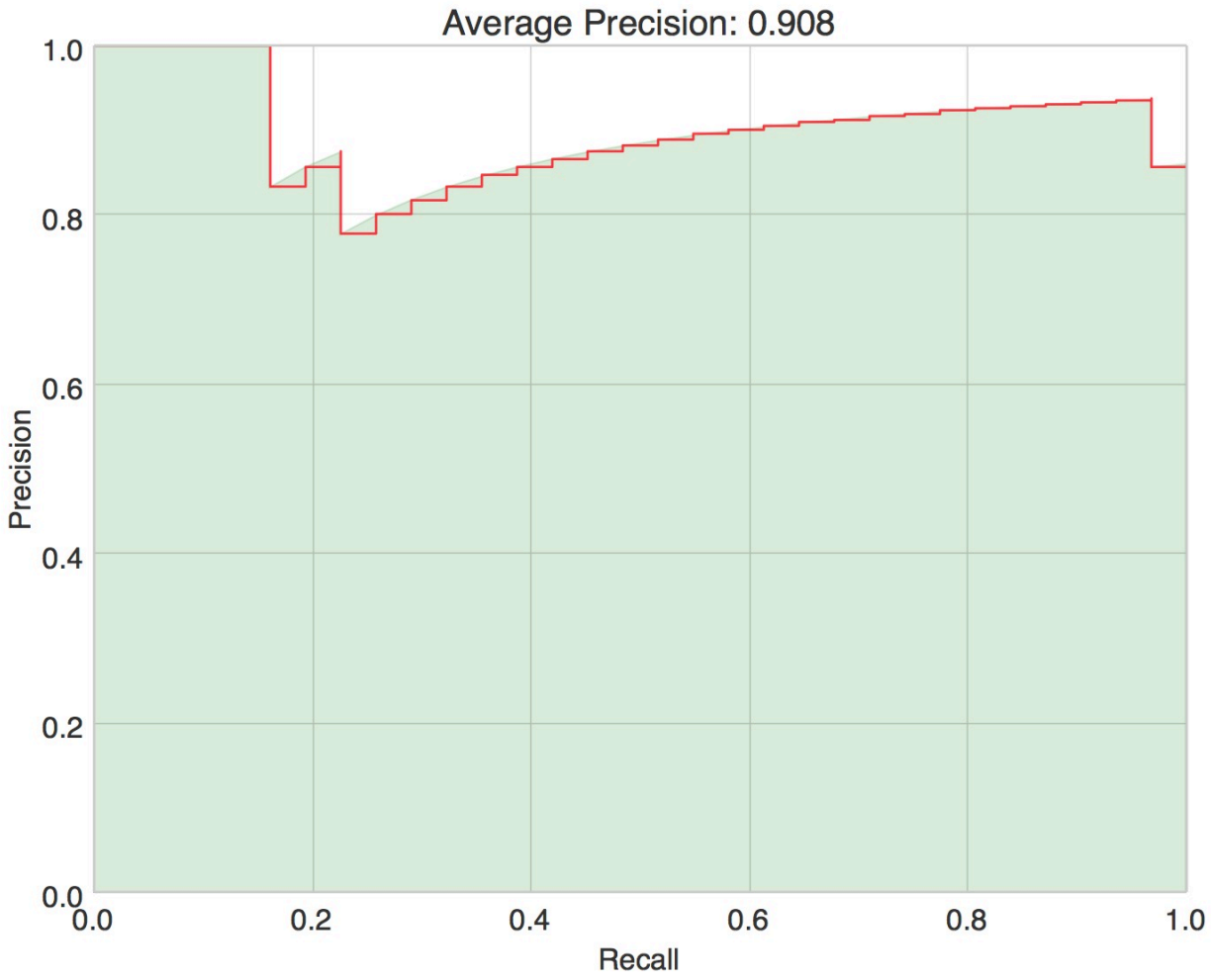

MAN UTD
3-0
FULHAM
63:04
SPORTS 2
LIVE

效能評估

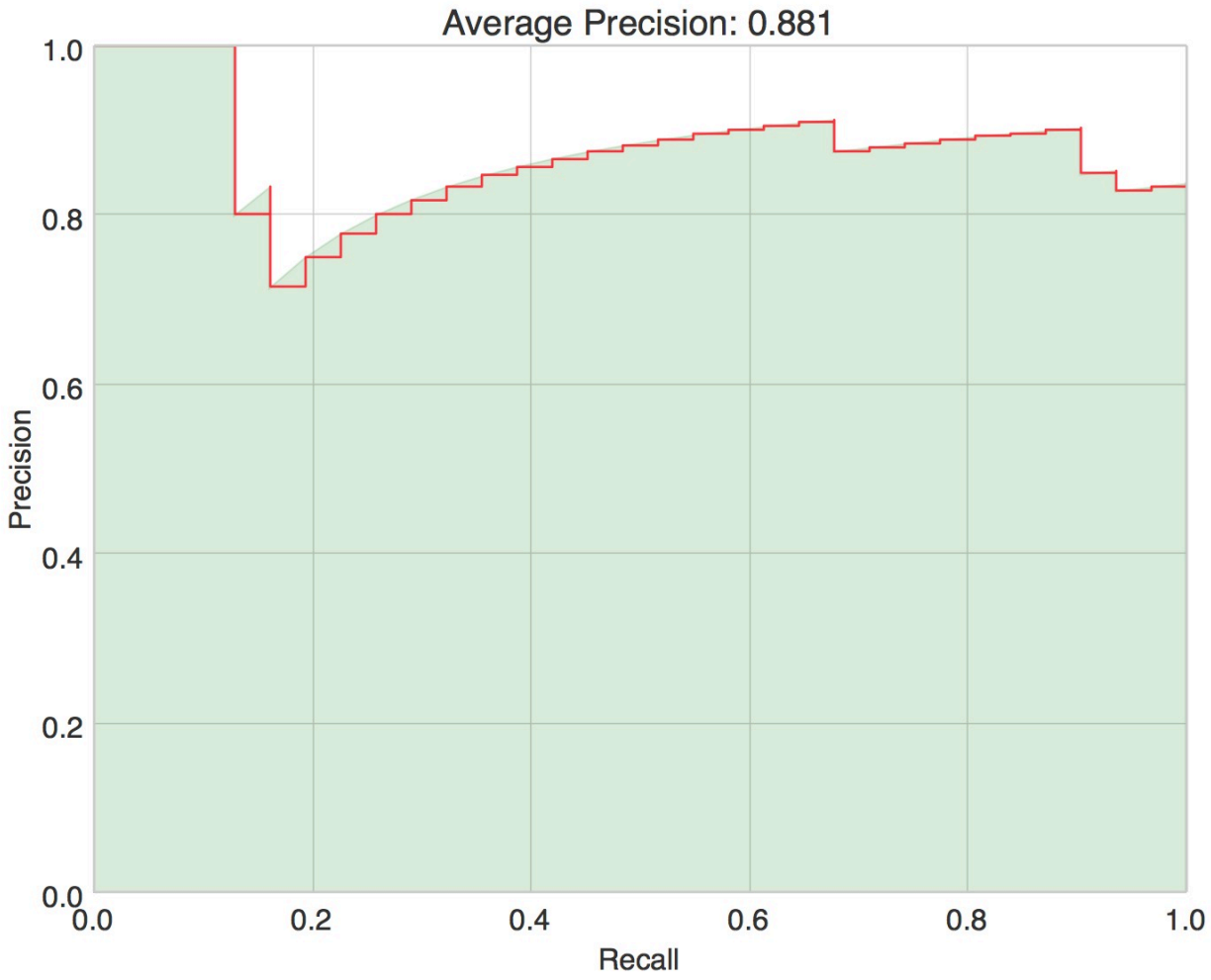
`friends.mpg` 資料



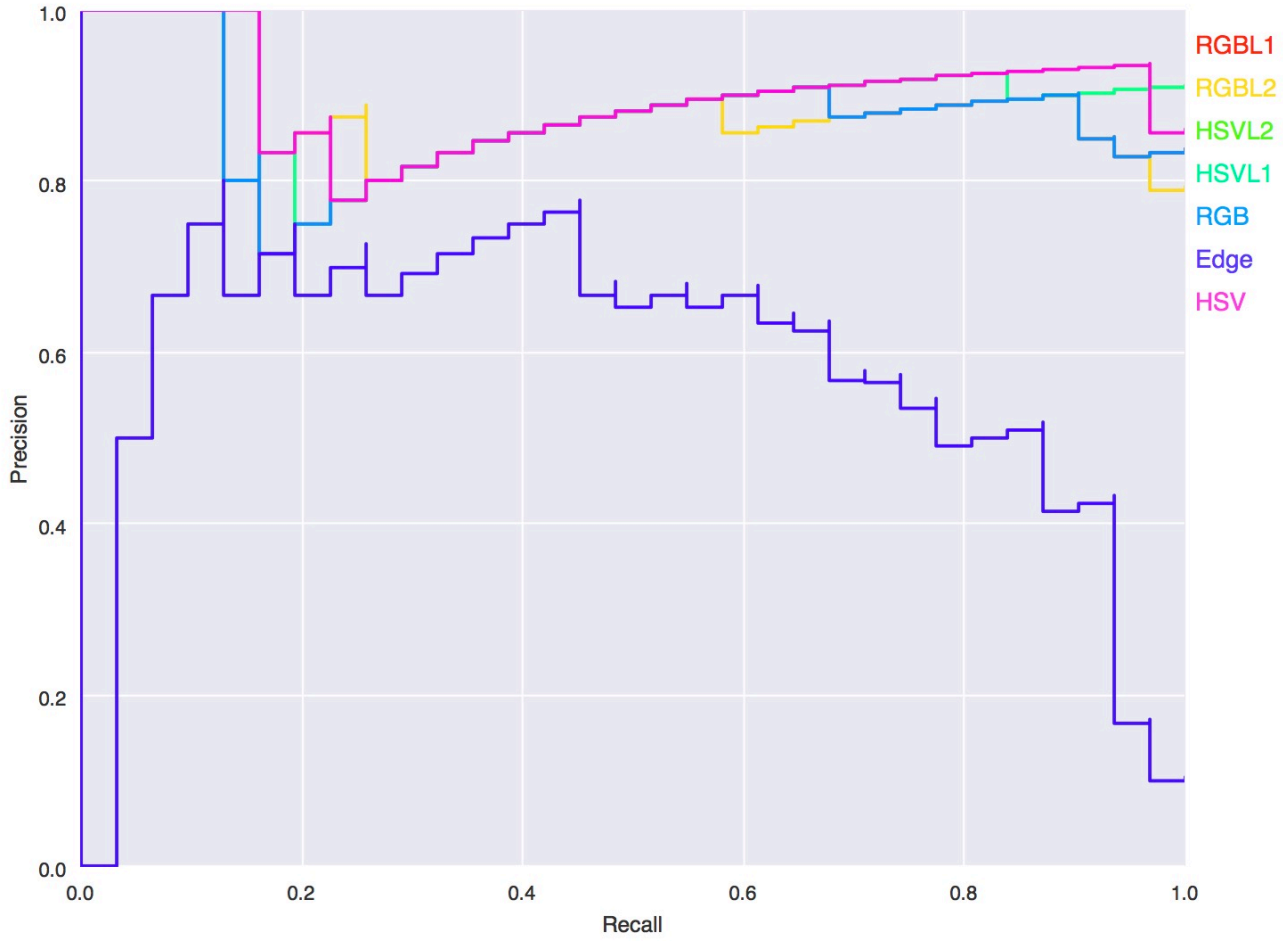
(`Edge`)



(`HSV` + `1-Histogram Intersection`)



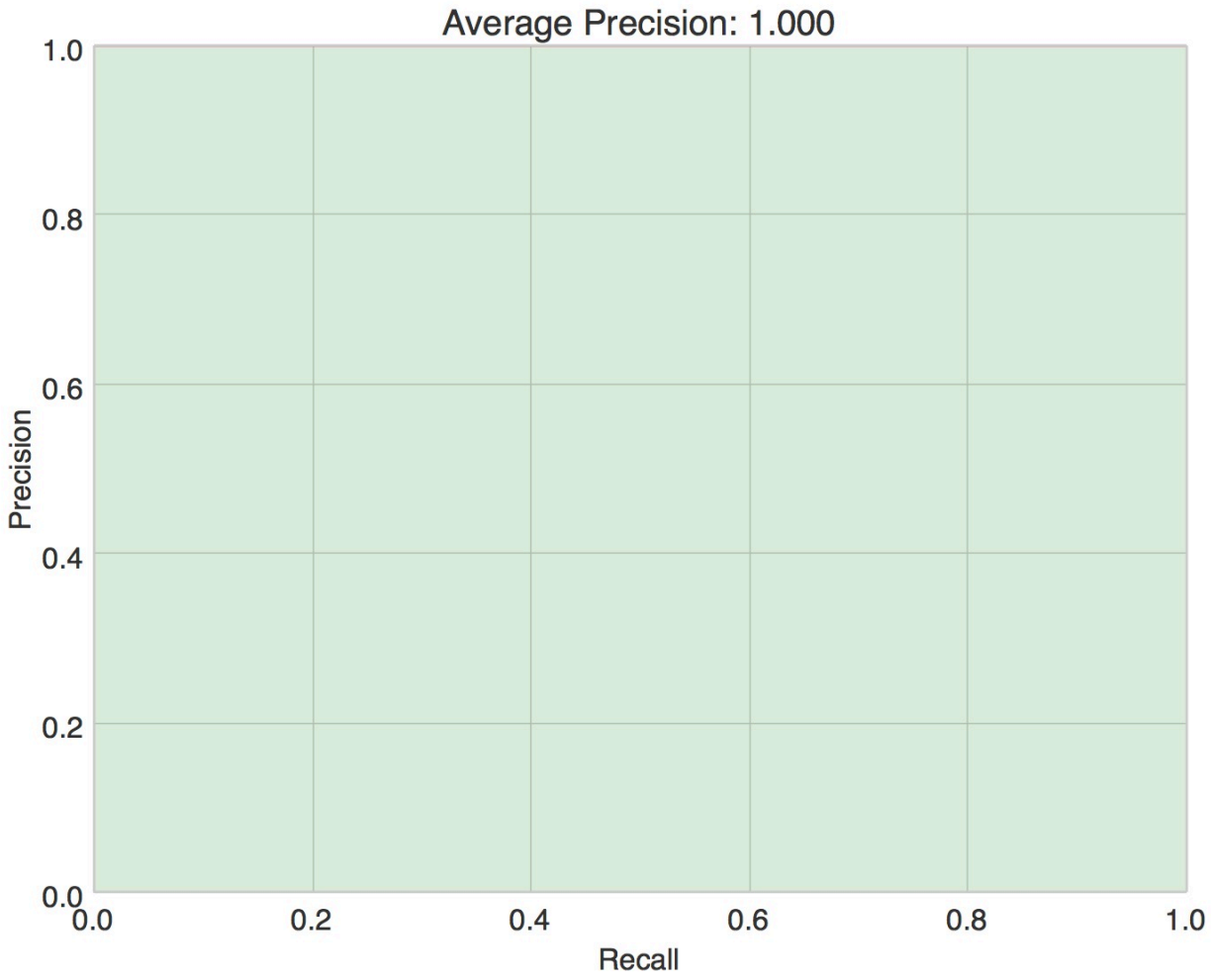
(`RGB` + `1-Histogram Intersection`)



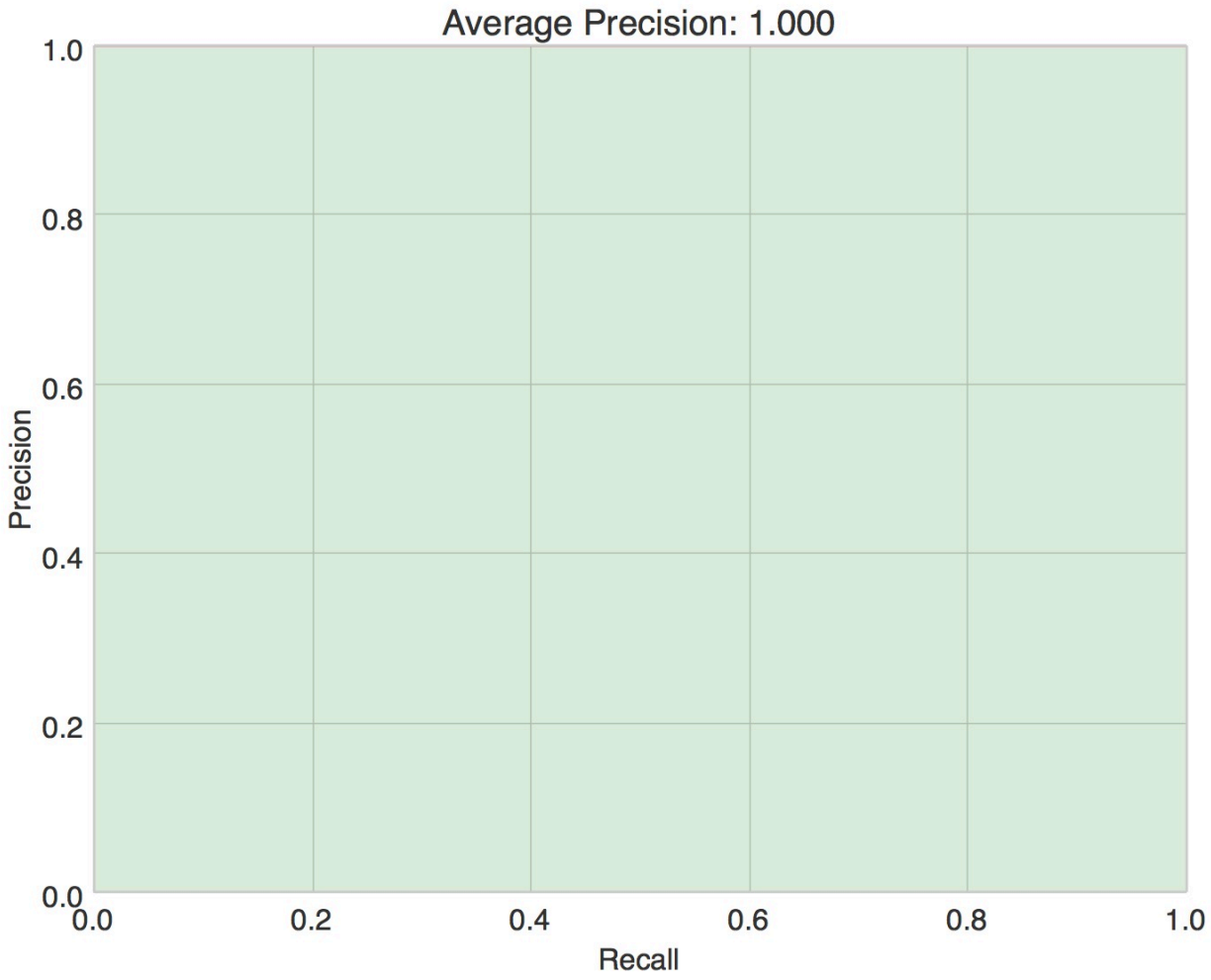
(`HSV` , `RGB` 兩條曲線都使用 `1-Histogram Intersection` ，後面的圖也用一樣的標記方式)

由上方圖片可以看出，一般來說 `Color Histogram` 的版本是較佳的、 `Edge` 的版本效果不好， `Color Histogram` 中，使用 `HSV` color space 的版本表現較好、使用 `HSV` + `1-Histogram Intersection` 的版本效果尤佳，而不同的距離計算方式，效果差異不大。

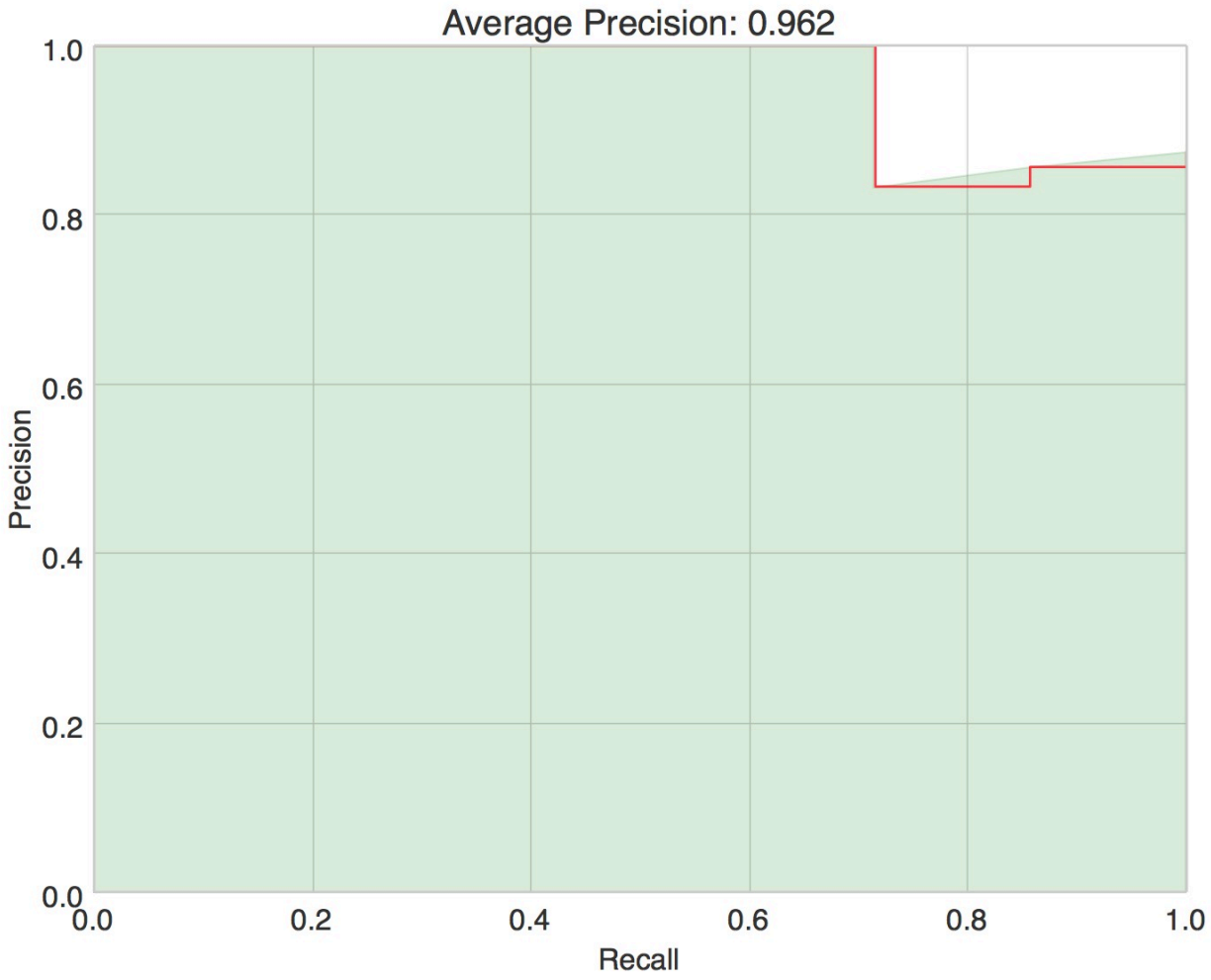
`news.mpg` 資料


Average Precision: 1.000
1.0
0.8
0.6
0.4
0.2
0.0
Precision
0.0
0.2
0.4
0.6
0.8
1.0
Recall

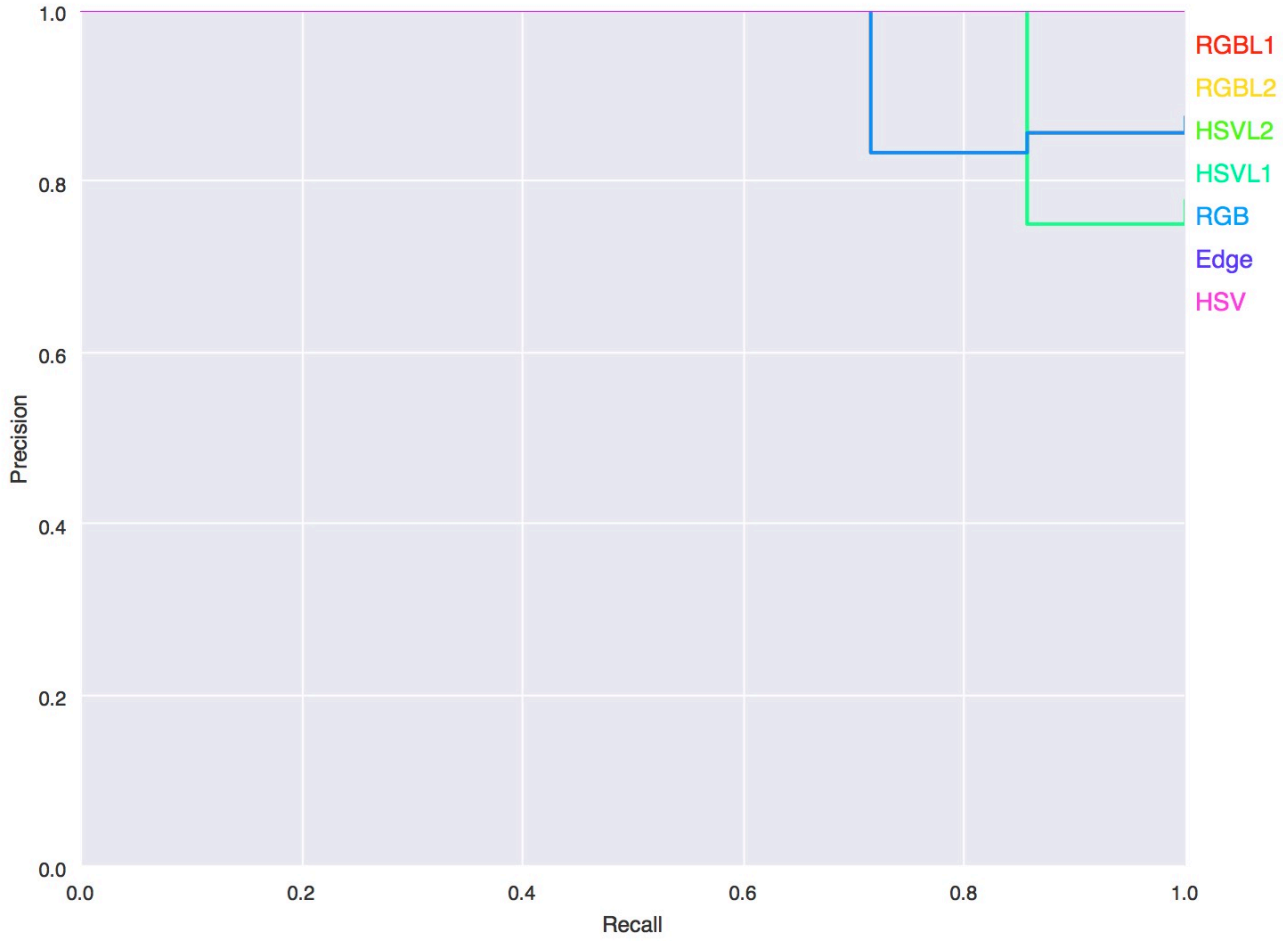
(`Edge`)



(`HSV` + `1-Histogram Intersection`)



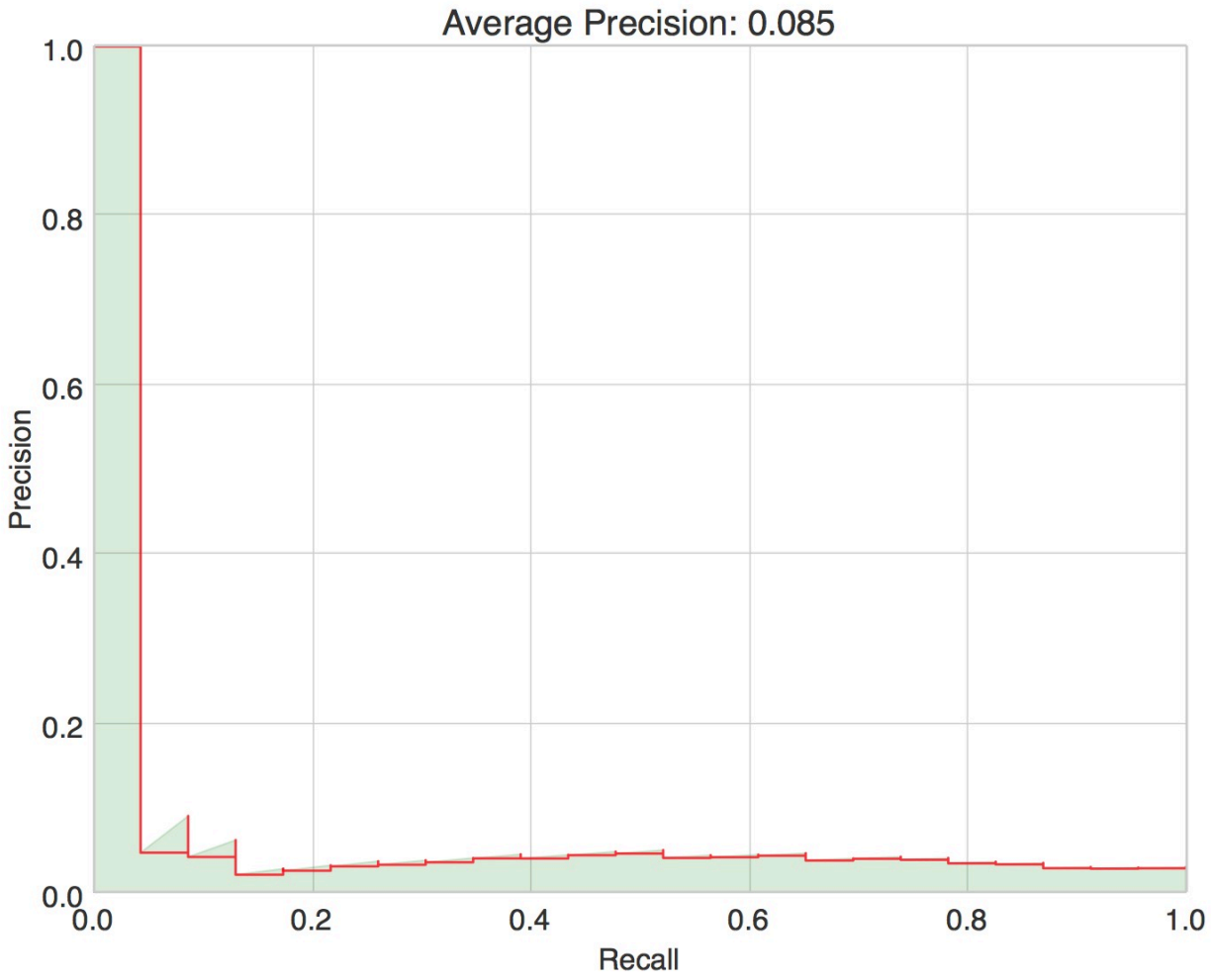
(`RGB` + `1-Histogram Intersection`)



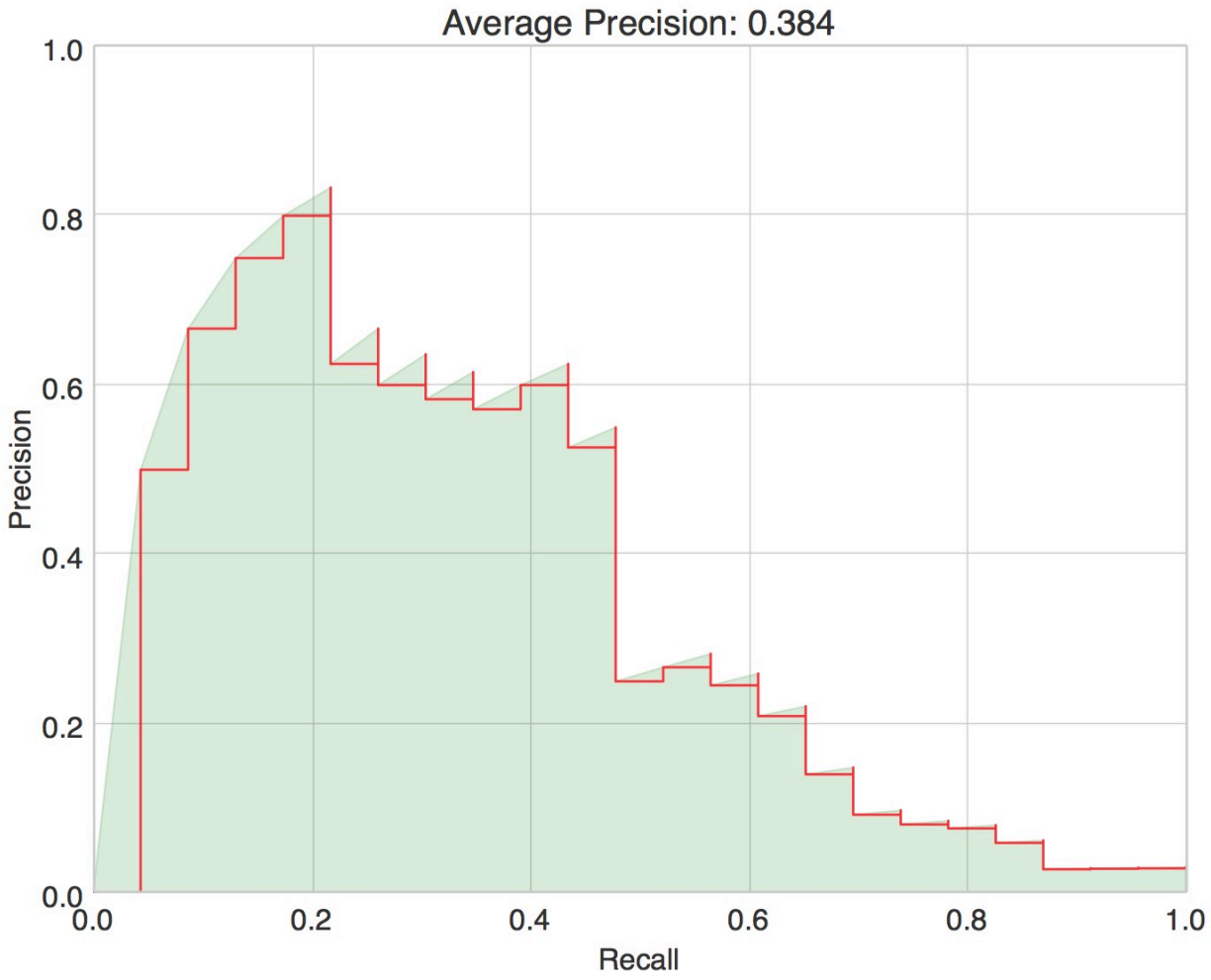
(總評)

由上圖可以看出，每個方法效果都非常好。

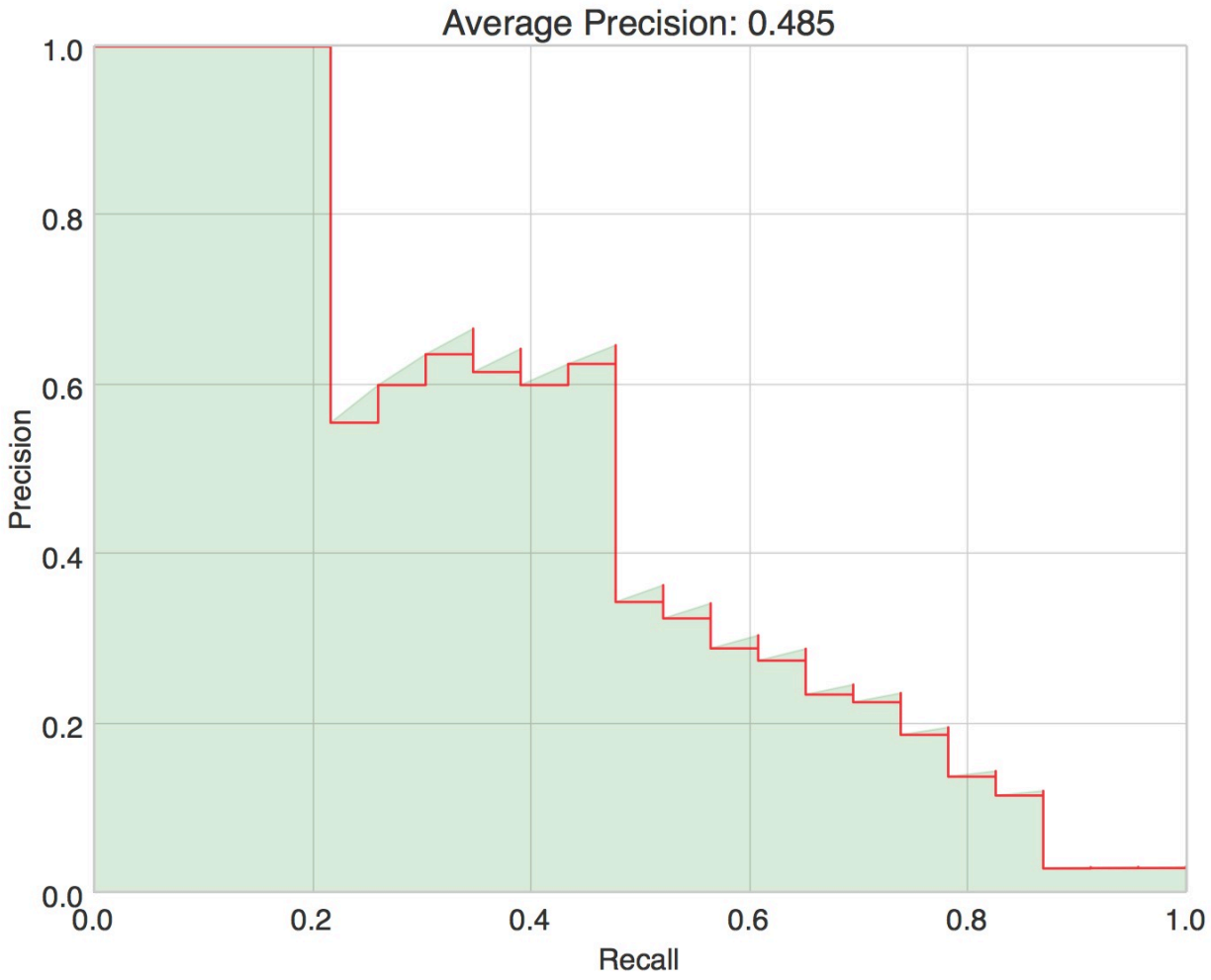
`soccer.mpg` 資料


Average Precision: 0.085
1.0
0.8
0.6
0.4
0.2
0.0
Precision
0.0
0.2
0.4
0.6
0.8
1.0
Recall

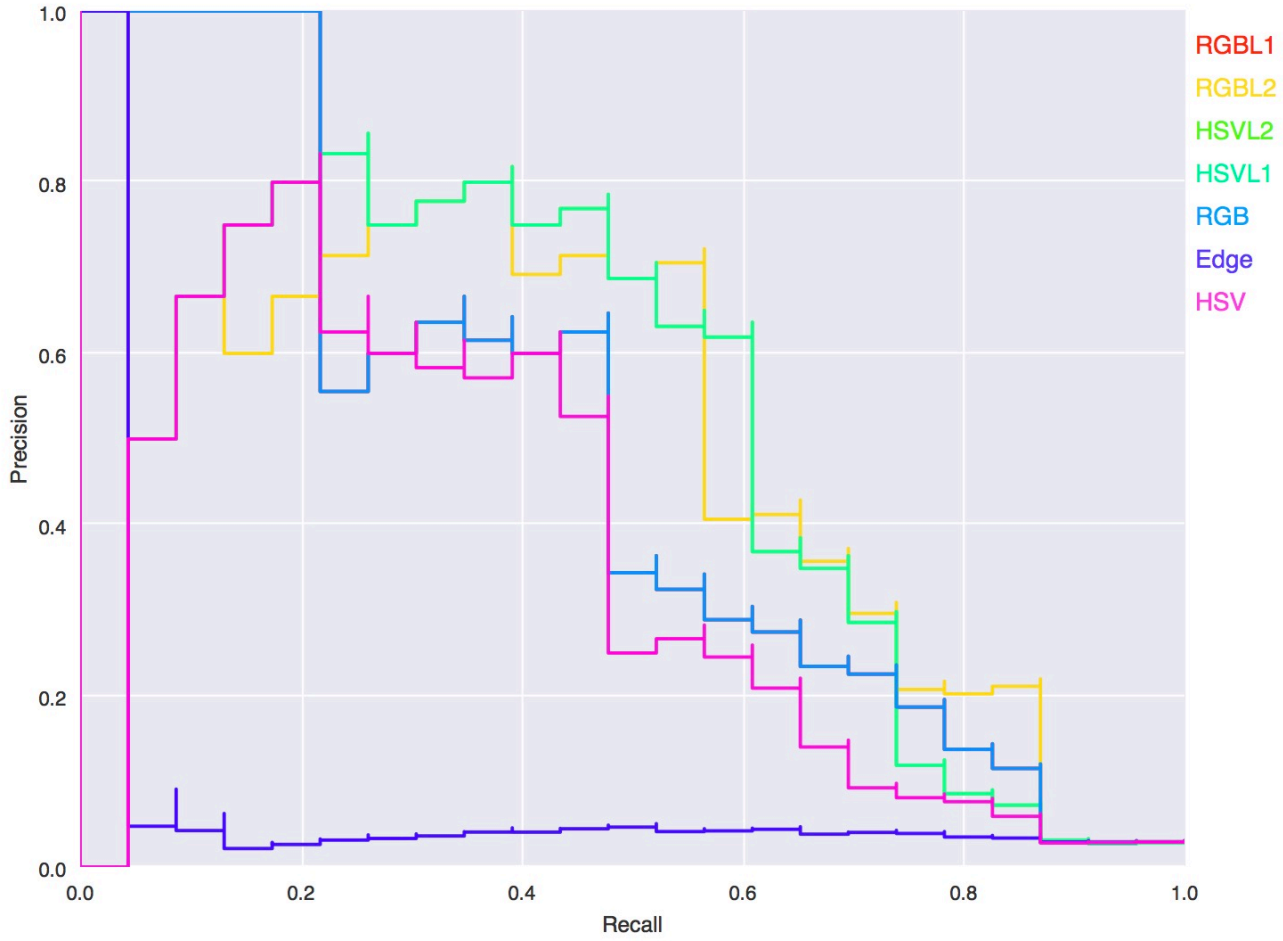
(`Edge`)



(`HSV` + `1-Histogram Intersection`)



(`RGB` + `1-Histogram Intersection`)



(總評)

由上圖可以看出，對於這段影片，使用 `RGB` color space 會優於 `HSV`，在同個 color space 下，距離使用 `L1` / `L2` 來計算，效果都優於 `1-Histogram Intersection`， `Edge` 的效果很差。

結論

- 使用基於 `Color Histogram` 的方法，得到的效果一般來說都不錯
- 不要使用 `Edge Change Ratio` ，效果不太好
- 在 `HSV` color space 使用 `Color Histogram` 偵測 shot change，效果也許會略優於在 `RGB` color space 好
- 在 `Color Histogram` 方法裡，利用 `Histogram Intersection` 計算距離，效果一般來說都不錯

附錄/其他

那麼，這隻程式在其他影片的表現如何呢？

以下是一些利用它收集到的 key frames：

節錄 shot change:

id	frame_n
0	63
1	100

2	139
3	285
4	395
5	550
6	595
7	688
8	712
9	795
10	840
11	1034
12	1111
13	1223
14	1289
15	1440
16	2015
17	2161
18	2180
19	2315
20	2668
21	2771
22	2895
23	2994
24	3007
25	3168
26	3328
27	3520

28	3624
29	3730
30	3762

影片來源：

youtube: 【MMD】Tda式ミクをトゥーンっぽくしたい「JEWEL」

niconico: 【MMD】Tda式ミクをトゥーンっぽくしたい「JEWEL」