影像處理作業

運用Canny邊緣偵測與遮罩實作Instagram濾鏡

系級：資工碩一

學號：407411056

姓名：林妤潔

2023年1月10日

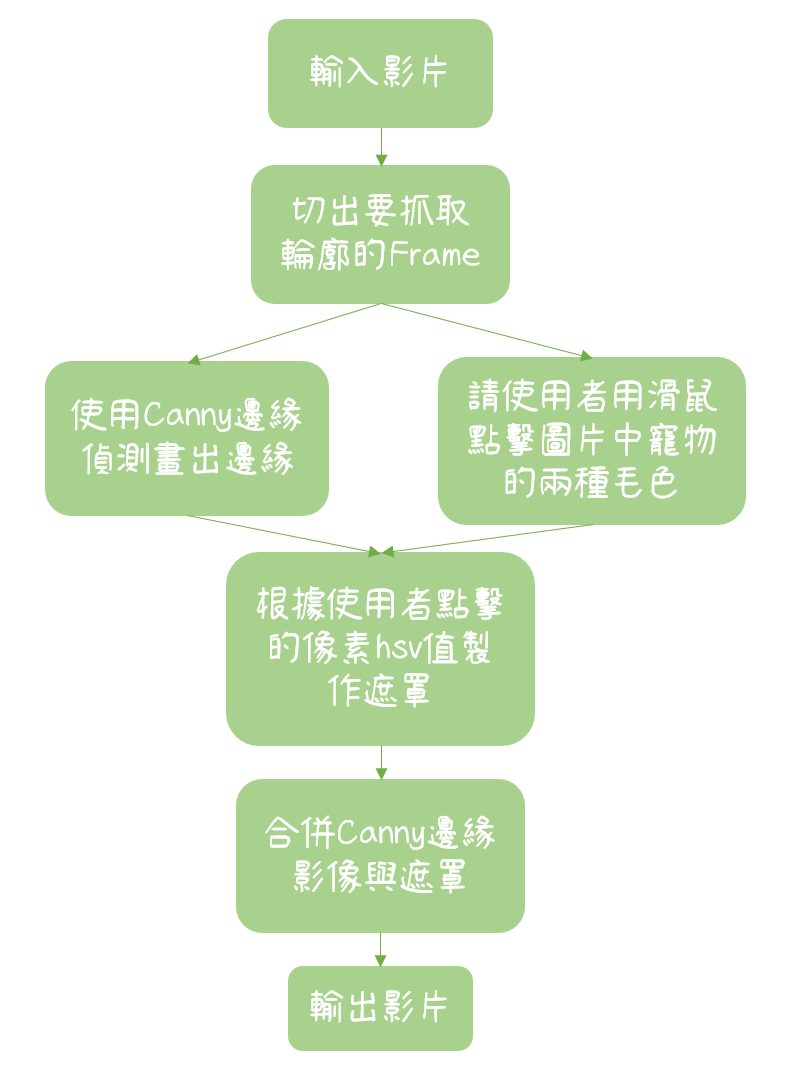
1. 動機與目標

知名社群軟體平台「Instagram」近幾年有許多人製作了一些短影片濾鏡，並透過此平台，分享給使用者使用。也有些人會使用繪圖軟體或剪輯軟體製作特色影片後上傳到平台分享。近期我看到有人使用Adobe Photoshop、Adobe Premiere Pro製作了可愛的輪廓寵物影片，參考影片連結如下面所示。但因為製作此影片需要手動描繪寵物的輪廓並加入音訊，所以製作一支影片的時間成本可能需要五到七分鐘不等。因此本次作業的目的是希望透過影像處理的方式，簡單的輸入一段影片，就會有處理過輪廓和音訊的影片出來，藉此減少人工作業的時間。

參考影片連結：<https://www.youtube.com/watch?v=mgJdd2LegS0>

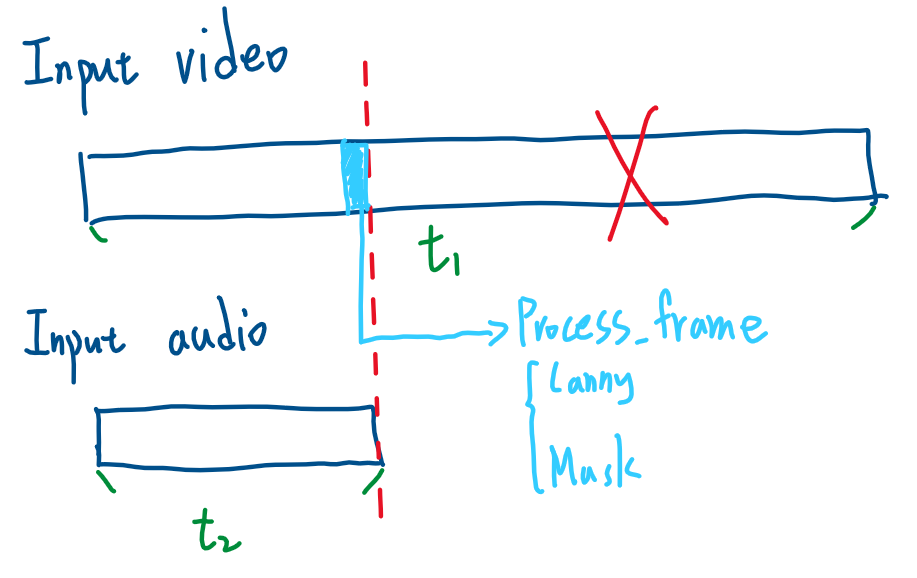
1. 實作過程與方法

本次作業希望使用者能夠輸入影片後，經過處理後，輸出完成的影片，整體流程如「圖一」所示。



圖一、整體流程圖

輸入影片後為了要達到上面參考影片的效果，所以影片的長度會切成與音效持續時間一樣的長度，並且抓出最後一幀影像進行輪廓的偵測與遮罩的製作，「圖二」為示意圖。



圖二、輸入影片處理示意圖

本次作業使用影像處理的方法有：

* 1. Canny邊緣偵測

Canny演算法首先使用了高斯濾波器去除雜訊，接著透過Sobel濾波器找出邊緣的強度與方向，並根據四種梯度方向計算梯度變化最大的點，除了最大的點之外，其餘的點皆設為0，最後透過自訂的高界線與低界線數值找出邊緣。

* 1. Morphological Closing

Closing是先經過膨脹(dilation)運算後再進行腐蝕(erosion)運算，可以將影像中的破孔或是分離但相距不遠的區塊連結再一起。

首先將要處理的影像顯示給使用者，並請使用者點擊寵物的兩種不同毛色(或是顏色光澤不一樣的地方)，這是為了利用影像的HSV值製作去邊緣遮罩，且通常寵物(這邊都用貓咪當輸入資料)會有超過一種以上的毛色，所以這邊才要請使用者點擊兩次，製作出來的遮罩才會比較完整。獲得使用者給的HSV值後，分別定義H、S、V區間值，得到兩組最高與最低的HSV值。

假設使用者點擊的兩個點之HSV值為：

Point1 = [150, 157, 13]

Point2 = [177, 32, 80]

我們自行定義的H、S、V的區間值為：

H\_range = 10 S\_range = 20 V\_range = 20

則兩組HSV值分別為：

Low\_Point1 = [150-10, 157-20, 13-20] → [140, 137, 0]

High\_Point1 = [150+10, 157+20, 13+20] → [160, 177, 33]

Low\_Point2 = [177-10, 32-20, 80-20] → [167, 12, 60]

High\_Point2 = [177+10, 32+20, 80+20] → [180, 52, 100]

因為HSV的區間為[0-180, 0-255, 0-255]，加上或減去區間值若超出範圍會直接設為最高或最低值，防止程式出現錯誤。另外通常貓咪最常出現的毛色是白色、黑色以及灰色，所以參考HSV的顏色對照表後，在區間值的設置上我覺得這組數字對出現白色、黑色、灰色的HSV值可能會有較好的表現。

得到兩組HSV值後我使用opencv中的inRange來製作遮罩，inRange會將低於Low\_Point的像素值和高於High\_Point的像素值設為黑色，在區間內的像素值設為白色，因此藉由inRange函式可以得到兩個只有黑和白的遮罩。而因為當初選的HSV值是兩個不同毛色的代表值，所以將這兩個遮罩使用opencv中的bitwise\_or函示將所有白色的像素保留合成一個遮罩，「表一」為實作遮罩的結果。

|  |  |
| --- | --- |
| 原始影像，黃圈為使用者點擊的像素 | merge\_mask = cv.bitwise\_or(遮罩一, 遮罩二) |
|  |  |
| 遮罩一  Point = [150, 157, 13]  Low = [140, 137, 0]  High = [160, 177, 33] | 遮罩二  Point = [177 32 80]  Low = [167, 12, 60]  High = [180, 52, 100] |
|  |  |

表一、藉由HSV值取得遮罩實作

由於合成出來的遮罩有許多的破洞，所以使用opencv中的morphologyEx函式的closing運算修補破洞，Kernal是使用5\*5的橢圓型矩陣

[ [0 0 1 0 0 ]

[1 1 1 1 1 ]

[1 1 1 1 1 ]

[1 1 1 1 1 ]

[0 0 1 0 0 ] ]

修補情形如「表二」所示。

|  |  |
| --- | --- |
| merge\_mask | closing\_mask |
|  |  |

表二、破洞修補

但因為修補的效果看起來不好，所以我又單獨做了膨脹和侵蝕，而因為要保留邊緣，所以在參數的設定上，我讓侵蝕的效果稍微小一些，確保邊緣不會因為誤差而被遮蓋，「表三」為再次做膨脹與侵蝕的效果。

|  |  |
| --- | --- |
| closing\_mask | mask |
|  |  |

表三、補強破洞並確保邊緣遮罩

接著將一開始經過Canny計算好的邊緣影像與我們製作好的遮罩使用opencv的bitwise\_and函式，過濾掉不需要的邊緣，保留遮罩內的邊緣，Canny的參數目前設為(100,200)，可以根據想要多少細節調參數，如「表四」所示。

|  |  |
| --- | --- |
| Canny計算出的邊緣影像 | 加上遮罩後的影像 |
|  |  |

最後將加上遮罩後的影像image\_mask寫入原本切好的影片中，並加上輸入的音訊一起輸出成影片，image\_mask和影像結合的影像如「圖三」所示。



圖三、最後原始影像與image\_mask合併

1. 實作結果與討論

本次作業實際拍攝家中貓咪作為輸入資料，總共有9支影片，為了方便觀看，我將9支影片接在一起變為1支影片，影片連結如下面所示。

影片連結：<https://youtube.com/shorts/ExR4HmZh508?feature=share>

也可以到data/output\_videos/audio的資料夾觀看實作結果。

實作結果一：

點擊之HSV值： HSV1:[3, 200, 14] HSV2:[21, 111, 53]

可以看到這張影像的背景複雜，導致Canny找到了很多邊緣，且這隻貓咪的毛色與後面的黑色鞋子相近，故在製作遮罩時容易將偏黑色的像素也納入範圍，造成有些背景點的邊緣被遮罩保留了下來，但也成功濾掉大部分的地板邊緣，貓咪的輪廓也基本上都被保留下來。

|  |  |  |  |
| --- | --- | --- | --- |
| Canny edge | mask | edge+mask | output |
|  |  |  |  |

實作結果二：

點擊之HSV值： HSV1:[155, 96, 16] HSV2:[0, 24, 73]

這隻貓咪的毛色僅有單一個顏色，且背景點沒有太雜亂也沒有太多像素與貓咪毛色相近，因此可以很容易的製作相對應的遮罩來將一些背景點的邊緣去掉。

|  |  |  |  |
| --- | --- | --- | --- |
| Canny edge | mask | edge+mask | output |
|  |  |  |  |

實作結果三：

點擊之HSV值： HSV1:[176, 16, 223] HSV2:[10, 89, 155]

這隻貓咪的毛色較為複雜，且右邊背景較為雜亂，因此Canny抓取的邊緣數較多，另外因為毛色與背景相似，所以製作遮罩時也容易將相似的像素保留，導致有一些邊緣還留在最後的結果上。

|  |  |  |  |
| --- | --- | --- | --- |
| Canny edge | mask | edge+mask | output |
|  |  |  |  |

實作結果四：

點擊之HSV值： HSV1:[131, 55, 51] HSV2:[106, 18, 245]

這張影像的背景為大馬路和樹葉，像素的變化很多，因此Canny抓取到了很多的邊緣。另外這隻貓咪的毛色雖然有兩種，但因為背景沒有太多與毛色相似的像素(除了右上角車牌之外)，因此可以看到遮罩幾乎是把背景點的邊緣濾掉了。

|  |  |  |  |
| --- | --- | --- | --- |
| Canny edge | mask | edge+mask | output |
|  |  |  |  |

實作結果五：

點擊之HSV值： HSV1:[5, 29, 53] HSV2:[15, 4, 240]

這張與上一張的背景很像，但因為亮度的不同所以相較於上一個結果來說，這次無法很好的將馬路的一些邊緣濾掉，但貓咪的輪廓還是有很好的保留下來。

|  |  |  |  |
| --- | --- | --- | --- |
| Canny edge | mask | edge+mask | output |
|  |  |  |  |

實作結果六：

點擊之HSV值： HSV1:[9, 87, 152] HSV2:[9, 184, 86]

這張影像有太多的像素與貓咪的毛色過於相近，且影像較模糊，Canny也沒有將貓咪的輪廓抓得很好，所以可以看到遮罩的效果並沒有太好，還是有很多非貓咪的輪廓出現。

|  |  |  |  |
| --- | --- | --- | --- |
| Canny edge | mask | edge+mask | output |
|  |  |  |  |

實作結果七：

點擊之HSV值： HSV1:[11, 63, 97] HSV2:[167, 255, 7]

這張影像的貓咪與背景的顏色較不相似，但因為Canny抓取輪廓並沒有將貓咪的右半邊輪廓找出，所以雖然遮罩看起來製作的不錯，出來的效果看起來卻感覺很多雜訊。這邊推測可能是Canny的參數必須微調，讓他找多一點輪廓看起來的效果才會好。

|  |  |  |  |
| --- | --- | --- | --- |
| Canny edge | mask | edge+mask | output |
|  |  |  |  |

實作結果八：

點擊之HSV值： HSV1:[0, 255, 14] HSV2:[15, 27, 229]

這張影像的輪廓就抓得很好，但因為這隻貓咪的毛色與背景有很多地方相似，所以在濾掉多餘的輪廓時遮罩並沒有起太大的作用。

|  |  |  |  |
| --- | --- | --- | --- |
| Canny edge | mask | edge+mask | output |
|  |  |  |  |

實作結果九：

點擊之HSV值： HSV1:[12, 89, 89] HSV2:[3, 187, 15]

這張影像的輪廓也沒有抓到很好，且因為光線關係有些影子和貓咪的毛色過於相似，因此在影子的地方還是可以看到沒有被濾掉的輪廓。

|  |  |  |  |
| --- | --- | --- | --- |
| Canny edge | mask | edge+mask | output |
|  |  |  |  |

1. 結語

本次作業利用影像處理的技術實作了一個小的製作寵物輪廓濾鏡之程式，從結果來看如果結合深度學習做影像辨識，框出要處理的部分後再進行邊緣偵測應該能夠減少更多不必要邊緣，可能比起現在的結果要好。且遮罩會因為使用者點擊的HSV值而有所偏差，萬一點擊的像素與想描繪輪廓的物體無關的話，遮罩的作用就變成保留與物體無關的輪廓了。未來也許可以嘗試變成分析整張圖的資訊(如HSV值)後找到物體特徵的範圍，但也因為本次作業僅使用影像處理的技術，讓我能更細膩的思考一些filter的選擇、參數的設置、遮罩的重要性、顏色的調整等等的問題，未來會嘗試結合深度學習的方法更準確的將需要的輪廓抓出來。