Homework#2：homography程式說明:

方法: **Direct Linear Transformation (DLT)** —— 利用奇異值分解singular value decompositio(SVD)，找出兩張圖片相對應的homography matrix。

附檔案說明:

|  |  |
| --- | --- |
| 原始圖片檔 | GroundTruth.jpg |
| 拍照圖片檔 | test.jpg |
| DTL轉換後圖片檔 | M10707307.jpg |
| 誤差BMP檔 | M10707307.bmp |
| 執行檔 | Project1.exe |

步驟1. 在**目標圖片GroundTruth.jpg**與實際圖片test.JPG中，均勻地標出9個對應特徵點，並存入A矩陣中，進行正規化(標準化)處理。

步驟2. 宣告並初始化SVD所用的矩陣，單位為long bouble

步驟3. 計算SVD的V矩陣，並將V矩陣最後一列元素放入H(33)矩陣中，迭代次數為200次，將條件判斷設為D(特徵值)>。

步驟4. 將H矩陣進行正規化，使得H(2,2)=1。

步驟5. 掃描test. jpg中符合範圍的所有像素，設為矩陣(包含X,Y,Z)，並將正規化，得到。

步驟6. 判斷投影過程是否落在圖像大小中，利用，把test.jpg所有像素投影到**M10707307.jpg**中。

步驟7. 將原本**目標圖片檔GroundTruth.jpg**與**DTL轉換後圖片檔M10707307.jpg**進行相減運算，比較轉換中所產生的誤差，並儲存成M10707307.bmp。

步驟8. 將檔案儲存成release檔。

宋俊賢

M10707307

讀檔案(開檔案測試)Flag =(|

邏輯判斷是要分開寫(不要寫在同一個) 效率還是不一樣

openCV矩陣乘法不會比double快

K[R|t]XP 不要放在for裡面 會多做1000000次

變成p0~p10

判斷方式以<image[i].at<uchar>ix,iy<128 這樣部會又漏判斷

Fprtintf(跟硬體相關的事情，不要算時間)

主要運算程式

然就getcpuTickCount 抓執行時間

%f\n”,1000.0f(宣告要注意)/cpu

可以做到realtime

A-B 要用絕對值

自動(兩張照片 點數固定)，情境是可以掌控 照片拍斜的

1.四個點的排序

2.binary threshold

3.find extrem

要找極端點

每一個座標去做內機(白色的部分 ex找最右上方的那一點

Max(X2d點成(1,1)) 1,1方向得投影

用相機的比例

如果很彎曲就會失敗

4.homograpy 兩張影像

UI手動 寫介面

點四個點