Report

1. explain your implementation.

本次作業使用講義所教之方法來實現圖片的拼接任務，實施步驟大略如下：

* SIFT尋找特徵點
* KNN做兩張圖片之特徵點配對
* RANSAC多次迭代尋找兩張圖片間之最佳轉移矩陣H
* 使用最佳轉移矩陣完成圖片拼接

接下來將解釋我在程式中如何實現這些步驟。

* 1. **SIFT尋找特徵點**

SIFT主要用來偵測並描述影像中的局部特徵，在不同的scale space中尋找極值點後，取得這些極值點的位置、尺度、旋轉等資訊，最後計算出所有局部特徵點的方向。

一張含有 文字, 螢幕擷取畫面, 字型 的圖片

自動產生的描述

此步驟使用cv提供的operation實現，首先將圖片轉至灰階作為輸入，接著透過cv提供的operation取得特徵點的座標、特徵向量兩項資訊，每個特徵向量都會是128維的形式。

每張欲拼接的照片都會透過此SIFT函式取得局部特徵點資訊。

* 1. **KNN特徵點配對**

該步驟透過KNN將每個特徵點與另一張圖片的每個特徵點descriptor做2-norm找出最小值(即相似度最高)，以此找出每個特徵點的匹配點。

由於特徵向量的維度高達128維，若使用暴力法直接將兩個128維的矩陣做2-norm距離計算，計算量將會非常大。因此我使用KDTree方法做KNN，善用KDTree的二元樹特性降低資料和時間複雜度，提高搜尋鄰近點的效率。

一張含有 文字, 字型, 螢幕擷取畫面 的圖片

自動產生的描述

首先將第一張圖片之特徵點轉換成KDTree之資料結構，接著使用sklearn提供之KDTree功能:tree.query，比較另一張圖中每個特徵點並取得最近點及次近點，存入match list。

一張含有 文字, 字型, 螢幕擷取畫面 的圖片

自動產生的描述

為避免最近點因其他點特徵相近導致錯誤匹配，Lowe’s Ratio test可以確保匹配點之準確性，比較最近點與次近點的比例，若小於閥值則判定為良好配對。此外，Lowe的論文中提到，經過他測試後的合適閥值建議在0.4~0.6間，由於baseline的匹配難度較低，設定較高的匹配閥值可以增加匹配點數量，提升匹配效果，因此我在baseline設定閥值為0.6。最後將良好的匹配組合記錄下來並回傳。

* 1. **RANSAC與Homography**

RANSAC在每次迭代中隨機採樣匹配點，計算出轉移矩陣H後，將第一張照片的每個特徵點經過H轉移到第二張照片的frame，比較每個轉移後的點與第二張照片的對應點之距離誤差值，小於指定閥值則為inlier，inlier數量越高，代表H越貼近理想轉移情形，記錄此H；經過多次迭代將記錄較佳的轉移矩陣，該轉移矩陣將用來進行最後拼接。

一張含有 文字, 螢幕擷取畫面, 字型 的圖片

自動產生的描述

此步驟程式主要實現方法如下：

* Random\_point函式在所有匹配點中隨機提取四組匹配點
* Homography函式使用該四組匹配點計算出轉移矩陣中之所有元素值，目的在於解出以下矩陣方程：

一張含有 文字, 白色, 字型, 黑與白 的圖片

自動產生的描述

由於矩陣並非n\*n形式，因此該處我使用pseudo inverse，以SVD解出h矩陣，並將h33標準化為1，對每個h元素值除以h33。函式實現如下：

一張含有 文字, 螢幕擷取畫面, 字型 的圖片

自動產生的描述

* Get\_error函式計算所有「照片一經H轉移到照片二之特徵點」與「照片二之匹配特徵點」做距離誤差比較，並記錄這些誤差值回傳給RANSAC做inlier數量比較。函式如下：

一張含有 文字, 螢幕擷取畫面, 字型, 軟體 的圖片

自動產生的描述

* 若這些匹配距離誤差值小於指定閥值，則加入inlier點，此閥值因圖片拼接難度設定不一，我在此設定為0.5。在每次迭代中比較該次的inlier數量是否比當前最高的還要多，以此紀錄最佳之H轉移矩陣。
  1. **圖片拼接(stitching)**

此步驟主要實現三件事情：重設圖片原點、圖片映射、邊界優化。

由於圖片在映射到目標圖片後，其圖片長寬值將會改變，因此須設定一個新的原點，讓圖片可以完整的將每個部份顯示出來並輸出，原理如下圖。

一張含有 文字, 圖表, 行, 螢幕擷取畫面 的圖片

自動產生的描述

此外，圖片拼接後，邊界因原始圖片rgb差異導致非常明顯，因此需對其做後處理，處理方法將會於後續說明。

我將整個拼接流程以一個函式實現，方法如下：

* 先將圖片二的8個轉角經由H映射到目標圖片，並從兩張圖共8個轉角找出x與y的最大與最小值。透過x、y的最小值當作兩張圖片之平移數值，取得平移矩陣，同時利用x、y的最大最小值設定圖片長寬值；接著將兩張圖片進行映射、平移及範圍調整，以取得接下來要用來合併的兩張輸入圖片。

一張含有 文字, 螢幕擷取畫面, 軟體 的圖片

自動產生的描述

* 合併過程同時也會處理邊界問題，處理方法透過兩張相同pixel的rgb資訊做2-norm比較，以rgb差異做不同的合併處理。

此方法分為三種情況：前兩種為當其中一張圖片為黑色，則該pixel之rgb值以有顏色之圖片為主；當兩張圖片皆有rgb值，則先將兩者的rgb值取2-norm，若數值大於所設定之閥值，則讓rgb比例傾向其中一張，反之則兩個rgb值比例各半。此方法實測後，邊界問題有顯改善。

一張含有 文字, 螢幕擷取畫面, 字型 的圖片

自動產生的描述

1. show the result of stitching 2 images.
   1. **baseline**

一張含有 天空, 視窗, 建築, Rectangle 的圖片

自動產生的描述

* 1. **bonus**

一張含有 天空, 戶外, 樹狀, 螢幕擷取畫面 的圖片

自動產生的描述

1. try to stitch more images as you can and compare with them.

我將6張baseline完成拼接，以及4張bonus的拼接。此部分我將說明如何完成多張圖的拼接，以及bonus的優化。

* 1. **多張圖的拼接**

首先，我同樣對每張圖片進行以SIFT找特徵點、KNN特徵點匹配的動作。接著透過ransac函式，找出所有圖片兩兩間的轉移矩陣H，如下圖：

一張含有 文字, 字型, 螢幕擷取畫面, 數字 的圖片

自動產生的描述

接下來將進行重複拼接，如下圖：

一張含有 文字, 字型, 螢幕擷取畫面 的圖片

自動產生的描述

此拼接方法的概念是，**將每張圖片都映射到第一張圖片做拼接**。當進行到第三張、第四張圖時，映射的輸入矩陣應為前述所有轉移矩陣的連乘，以及乘上第一張圖的平移A1，如此才能將後續的圖片完整映射到第一張圖。進行多次拼接後，最後一次的圖片輸出即為合併後的最終結果。

Baseline之6張圖片也是如此，然而由於實測發現其加入平移後的拼接效果稍差，因此最佳的版本中並未使用到平移矩陣A，僅使用H的連乘。程式實現如下：

一張含有 文字, 字型, 螢幕擷取畫面, 數字 的圖片

自動產生的描述

拼接結果如下：

一張含有 天空, 戶外, 螢幕擷取畫面, 視窗 的圖片

自動產生的描述

一張含有 天空, 螢幕擷取畫面, 戶外 的圖片

自動產生的描述

可以發現越後面拼接的圖片，其拼接效果會略為下降且扭曲情況較大，此部分由於本身拍攝角度之旋轉差異，導致後面的圖本來就會相對有扭曲情況，因此可以合理解釋。

* 1. **Bonus優化**

因bonus每張圖片之差異性較大，因此我在特徵點匹配部分提升了匹配的難度，Lowe’s Ratio test的閥值從baseline的0.6調整為0.4；在ransac的inlier判定中，也將其閥值由0.5降低為0.4，提升inlier之判定難度，同時將迭代次數增加，嘗試提升H準確性。

此外，也修正在拼接後之邊界處理方法，增加兩張圖pixel間的rgb差異範圍處理條件，以適應不同差異的rgb之融合比例。

經過上述調整前後比較如下：

調整前↓

一張含有 天空, 螢幕擷取畫面, 戶外, Rectangle 的圖片

自動產生的描述

調整後↓

一張含有 天空, 螢幕擷取畫面, 戶外 的圖片

自動產生的描述

可以觀察到「館」字的疊合以及樓梯部分有明顯改善，然而其邊界仍較為明顯，相對於原本之處理方法，已有稍微改善。