Merge branch 'release/a3'

drakeguan (author)
5 minutes ago

commit 67952f501f8c46307206
tree 4ac4b3ed50dbc5a36430
parent 48a857a91ae78ea4ccc6 parent 55c2c0863e6040a01c3e

vfx11spring_project2 /

name	age	message	history
artifact/	5 minutes ago	update readme. [drakeguan]	
image/	5 minutes ago	update readme. [drakeguan]	
program/	about an hour ago	... [drakeguan]	
README.md	5 minutes ago	update readme. [drakeguan]	

README.md

Digital Visual Effects, Spring 2011

project #2: Image Stitching (original link)

D99944013, Shuen-Huei (Drake) Guan, (drake.guan@gmail.com)

doc online: https://github.com/drakeguan/vfx11spring_project2/

Digital Visual Effects 2011 Spring @ CSIE.NTU.EDU.TW

Objective of the project

This project is my testing plan of examining the power and possibility of Open source for an university course, VFX, lectured by Prof. Yung-Yu Chuang. The project contains 4 sub-folders, each for an assignment. Some is a programming assignment while others might focus on visual effects demo. This project will host my working history of assignments and hopefully, there is something informative or helpful to myself and others :)

Introduction to the course

> With the help of digital technology, visual effects have been widely used in film production lately. For example, up to April 2004, the top ten all-time best selling movies are so-called "effects movies." Six of them even won Academic awards for their visual effects. This course will cover the techniques from computer graphics, computer vision and image processing with practical or potential usages in making visual effects.

Project description

Image stitching is a technique to combine a set of images into a larger image by registering, warping, resampling and blending them together. A popular application for image stitching is creation of panoramas. Generally speaking, there are two classes of methods for image stitching, direct methods and

feature-based methods. An example of direct methods is Szeliski and Shum's SIGGRAPH 1997 paper. Brown and Lowe's ICCV2003 paper, Recognising Panoramas, is a cool example for feature-based methods.

In this project, you will implement part of the "Recognising Panoramas" paper. There are basically five components in that paper, feature detection, feature matching, image matching, bundle adjustment and blending. You are required to do feature detection, feature matching, image matching and blending. For feature matching, we have talked about several options, SIFT, Harris and MSOP. You are free to make your own choice. If you want to implement SIFT, you can refer to this matlab implementation as a reference. For feature matching, if you want to speed up matching, you can use some kd-tree library such as ANN. You have four weeks to finish this project. However, to encourage you not to wait until the last minute, you are asked to submit your feature detection and matching part at the checkpoint. For checkpoint submission, it is enough to submit images with features displayed on them.

Algorithm

The following algorithms are implemented:

- Feature detector:
  - Moravec
  - Harris
  - rejector
  - features with low contrast
  - features on edge
  - features near image boundaries
- Cylindrical warping
- Feature descriptor:
  - SIFT descriptor
  - RANSAC on descriptors
- Feature matching
- Image matching
- Image blending
- Image reader: There is a bug in octave's imread such that I choose to use *jpgread*.

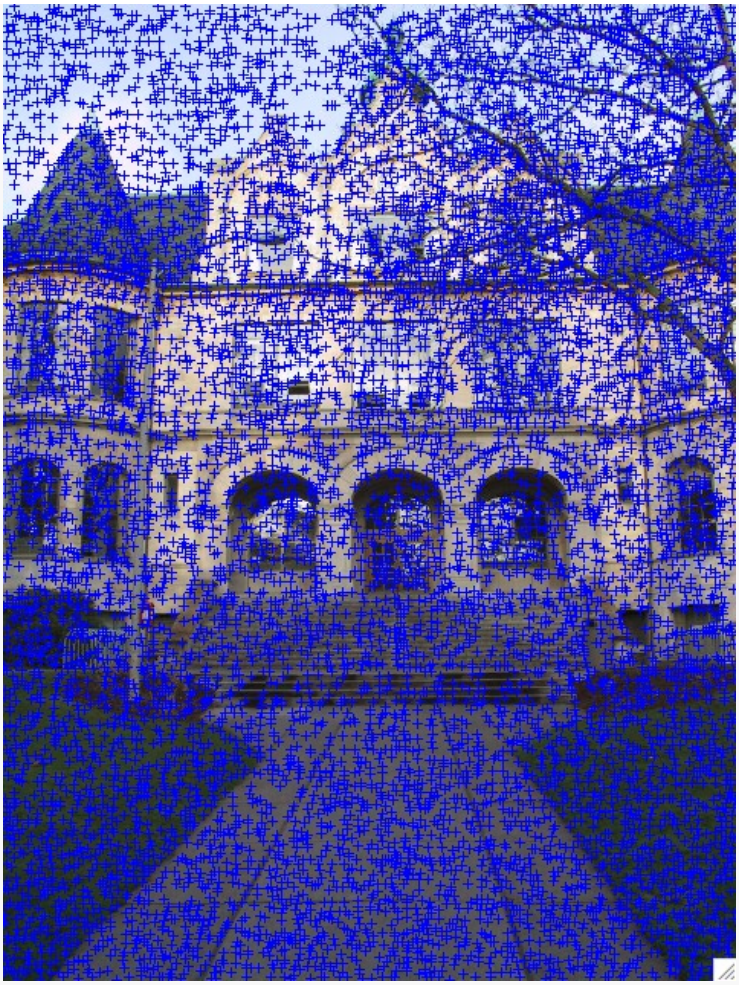
Approach

Taking photos

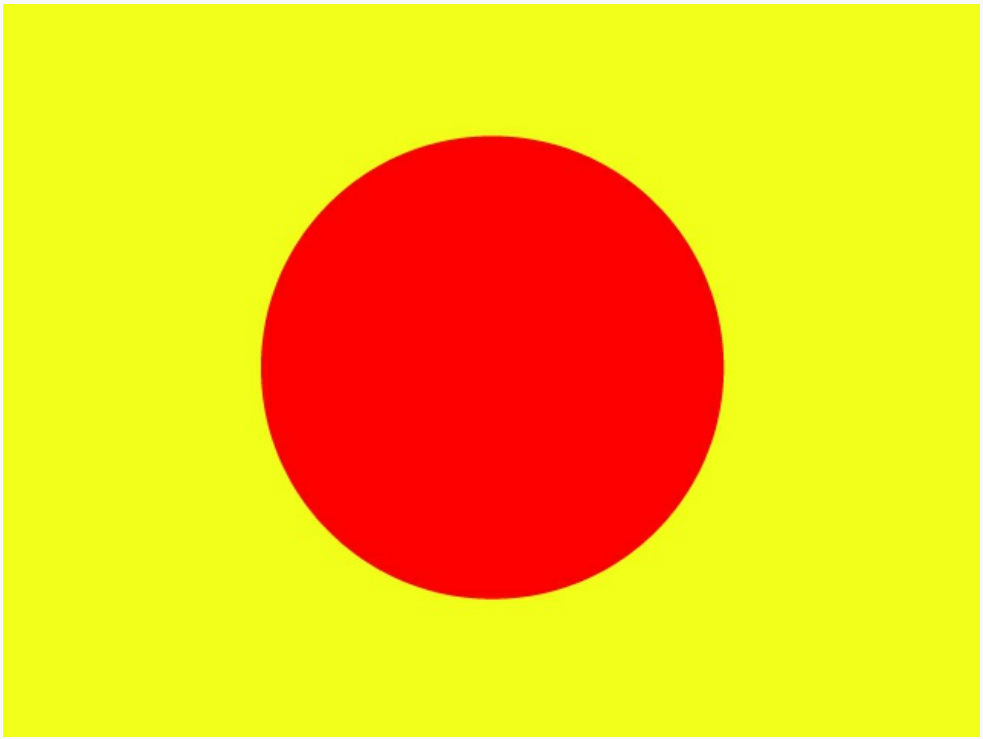
Photos are shot with a tripod in Taipei Maple (台北奧萬大) although you can't see any maple leaves.

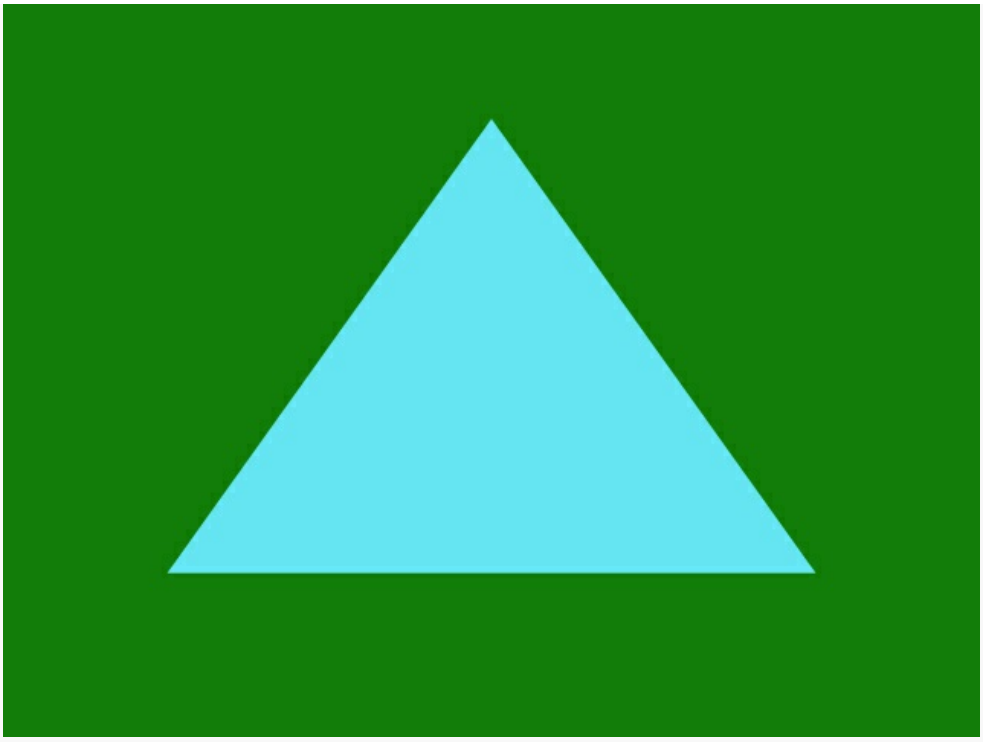
Feature detection

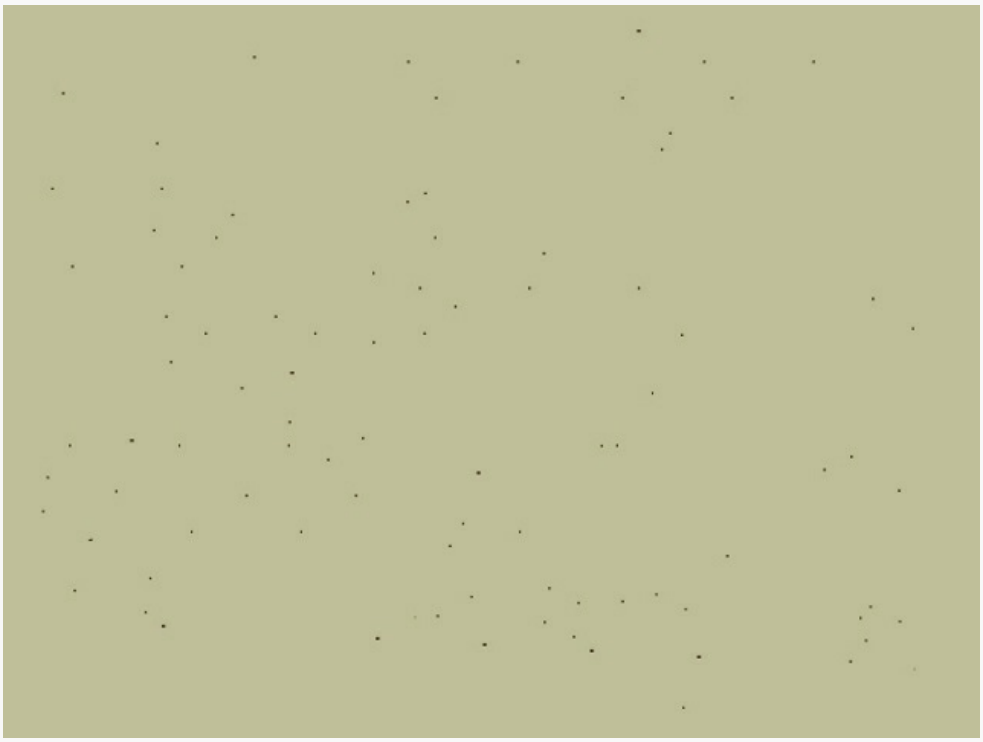
Moravec and Harris feature detectors are both implemented. I implemented Moravec to have the first idea about how feature detector works. The first trial is really terrible. Without any feature removal techniques, there are thousands of features in the following image. It's really hard to tell they are features or not.

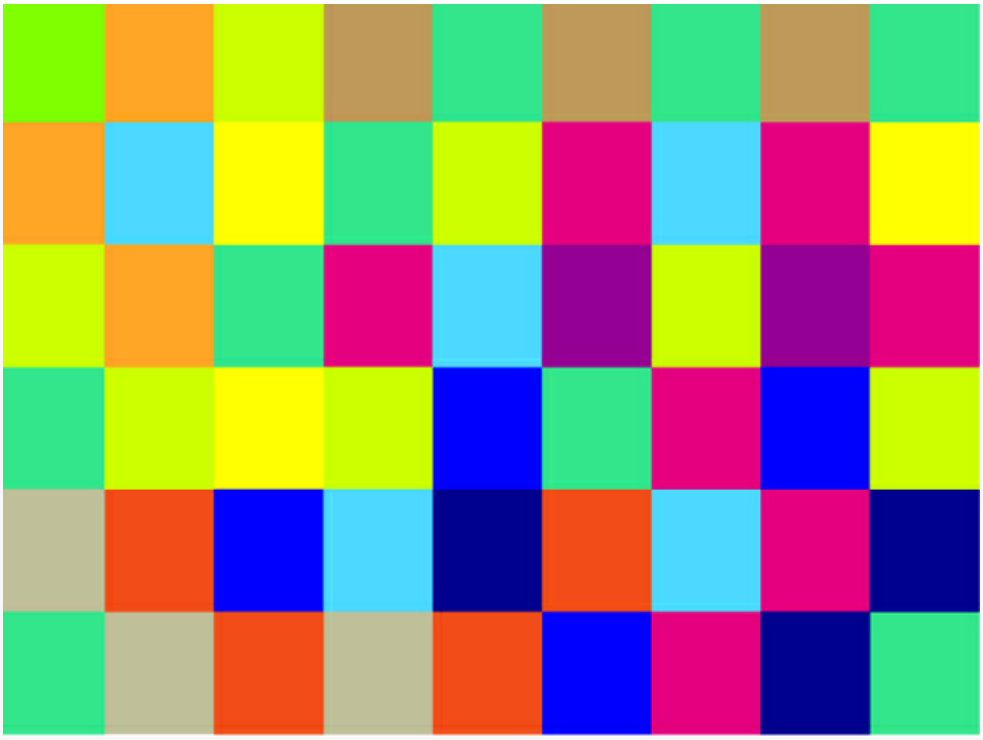


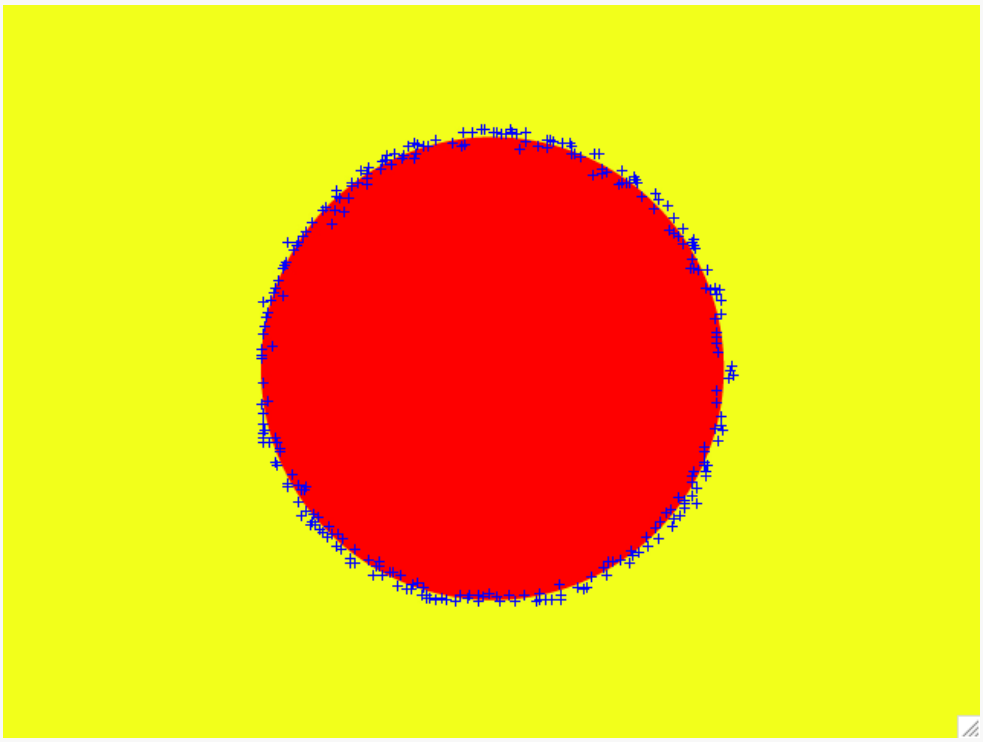
In order to make it much clear and helpful for begineers like me, I designed 4 testing images and feed them into Moravec detector.

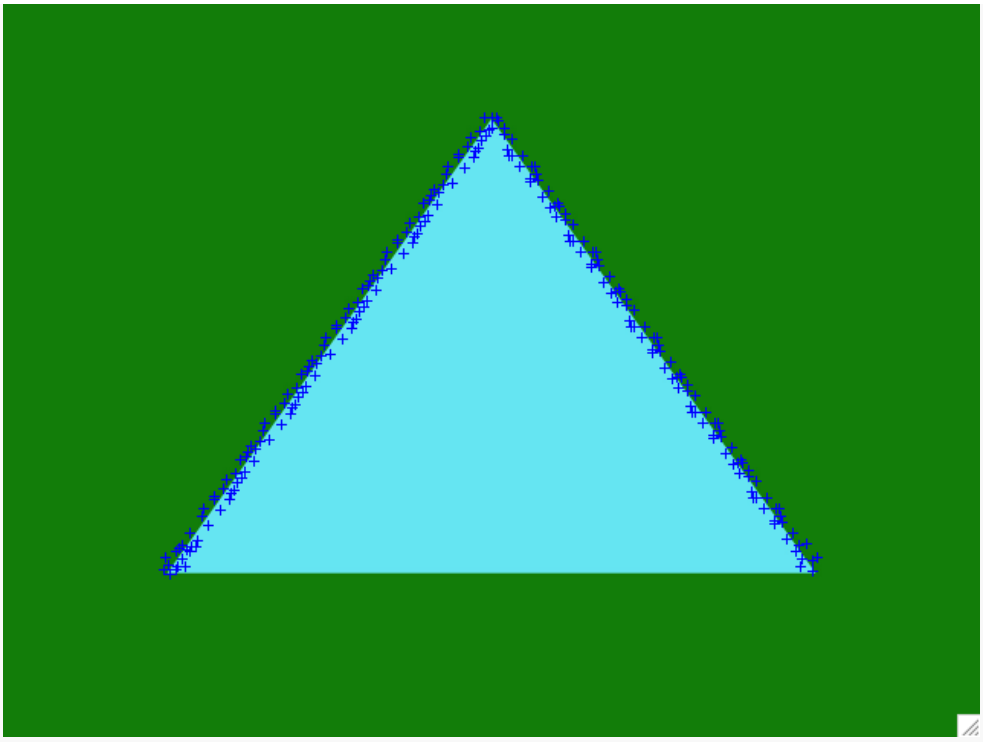


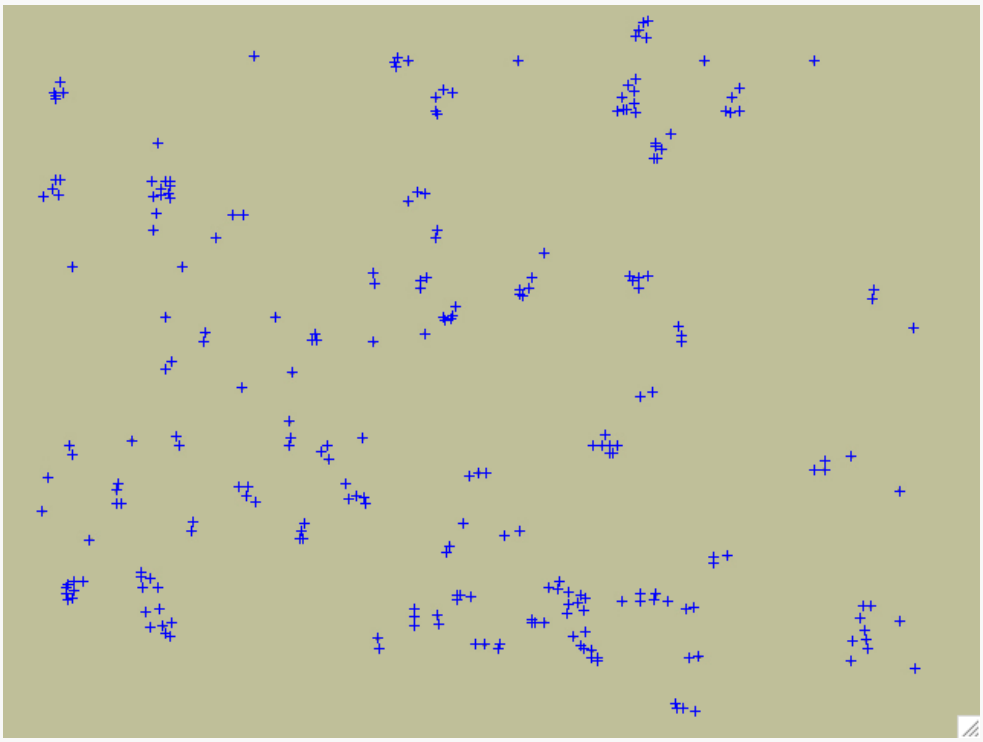


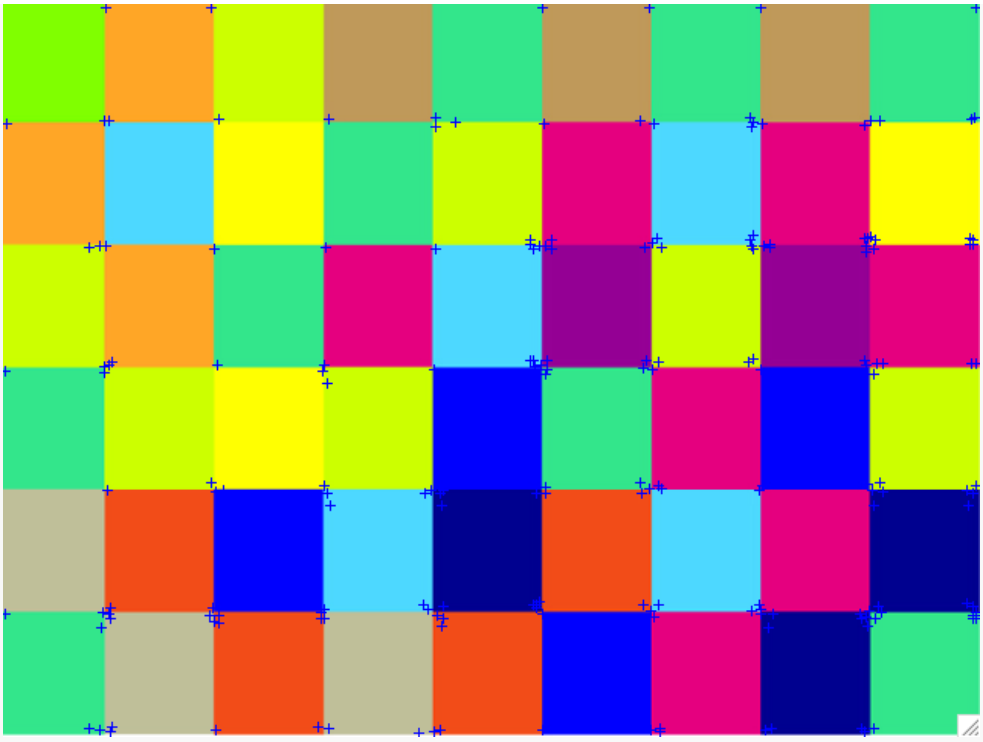












There are still too many features on those simple testing images but at least, they reveals much clear sense of how it works.

Feature descriptor

The SIFT descriptor is adopted cause it looks much stable and reliable.

Feature matching

I use L2 distance to measure the similarity of features. The following is the idea suggested by someone.

> For each feature p_A in image A, you will try to find its best match in image B. You can first find the closest match p_B in terms of L2 distances of their SIFT descriptors. If the distance is smaller than 0.4*(the distance to the second best match), then p_B is a correct match for p_A. Otherwise, we assume that there is no match for p_A in image B.

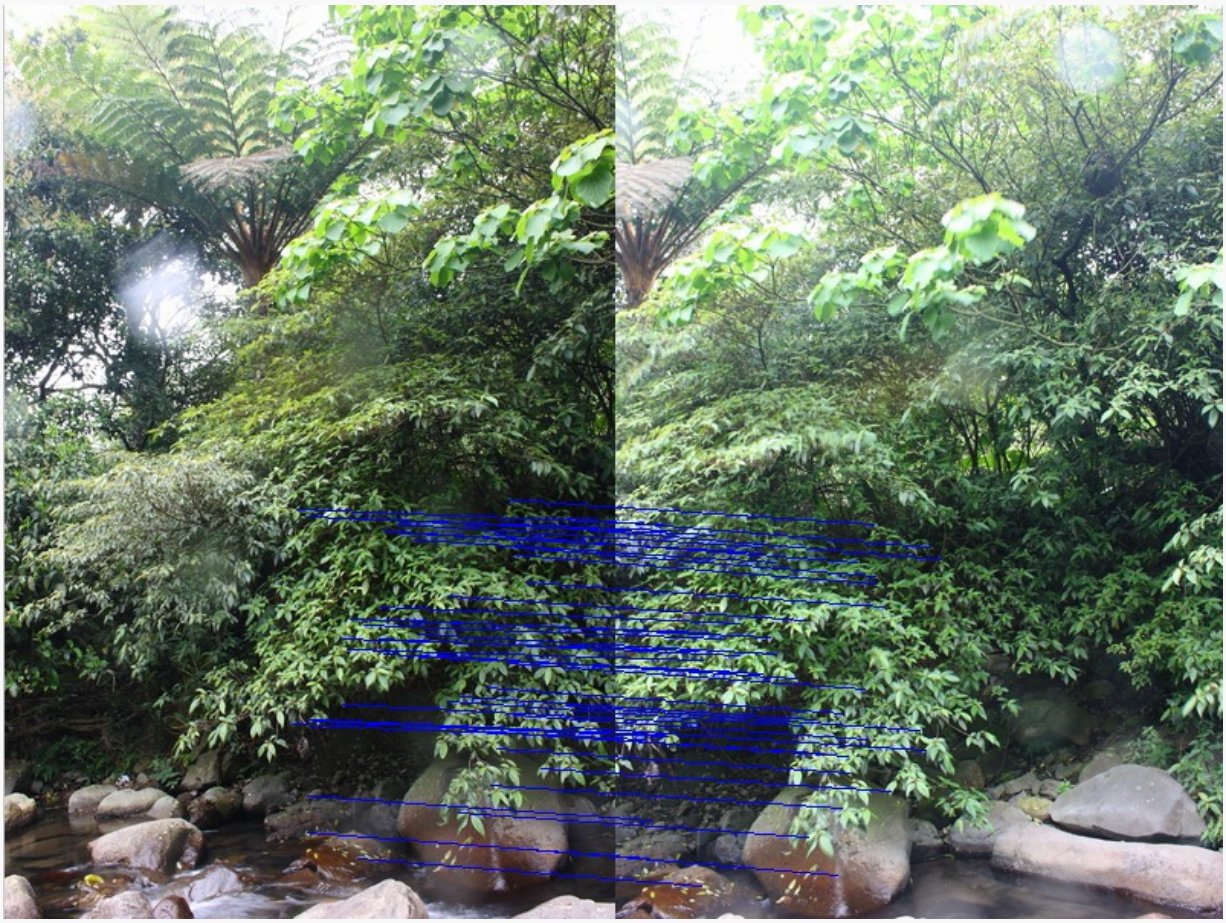


Image matching

The translation warping is used for image blending modeled by Ax = b, which is solved by matlab's *x = Ab.

Image blending

A linear alpha blending is used to blend 2 adjacent images.

Result

Discussion

There are several weakness in this implementation:

- It would be better if the sub-pixel accurate feature position is used.
- The solver to model the warping between adjacent images is really naive, just the 2D translation. A more general affien transformation model can be used and feed into Ax = b. But I think it would add some more efforts on image blending later on.
- The most need-to-enhance part is image blending!

Codes

- main.m: main entry.
- blendImage.m: image blending.
- descriptorDistance.m: L2-norm for distance between descriptors.
- descriptorSIFT.m: SIFT descriptor representation.
- display2MatchingImage.m: display matching points over the image.
- display_keypoints.m: display keypoints with orientation.
- displayMatchInTerminal.m: display matching directly on terminal for debugging.
- featureDetection.m: feature detection.
- featureHarris.m: Harris feature detector.
- featureMatching.m: feature matching.
- featureMoravec.m: Moravec feature detector.
- featureSIFT.m: SIFT feature detector, empty.
- filterGaussian.m: Gaussian filter.
- imageMatching.m: image matching.
- imRead.m: image reader.
- plotFeaturesOverImage.m: plot features as '+' on the image.
- ransac.m: RANSAC for feature descriptors.
- rejectBoundary.m: feature removal by boundary.
- rejectEdge.m: feature removal by edge.
- rejectLowContrast.m: feature removal by low contrast.
- solverTranslation.m: image matching with model of 2D translation.
- unwarpCylindrical.m: cylindrical unwarping.
- warpCylindrical.m: cylindrical warping.