Panorama Stitching

Problem Description

Get two images I1 and I2 of our campus and make sure that the major parts of I1 and I2 are from the same physical plane. Stitch I1 and I2 together to get a panorama view using LoG (or DoG) based interest point detector and SIFT descriptor. You can use OpenCV or VLFeat.

Development Environment

- **Operating System**: macOS Catalina 10.15.4
- **IDE**: MATLAB_R2019a
- **Programming Language**: matlab
- **Dependence**: vlfeat-0.9.21

How to Run

1. Install dependence vlfeat-0.9.21 to your root folder.
2. Put the folling matlab script into your working path.
   - `main.m`: sample driver for the project
   - `SetupVlfeat.m`: set up vlfeat for SIFT
   - `SIFT.m`: SIFT scale invariant feature transform
   - `MatchDescriptors.m`: matches the two sets of SIFT descriptors

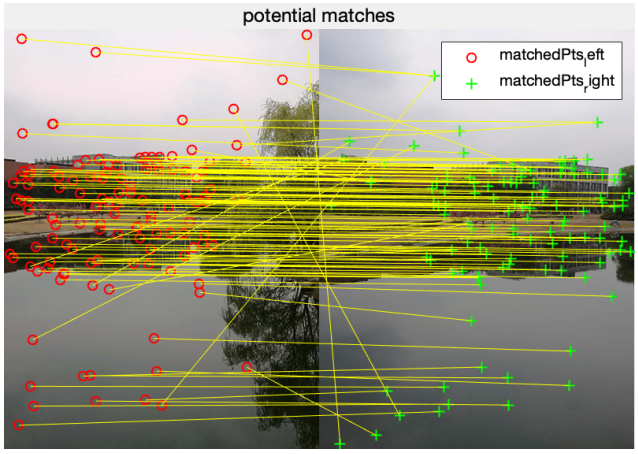
- `RANSAC_based_HomographyEstimation.m`: estimate homograpyh matrix based on RANSAC
- `CorrespondenceEstimation.m`: display correspondence estimation feature points
- `ImageStitching.m`: use homography matrix to stitching image

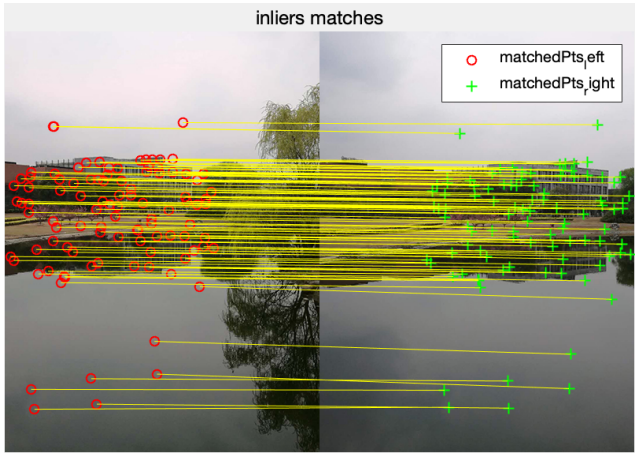
3. Put your testing image into the folder `img/`
4. Change the `file_name_l`, `file_name_r` in `main.m`
5. Run `main.m` and waiting for output image.
   - `image SIFT features`: feature points on image
   - `potential matches`: potential correspondences estimation feature points
   - `inlier matches`: inlier correspondences estimation feature points
   - `stitched image`: final result of stitched image

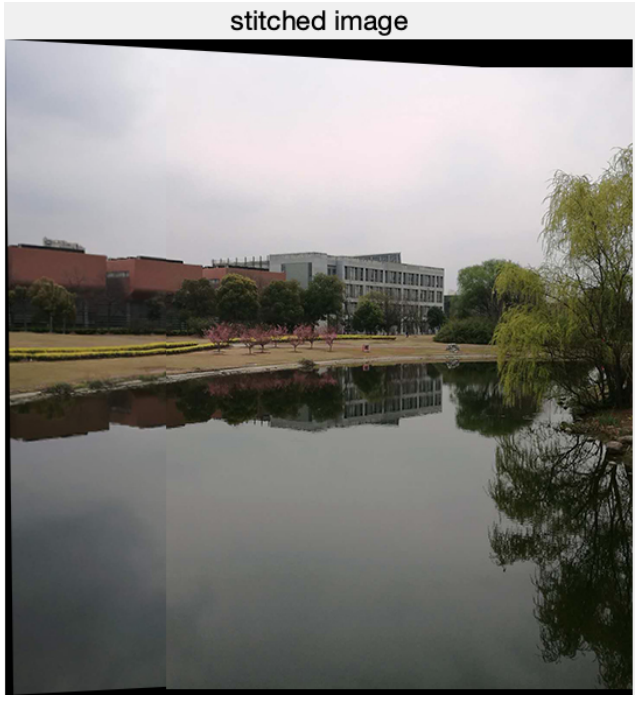
Algorithm

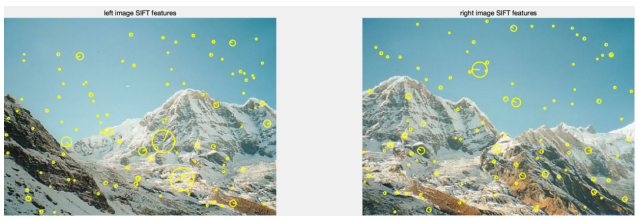
1. interest points detection using SIFT
   - `vl_sift()`
2. matching the two sets of SIFT descriptors
   - `vl_ubcmatch()`
3. calculate homography matrix estimation by using the correspondence pairs with RANSAC
   - initial
   - iterations
     - randomly select 4 points
     - instantiate the model
     - calculate the error
     - optimization
4. correspondence estimation
   - `showMatchedFeatures()`
5. stitch the transformed images

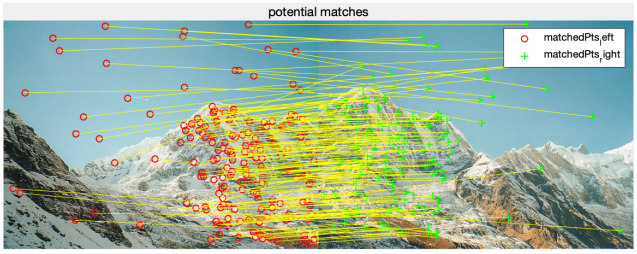
Realization

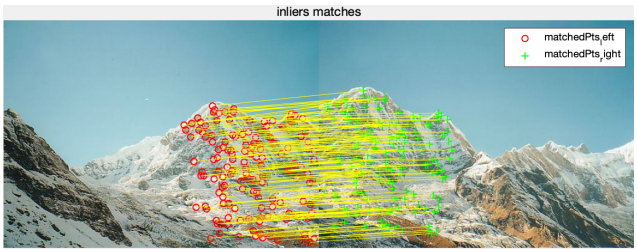

potential matches
matchedPts_left
matchedPts_right

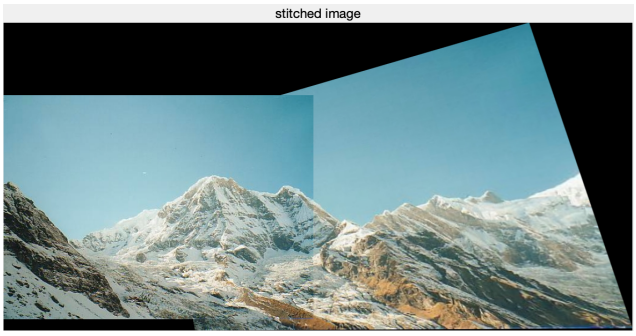

inliers matches
matchedPts_left
matchedPts_right


stitched image


left image SIFT features
right image SIFT features







SIFT

```
%% SIFT
[F, D] = vl_sift(im2single(rgb2gray(img)));     % compute the SIFT frames and
descriptors

%% pick random feature points
total_feature_num = 100;         % total number of picked feature points
rand_perm = randperm(size(F, 2));          % random permutation from 1~frames
number
rand_features_index = rand_perm(1:total_feature_num);    % take the first 100
feature points
rand_features = F(:, rand_features_index); % random features
```

Match Descriptors

```
[matches, distances] = vl_ubcmatch(D_l, D_r);  % match SIFT features(matches | squared Euclidean distance between the matches)

[~, distances_index] = sort(distances, 'descend');     % descend sort for distances
matches = matches(:, distances_index);      % sort matches according to the order of scores
distances = distances(distances_index);     % sort distances

match_l = F_l(1:2, matches(1,:));     % coordinates of the matches point in left image
match_r = F_r(1:2, matches(2,:));     % corrdinates of the matches point in right image
```

RANSAC based Homography Estimation

```
%% hyperparameters
N = 100;          % number of iterations
threshold = 10;   % error tolerance threshold

%% initial
match_num = size(match_l, 2);
inlier_num_max = 0;
inlier_num = 0;
H_best = zeros(3,3);

%% iterations
for i=1:N
    %% randomly select 4 points
    [~,sample_index] = datasample(match_l(1,:), 4);
    l = match_l(:, sample_index)';   % 4 random points from left image features to perform H matrix
    r = match_r(:, sample_index)';  % corresponding 4 matches from the right image

    %% instantiate the model
    l1 = l(1,:); l2 = l(2,:); l3 = l(3,:); l4 = l(4,:);
    r1 = r(1,:); r2 = r(2,:); r3 = r(3,:); r4 = r(4,:);
    % Direct Linear Transformation
    M = [
           r1(1) r1(2)  1    0     0     0  -r1(1)*l1(1) -r1(2)*l1(1) -l1(1);
            0     0     0   r1(1) r1(2)  1  -r1(1)*l1(2) -r1(2)*l1(2) -l1(2);
           r2(1) r2(2)  1    0     0     0  -r2(1)*l2(1) -r2(2)*l2(1) -l2(1);
            0     0     0   r2(1) r2(2)  1  -r2(1)*l2(2) -r2(2)*l2(2) -l2(2);
           r3(1) r3(2)  1     0     0     0  -r3(1)*l3(1) -r3(2)*l3(1) -l3(1);
            0     0     0   r3(1) r3(2)  1  -r3(1)*l3(2) -r3(2)*l3(2) -l3(2);
           r4(1) r4(2)  1     0     0     0  -r4(1)*l4(1) -r4(2)*l4(1) -l4(1);
            0     0     0   r4(1) r4(2)  1  -r4(1)*l4(2) -r4(2)*l4(2) -l4(2);
```

```
        ];
    [~,~,V] = svd(M);   % singular value decomposition
    smallest_eigenvector = V(:, end);      % smallest eigenvalue corresponding
to eigenvector

    H = [smallest_eigenvector(1:3,1)'; smallest_eigenvector(4:6,1)';
smallest_eigenvector(7:9,1)'];
    H = H / H(end);     % normization a33 to 1(8 dof)

    %% calculate the error
    points_ = H * [match_r; ones(1,match_num)];
    points_ = points_ ./ points_(end, :);   % bitwise normalize
    error = points_ - [match_l; ones(1,match_num)];
    error = sqrt(sum(error.^2, 1));     % eucledian distance

    %% optimization
    % keep the H matrix corresponding to maximun inliers
    inliers = error < threshold;
    inlier_num = size(find(inliers), 2);  % number of inliers feature points
    if inlier_num > inlier_num_max      % better choice of homography matrix
        inlier_num_max = inlier_num;
        inlier_pos = find(inliers);     % position of inliers feature points
        H_best = H;
    end
end
```

Correspondence Estimation

```
showMatchedFeatures(img_l, img_r, match_l(:,inlier_pos)',
match_r(:,inlier_pos)', 'montage');
```

Image Stitching

```
%% build TFORM struct for N-dimensional projective transformation.
tform = maketform('projective', H');
img_r_ = imtransform(img_r, tform);

%% display right image after homography
% figure;
% imshow(img_r_);

[M_l, N_l, ~] = size(img_l);
[M_r, N_r, ~] = size(img_r);

%% border position under homography
border_M = [
    1 N_r N_r  1;
```

```
    1   1   M_r M_r;
    1   1    1   1;
];
border_M_ = zeros(3,4);
border_M_ = H * border_M;

%% normize border position
x_ = border_M_(1,:) ./ border_M_(3,:);
y_ = border_M_(2,:) ./ border_M_(3,:);

%% determine up and left
up = round(min(y_));
y_offset = 0;
if up<=0
    y_offset = -up + 1;
    up = 1;
end

left = round(min(x_));
x_offset = 0;
if left<=0
    x_offset = -left + 1;
    left = 1;
end

%% display final stitchied image
[M_, N_, ~] = size(img_r_);
imgout(up:up+M_-1, left:left+N_-1,:) = img_r_;
imgout(y_offset+1:y_offset+M_l, x_offset+1:x_offset+N_l, :) = img_l;
```

About the Author

Item	Info
Name	Zhe Zhang(张喆)
ID	1754060
Adviser	Prof. Lin Zhang
Course Name	Digital Image Processing
Course Time	Mon. 2-4 [1-17]
Email	dbzdbz@tongji.edu.cn

Project Structure

```
.
├── README.md
├── doc
│   └── Report of Panorama Stitching.pdf
├── img
│   ├── jiading1.jpg
│   ├── jiading2.jpg
│   ├── mountain1.jpg
│   └── mountain2.jpg
├── src
│   ├── CorrespondenceEstimation.m
│   ├── ImageStitching.m
│   ├── MatchDescriptors.m
│   ├── RANSAC_based_HomographyEstimation.m
│   ├── SIFT.m
│   ├── SetupVlfeat.m
│   └── main.m
└── vlfeat-0.9.21 (dependance)

4 directories, 13 files
```