Report

Author: Zhengqi Yang
Date: 10/31/2019

Part 1: Point Feature Detection

In this project, the testing reference image (Overhead_1.jpg) and registering image (Overhead_2.jpg) are provided by Gauchospace, shown in Figure 1. They are 640 x 480 RGB images.

Figure 1: Reference image (left) and registering image (right).

The feature detection and localization were achieved by SIFT algorithm provided by VLFEAT open source library.[1][2] First of all, the RGB images were converted into single precision and then converted into grayscale. After this, vl_sift and vl_plotframe were used to compute and plot the keypoints and descriptors. The feature detected images are shown in Figure 2. The sample code is provided in Appendix A.

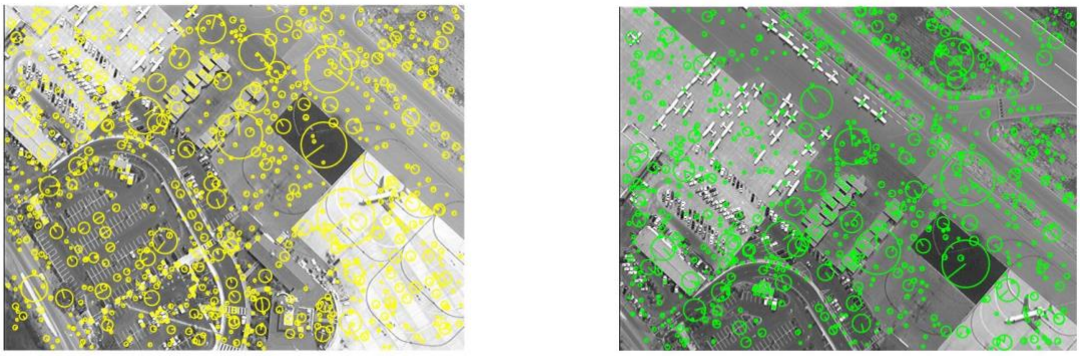

Figure 2: Feature detected images, reference one (left), registering one (right).

Part 2: Establishing Correspondences

In this part, the vl_ubcmatch function was used to extract correspondences and calculate Euclidean distances of the corresponding features.[2][3] Corresponding features were saved to f1match and f2match matrices. The feature descriptors that are within matches were chosen. In the reference image, 1065 features were detected, and 915 features were detected in the registering image. By applying

match generated by vl_ubcmatch function, 361 putative correspondences were established in the end, shown in Figure 3. The code is in Appendix B.

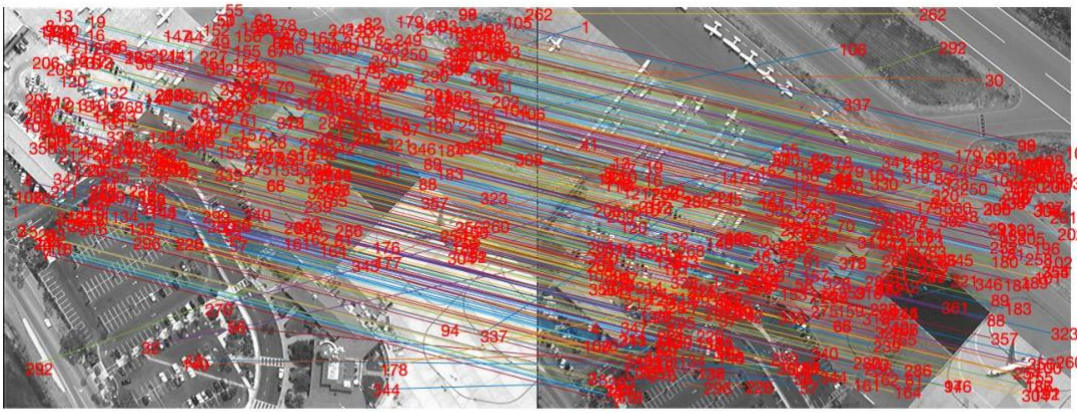

Figure 3: Established Correspondences.

Part 3 and 4: Estimate the Homography and RANSAC

In this part, there were three algorithms being experimented. The first one was to estimate the homography via Direct Linear Transform (DLT) algorithm.[4] The DLT algorithm was achieved by implementing vgg_H_from_x_lin function.[4] In this case, the generated H is:

-1.58554635989681 -0.116233349480623 33.5519822069193
-10.1294645673464 -0.631864015480568 201.928522319643
-0.0467688754614675 -0.00345351091478038 1

The second experiment was to estimate the homography via normalized DLT algorithm.[5] The normalized DLT algorithm was achieved by implementing normalise2dpts function.[5] Then the obtained H is:

-0.0358566407944370 -0.209996001489194 -0.371370038306976
-14.6249632557706 -1.45238626635029 5.42510907166490
-9.09480064699582 1.67692854382944 1

The last experiment was to estimate the homography via DLT and RANSAC pruning.[2] This was achieved by implement the RANSAC estimate homography model in sift_mosaic function. [2] In this case, H is:

0.00527112548456007 0.000373390641608535 0.157470850504073
-0.000324460583355497 0.00543798964408685 0.987479117644620
-4.52335662392518e-07 -1.67308059586647e-08 0.00550656577524863

The core part of algorithm for these three approaches is in Appendix C.

Part 5: Image Warping

In this part, the image warping approach was modified from VLFEAT.[2] The algorithm is in Appendix D. The result images via these three approaches are shown in Figure 4.

Figure 4: Result images

Reference

[1] D. Lowe, *Distinctive Image Features from Scale-Invariant Keypoints*, International Journal of Computer Vision, 2004.
[2] VLFEAT Library: see http://www.vlfeat.org/
[3] W. Hoff, *Lecture 12 SIFT Samples*, Colorado School of Mines, 2016
[4] MATLAB Functions for Multiple View Geometry Library: see http://www.robots.ox.ac.uk/~vgg/hzbook/code/ by David Capel, Andrew Fitzgibbon, Peter Kovesi, Tomas Werner, Yoni Wexler, and Andrew Zisserman.
[5] Peter Kovesi Matlab Image Processing functions: http://www.peterkovesi.com/matlabfns/ index.html\

Appendix A

```
%% ---------------------------------------------------------------------
% Load reference image
% ---------------------------------------------------------------------

im1 = imread('overhead_1.jpg') ;
% Make it single precision
im1 = im2single(im1) ;
% Make it grayscale
if size(im1,3) > 1, im1g = rgb2gray(im1); else im1g = im1 ; end
figure(1), imshow(im1g);

% ---------------------------------------------------------------------
% Compute the SIFT frames (keypoints) and descriptors for the Ref Im
% ---------------------------------------------------------------------

[f1,d1] = vl_sift(im1g) ;
fprintf('Number of frames (features) detected: %d\n', size(f1,2));
h = vl_plotframe(f1);
set(h,'color','y','linewidth',1);
```

Appendix B

```
%% ---------------------------------------------------------------------
% Extract and Match the descriptors
% ---------------------------------------------------------------------

[matches, scores] = vl_ubcmatch(d1,d2) ;
fprintf('Number of matching frames (features): %d\n', size(matches,2));
indices1 = matches(1,:); % Get matching features
f1match = f1(:,indices1);
d1match = d1(:,indices1);
indices2 = matches(2,:);
f2match = f2(:,indices2);
d2match = d2(:,indices2);

figure(3), imshow([im1g,im2g]);
o = size(im1g,2) ;
line([f1match(1,:);f2match(1,:)+o],[f1match(2,:);f2match(2,:)]) ;
for i=1:size(f1match,2)
 x = f1match(1,i);
```

```
 y = f1match(2,i);
 text(x,y,sprintf('%d',i), 'Color', 'r');
end

for i=1:size(f2match,2)
 x = f2match(1,i);
 y = f2match(2,i);
 text(x+o,y,sprintf('%d',i), 'Color', 'r');
end
```

Appendix C

DLT:

```
function H = vgg_H_from_x_lin(xs1,xs2)
% H = vgg_H_from_x_lin(xs1,xs2)
%
% Compute H using linear method (see Hartley & Zisserman Alg 3.2 page 92 in
%                                1st edition, Alg 4.2 page 109 in 2nd edition).
% Point preconditioning is inside the function.
%
% The format of the xs [p1 p2 p3 ... pn], where each p is a 2 or 3
% element column vector.

[r,c] = size(xs1);

if (size(xs1) ~= size(xs2))
 error ('Input point sets are different sizes!')
end

if (size(xs1,1) == 2)
  xs1 = [xs1 ; ones(1,size(xs1,2))];
  xs2 = [xs2 ; ones(1,size(xs2,2))];
end

% condition points
C1 = vgg_conditioner_from_pts(xs1);
C2 = vgg_conditioner_from_pts(xs2);
xs1 = vgg_condition_2d(xs1,C1);
xs2 = vgg_condition_2d(xs2,C2);

D = [];
ooo  = zeros(1,3);
for k=1:c
  p1 = xs1(:,k);
  p2 = xs2(:,k);
  D = [ D;
    p1'*p2(3) ooo -p1'*p2(1)
    ooo p1'*p2(3) -p1'*p2(2)
   ];
end

% Extract nullspace
[u,s,v] = svd(D, 0); s = diag(s);
```

```
nullspace_dimension = sum(s < eps * s(1) * 1e3);
if nullspace_dimension > 1
  fprintf('Nullspace is a bit roomy...');
end

h = v(:,9);

H = reshape(h,3,3)';
```

Normalized DLT:

```
function [newpts, T] = normalise2dpts(pts)

    if size(pts,1) ~= 3
        error('pts must be 3xN');
    end

    % Find the indices of the points that are not at infinity
    finiteind = find(abs(pts(3,:)) > eps);

%    if length(finiteind) ~= size(pts,2)
%        warning('Some points are at infinity');
%    end

    % For the finite points ensure homogeneous coords have scale of 1
    pts(1,finiteind) = pts(1,finiteind)./pts(3,finiteind);
    pts(2,finiteind) = pts(2,finiteind)./pts(3,finiteind);
    pts(3,finiteind) = 1;

    c = mean(pts(1:2,finiteind)')';            % Centroid of finite points
    newp(1,finiteind) = pts(1,finiteind)-c(1); % Shift origin to centroid.
    newp(2,finiteind) = pts(2,finiteind)-c(2);

    dist = sqrt(newp(1,finiteind).^2 + newp(2,finiteind).^2);
    meandist = mean(dist(:));  % Ensure dist is a column vector for Octave
3.0.1

    scale = sqrt(2)/meandist;

    T = [scale   0   -scale*c(1)
         0     scale -scale*c(2)
         0       0      1      ];

    newpts = T*pts;
```

DLT and RANSAC:

```
numMatches = size(matches,2) ;

X1 = f1(1:2,matches(1,:)) ; X1(3,:) = 1 ;
X2 = f2(1:2,matches(2,:)) ; X2(3,:) = 1 ;
clear H_ransac score ok ;
```

```
for t = 1:100
  % estimate homograpyh
  subset = vl_colsubset(1:numMatches, 4) ;
  A = [] ;
  for i = subset
    A = cat(1, A, kron(X1(:,i)', vl_hat(X2(:,i)))) ;
  end
  [U_r,S_r,V_r] = svd(A) ;
  H_ransac{t} = reshape(V_r(:,9),3,3) ;

  % score homography
  X2_ = H_ransac{t} * X1 ;
  du = X2_(1,:)./X2_(3,:) - X2(1,:)./X2(3,:) ;
  dv = X2_(2,:)./X2_(3,:) - X2(2,:)./X2(3,:) ;
  ok{t} = (du.*du + dv.*dv) < 6*6 ;
  score(t) = sum(ok{t}) ;
end

[score, best] = max(score) ;
H_ransac = H_ransac{best} ;
ok = ok{best} ;
```

Appendix D

```
box2 = [1  size(im2,2) size(im2,2)  1 ;
        1  1           size(im2,1)  size(im2,1) ;
        1  1           1            1 ] ;
box2_ = inv(H) * box2 ;
box2_(1,:) = box2_(1,:) ./ box2_(3,:) ;
box2_(2,:) = box2_(2,:) ./ box2_(3,:) ;
ur = min([1 box2_(1,:)]):max([size(im1,2) box2_(1,:)]) ;
vr = min([1 box2_(2,:)]):max([size(im1,1) box2_(2,:)]) ;

[u,v] = meshgrid(ur,vr) ;
im1_ = vl_imwbackward(im2double(im1),u,v) ;

z_ = H(3,1) * u + H(3,2) * v + H(3,3) ;
u_ = (H(1,1) * u + H(1,2) * v + H(1,3)) ./ z_ ;
v_ = (H(2,1) * u + H(2,2) * v + H(2,3)) ./ z_ ;
im2_ = vl_imwbackward(im2double(im2),u_,v_) ;

mass = ~isnan(im1_) + ~isnan(im2_) ;
im1_(isnan(im1_)) = 0 ;
im2_(isnan(im2_)) = 0 ;
mosaic_DLT = (im1_ + im2_) ./ mass ;

figure(4) ; clf ;
imagesc(mosaic_DLT) ; axis image off ;
title('Mosaic DLT') ;
```