Report for image stitcher

Author: Li Zehun 李泽寰 119370910018

In this practicing, I will implement an image stitcher that uses image warping and homographies to automatically create an image mosaic. We will focus on the case where we have two input images that should form the mosaic, where we warp one image into the plane of the second image and display the combined views. This problem will give some practice manipulating homogeneous coordinates, computing homography matrices, and performing image warps.

1. **Getting correspondences [10 points]**: First write code to get manually identified corresponding points from two views.
   By using the built-in function cpselect, I could select the two corresponding points;

```
clear;
image1 = imread("uttower1.jpg");
image2 = imread("uttower2.jpg");
[mp,fp] = cpselect(image2,image1,'Wait',true);
mp = mp';
fp = fp';
```

   mp and fp is the selected corresponding points matrix.
   mp contains the point coordinates in the image2, and is a 2*N matrix.
   fp contains the point coordinates in the image1, and is a 2*N matrix.

2. **Computing the homography parameters**: we need to calculate the homography matrix from the above corresponding points;

```
function v = Homo_solve(pin,pout)
% return the homography matrix via homogeneous solution 2xN 的矩阵

if ~isequal(size(pin), size(pout))
    error('Points matrices different sizes');
end
if size(pin, 1) ~= 2
    error('Points matrices must have two rows');
end
n = size(pin, 2);
if n < 4
    error('Need at least 4 matching points');
end
% solve equation using svd
x = pout(1, :); y = pout(2,:); X = pin(1,:); Y = pin(2,:);
```

```
rows0 = zeros(3, n);
rowsXY = -[X; Y; ones(1,n)];
hx = [rowsXY; rows0; x.*X; x.*Y; x];
hy = [rows0; rowsXY; y.*X; y.*Y; y];
h = [hx hy];
%svd,奇异值分解
if n == 4
    [U, ~, ~] = svd(h);
else
    [U, ~, ~] = svd(h, 'econ');
end
v = (reshape(U(:,9), 3, 3)).';
end
```

I wrote the Homo_solve function to obtain the homography parameters. The function required the pin (2*N matrix, corresponding points in the image2, since image2 is transformed to image1 reference).
The main idea of Homo_solve function is svd decomposition. The function requires at least 4 pairs corresponding points to calculate homography matrix.
Write the H matrix in vector form as $h = (h_{11}, h_{12}, h_{13}, h_{21}, h_{22}, h_{23}, h_{31}, h_{32}, h_{33})^T$, then the homogeneous equations for n points become $Ah = 0$, with A the 2N*9 matrix;

$$A = \begin{pmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x_1X_1 & -y_1X_1 & -X_1 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -x_1Y_1 & -y_1Y_1 & -Y_1 \\ x_2 & y_2 & 1 & 0 & 0 & 0 & -x_2X_2 & -y_2X_2 & -X_2 \\ 0 & 0 & 0 & x_2 & y_2 & 1 & -x_2Y_2 & -y_2Y_2 & -Y_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_n & y_n & 1 & 0 & 0 & 0 & -x_nX_n & -y_nX_n & -X_n \\ 0 & 0 & 0 & x_n & y_n & 1 & -x_nY_n & -y_nY_n & -Y_n \end{pmatrix}$$

It is a standard result of linear algebra that the vector h that minimizes the algebraic residuals $|Ah|$, subject to $|h| = 1$, is given by the eigenvector of least eigenvalue of A^TA. The eigenvector can be obtained directly from the SVD of A.

The following function requires the input points coordinate and homography matrix, it returns the corresponding points by multiplying the coordinate and homography matrix.

```
function y = Homo_tran(x,v)
% a series of points
q = v * [x; ones(1, size(x,2))];
p = q(3,:);
y = [q(1,:)./p; q(2,:)./p];
end
```

3. **Warping between image planes and Create the output mosaic:** The Warp_Image function requires the two RGB image (read by command “imread”), and homography matrix. It returns the output mosaic and index of reference image (1,1) corresponding position.

```
function [res,index]= Warp_Image(img1,img2,H_Matrix)
%%%输入一张图 img2，和把它变换成另一张图的单应性矩阵 H_Matrix,得到变换之后的图
%%%然后把 img1 和 img2 合成一下
%%%使用的是 reverse warp
%%%index 返回的是作为 frame/ reference 的那张图原来（1,1）在变换后的大图中的位置
index [行;列];

[row2, col2] = size(img2(:,:,1));
[row1, col1] = size(img1(:,:,1));
%%%边角的位置
corner_P = [1,col2,1,col2;1,1,row2,row2];
change_P = Homo_tran(corner_P,H_Matrix);
change_P = round(change_P);

x_min = min([min(change_P(1,:)),1]);
y_min = min([min(change_P(2,:)),1]);
x_max = max([max(change_P(1,:)),col1]);
y_max = max([max(change_P(2,:)),row1]);

[X,Y] = meshgrid(x_min:x_max,y_min:y_max);
[row3,col3] = size(X);

%%%%使用 reshape 一次计算出来;
X = reshape(X,1,[]);
Y = reshape(Y,1,[]);
P = [X;Y];
P_tran = Homo_tran(P,inv(H_Matrix));
tran_X = reshape(P_tran(1,:),row3,col3);
tran_Y = reshape(P_tran(2,:),row3,col3);
index = [2-y_min;2-x_min];
res = zeros(row3,col3,3);
for color = 1:3
    output = interp2(double(img2(:,:,color)),tran_X,tran_Y);
    output = round(output);
    nan_pos = isnan(output);
    output(nan_pos) = 0;%%%img2 变换之后的图像
    output2 = zeros(row3,col3);%%img1 在该图中的位置;
    output2((2-y_min):(1-y_min+row1),(2-x_min):(1-
x_min+col1))=double(img1(:,:,color));
    pos = (output2~=0);
    output(pos)=0;
```

```
    res(:,:,color) = imadd(output,output2);
end
res = uint8(res);
end
```

The main idea of Warp_Image function is inverse warp. The function first calculates the four corner points of the image need being transformed (image2), then compares the four transformed corresponding points with the referenced image four corner points. Then the function extended the output image size to those corner points' max and min values. Then we use "meshgrid" command to form a coordinate tabula. By invoking the Homo_tran function and inputting the tabula and inverse matrix of homography matrix, the function could obtain the corresponding position in the image2, Then, the "interp2" command could interpolate the value in the output image from image2, since lots of position is sub-pixel. Afterwards, the function merge the two images – transformed image2 and referenced original image1. The function uses the image1 part to put on the overlay part.

4. **The main function:**

```
clear;
image1 = imread("uttower2.jpg");
image2 = imread("uttower1.jpg");
[mp,fp] = cpselect(image2,image1,'Wait',true);
mp = mp';
fp = fp';

H_Matrix = Homo_solve(mp,fp);
[output,index] = Warp_Image(image1,image2,H_Matrix);
imshow(output);
hold on
plot(fp(1,:)+index(1)-1,fp(2,:)+index(2)-1,'r+','LineWidth',2);
hold on;
img2_tran = Homo_tran(mp,H_Matrix);
plot(img2_tran(1,:)+index(1)-1,img2_tran(2,:)+index(2)-
1,'bo','LineWidth',2);
```

The image1 is referenced image and image2 need being transformed. And I use the red add sign-"+" as the original selected points in the image1, and the blue sign-"o" as transformed selected points in the image2.

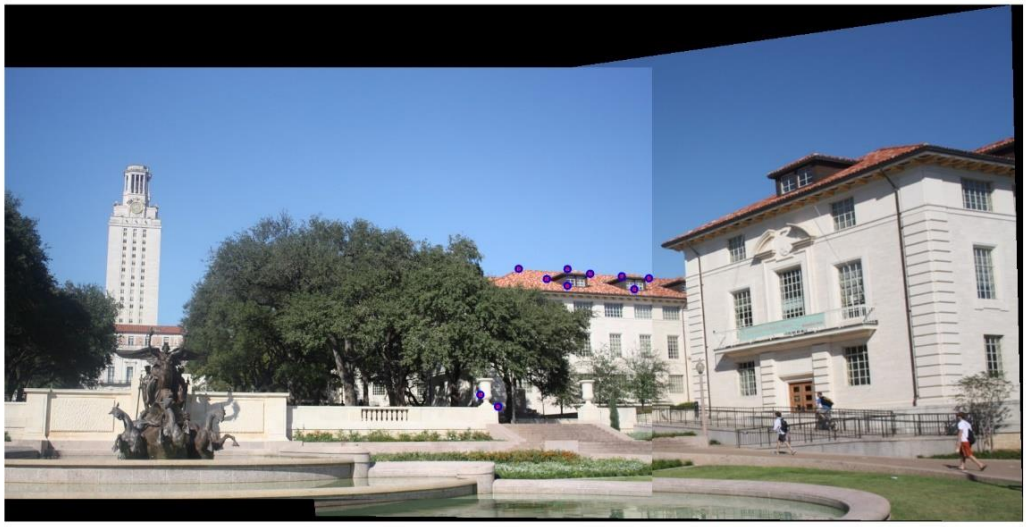
5. **Test:**

Part(1):

Reference image with selected points: (image1) :uttover2.jpg

Transformed image with selected points: (image2) :uttover1.jpg

Result image:

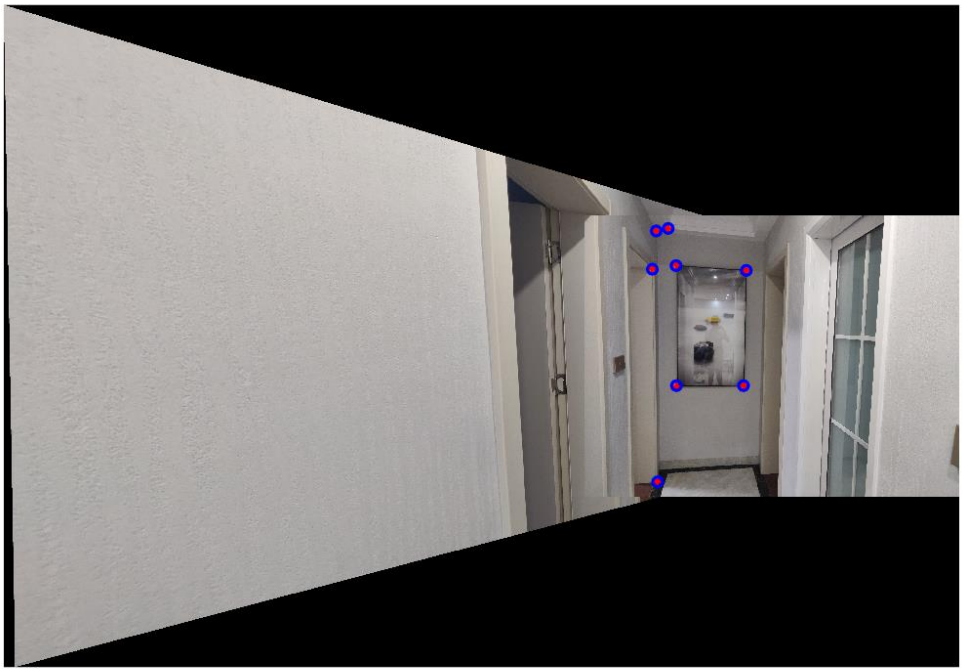


Part(2): additional example

Reference image: (image1) :

Transformed image:

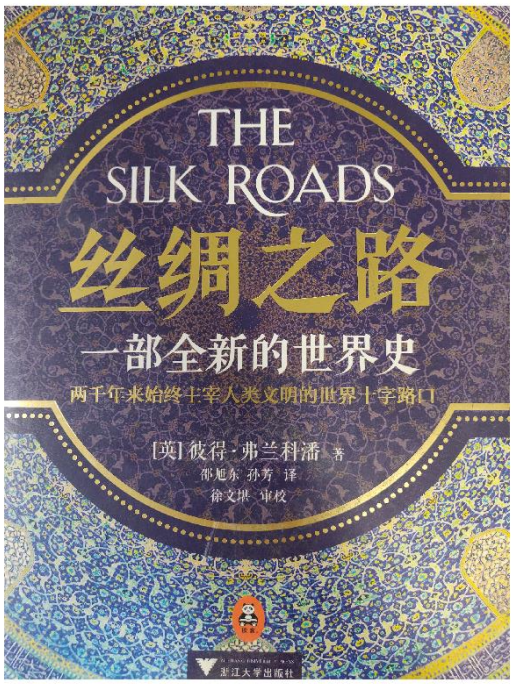
Result image:



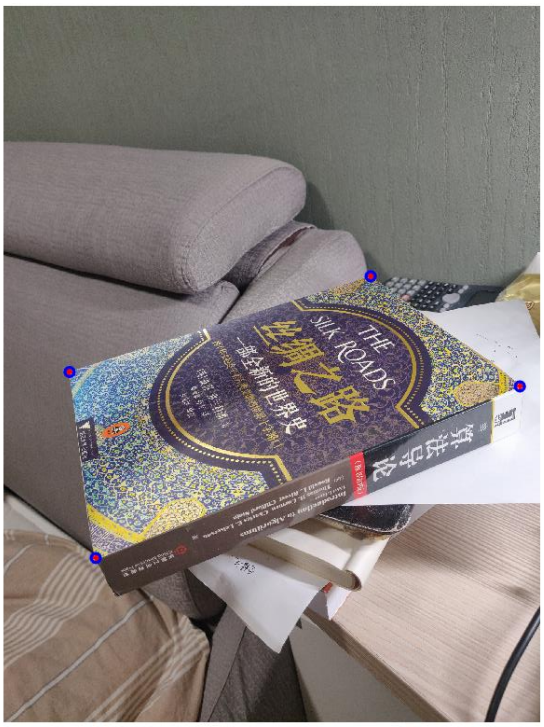
6. **Stitch image**

Referenced image with selected points:

Transformed image:



Result image;




```
clear;
image1 = imread("referenced_image.jpg");%%reference image
image2 = imread("transformed_image.jpg");%%need being tranformed image

[mp,fp] = cpselect(image2,image1,'Wait',true);
mp = mp';
fp = fp';
H_Matrix = Homo_solve(mp,fp);

[output,index] = Warp_Image(image1,image2,H_Matrix);
imshow(output,[]);
hold on
plot(fp(1,:)+index(1)-1,fp(2,:)+index(2)-1,'r+','LineWidth',2);
hold on;
img2_tran = Homo_tran(mp,H_Matrix);
plot(img2_tran(1,:)+index(1)-1,img2_tran(2,:)+index(2)-
1,'bo','LineWidth',2);
```

We use almost the same code with the previous main function code. But when we need to select the four boundary corner points to “mp” of transformed image, and select the corresponding four point “fp”.

Also, we need to change the Warp_Image function, since we use the referenced image part for overlay part. Now, we need to use the transformed part as overlay part. It is pretty easy. The code in the loop body in the Warp_Image should be changed to :

```
for color = 1:3
    output = interp2(double(img2(:,:,color)),tran_X,tran_Y);
    output = round(output);
    nan_pos = isnan(output);
    output(nan_pos) = 0;%%%img2 变换之后的图像
    output2 = zeros(row3,col3);%%img1 在该图中的位置;
    output2((2-y_min):(1-y_min+row1),(2-x_min):(1-
x_min+col1))=double(img1(:,:,color));
%     pos = (output2~=0);
%     output(pos)=0;
    pos = (output~=0);
    output2(pos)=0;
    res(:,:,color) = imadd(output,output2);
end
```

The dark green part should be annotated and change to the red color code.