Mayank Gupta
SR Number: 17112
M.Tech AI

Computer Vision

Homework 2 Report

Introduction:

We are given multiple images taken from a camera by rotating it with its camera center fixed. We have to combine all these images to form a mosaic using homography.

Key Steps:

1) To find the homography between 2 images we need corresponding points(which are images of the same 3-D point) in the images. For finding the corresponding points I am using SURF features detection and matching in MATLAB.
2) If the camera center is same and camera is just rotated then we can relate the location of point P in image1 which P1 and in image2 which is P2 by

   $$P2=H*P1$$

   where H is a 3*3 homography matrix

   Ignoring the scaling, we have 8 degrees of freedom for H, so we need 8 equations. And each matched point gives us 2 equations, so we need at least 4 equations to solve for H.
3) Let $P1=[x1,y1,1]^T$ and $P2=[x1',y1',1]^T$, then solving P2=H*P1, we get

   $a^T*h = 0$ and $b^T*h=0$

   where a = [-x1, -y1, -1, 0, 0, 0, x1*x1', y1*x1',x1']

   and b = [0, 0, 0, -x1, -y1, -1, x1*y1', y1*y1',y1']

   And h is flattened H matrix

   So if we have 4 corresponding points in both images then we have 8 such equations and we can stack these equations in matrix A which is 8*9 matrix.

   $$A*h=0$$

   We can solve for h using SVD as h should be the right singular vector of A.
4) Points that we chose for solving for h may have some error in location due to various factors, so we use RANSAC to make our estimate of H more robust. We randomly choose 4 matched points from all matched points and get A and compute H from A using SVD, then we see how many other matched points agree with that homography matrix. We need to set some threshold(for considering points as inliers) and parameters for RANSAC to do this step.
5) Then we take the homography which gives the maximum inliers. This homography matrix can be used as H but we can improve our estimate of H by taking all the inliers and using those to calculate H. I am using all inliers to calculate H as they give better results.
6) Matched points in the images may occupy small region because of very less overlap in images or some other reasons, so we apply some transformations

before calculating the H. So I shift the origin to the centroid of the matched points and also scale the mean distance from origin to a point to sqrt(2). We can use the inverse of this transformation matrices and H(calculated after transformation of points) to calculate the homography matrix for original points in the image.

7) After obtaining homography, we need to join the images together. For this we need to fix one image as reference frame and transform other images to the reference frame of that image using H matrix. For example, to bring the points in second image to first image use:

$$P=H_{12}^{-1}*P2$$

Problem is that they may not be an integer value, so we have to interpolate these values. I am not using interpolation(i.e. I am just round P to integers and using values there) because it made the code slow and did not affect the result much. To use interpolation just uncomment the corresponding lines in the code.

8) After having all the images in the same frame of reference, we can use blending to compute the intensities in the overlapping regions. I am using alpha blending for this.

Some Results :

Homography matrices:

H_{12}= 0.5813 -0.0300 -258.7310
0.0543 0.5742 -131.5090
0.0000 0.0000 0.5390

H_{23}= 0.6010 -0.0127 -360.4556
0.0564 0.5834 -161.4918
0.0000 0.0000 0.5282

H_{34}= 0.6091 -0.0392 -389.6853
0.0977 0.5716 -220.9301
0.0000 0.0000 0.5085

H_{45}= -0.6176 0.0287 351.6324
-0.0823 -0.5808 157.6396
-0.0000 0.0000 -0.5282

I have 2 codes for mosaic. One uses feathering/alpha blending and other uses averaging over overlapping regions. Results for both are presented below:

Fig) Mosaic with 3rd image set as reference image.

Fig) Mosaic obtained using first 3 images with alpha blending and 1st image set as reference image.

For mosaic with blending, I have to pass image sequentially and therefore mosaic with blending gets very skewed if all 5 images passed sequentially. Mosaic with blending only works good for 3 images after that mosaic gets very skewed.

Conclusions:

We can clearly see that blending improves the result. If we just average the values in the overlapping region, we get blocky effects. If we choose the first image as reference image, then the resulting mosaic is very stretched and does not look good, choosing 3rd image as reference improves the result. Result also depends a lot on the hyper-parameters(e.g. Threshold for inliers) that we choose.

Code:

Coding is done in MATLAB for this assignment.

Calculating N for RANSAC:

```
function N=ransac(p,w,s)
        N=log(1-p)/log(1-w^s);
```

Transforming points before calculating Homography:

```
function [t, points]=transform(points1)
centroid=mean(points1,1);
points=points1-centroid;
t11=eye(3);
t11(1,3)=-centroid(1,1);
t11(2,3)=-centroid(1,2);
c= mean(sqrt(points(:,1).^2+points(:,2).^2));
points=sqrt(2)*points/c;
t2=diag([sqrt(2)/c sqrt(2)/c 1]);
t=t2*t11;
```

Calculating Homography from 4 points:

```
function H=homography(points1,points2)
%returns homography with last row normalized
A=zeros(2*size(points1,1),9);
for i=1:size(points1,1)
        A(2*i-1,1)=-points1(i,1);
        A(2*i-1,2)=-points1(i,2);
        A(2*i-1,3)=-1;
        A(2*i-1,7)=points2(i,1)*points1(i,1);
        A(2*i-1,8)=points2(i,1)*points1(i,2);
        A(2*i-1,9)=points2(i,1);
        A(2*i,4)=-points1(i,1);
        A(2*i,5)=-points1(i,2);
        A(2*i,6)=-1;
        A(2*i,7)=points2(i,2)*points1(i,1);
        A(2*i,8)=points2(i,2)*points1(i,2);
        A(2*i,9)=points2(i,2);
end
[~,~,v]=svd(A);

H=reshape(v(:,end),3,3)';
```

Calculating best Homography matrix:

```
function H=get_best(points1,points2,N,threshold)
best=0;
[t1,points1]=transform(points1);
[t2,points2]=transform(points2);
H=zeros(3,3);
        for i=1:N
        r=randperm(size(points1,1),4);
```

```
        current_p1=points1(r,:);
        current_p2=points2(r,:);
        h_temp=homography(current_p1,current_p2);
        in_number=inliers(points1,points2,h_temp,threshold);
        if(in_number>best)
        best=in_number
        H=h_temp;
        end
        end


        points3=[points1,ones(size(points1,1),1)];
        points=points3*(H');
        points3=points(:,1:2)./points(:,3);
        n=sqrt(sum((points3-points2).^2,2))<threshold;
        H=homography(points1(n,:),points2(n,:));
        H=pinv(t2)*H*(t1);
end
```

Combining Images with blending(uncomment corresponding commented lines for interpolation):

```
function img=final_combine(img1,img2,H)
c1=ones(3,1);
c2=[1;size(img2,1);1];
c3=[size(img2,2);1;1];
c4=[size(img2,2);size(img2,1);1];

p1=pinv(H)*c1;
p1=p1/p1(3,1);
p2=pinv(H)*c2;
p2=p2/p2(3,1);
p3=pinv(H)*c3;
p3=p3/p3(3,1);
p4=pinv(H)*c4;
p4=p4/p4(3,1);

min_x=floor(min([1,p1(1,1),p2(1,1),p3(1,1),p4(1,1)]));
min_y=floor(min([1,p1(2,1),p2(2,1),p3(2,1),p4(2,1)]));
max_x=ceil(max([size(img1,1),p1(1,1),p2(1,1),p3(1,1),p4(1,1)]));
max_y=ceil(max([size(img1,2),p1(2,1),p2(2,1),p3(2,1),p4(2,1)]));

%%%%%%%
min_x_img2=floor(min([p1(1,1),p2(1,1),p3(1,1),p4(1,1)]));


n1=ones(3,1);
```

```
n2=[1;size(img1,1);1];
n3=[size(img1,2);1;1];
n4=[size(img1,2);size(img1,1);1];

temp=[p1,p2,p3,p4];
[min_x_img2,i]=min([p1(1,1),p2(1,1),p3(1,1),p4(1,1)]);
min_x_img2=floor(min_x_img2);
if ~(temp(i,1,1)>0 &&temp(i,1,1)<=size(img1,1))&&(temp(i,2,1)>0
&&temp(i,2,1)<size(img1,2))
        out1=intersection(n1,n3,p1,p2);
        out2=intersection(n2,n4,p1,p3);
        if ((out(1,1)>-1) &&out(1,1)<=(size(img1,1)+1)&&(out(2,1)>-1)
&&out(2,1)<(size(img1,2)+1))
        min_x_img2=out1(1,1);
        else
        min_x_img2=out2(1,1);
        end
end

min_y_img2=floor(min([p1(2,1),p2(2,1),p3(2,1),p4(2,1)]));
img=zeros(max_y-min_y+1,max_x-min_x+1,size(img1,3));

v_q=zeros(1,size(img1,3));

for i=min_x:max_x
        for j=min_y:max_y
        p=[i;j;1];
        p=H*p;
        p=p/p(3,1);
        if ((p(1,1)>1 && p(1,1)<size(img2,2)) &&(p(2,1)>1 && p(2,1)<size(img2,1)))
%       v=[img2(floor(p(2,1)),floor(p(1,1))) img2(floor(p(2,1)),ceil(p(1,1)));
%       img2(ceil(p(2,1)),floor(p(1,1))) img2(ceil(p(2,1)),ceil(p(1,1)))] ; % v = [xmin,
xmax;ymin; ymax]
%       v_q=interp2(double(v),p(1,1)-floor(p(1,1))+1,p(2,1)-floor(p(2,1))+1,'linear');
        v_q=img2(round(p(2,1)),round(p(1,1)),:);
        else
        v_q=0;
        end
        if (v_q>0)
        % if ((i-min_x+1)>0&&(i-min_x+1)<size(img1,1) &&
((j-min_y+1)>0&&(j-min_y+1)<size(img1,2)))
        if ((i>0 && (i<size(img1,2)))&&(j>0 &&(j<size(img1,1))))
        alpha=double((i-min_x_img2)/(size(img2,1)-min_x_img2));
        %alpha=1/2;
        img(j-min_y+1,i-min_x+1,:)=(1-alpha)*img1(j,i,:)+(alpha)*v_q;
```

```
        else
        img(j-min_y+1,i-min_x+1,:)=v_q;
        end
        elseif ((i>0 && (i<size(img1,2)))&&(j>0 &&(j<size(img1,1))))

        img(j-min_y+1,i-min_x+1,:)=img1(j,i,:);
        end
        end

end
```

Function which returns mosaic with blending given 2 images:

```
function i=mos2(I1,I2)
Ia = rgb2gray(I1);
points1 = (detectSURFFeatures(Ia));
%points1=points1.selectStrongest(6000);

Ib = rgb2gray(I2);
points2 = detectSURFFeatures(Ib);
%points2=points2.selectStrongest(6000);
[f1,vpts1] = extractFeatures(Ia,points1);
[f2,vpts2] = extractFeatures(Ib,points2);
indexPairs = matchFeatures(f1,f2) ;
matchedPoints1 = vpts1(indexPairs(:,1));
points1=matchedPoints1.Location;
matchedPoints2 = vpts2(indexPairs(:,2));
points2=matchedPoints2.Location;
%figure; showMatchedFeatures(Ia,Ib,matchedPoints1,matchedPoints2);

N=floor(ransac(0.99,0.3,4));
H=get_best(points1,points2,N,0.028);
i=uint8(final_combine(I1,I2,H));
```

Function to get coordinates of corners with reference to other image:

```
function a=get_corners(H,img2)
c1=ones(3,1);
c2=[1;size(img2,1);1];
c3=[size(img2,2);1;1];
c4=[size(img2,2);size(img2,1);1];

p1=pinv(H)*c1;
p1=p1/p1(3,1);
p2=pinv(H)*c2;
p2=p2/p2(3,1);
p3=pinv(H)*c3;
p3=p3/p3(3,1);
```

```
p4=pinv(H)*c4;
p4=p4/p4(3,1);
a=[p1,p2,p3,p4];
```

Function to get value of image at transformed coordinates:

```
function v_q=check(H,p,img2)
p=H*p;
p=p/p(3,1);
v_q=zeros(1,3);
        if ((p(1,1)>1 && p(1,1)<size(img2,2)) &&(p(2,1)>1 && p(2,1)<size(img2,1)))
        v_q=img2(round(p(2,1)),round(p(1,1)),:);
        else
        v_q=(zeros(1,1,3));
        end
end
```

Function to combine 5 images given corresponding homographies:

```
function img=final_combine_without_blend(img1,img2,img3,img4,img5,H12,H23,H34,H45)
H35=H34*H45;
H32=inv(H23);
H31=H32*inv(H12);

cor4=get_corners(H34,img4);
cor5=get_corners(H35,img5);
cor2=get_corners(H32,img2);
cor1=get_corners(H31,img1);

min_x=floor(min([1,cor1(1,:),cor2(1,:),cor4(1,:),cor5(1,:)]));
min_y=floor(min([1,cor1(2,:),cor2(2,:),cor4(2,:),cor5(2,:)]));
max_x=ceil(max([size(img3,1),cor1(1,:),cor2(1,:),cor4(1,:),cor5(1,:)]));
max_y=ceil(max([size(img3,2),cor1(2,:),cor2(2,:),cor4(2,:),cor5(2,:)]));

%%%%%%%
% min_x_img2=floor(min([p1(1,1),p2(1,1),p3(1,1),p4(1,1)]));
%
%
% n1=ones(3,1);
% n2=[1;size(img1,1);1];
% n3=[size(img1,2);1;1];
% n4=[size(img1,2);size(img1,1);1];
%
% temp=[p1,p2,p3,p4];
% [min_x_img2,i]=min([p1(1,1),p2(1,1),p3(1,1),p4(1,1)]);
% min_x_img2=floor(min_x_img2);
%           if          ~(temp(i,1,1)>0          &&temp(i,1,1)<=size(img1,1))&&(temp(i,2,1)>0
&&temp(i,2,1)<size(img1,2))
```

```
%       out1=intersection(n1,n3,p1,p2);
%       out2=intersection(n2,n4,p1,p3);
%       if              ((out(1,1)>-1)              &&out(1,1)<=(size(img1,1)+1)&&(out(2,1)>-1)
&&out(2,1)<(size(img1,2)+1))
%       min_x_img2=out1(1,1);
%       else
%       min_x_img2=out2(1,1);
%       end
% end
%
% min_y_img2=floor(min([p1(2,1),p2(2,1),p3(2,1),p4(2,1)]));


img=zeros(max_y-min_y+1,max_x-min_x+1,3);
v_q=zeros(1,3);
i1=0;
i2=0;
i3=0;
i4=0;
i5=0;
H33=eye(3);
num=0;
for i=min_x:max_x
        for j=min_y:max_y
        p=[i;j;1];
        v1=int16(check(H31,p,img1));
        i1=(sum(v1)>5);
        v2=int16(check(H32,p,img2));
        i2=(sum(v2)>5);
        v3=int16(check(H33,p,img3));
        i3=(sum(v3)>5);
        v4=int16(check(H34,p,img4));
        i4=(sum(v4)>5);
        v5=int16(check(H35,p,img5));
        i5=(sum(v5)>5);
        num=i1+i2+i3+i4+i5;
        if (num>0)
        img(j-min_y+1,i-min_x+1,:)=(v1+v2+v3+v4+v5)/num;
        end
        end
end
end
```

Calculating mosaic of 5 images without blending:

Comment last line of mos2 function and return H instead of i from mos2 function for this.

```
I1 = imread('image01.jpg') ;
I2 = imread('image02.jpg') ;
I3 = imread('image03.jpg') ;
I4 = imread('image04.jpg') ;
I5 = imread('image05.jpg') ;
H12 = mos2(I1,I2);
H23 = mos2(I2,I3);
H34 = mos2(I3,I4);
H45 = mos2(I4,I5);
img=final_combine_without_blend(I1,I2,I3,I4,I5,H12,H23,H34,H45);
im=uint8(img);
```