Stiching

Estimate an homography to stich togheter two images acquired by a rotating camera: - Normalization of the points - Direct Linear Transform - Transfer error minimization

References: Hartley Zisserman, Multiview Geometry Szeliski, Computer Vision .\\°//.\\°//.\\°//.\\°//.\\°//.\\°//.\\°//.\\°//.\\°//.\\°//.\\°//.\\°//.

Image Analysis and Computer Vision Politecnico di Milano

Luca Magri for comments and suggestions please send an email to luca.magri@polimi.it

.\\°//.\\°//.\\°//.\\°//.\\°//.\\°//.\\°//.\\°//.\\°//.\\°//.\\°//.\\°//.

Contents

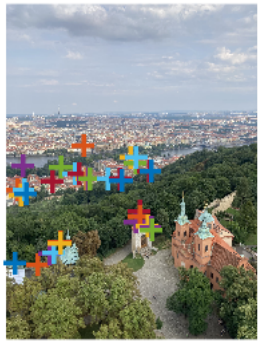
load images

```
clc;
close all;
clear;
```

```
im1 = imread("E3_data/pragaA.jpeg");
im2 = imread("E3_data/pragaB.jpeg");
im1 = imresize(im1,0.6);
im2 = imresize(im2,0.6);



%data = load('E3_data/ella_homo.mat');
%H0 = data.H;
```

get correspondences

```
do_load_correspondences = 1;

if(do_load_correspondences)
    data = load("E3_data/praga_matches.mat");
    m1 = double(data.m1);
    m2 = double(data.m2);

    % add a little bit of noise
    sigma = 0;
    m1(1:2,:)= m1(1:2,:) +sigma.*randn(2,size(m1,2));
    m2(1:2,:)= m2(1:2,:) +sigma.*randn(2,size(m2,2));
end
figure;
subplot(1,2,1);
imshow(im1);
hold all;
for i=1:size(m1,2)
    plot(m1(1,i),m1(2,i),'+','MarkerSize',15,'Linewidth',4);
end
subplot(1,2,2);
imshow(im2);
hold all;
for i=1:size(m2,2)
    plot(m2(1,i),m2(2,i),'+','MarkerSize',15,'LineWidth',4);
end
```



Preconditioning

to improve numerical stability data should be normalized in order to have zero mean and and mean distance to the center equal to sqrt(2). let's perfomr this transformation via a planar transformation Ti that translate the points and rescale them Reference: Hartley, Zisserman

```
% let's incapsulate this in a function normalize2D points
% and let's use it on m2
num_points = size(m1,2);
m1 = m1./repmat(m1(3,:),3,1);

centroid = mean(m1,2);
% distance from the centroid
scale = mean(sqrt(sum((m1-repmat(centroid,1,num_points)).^2,1)));

T1 = diag([sqrt(2)/scale;sqrt(2)/scale;1]);
T1(1:2,3)=-sqrt(2)*centroid(1:2)/scale;
m1_cond = T1*m1;

% let's encode this preconditioning step in a function
[m2_cond,T2] = precondition(m2);
```

compute the homography via DLT

```
H0 = dlt_homography(m1_cond,m2_cond);

% Once we get the homography we have to get rid of the normalization
% process by applying the inverse transformations
H0 = T2\H0*T1;
H0 = H0./norm(H0);
```

```
lambda = 0.5;
[mosaic_big,offset_x,offset_y] = image_mosaic_big(im1,im2,H0,lambda);
```

```
Warning: Integer operands are required for colon operator when used as index.
Warning: Integer operands are required for colon operator when used as index.
Warning: Integer operands are required for colon operator when used as index.
Warning: Integer operands are required for colon operator when used as index.
Warning: Integer operands are required for colon operator when used as index.
Warning: Integer operands are required for colon operator when used as index.
```

compute the errors

```
%H0(3,3) = 1e-9*randn(1,1);

% we consider the symmetric transfer error
hm1 = H0*m1;
hm1 = hm1./repmat(hm1(3,:),3,1);
invhm2 = inv(H0)*m2;
invhm2 = invhm2./repmat(invhm2(3,:),3,1);
transfer_error = 0;
for i = 1:size(m1,2)
    d1 = norm(m1(:,i)-invhm2(:,i));
    d2 = norm(m2(:,i)-hm1(:,i));
    transfer_error = transfer_error + (d1 + d2);
end
disp(transfer_error)
```

```
  113.5701
```

```
cmap = brewermap(size(m1,2),'Paired');
figure;
imshow(mosaic_big);
hold all;
for i=1:size(m1,2)
    plot(m2(1,i)+offset_x,m2(2,i)+offset_y,'o','MarkerSize',25,'Linewidth',4,'Color',cmap(i,:));
    plot(hm1(1,i)+offset_x,hm1(2,i)+offset_y,'+','MarkerSize',25,'Linewidth',4,'Color',cmap(i,:));
end
```

iterative minimization

```
MaxIterations = 1e5;
FunctionTol = 1e-8;
StepTol = 1e-16;

options = optimoptions('lsqnonlin','Algorithm','levenberg-marquardt',...
    'MaxIterations',MaxIterations,...
    'StepTolerance', StepTol, 'FunctionTolerance', FunctionTol,'Display','iter','Diagnostics','on');

fobj = @(h)get_transfer_errors(h,m1,m2);
h0 = H0(:);
h = lsqnonlin(fobj,h0,[],[],options);
H = reshape(h,3,3);
H = H./norm(H);
```

```
                                                         
   Diagnostic Information

Number of variables: 9

Functions
Objective:                              @(h)get_transfer_errors(h,m1,m2)
```

```
Gradient:                            finite-differencing

Number of lower bound constraints:          0
Number of upper bound constraints:          0

Algorithm selected
   levenberg-marquardt


_____________________________________________________________
   End diagnostic information

                                       First-Order                    Norm of
 Iteration  Func-count     Residual       optimality      Lambda           step
     0          10          391.935        4.38e+11         0.01
     1          41                                                  5.70508e-17

Local minimum possible.
lsqnonlin stopped because the relative size of the current step is less than
the value of the step size tolerance.
```

```
lambda = 0.5;
[mosaic_big,offset_x,offset_y] = image_mosaic_big(im1,im2,H,lambda);
```

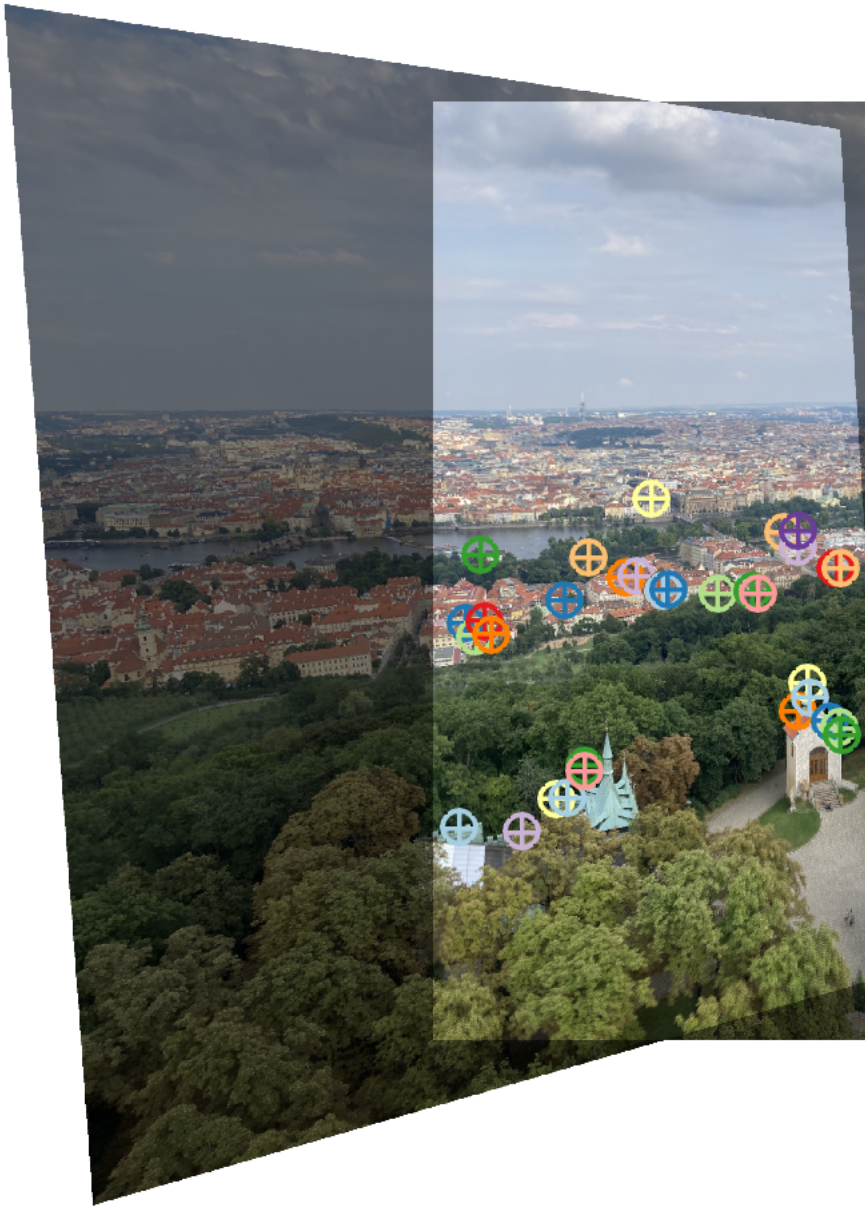
```
Warning: Integer operands are required for colon operator when used as index.
Warning: Integer operands are required for colon operator when used as index.
Warning: Integer operands are required for colon operator when used as index.
Warning: Integer operands are required for colon operator when used as index.
Warning: Integer operands are required for colon operator when used as index.
Warning: Integer operands are required for colon operator when used as index.
```

compute the errors

we consider the symmetric transfer error

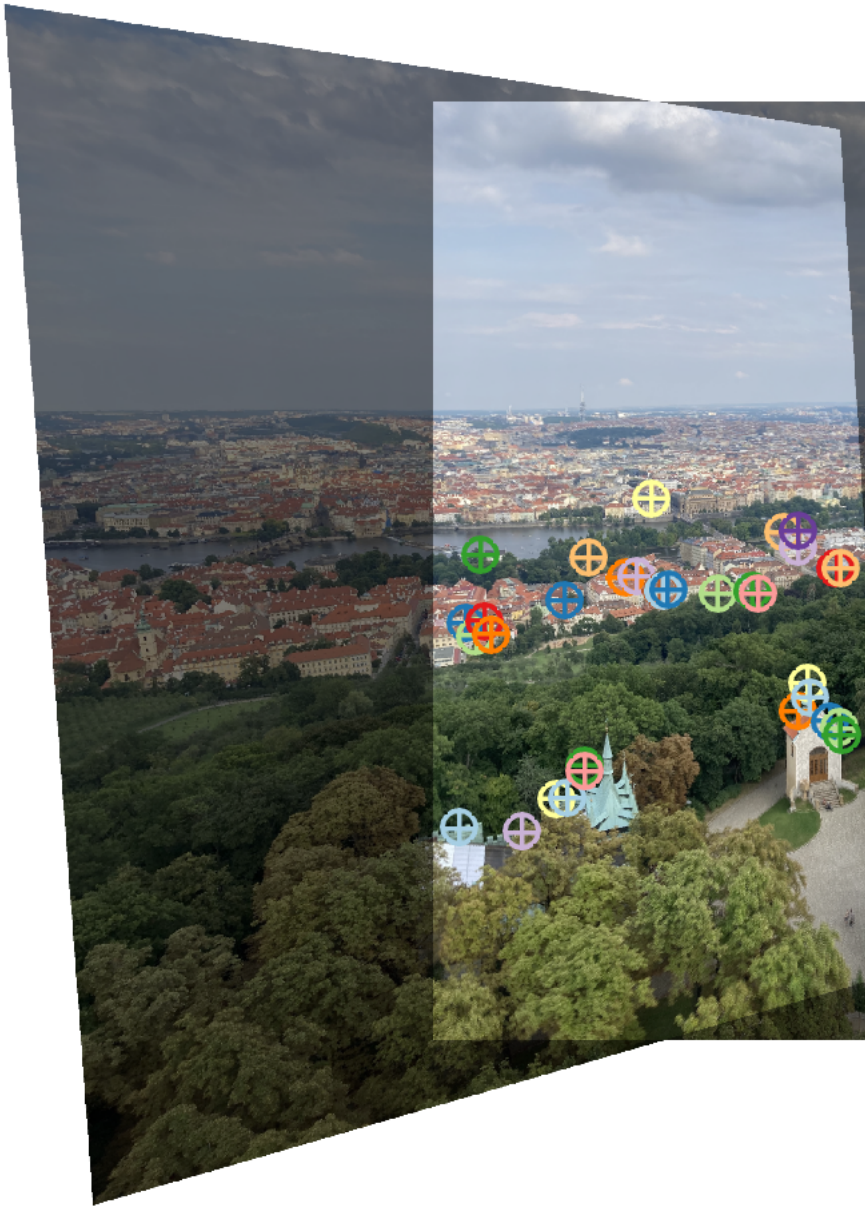
```
hm1 = H*m1;
hm1 = hm1./repmat(hm1(3,:),3,1);
invhm2 = inv(H)*m2;
invhm2 = invhm2./repmat(invhm2(3,:),3,1);
transfer_error = 0;
for i = 1:size(m1,2)
    d1 = norm(m1(:,i)-invhm2(:,i));
    d2 = norm(m2(:,i)-hm1(:,i));
    transfer_error = transfer_error + (d1 + d2);
end
disp(transfer_error)
```

```
  113.5701
```



```
cmap = brewermap(size(m1,2),'Paired');
figure;
imshow(mosaic_big);
hold all;
for i=1:size(m1,2)
    plot(m2(1,i)+offset_x,m2(2,i)+offset_y,'o','MarkerSize',25,'Linewidth',4,'Color',cmap(i,:));
    plot(hm1(1,i)+offset_x,hm1(2,i)+offset_y,'+','MarkerSize',25,'Linewidth',4,'Color',cmap(i,:));
end


H./H(3,3)
% Affine homographies have their third row fixed to
% [0, 0, 1] and arise when the image displacements come from a small
% image region or a large focal length lens is being used. When
% estimating affine homographies, the non-linear refinement is omitted: Non-linear refinement is unnecessary
% for them because in that case, the algebraic error equals the
% geometric one.
```

```
ans =

   1.0e+03 *

    0.0014    0.0001   -1.1085
    0.0002    0.0013   -0.2522
    0.0000    0.0000    0.0010
```



Published with MATLAB® R2021b