CVPR 2018, Lab class#2

Felice Andrea Pellegrino

December 4, 2018

Keypoint detection and matching and estimation of the fundamental matrix

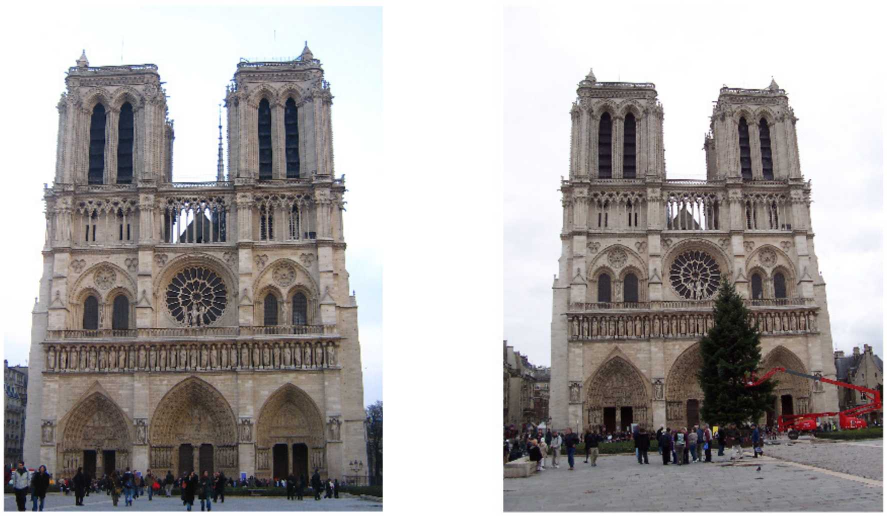
In this example we will detect keypoints in an image pair using SIFT detector, we will match the keypoints based on the SIFT descriptor, and use RANSAC for estimating the fundamental matrix in a robust manner.

Table of Contents

Load and display images

Let the images be located in a subfolder of the current folder called `images`.

```
% Load the pair of images
Ia = imread(fullfile('images','notredame1.jpg'));
Ib = imread(fullfile('images','notredame2.jpg'));
% Display the images in the same figure
figure;
subplot(1,2,1)
imshow(Ia)
subplot(1,2,2)
imshow(Ib)
```

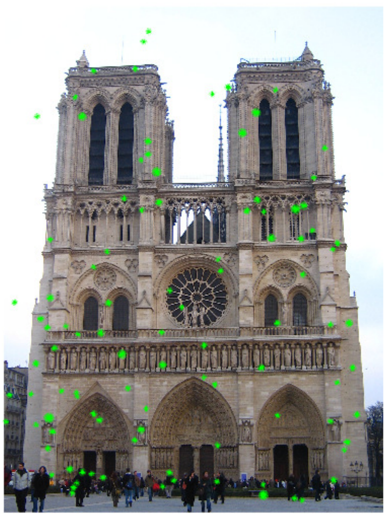


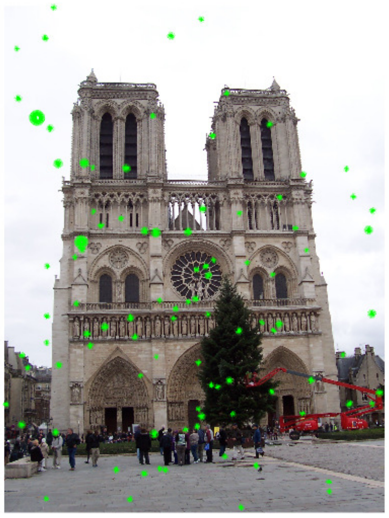
Detecting Keypoints using SIFT

The VLFeat library (http://www.vlfeat.org/) provides a very good implementation of many detectors, including SIFT. It also provides descriptors, such the SIFT descriptor. We find keypoints (represented as the four element columns `[x;y;s,o]`of matrices `frames_a` and `frame_b,`. containing the coordinates x and y , the scale and the dominant orientation of the keypoint). We also extract the 128-dimensional SIFT descriptors (the columns of matrices `descriptors_a` and `descriptors_b`). Notice that the images must be converted to grayscale *single precision* before being used by `vl_sift`.

```
% extract features using:
% [frames,descriptors] = vl_sift(image);
% The matrix frames has a column for each frame. A frame is a ...
    disk of center frames(1:2), scale frames(3) and orientation ...
    frames(4).

[frames_a,descriptors_a] = vl_sift(im2single(rgb2gray(Ia)));
[frames_b,descriptors_b] = vl_sift(im2single(rgb2gray(Ib)));
```

```
% visualize 50 randomly selected frames
subplot(1,2,1)
hold on
vl_plotframe(frames_a(:,randperm(size(frames_a,2),100)));

subplot(1,2,2)
hold on
vl_plotframe(frames_b(:,randperm(size(frames_b,2),100)));
```

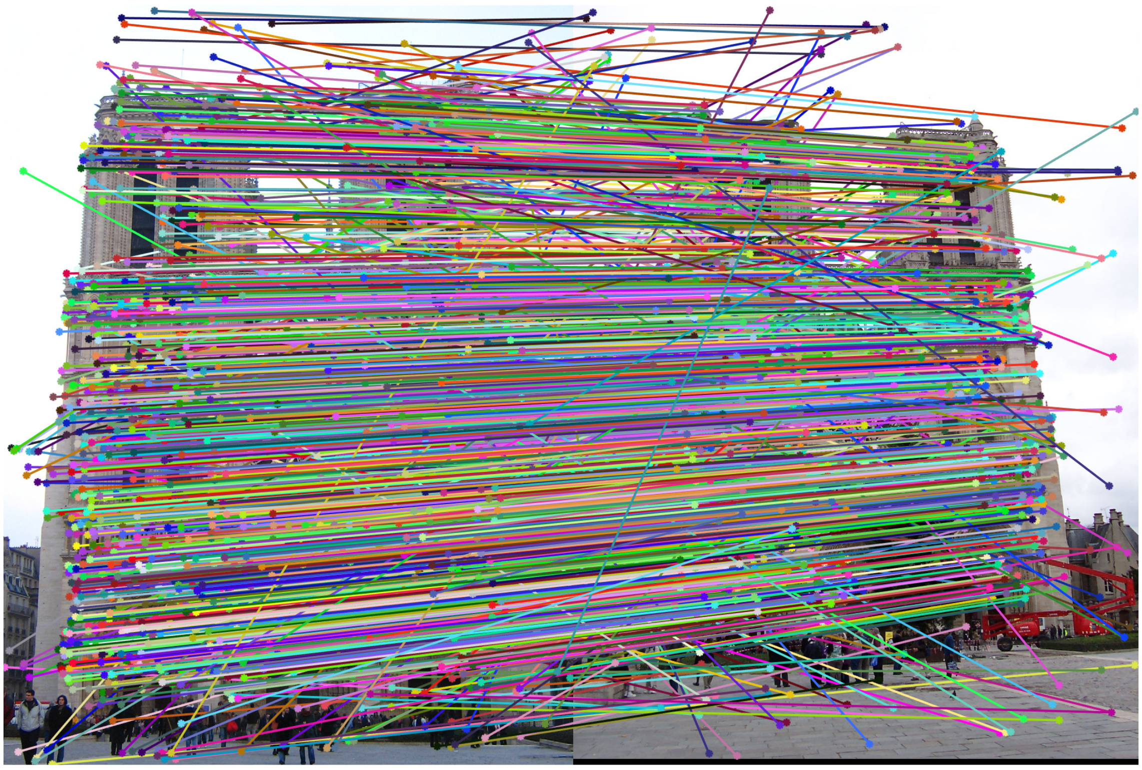




Matching keypoints

Using `vl_ubcmatch` we perform the matching, based on the Euclidean distance between descriptors. The approach is the *nearest neighbor distance ratio matching*. A descriptor D1 is matched to a descriptor D2 only if the distance d(D1,D2) multiplied by THRESH is not greater than the distance of D1 to all other descriptors. The default value of THRESH is 1.5.

```
% perform matching

[matches, scores] = vl_ubcmatch(descriptors_a,descriptors_b) ;

% show matches as colored segments between keypoints
figure;
show_correspondence(Ia, Ib, frames_a(1,matches(1,:))', ...
    frames_a(2,matches(1,:))', frames_b(1,matches(2,:))', ...
    frames_b(2,matches(2,:))');
```



Visualizing the matches it is clear that many of them are certainly wrong. Many wrong correspondences prevent to use the whole set of corrspondences to estimate the fundamental matrix F. A robust method is required, to mitigate the effect of the outliers.

Using RANSAC for robust estimation of the fundamental matrix

Recall that the fundamental matrix $F$is the matrix such that ${m'_b}^\top F m'_a = 0$ for pixel coordinates of conjugate pairs. It can be estimated by 8 correspondences

$$\underbrace{\begin{bmatrix} {m'^{(1)}_a}^\top \otimes {m'^{(1)}_b}^\top \\ {m'^{(2)}_a}^\top \otimes {m'^{(2)}_b}^\top \\ \vdots \\ {m'^{(8)}_a}^\top \otimes {m'^{(8)}_b}^\top \end{bmatrix}}_{A} \operatorname{vect}(F) = 0.$$

The solution is the nullspace of the matrix A, i.e. the nineth column of matrix Vsuch that $A = U\Sigma V^\top$. Once F has been computed, the singularity constraint can be enforced by setting to zero the smallest singular value of F. The function `estimateF(Ma,Mb)` does the job. Columns of matrices Ma and Mb contain the coordinates of corresponding points in image a and b.

```
function F=estimateF(Ma,Mb)
%Ma and Mb are 2x8
    A=zeros(8,9);
    for ii=1:8
        A(ii,:)=kron([Ma(:,ii); 1]',[Mb(:,ii); 1]'); %Kronecker product
    end

    [U,S,V]=svd(A);

    F=reshape(V(:,9),[3 3 ]);

    %enforce singularity by zeroing the smallest singular value
    [U,S,V]=svd(F);
    F=U*diag([S(1,1),S(2,2),0])*V';
end
```

The basic RANSAC loop consist of the following steps:

1. randomly draw a minimal sample from the set of data;
2. compute the fit based on the selected sample;
3. find inliers (data points that are well explained by the computed primitive);
4. repeat steps 1-3 for a fixed number of iterations;
5. the estimated $F$is the one with the greater consensus (number of inliers)

In our case it becomes:

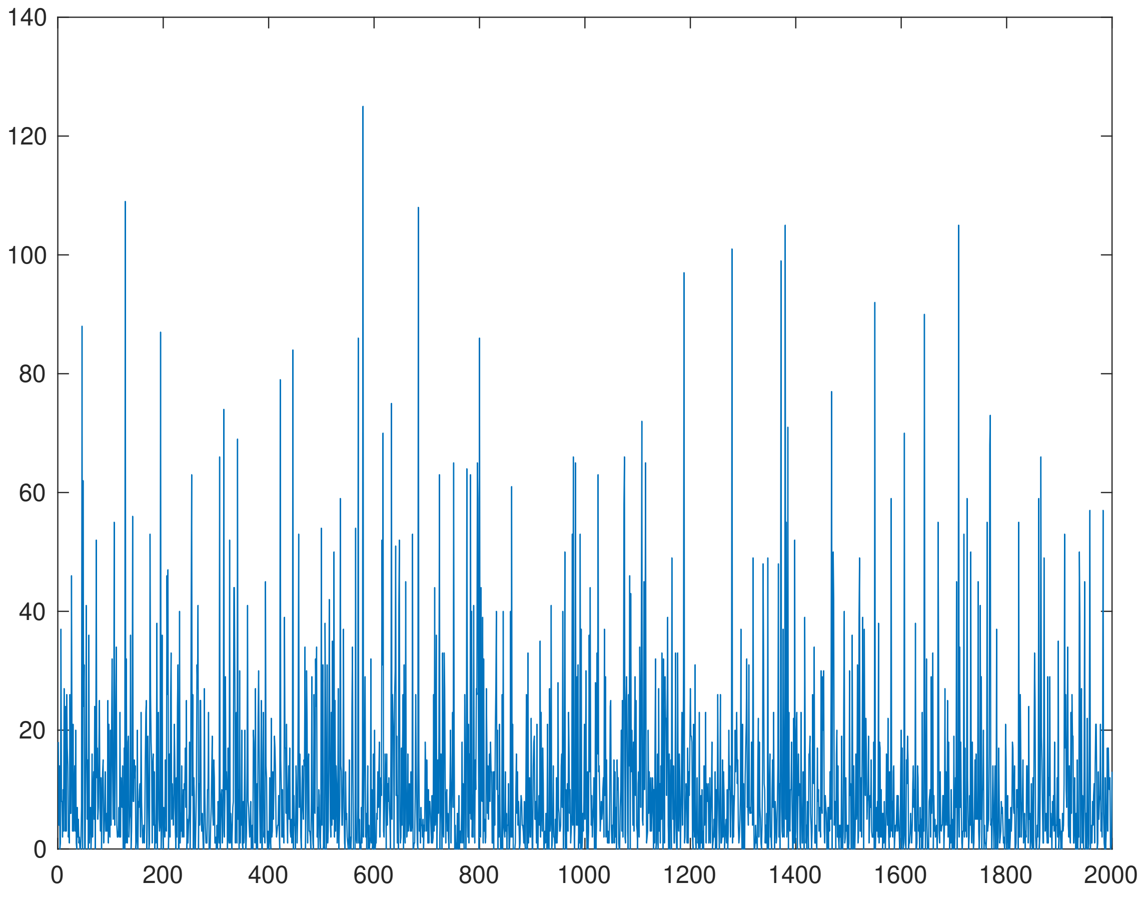
1. randomly select 8 pairs of corresponding points;
2. estimate F using the eight-point algorithm;
3. find correspondences that are in agreement with the estimated F;
4. repeat steps 1-3 for a fixed number of iterations;

Step 3 requires **a criterion** for a pair of corresponding points to be considered in accordance with the estimated F. A possibility is to put a threshold on $|{m'_b}^\top F m'_a|$, since it should be zero for conjugate points.

```
ransac_th = 0.001;  %algebraic  threshold

nmatches  =  size(matches,2);

ntrials = 2000;
FF = zeros(3,3,ntrials);  %  to  store  the  F  matrices
consensus = zeros(1,ntrials);  %  to  store  the  consensus

for  ii = 1:ntrials

    selected = randperm(nmatches,8);

    Ma= frames_a(1:2,matches(1,selected));
    Mb= frames_b(1:2,matches(2,selected));

    F = estimateF(Ma,Mb);
    FF(:,:,ii) =F;

    %  cast  votes  from  all  (including  the  selected)
    Ma_all = frames_a(1:2,matches(1,:));
    Mb_all = frames_b(1:2,matches(2,:));

    consensus(ii) = 0;
    for  jj = 1:nmatches

        %  ideally  mb^T  *  F  *ma  should  be  zero
        %  thus  we  put  a  threshold  on  the  absolute  value
        if  abs([Mb_all(:,jj);1]'*F*[Ma_all(:,jj);1]) < ransac_th
            consensus(ii) = consensus(ii) + 1;
        end

    end

end
```

Now we find the winning F:

```
figure
plot  (consensus)
```



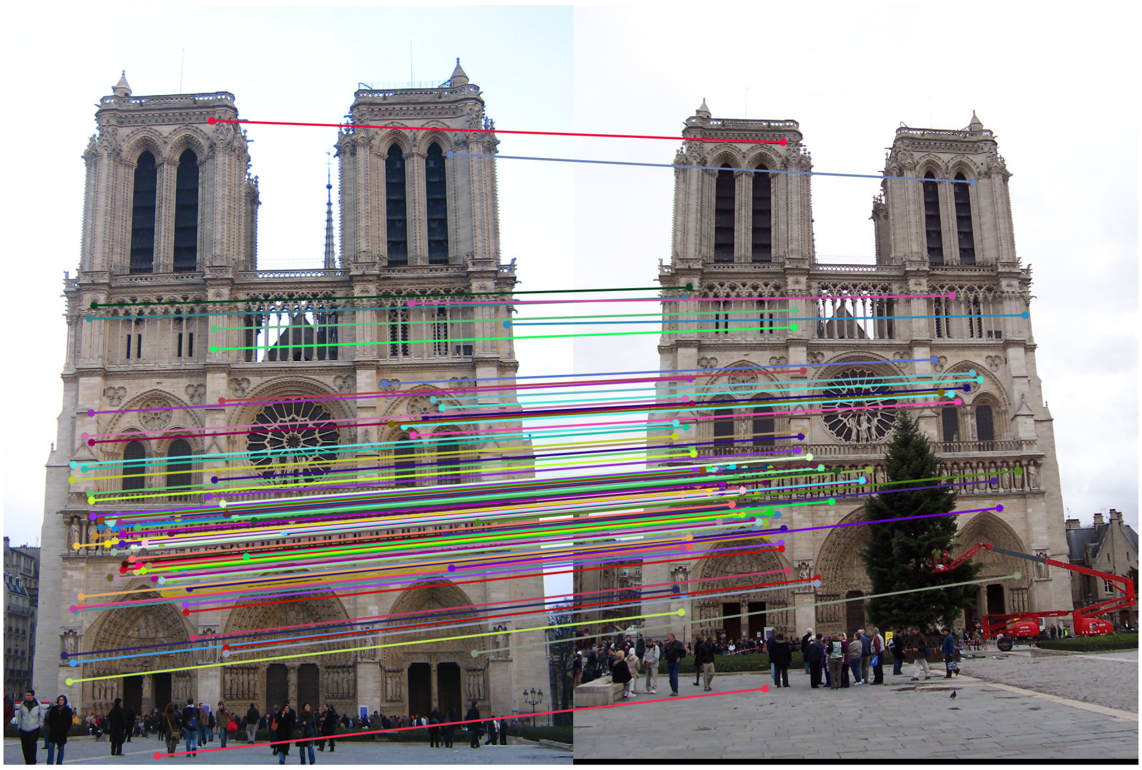
```
[~,imax]=max(consensus);
F=FF(:,:,imax); % the fundamental matrix with largest consensus
```

Finally, we discard all the outliers, based on the same criterion

```
Ma= frames_a(1:2,matches(1,:));
Mb= frames_b(1:2,matches(2,:));

 for ii = size(matches,2):-1:1

      if abs([Mb(:,ii);1]'*F*[Ma(:,ii);1]) > =ransac_th
           matches(:,ii) = []; % discard as outlier
     end

 end
```

If we visualize the inliers only, the result is much better.

```
 % plot all the inliers
hnd= figure;
figure(hnd)
show_correspondence(Ia, Ib, frames_a(1,matches(1,:))', ...
     frames_a(2,matches(1,:))', frames_b(1,matches(2,:))', ...
     frames_b(2,matches(2,:))')
```



```
ans =
  Figure (6) with properties:

      Number: 6
        Name: ''
       Color: [0.9400 0.9400 0.9400]
    Position: [210 164 1020 683]
       Units: 'pixels'

  Show all properties
```

Homework

1. Instead of the algebraic criterion use a **geometric criterion** for spotting outliers: implement a criterion based on both the distance of m'_b from the epipolar line corresponding to m'_a and the distance of m'_a from the epipolar line corresponding to m'_b. Recall that given a line $ax + by + c = 0$, the distance from the point u, v to the line is $|au + bv + c|$ provided that $a^2 + b^2$=1.
2. implement the **normalized eight point algorithm** and compare the results with the un-normalized case.

```
function F=estimateF(Ma,Mb)
%Ma and Mb are 2x8
    A=zeros(8,9);
    for ii = 1:8
        A(ii,:) = kron([Ma(:,ii); 1]',[Mb(:,ii); 1]'); %Kronecker ...
            product
    end

    [U,S,V] = svd(A);

    F=reshape(V(:,9),[3 3 ]);

    %enforce singularity by zeroing the smallest singular value
    [U,S,V] = svd(F);
    F=U*diag([S(1,1),S(2,2),0])*V';
end
```