

**Visual Recognition and Machine Learning Summer School, Paris, 2011**  
**Instance-level recognition: Practical session**

## **Stitching photo mosaics**

**Ivan Laptev, Cordelia Schmid, Josef Sivic and Andrew Zisserman**

The goal of the practical session is to automatically stitch images acquired by a panning camera into a mosaic as illustrated in images below.



### **Algorithm outline:**

1. Choose one image as the reference frame.
2. Estimate homography between each of the remaining images and the reference image. To estimate homography between two images use the following procedure:
  - a. Detect local features in each image.
  - b. Extract feature descriptor for each feature point.
  - c. Match feature descriptors between two images.
  - d. Robustly estimate homography using RANSAC.
3. Warp each image into the reference frame and composite warped images into a single mosaic.

### **Detailed steps:**

1. Download the (partial) example Matlab code and images in a single zip file from: <http://www.di.ens.fr/willow/events/cvml2011/mosaic.zip>
2. Open file mosaic.m in the Matlab text editor. Follow the instructions below by executing the provided code. Note that the code is also given in mosaic.m, so that you don't have to cut and paste from this pdf. Note that the code is partial and you will have to complete it.
3. Read in the three images of Keble college and look at them

```
imargb = double(imread('keble_a.jpg'))/255;  
imbrgb = double(imread('keble_b.jpg'))/255;  
imcrgb = double(imread('keble_c.jpg'))/255;  
...
```

These images are acquired by a panning camera and, as you will recall from the lecture, this means that the images are related by a planar homography.

4. Detect local features (single scale Harris corners) in each image. Use function "harris.m". Is the choice of single scale features (vs. multi-scale features such as Harris-Laplace) reasonable for this example?

```
topn = ??; % how many Harris corners points?
[xa,ya,strengtha] = harris(ima,topn);
[xb,yb,strengthb] = harris(imb,topn);
[xc,yc,strengthc] = harris(imc,topn);
```

Use the provided code to display the detected features.

```
% show detected points
figure(1); clf;
imagesc(imargb); axis image; hold on;
plot(xa,ya,'+y');
```

5. Extract feature descriptor for each feature point. Descriptors are simple 21x21 grayscale pixel patches extracted from an image blurred by a (fairly large) Gaussian (sigma=3 pixels). The blurring is done to obtain some tolerance to feature localization errors. This is a very simple version of the descriptor used in the paper "*Multi-Image Matching using Multi-Scale Oriented Patches*" by Brown et al., CVPR'05.

```
[Da,xa,ya] = ext_desc(ima,xa,ya);
[Db,xb,yb] = ext_desc(imb,xb,yb);
[Dc,xc,yc] = ext_desc(imc,xc,yc);
```

6. Compute the planar homography map between the images. We will start with the homography between images 'c' and 'b'. First, we will compute tentative correspondences, That is, you will need to find pairs of features that look similar (have similar descriptors) and are thus likely to be in correspondence.

```
rt      = ??          % 1NN/2NN distance ratio threshold (between 0 and 1)
Db      = dist2(Da',Db'); % compute pair-wise distances between descriptors
[Y,I]   = sort(Db,2);   % sort distances
rr      = Y(:,1)./Y(:,2); % compute D. Lowes' 1nn/2nn ratio test
inDa2   = find(rr<rt);  % take only points with a 1nn/2nn ratio below rt
I       = I(inDa2);     % select matched points
xat     = xa(inDa2);
yat     = ya(inDa2);
xbt     = xb(I);
ybt     = yb(I);
```

To obtain distinctive features, we use the ratio of distances to the first and the second nearest neighbour, instead of simply finding the best match (nearest neighbour) for each feature. For distinctive features the ratio should be small (e.g around 0.8) as the best match (the 1<sup>st</sup> nearest neighbour) should be significantly closer than the next best match (the 2<sup>nd</sup> nearest neighbour).

Visualize the tentative correspondences. Can you guess which matches are correct and which matches are wrong? Try varying the ratio threshold from 0.8 to e.g. 0.6 and 0.9. How does it effect the number of tentative matches and the proportion of correct matches?

```
figure(1); clf;
imagesc(imargb); hold on;
plot(xat,yat,'+g');
hl = line([xat; xbt],[yat; ybt],'color','y');
title('Tentative correspondences');
axis off;
```

7. Robustly fit homography using RANSAC. Use the provided code by P. Kovesi, which implements RANSAC homography fitting, where in each step the homography is estimated from 4-point correspondences using the function "homography2d.m". This function computes the homography as the null-space of a matrix A composed from the correspondences using the SVD. Inspect the function and make sure you understand it. You will need to specify the inlier threshold in normalized image co-ordinates where image dimensions are (roughly) [0 1] x [0 1].

```
t = ???; % inlier threshold (in normalized image co-ordinates)
[H12, inliers] = ransacfitHomography([xat; yat], [xbt; ybt], t);
```

Visualize the inliers:

```
% show inliers
figure(4); clf;
imagesc(imargb); hold on;
hl = line([xat(inliers); xbt(inliers)], [yat(inliers); ybt(inliers)]);
set(hl, 'color', 'y');
plot(xat(inliers), yat(inliers), '+g');
title('Inliers');
```

8. View the homography. The accuracy of the estimated homography can be visualized using the Graphical User Interface function `vgg_gui_H(ib,ic,Hbc)`. Press the left mouse button down in the left image, and the cursor should appear at the corresponding position in the right image.

```
vgg_gui_H(ima, imb, Hab)
```

9. Warp the images using the estimated homography. First we will define a mosaic image to warp all the images onto. We will use image 'b' as the reference image, and map this image to the origin of the mosaic image using the identity homography.

```
bbox=[-400 1200 -200 700] % image space for mosaic
iwb = vgg_warp_H(ib, eye(3), 'linear', bbox); % warp image b to
mosaic image
imshow(iwb)
```

Now warp image 'c' to a separate mosaic image

```
imshow(iwb)
iwc = vgg_warp_H(ic, inv(Hbc), 'linear', bbox);
imagesc(iwc)
```

and finally combine the mosaic images

```
imagesc(double(max(iwb,iwc))/255);
```

10. Compute the homographies for the other images. Repeat the above operations to compute the homography between images 'b' and 'a'. Finally combine all the warped images into a common mosaic using

```
imagesc(double(max(iwa,max(iwb,iwc)))/255);
```