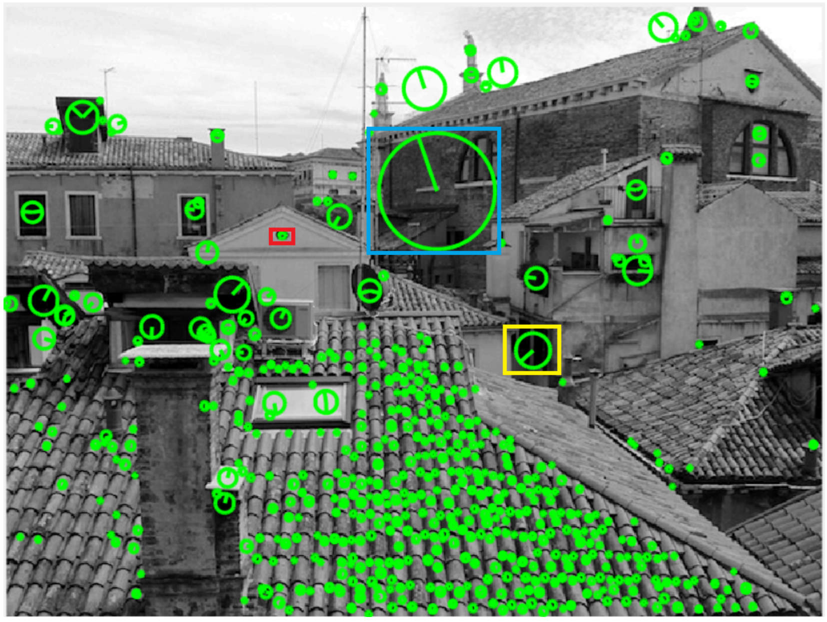
Practicum 1
Computational Vision

Julià Camps
María Leyva

November 15, 2015

1 Feature detection

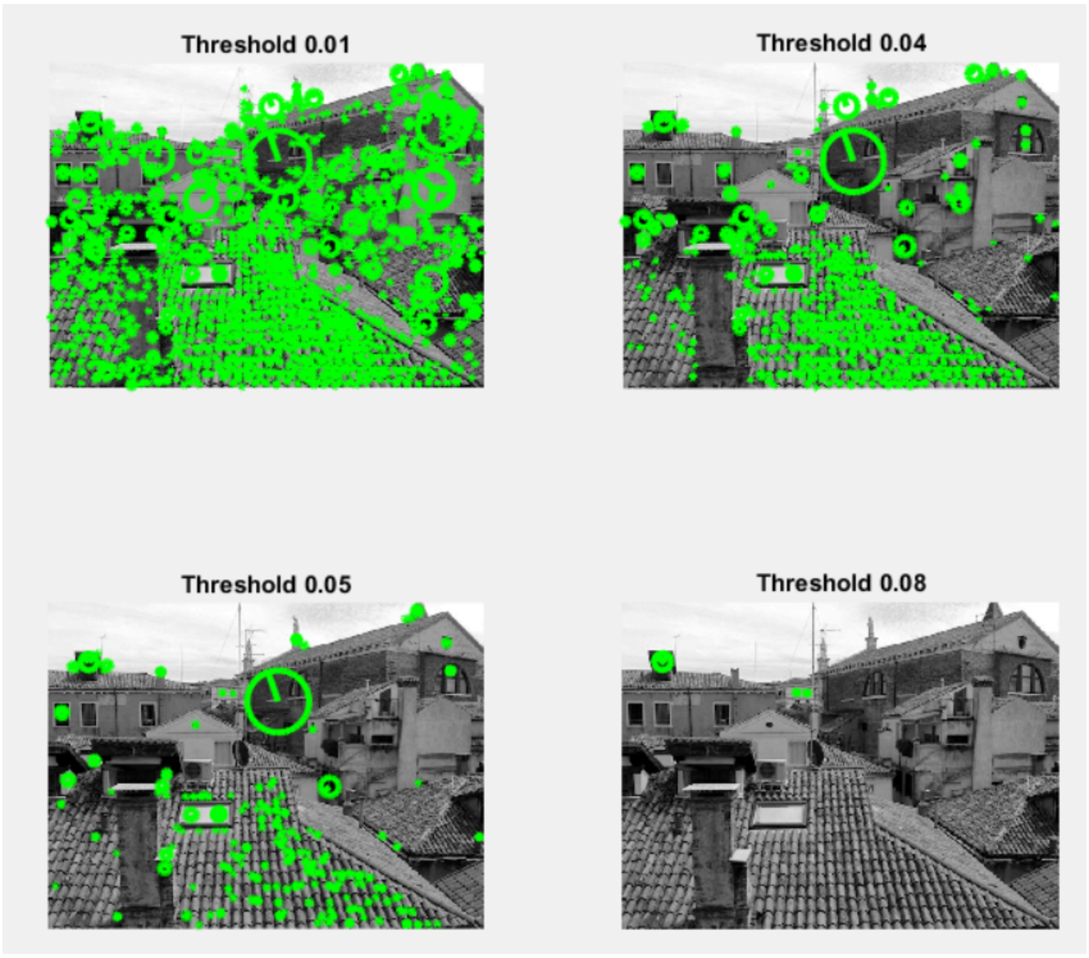
***Question**: Take at your own choice several keypoints that have been detected at different scales. Using the theory given in the lectures, comment on the reasons of why do think that a keypoint has been detected at that position and at that particular scale. You may repeat the experiment with another image (such as 'river1') to understand what a significant keypoints is.*



1. **Red feature:** This feature corresponds to a small window and it is selected it's exact size.
2. **Blue feature:** It is selected because it's different to the closest neighbors. As we can see, the top ends at the cornice, left, to the end of the wall, and to the right the other roof.
3. **Yellow feature:** it selects the two windows and it ends at the end of the windows.

***Question**: Try to slowly increase or decrease the threshold. Comment why the number of detected keypoints decreases when the threshold is increased. Is this the expected behavior according to the way the threshold is defined?*

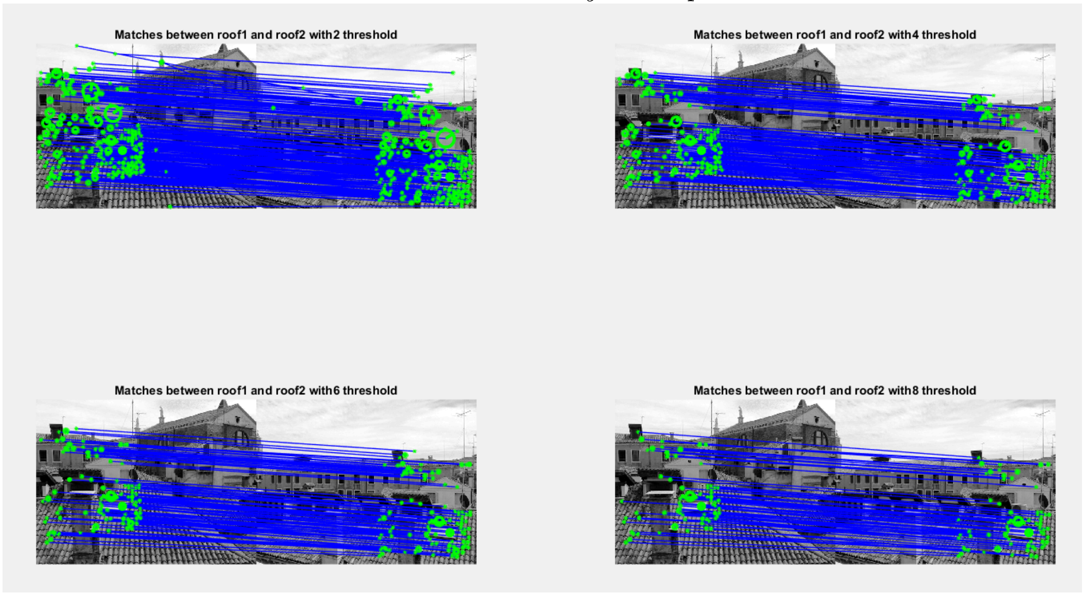
We set different values to the threshold and we observe the results.



As we can see when we decrease the threshold we obtain many more features, and when we increase it we get many less.

2 Feature matching

Question: *Modify the previous threshold. What is the effect of this threshold? Why do results improve as the threshold is increased? Comment your response.*


Matches between roof1 and roof2 with2 threshold
Matches between roof1 and roof2 with4 threshold
Matches between roof1 and roof2 with6 threshold
Matches between roof1 and roof2 with8 threshold

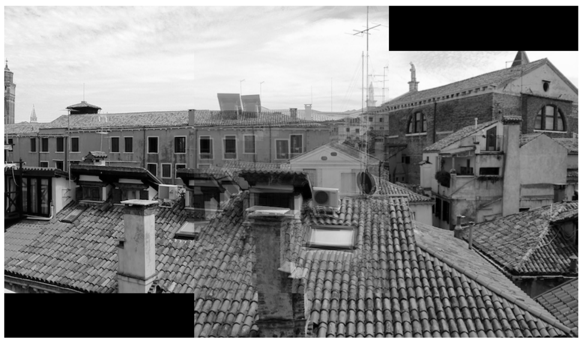
As we can see, when we increase the threshold we're getting only the most alike pairs of features. When we increase it, we get only the most similar ones.

Question: *Comment on the obtained result. Is the resulting model (more or less) correct? By looking at the original images, do you think a linear model is enough to model the transformation between the two images?*

The similar features are retrieved quite correctly. However, this might not be enough to model the transformation between roofs1 and roofs2. If we want to get this information we should consider some other methods.

3 Panorama creation

We tested the panorama.m script with roofs1.jpg and roof2.jpg images. As we can see, the common features (the chimney, windows, etc) have been detected and overlapped to form a bigger, panoramic image.



3.1 Panorama.m script

Assume we load two images from which we want to construct the panoramic image. Compute then the SIFT descriptors for each of both images and compute the correspondences between both images. Let xa be the matched SIFT keypoint coordinates of the first image, and xb the matched SIFT keypoint coordinates of the second image.

```
function Panorama(M)

    Ia = imread('images/roofs2.jpg');
    Ib = imread('images/roofs1.jpg');
    Ia = single(rgb2gray(Ia));
    Ib = single(rgb2gray(Ib));
    [fa,da] = vl_sift(Ia);
    [fb,db] = vl_sift(Ib);
    [matches, scores] = vl_ubcmatch(da,db);
    numMatches = size(matches,2);
    xa = fa(1:2,matches(1,:));
    xb = fb(1:2,matches(2,:));
```

Apply the RANSAC algorithm: loop for M times, where M is a user specified parameter (e.g. M = 100), and perform the next task:

Select randomly some (in this example we select 10) of the previous correspondences:

```
    for k=1:M
        subset = vl_colsubset(1:numMatches, 10);
```

Assume that the two images correspond to each other according to a linear model (i.e. a translation):

```

        d = xb(1:2,subset) - xa(1:2,subset);
        p = mean(d,2);
```

Apply the previous obtained translation to all the correspondences of the first image

```

        xb_ = zeros(size(xb));
        for i=1:numMatches,
            xb_(:,i) = xa(:,i) + p;
        end
```

Check how many of these new points are near the original ones

```

        n = 0;
        for i=1:numMatches,
            e = xb_(:,i) - xb(:,i);
            if (norm(e) < 5), n = n + 1; end
        end
```

Among all randomly selected displacements, select the one with largest value of n. Use this displacement to create the panoramic image.

```

        if n>max_n
            max_n=n;
            best_p=p;
        end
    end %loop end
```

panoramic image by merging the two images. We discuss two possible solutions:

we have to apply to a point at image Ia to obtain the corresponding point at (the coordinate system of) Ib. We may try to apply this transformation to map each point of Ia to (the coordinate system of) Ib. To construct the panoramic image we just need to paint all pixels that do not have a color assigned. This procedure is not a good idea, since an integer position of Ia will fall on a non-integer position of Ib. How do we draw a pixel on a non-integer position ? Do we just round to the nearest integer ? Just think a bit about it and you'll see that this is not a good way to proceed.

to compute the panoramic image. Assume we take the coordinate system of Ib. For each integer pixel of Ib (for which we do not know its color) we ask where from Ia it comes from. Again, the resulting pixel at Ia may be a non-integer, but this problem is easy to solve. Just use interpolation techniques. The algorithm is as follows. First, we take the corners of Ia and transform them to the coordinate system of Ib (which is our reference image). This allows us to compute the bounding box for the panoramic image

```

    box2 = [1 size(Ia,2) size(Ia,2) 1;
            1 1 size(Ia,1) size(Ia,1)];
    box2_ = zeros(2,4);
    for i=1:4,
        box2_(:,i) = box2(:,i) + best_p;
    end
    min_x = min(1,min(box2_(1,:)));
    min_y = min(1,min(box2_(2,:)));
    max_x = max(size(Ib,2),max(box2_(1,:)));
    max_y = max(size(Ib,1),max(box2_(2,:)));
```

We now create the panoramic image by sampling the image Ib and Ia at the corresponding points. Note that the inverse transformation is used for Ia.

```

    ur = min_x:max_x;
    vr = min_y:max_y;
    [u,v] = meshgrid(ur,vr);
    Ib_ = vl_imwbackward(im2double(Ib),u,v);
    p_inverse = -best_p;
    u_ = u + p_inverse(1);
    v_ = v + p_inverse(2);
    Ia_ = vl_imwbackward(im2double(Ia),u_,v_);
    mass = ¬isnan(Ib_) + ¬isnan(Ia_);
    Ib_(isnan(Ib_)) = 0;
    Ia_(isnan(Ia_)) = 0;
    panoramic = (Ib_ + Ia_) ./ mass;
    imshow(panoramic,[]);
end %function
```