Computer Vision Lab 4

Authors: Emer Rodriguez Formisano and Jorge Alexander

Date: 06/11/2017

1. Introduction

Checking if vlfeat code is correctly installed

```
run('vlfeat-0.9.16/toolbox/vl_setup');
vl_version
```

```
0.9.16
```

2. Feature matching

Load and observe the images for the exercise. Ia and Ib images of the same scene from different angles.

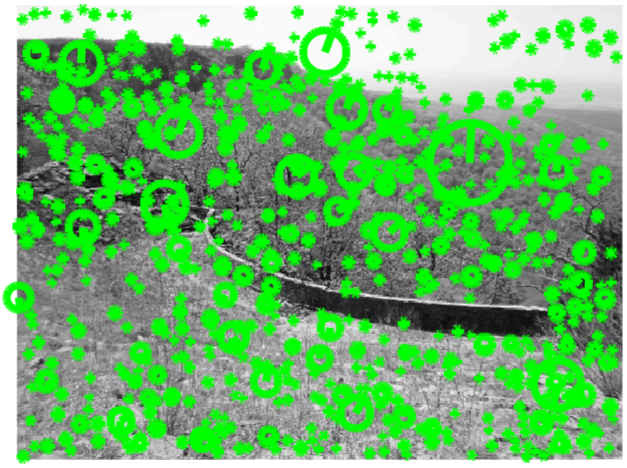
```
Ia = imread('images/im2_ex1.jpg');
Ib = imread('images/im1_ex1.jpg');
Ia = single(rgb2gray(Ia));
Ib = single(rgb2gray(Ib));
imshow([Ia Ib],[])
```

Computing SIFT and showing keypoints

```
[fa, da] = vl_sift(Ia);
[fb, db] = vl_sift(Ib);
```

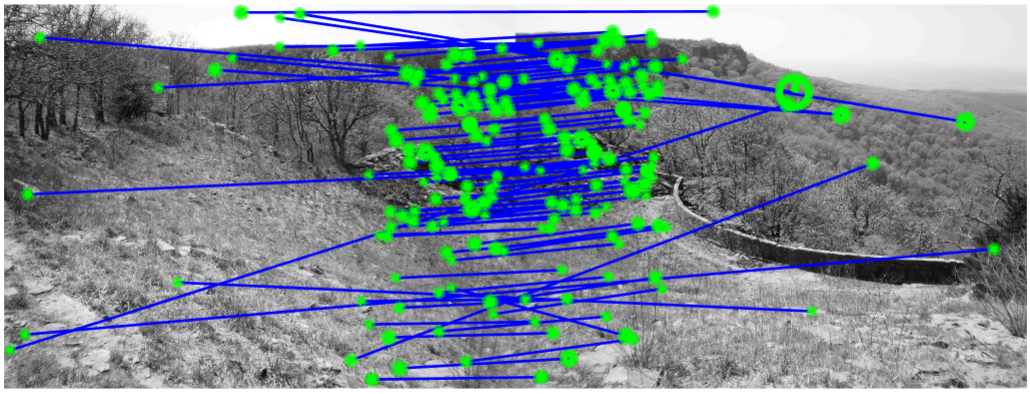
```
show_keypoints(Ia,fa)
```

```
show_keypoints(Ib,fb)
```



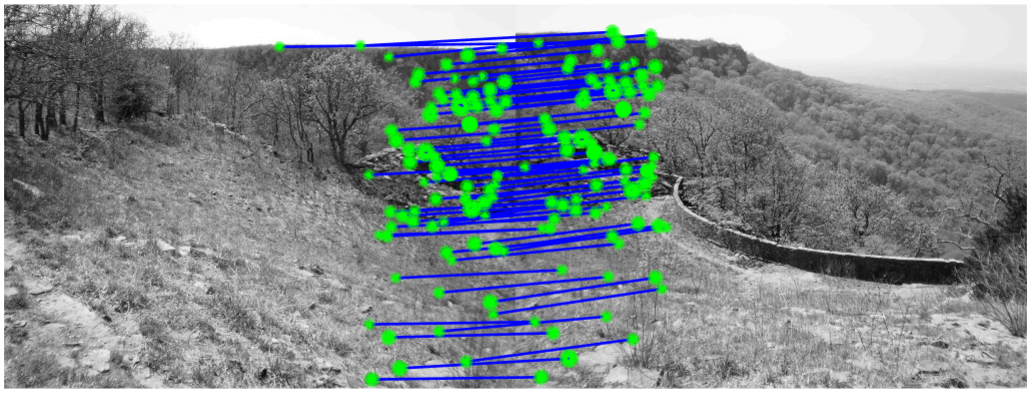
Match SIFT features using default threshold of 1.5 and show them

```
[matches, scores] = vl_ubcmatch(da, db);
show_matches(Ia, Ib, fa, fb, matches);
```



Modify treshold to values 2.0

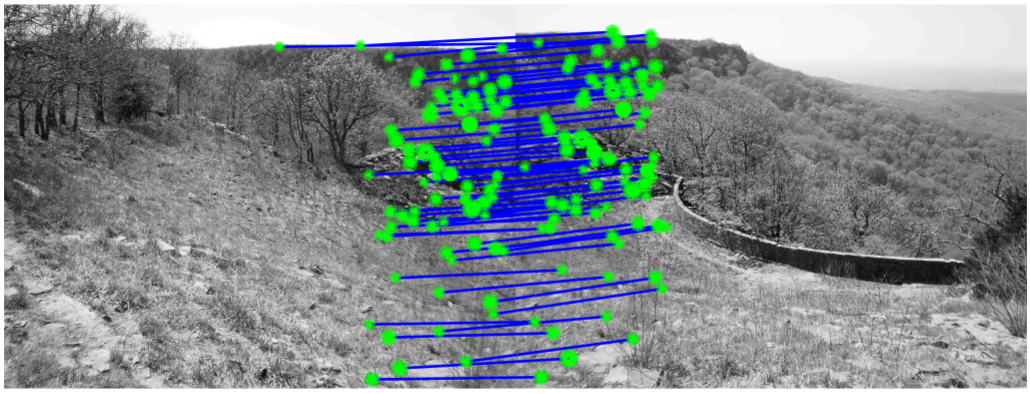
```
[matches2, scores2] = vl_ubcmatch(da, db, 2.0); show_matches(Ia,Ib,fa,fb,matches2)
```



***Question**: Modify the previous threshold. What is the effect of this threshold? Comment your response.*

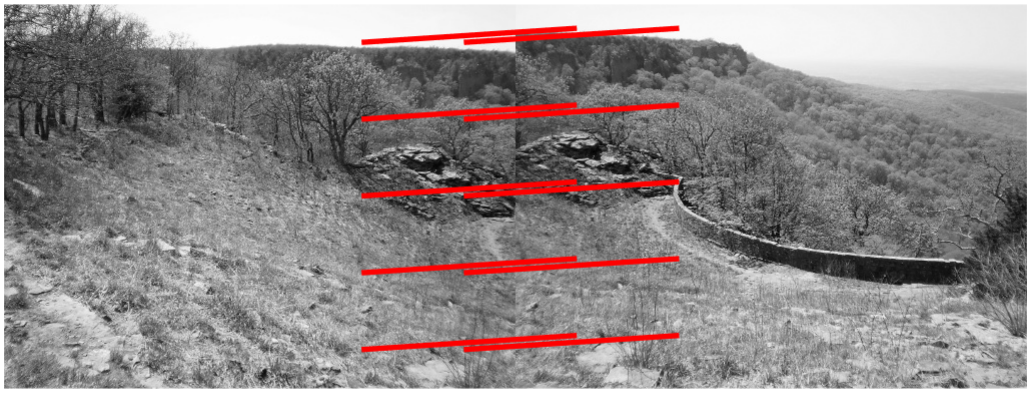
Given a keypoint the match function checks if thw following condition is true: $d_1 \cdot \text{Threshold} \leq d_2$ where d_1 is the distance to the closest match to the reference keypoint and d_2 is the second closest. The higher the threshold the more restrictive the comparison of distances will be. For example, if Threshold = 2, means that d_1 should be the half of the d_2. For this particular exercice it is better to use a higher value such as 4.0

```
[matches4, scores4] = vl_ubcmatch(da, db, 4.0); show_matches(Ia,Ib,fa,fb,matches2)
```



Assuming that both images are related according to a translational model, a mean displacement $\bar{d}$ can be estimated. The mean displacement minimes the quadratic error in the translational linear model.

```
d = fb(1:2,matches(2,:))-fa(1:2,matches(1,:));
p = mean(d,2);
show_matches_linear_model(Ia,Ib,fa,fb,p)
```



***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 model is not good enough as it was fitted using the matches obtained with the default threshold. If the sample code is updated in order to use the matches with a threshold of 4.0, the results are better. The model is now estimated with only accurate matches.

```
d4 = fb(1:2,matches4(2,:))-fa(1:2,matches4(1,:));
p4 = mean(d4,2);
show_matches_linear_model(Ia,Ib,fa,fb,p4)
```

Affine model is presented in order to obtain better results than the linear model.

```
d = fb(1:2,matches(2,:))-fa(1:2,matches(1,:));
i = size(d, 2);
A = zeros(6,6);
b = zeros(6,1);
for j = 1:i
    x = fa(1:2,matches(1,j));
    J = [x(1), x(2), 0, 0, 1, 0; 0, 0, x(1), x(2), 0, 1];
    A = A + J'*J;
    b = b + J'*d(1:2,j);
end
p = inv(A)*b;
show_matches_affine_model(Ia,Ib,fa,fb,p)
```

The model is not good for two main reasons. First, the quality of the input data is not very good, it has outliers similar to the first linear model estimated. Secondly, as the model is more complex and harder to estimate due to a higher degree of freedom.

If we stimate the model using the matches obtained using a threshold of 4.0, the results are improved. This is similar to selecting the first best 10 matches.

```
i = size(d4, 2);
A = zeros(6,6);
b = zeros(6,1);
for j = 1:i
    x = fa(1:2,matches4(1,j));
    J = [x(1), x(2), 0, 0, 1, 0; 0, 0, x(1), x(2), 0, 1];
    A = A + J'*J;
    b = b + J'*d4(1:2,j);
```

```
end
p = inv(A)*b;
show_matches_affine_model(Ia,Ib,fa,fb,p)
```

3. Panorama creation

ANSAC algorithm is implemented in order to create panorama images using the SIFT method.

```
panorama('images/im2_ex1.jpg','images/im1_ex1.jpg', 4, 100)
```

Note: please ignore the appended part at the right hand side. It is a bug in the LiveScript rendering engine. You can confirm that by running the panorama_workbench.m script manually.

```
function panorama(filenameA, filenameB, thres, max_iter)
    % Load images
```

```
    Ia_rgb = imread(filenameA);
    Ib_rgb = imread(filenameB);
    Ia = single(rgb2gray(Ia_rgb));
    Ib = single(rgb2gray(Ib_rgb));

    % Calculate SIFT features and descriptors
    [fa, da] = vl_sift(Ia);
    [fb, db] = vl_sift(Ib);
    [matches, scores] = vl_ubcmatch(da, db, thres);
    numMatches = size(matches, 2);
    xa = fa(1:2, matches(1, :));
    xb = fb(1:2, matches(2, :));

    % RANSAC
    sample_size = 10;
    prev_p = 0;
    prev_n = 0;
    for m = 1:max_iter
        % 2.a
        subset = vl_colsubset(1:numMatches, sample_size);
        % 2.b
        d = xb(1:2, subset) - xa(1:2, subset);
        p = mean(d, 2);
        % 2.c
        xb_ = zeros(size(xb));
        for i = 1:numMatches
            xb_(:,i) = xa(:,i) + p;
        end
        % 2.d (it can be combined with the previous loop)
        n = 0;
        for i = 1:numMatches
            e = xb_(:, i) - xb(:, i);
            if (norm(e) < 5)
                n = n + 1;
            end
        end
        % 2.e
        % Select the mean displacement with largest number of points near
        % to the original ones.
        if (n > prev_n)
            prev_n = n;
            prev_p = p;
%             disp(prev_n)
        end
    end
    p = prev_p;
    % 3.b
    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) + 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, :)));

    ur = min_x:max_x;
    vr = min_y:max_y;
    [u, v] = meshgrid(ur, vr);
    Ib_ = vl_imwbackward(im2double(Ib), u, v);

    p_inverse = -p;
    u_ = u + p_inverse(1);
    v_ = v + p_inverse(2);
    Ia_ = vl_imwbackward(im2double(Ia), u_, v_);

    Ib_(isnan(Ib_)) = 0;
    Ia_(isnan(Ia_)) = 0;

    panoramic = max(Ib_, Ia_);
    imshow(panoramic,[]);

end
```