Feature detection and matching

October 2015

Abstract

This laboratory is devoted to panorama creation using SIFT descriptors.

Contents

1 Introduction

In order to perform this laboratory, you need to use the `vlfeat` code. You are asked to download the code and configure it as explained in the introduction section of the previous lab. Remember that you need to run the following command in the MATLAB terminal:

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

Make sure that the code works properly before going on.

2 Feature matching

In order to compute the SIFT correspondences between two images you need to introduce the following code in the MATLAB terminal:

```
Ia = imread('images/im2_ex1.jpg');
Ib = imread('images/im1_ex1.jpg');
Ia = single(rgb2gray(Ia));
Ib = single(rgb2gray(Ib));
imshow([Ia Ib],[])
[fa, da] = vl_sift(Ia);
[fb, db] = vl_sift(Ib);
[matches, scores] = vl_ubcmatch(da, db);
```

Observe that the two images, `Ia` and `Ib`, correspond to a different view of the same scene. For each descriptor in `da`, `vl_ubcmatch` finds the closest descriptor in `db` (as measured by the L2 norm of the difference between them). The index of the original match and the closest descriptor is stored in each column of matches and the distance between the pair is stored in scores. The function uses the algorithm suggested by D. Lowe [1] to reject matches that are too ambiguous.

Two view the matchings we use the following command

```
show_matches(Ia,Ib,fa,fb,matches);
```

As before, one may view a reduced set of matchings by executing

```
show_matches(Ia,Ib,fa,fb,random_selection(matches,50));
```

Note that not all the matches are correctly performed. Matches can be filtered for uniqueness by passing a third parameter to `vl_ubcmatch` which specifies a threshold. Here, the uniqueness of a pair is measured as the ratio of the distance between the best matching keypoint and the distance to the second best one (see `vl_ubcmatch` for further details). We may thus improve the matching strategy by using a threshold

```
[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? Why do results improve as the threshold is increased? Comment your response.*

Another way to improve the matching between the two images is to assume that both images `Ia` and `Ib` are related according to a motion model and, in particular, translational model. As can be seen next, observe that the translational model `p` is assumed to be the mean displacement `d` between both images. The mean displacement is indeed the best approximation we can do in order to minimize the quadratic error between `p` and the displacements `d`.

```
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?*

We will now compute an affine model between the two images. As before we compute the affine model `p` (made up of 6 parameters) which minimizes the error between the model and the corresponding .

```
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);
```

Observe the result. It seems awful, right? It seems not possible that using a better model than before may degrade the result. This is due to the fact that with the affine model we have a larger number of degrees of freedom than before. And since we are approximating an affine model using good and bad matches the resulting affine model is indeed bad! How can be improve this? Let us try take only some of the best matchings!

```
N = 10;
[Y I] = sort(scores);
matches_sorted = matches(:,I);
show_matches(Ia,Ib,fa,fb,matches_sorted(:,1:N));
```

Observe how we do with the linear model using the 10 best matchings.

```
N = 10;
d = fb(1:2,matches_sorted(2,1:N))-fa(1:2,matches_sorted(1,1:N));
p = mean(d,2);
show_matches_linear_model(Ia,Ib,fa,fb,p);
```

Compare this result with the result we have obtained previously taking all the matchings. As can be seen, this second result is better than before.

We can try to use the affine model

```
N = 10;
d = fb(1:2,matches_sorted(2,1:N))-fa(1:2,matches_sorted(1,1:N));
i = size(d, 2);
A = zeros(6,6);
b = zeros(6,1);
for j = 1:i,
    x = fa(1:2,matches_sorted(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);
```

As can be seen, the result is not as good as expected. This is due to the fact that, as before, the affine model has a higher number of degrees of freedom. And any not perfect matching of `matched_sorted` may influence negatively when computing the model. In addition, it is known that the least squares method (used here to compute the affine model) is to sensible to outliers. Thus, other methods are usually used to cope with such issues. This is however out of the scope of this lab.

We will focus now on another way of selecting a "good set of matches". Rather than sorting the matches according to the score, we will use the RANSAC method.

3 Panorama creation

An interesting application of the use of SIFT descriptors is the panorama image creation. You are requested to implement a MATLAB script that constructs a

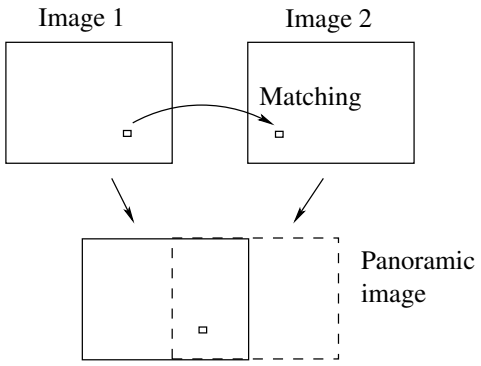

Figure 1: Panoramic image construction using SIFT descriptor matching.

panoramic image from two images, see Figure 1. For that we are going to use the RANSAC algorithm.

The objective of the algorithm you are going to implement is to automatically find the motion model between the two images and construct a panoramic image. Note that there may be some perspective distortion between the two images. Perspective distortion is more complicated to handle than a linear or affine model and is not the purpose of this lab.

1. 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.

```
Ia = imread('images/im2_ex1.jpg');
Ib = imread('images/im1_ex1.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,:));
```

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:

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

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

(b) 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);
```

(c) 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
```

(d) 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
```

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

3. Once the best displacement has been found, construct the associated panoramic image by merging the two images. We discuss two possible solutions:

   (a) Note that the linear transformation gives us the transformation that 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.

   (b) The second solution, which is usually used, is to use the inverse transform 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) + 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 = -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,[]);
```

Task: *You are requested to deliver the associated code and the experimental results obtained with your algorithm.*

4 Practicum submission

You are requested to deliver a PDF document including the experimentation performed during the first part of the lab and the second part. The report should included the answers to the questions proposed in the labs. The PDF document should include all necessary images and code to fully understand your discussion.

Deadline: 16th november for the the labs of the 20th, 27th of October (with Luis Garrido) and the lab of the 3rd of November (with Eloi Puertas).

References

[1] Lowe, "Distinctive image features from scale-invariant keypoints", International Journal of Computer Vision, 60, 2 (2004), pp. 91-110.