

# Assignment 2: Geometry

**Computer Vision, Fall 2016**

**Yael Moses**

Submission date**: 26.12.2016**

You can submit this assignment in pairs.

**A: Hands on Projection & Epipolar Geometry**

Follow the attached script, scr2.m, add the requested functions and answer

the questions.

1. Compute the intrinsic projection matrices: Mint1 and Mint2.
2. Compute the projection matrices: M1 and M2.
3. ***Question***: What is the distance between the COPs of the first and second cameras?
4. ***Question***: What are the COPs of the first and the second cameras?
5. Given object points in the world coordinate system, P=(-140,50,1200) and Q=(30,100,2000),
   1. What are the coordinates (non-homogenous) of the points in the first camera coordinate system?
   2. What are the coordinates (non-homogenous) of the points in the second camera coordinate system?
6. Project and display P=(-140,50,1200) and Q=(30,100,2000) to the two images.
7. **Question:** Is it possible to decide according to the location of the projected points in the image whether P or Q are points on the scene surface (e.g., the face or the hand)?
8. Compute the epipoles, e1 and e2, directly from M1 and M2 and the COPs.
9. Compute the fundamental matrix.
10. Check that the computed epipoles are consistent with F.
11. Choose a set of points from im1, compute the epipolar lines, and display them on the images.
12. Use the corner detector + matching from HW1, remove all matching pairs that do not satisfy the epipolar constraints using the Sampson distance (attached matlab function – sampsonDistance.m). Remove all matching pairs for which the Sampson distance is larger than 10. Display the new matching results again.

**Usefull matlabe tips:**

1. For displaying an epipolar line compute the y values of the two points  
    q=[1,y] and p=[ size(im,2)),y]. Then plot([q(1), p(1)],[q(2),p(2)].
2. Note that the y axis in Matlab is top down rather than bottom up - therefore to be consistent with the image coordinates, also the Y in the   world coordinates is top down.
3. To compute the null space of a matrix, you can use the matlab function "null".

**Part B: Hands on Triangulation**

**Triangulation**

1. Write a stereo function that receives a matrix with a set of points location in the left image, p\_L, and its corresponding points in the right image, p\_R. The function returns a set of points, P, in 3D.

P=stereo\_list(ps1,ps2, ML,MR),

The dimensions of and are *2 × m* where *m* is the number of points.

The dimensions of P are *3 × m* where *m* is the number of points.

**Note:** in class we learned how to solve for λ of each of the cameras. The solution can therefore give two different points, one from each camera. Use the average of these points to compute the 3D point in space.

1. To verify that your stereo function is correct:

Choose a pair of corresponding points by hand, compute its 3D location, and project it back to the two images.

**Usefull matlabe tips:**

To solve the triangulation: you should find the lambdas that minimize ||Ax-b||^2 where and and . To do so you can use the matlab function x=A\b. Then you should use the lambdas to compute the two points, one on each line, and average between the, to compute the desired P.

**Part C: Hands on Correspondence and simple Triangulation**

In this part of the assignment, you will write a stereo algorithm.

1. Read the two images view1.png and view5.png
2. The image planes are co-planar. The distance between the cameras is 160mm.
3. Write an algorithm that receives two images, and compute a naive disparity along corresponding epipolar lines:  
   For each pair of pixels (one from each image) compute the distance between a rectangles patches around it using the cosine distance from last HW  
    defined by where and are vectors with the pixels in a patch.  
   The correspondence is defined to be the one with the minimal distance. The algorithm should accept as a parameter the size of the patch (two numbers).

Assume that the order of points in the two images is preserved.   
The output of this algorithm is a matrix D that consists of the disparity of each pixel.

1. Apply your function to view1.png and view5.png, and display the disparity map as an image. Note, use imshow(D,[]);
2. Compute the depth map using the disparity. Add to your disparity depth map the value 100, since images were cropped. Note – simple triangulation can be applied here. Display it as an image.
3. Repeat (c) and (e) without using the assumption that the order of points is preserved.
4. Question: discuss the difference in the results. How the patch size affects the results? How the assumption of order preserving affects the results?
5. **Question:** How does the window size affect the results?
6. **Question:** What are the problems with the recovered depth?
7. To test your algorithm, you can back project the points of view5.png to view1.png, and look at the absolute value of their differences. By ‘back project’ we mean to compute view1.png using view5.png by using the computed disparity.
8. **BONUS:** use a distance function that is based on histogram of gradients to test correspondence of patches, and compare the results to the simple cosine distance. Discuss the results.

**Submit:**

A zip file which includes the following files:

1. All of the documented functions you wrote and all the files required to run them (except for the input images and the toolbox).
2. Assignment2.doc :

Your name and id.

Answers to questions.

1. Scr2.m :

The script file filled with all the missing parts.

**Note:** When choosing points by hand – please save the list of points for the grader.