Solutions to Exercise Session 6: Stereo Vision

1 Cross-correlation

- Which image transformations this method invariant to?

  Ilumination changes.

- Implement normalized cross-correlation matcher for $N \times N$ patches of an image. Fill in the missing parts in `NCC.m` and test your implementation using `ex0.m`

  The matlab code is availabe in the file NCC complete.m

2 The Automated Correspondence Problem

In order to match a point automatically to a point in another image, one can use different methods. In the last exercise session we have seen how combining a feature detector like a Harris corner detector with a simple comparision function like normalized cross-correlation can already yield nice results. Let us review the definition of NCC:

$$\begin{aligned} NCC &= \frac{N(I_1, I_2)}{\sqrt{N(I_1, I_1)N(I_2, I_2)}} \\ N(I, J) &= \iint_W (I(x, y) - \overline{I})(J(x, y) - \overline{J})dxdy \end{aligned}$$

In practice the double integral is computed as a summation over a window of size 5x5, 7x7, 9x9 or larger.

- Why will this work?

  You compute a correlation between two image windows. It's the matched filter theorem. First we unbias the mean intensity, making it invariant to illumination and allowing centered variance measures. If the two images are randomly different, the correlation will sum as many positive as negative elements, making a small sum. When the images are more or less the same, positive variances will multiply with positive and negative with negative ones, making the overall sum much larger. Hence, the NCC is a good measure for resemblance.

- What pre-conditions does the algorithm take? When will it in fact work?

  It requires that not many other parts of the image show the same resemblance. For instance, it will not work for highly monotonely textured images, like bricks or trees. It does require some texture too, because it won't work on large surfaces of the same colour. Next, it is not very invariant to scale, rotation or deformation of any kind, meaning the images must be taken from approximately the same angle, same focal length and close viewpoints.

- Why is this function called *normalized* cross correlation and what is the purpose of this normalization?

  It is called normalized, because of the division by the auto-correlations. This is necessary because high cross-correlation may result from large variations in the scene itself, too. Small changes in intensity can never give rise to high correlation measures. This is compensated by normalizing with the auto-correlation.

Run the `ex1.m` script to test the results (you can also exchange the files `harris_filter.m` and `NCC.m` with your own implementation from last week). Test it on the kasteel and esat images.

- Which images work best? Why? *The Kasteel sequence should work much better. The viewpoints are closer together, creating less deformation of features and a higher resemblance.*

- Play with the window kernel size. Which one is best?

  Small kernel sizes, like 5×5*, take only few features into account, which may be found in many places again, but computation is faster.Large kernel sizes, like* 13×13*, take a large amount of computational time. They contain so many features, the deformations between the two images become too big to resemble anything useful. Therefore it is best to choose a moderate kernel size. The best kernel size depends on the images themselves, in our case a kernel size of* 9×9 *works quite well. If the viewpoints are far apart, more weight should be taken into account for the deformation term, and smaller windows should perform better.*

- Does it work for all regions in the images? Which ones are difficult?

  The bricks, the trees, the grass, the roof, the sky ... All are too monotone or too texturized. Try selecting a point of the gutter. It's almost parallel to the epipolar line. The NCC measure is high throughout the whole line. The small negative spikes in the plot are due to discretization effects of the windowing.

For the following exercises, work with the images **esat3.jpg** and **esat4.jpg**. Run `ex1.m` again to find good correspondeces. You can add more points by left-clicking near a feature point in the left image. If the match found by NCC is not correct, you can change it by left-clicking near the correct feature point in the right image. You can remove bad matches by right-clicking near one the two corresponding feature points. Click on the right triangle to save your points for the next exercise.

3 Computing Fundamental Matrices

The fundamental matrix comprises the relationship between two images. It describes the epipolar geometry constructed by the two camera setups. As we finally want to reconstruct the scene in 3D, we'll need internal and external calibration from both camera's. Estimating the fundamental matrix is one way to go.

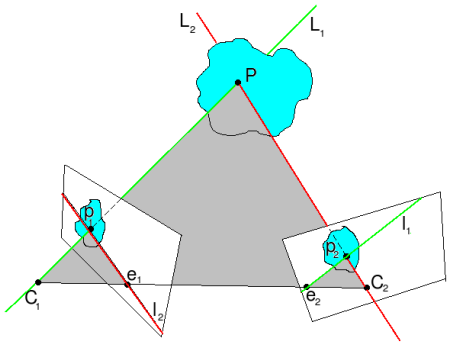

Figure 1: *Point correspondence and epipolar geometry.*

- Which relationship is defined by the fundamental matrix F? Write down the mathematical equation and draw the geometrical consequences.

  A world point P is projected onto points $p_{1,2}$ in the image planes of both cameras. In fact, any point along the line L_i through the camera center C_i and the world point P gets projected onto the same image point p_i for both cameras, i.e. $i = 1, 2$. The lines L_i are referred to as projection rays in the following. A projection ray L_i appears as point p_i in the view of camera i and as line l_i in the view of the other camera. See Figure 1 for an illustration.
  The fundamental matrix F defines the relation of the two projected points $p_{1,2}$. Each point in view i lies on a line in the other view. Those lines are called epipolar lines. Mathematically this relation is expressed by $l_1^T p_2 = 0$. Thus the line can be expressed as $l_1 = F^T p_1$ or put together:

$$p_1^T F p_2 = 0$$

- What is the rank of F? Why?

  In theory the rank of F is two. All possible epipolar lines span a subspace of all possible lines in a view. More precisely, they construct a fan of lines going through one point, the epipole. This can readily be seen, because all epipolar lines are images of projection rays of one camera onto the other one. By definition all projection rays pass through the focal point of the corresponding camera, so their images in the other view pass through the image of the focal point, too.

- How can we use corresponding points between two images to estimate F?

Each point pair (p_1, p_2) provides one linear constraint $p_1^T F p_2 = 0$ on F. Many point pairs construct a system of equations, which can be solved for the elements of F. In general this is an over-determined homogenous system, which must be solved in a least-squares way, e.g. by computing the nullspace by singular value decomposition.

- How many correspondence point pairs will we need?

  The fundamental matrix F consists of nine elements. At first sight, one might say one needs nine constraints, i.e. 9 point pairs. But F is only defined up to a scalar factor, so 8 points will do. Furthermore, we know that the rank of F is two, so not all elements are independent, leaving only 7 linear constraints and the non-linear rank constraint. But this is hard to solve analytically. You'll need at least 8 point pairs to do it linearly and the more the better your estimate will become.

Make sure you found some good matches using the `ex1.m script`. The points are only saved if you click on the red triangle in the upper corner of the image. The `ex2.m` matlab script then loads those correspondence pairs, computes the fundamental matrix, gives an error measure and saves the result. You will have to complete it with your answers of the previous questions at the `TODO` marks to make it work.

- Do you have a good estimate for F?

  If you selected the point correspondences carefully enough the estimate should be sufficient.

- What error measure was used? Why shouldn't we use $e = \sum {p^T}_{i,1} F p_{i,2}$?

  It is good to minimize the least-square error to find the parameters of F, but you can't use the above error measure, because it is not scaled. Remember p_1 and p_2 are in homogenous coordinates and F is defined up to a constant. The proposed measure in the `ex2.m` script is scaled though. It is the mean projection error, meaning the estimated distance you can expect between a point p_2 and the epipolar line l_1 of its corresponding point in the right view. This is formulated as

$$\hat{d} = \frac{1}{n} \sum \left[\frac{\left| l_1^T p_2 \right|}{p_2^{(3)} \sqrt{{l_1^{(1)}}^2 + {l_1^{(2)}}^2}} \right]$$

- Look at the numerical conditioning of the problem. Do you see the problem?

  The 3 components of the coordinate vectors are of relative order 100:100:1. This results in order gaps of 10000:1 in the linear equations (uses multiplications of p_1 and p_2 components). And even higher order gaps in the $A^T A$ squared system matrix give numerical problems during singular value decomposition. The effect of this badly conditioned problem is particulary visible when low-precision floating point representations are used, as the

single float variables in the `ex2.m` *Matlab script. A more detailed explanation is given in "Hartley, 'In Defence of the 8-Point Algorithm', PAMI 1997".*

- Adapt your code to overcome this problem. Is your estimate better now (look at the error measure)?

  You can pre-transform the point vectors to eliminate the order gap, e.g. by normalizing the x and y coordinates to be in the order of 1*, making the relative order 1:1:1. Following we will see how this affects our fundamental matrix, and how to undo this pre-transformation:*

$$\begin{aligned} p_1^T F p_2 &= 0 \\ p_1^T T^T {T^T}^{-1} F T^{-1} T p_2 &= 0 \\ (Tp_1)^T \left({T^T}^{-1} F T^{-1} \right) (Tp_2) &= 0 \\ \hat{p}_1^T \hat{F} \hat{p}_2 &= 0 \end{aligned}$$

  This means that if you pre-transform your points $\hat{p}_i = Tp_i$, you will end up with a scaled $\hat{F}$. But you can undo this scaling using $F = T^T \hat{F} T$. In our case at hand, you could choose the following transformation to condition the problem properly:

$$T = \begin{bmatrix} \frac{1}{100} & 0 & 0 \\ 0 & \frac{1}{100} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

- What is the rank of your estimated F? Why does it differ from the theoretical one?

  The rank of the estimated F is 3, i.e. the epipolar lines will not go through a single point. Its cause is that the points you selected in the views are discretisations of the real projections. They do not perfectly follow the pinhole camera model, e.g. due to radial distortion. All this adds noise to the equations, which now have no solution, certainly not one of rank 2. But if you use enough point pairs, you'll see it gets quite good, since the third singular value will be very small.

4 Epipolar Geometry

The `ex3.m` script allows you to see the fundamental matrix in action. As you learned from exercise 1, F defines the epipolar geometry between the image pair. This means that every point in one image is projected onto a line in the other one.

- What's the equation of that line (for both views)?

The equation of the right-view epipolar line l_1 of a point p_1 in the left view is

$$l_1^T p_2 = 0 \rightarrow l_1 = F^T p_1$$

The equation of the left-view epipolar line l_2 of a point p_2 in the right view is

$$l_2^T p_1 = 0 \rightarrow l_2 = F p_2$$

Fill in the `TODO 1`'s in `ex3.m` as such and run the script using the images and fundamental matrix of the previous exercise. Click on some interest points.

- Do the epipolar lines go through the corresponding points in the other view? Try both views.

  A well estimated fundamental matrix shows exactly that behaviour.

- What do you notice about the epipolar lines?

  They all go through the same point.

- How can that be explained geometrically?

  All epipolar lines are images of projection rays of one camera onto the other one. Hence all projection rays must pass through the focal point of the other camera, so their projection intersects the same point, too. That point is the image of the focal point of the other camera.

- What is its mathematical cause?

  The rank of F is 2. This means the relationship between point and epipolar line is degenerate. All possible epipolar lines will span a subspace of all possible lines in a view. More precisely, they will construct a fan of lines going through one point, the epipole.

- Click on interest points all on the same epipolar line. What do you see? Can you explain why?

  All points on one epipolar line in view i will result in exactly one (and always the same) epipolar line in the other view. This is because, seen in three dimensions, you stay in the same plane (gray in Figure 1 above) intersecting the two focal points, which has a single epipolar line projection in each view.

The intersection point is called the epipole.

- What is the geometrical meaning of the epipole?

  It is the image of the focal point of one camera in the other camera. As projection rays will always pass through the camera's focal point, the epipolar lines will always pass through the epipole.

- Can you compute both epipoles from F?

  Every epipolar line l_1 will have to pass through the epipole e_2, satisfying

$l_1^T e_2 = p_1^T F e_2 = 0$ for every choice of p_1. There is only one solution and that is if

$$Fe_2 = 0$$

Similarly,

$$F^T e_1 = 0$$

This means e_1 and e_2 span the nullspace of matrices F^T and F respectively, which is not empty since the rank of F is 2.

- What if the estimated F doesn't have the theoretical rank?

  Then you will have to look for the spanning vector corresponding to the smallest singular value.

Fill in `TODO 2` accordingly in the `ex3.m` script.

- What does the epipole tell you about the used camera setup?

  It gives you some indication of where the other camera is located, though not the exact location, since an epipole to the left can mean the other camera is left and in front of the current camera, but it could also mean it is to the right and behind. It doesn't tell you anything about the other camera's rotation either. However, the other camera's position is narrowed down from an arbitrary point in 3D to a location somewhere on a 1D line, which is not bad after all.

- What do parallel epipolar lines tell you?

  It indicates that the other camera is not in front of or behind the current camera, but right next to it. From looking at the epipoles alone, it is impossible to find out how far apart the cameras are.

Run the `ex1.m` and `ex2.m` scripts again, but now only click on points from a small neighbourhood in the image, but still with enough features (e.g. a window).

- Is F well estimated?

  We already mentioned that the discretisation of the image introduces noise. If only a small part of the view is used, the relative signal-to-noise ratio is much smaller, making the estimation of F less accurate. Also, small parts of the view don't contain much 3D information, which results in degenerate cases as in the next question.

- Does the epipolar geometry fit well for the whole image?

  Epipolar geometry might fit in the neighbourhood of the indicated point pairs, but will not extrapolate well to the rest of the image.

- Why not? Where in your code would you have noticed?

  In your code, you will not notice a significant change of the error measure, because the test points used to measure it, were the ones to estimate F.

Run the `ex1.m` and `ex2.m` scripts again, but now only click on points within one plane.

- Is F well estimated?

  Not for the whole image, see below.

- Does the epipolar geometry fit well for the whole image?

  F will fit very well for the points in the used plane, but it does not extend to other parts of the image.

- Why not? Where in your code would you have noticed?

  This is because F is the solution of a degenerate system of equations (you won't have eight linearly independent equations). This results in a bigger subspace, spanned by more than one possible solution. There is an infinite number of possible solutions for F and the one retreived is arbitrarly chosen, depending on data-noise. You will notice in your code that more than one singular value of the squared system matrix $A^T A$ tends towards zero, implying a larger-dimensional nullspace and multiple possible solutions.