M4 - Lab 3: The geometry of two views

Sergio Montoya, Laia Albors, Ibrar Malik, Adam Szummer

Pompeu Fabra University
{ sergio.montoyadepaco, laia.albors, ibrar.malik, adam.szummer
}@e-campus.uab.cat

1 Introduction

This week's goal is to compute the fundamental matrix in a robust way and apply it to a specific application which is photo-sequencing. In section 2, we discuss the computation of the fundamental matrix given noisy correspondences between a pair of images. In section 3, we apply the algorithm to estimate the fundamental matrix to photo-sequencing.

2 Estimation of the fundamental matrix

2.1 Normalized DLT algorithm

Description of the 8 point algorithm. The fundamental matrix is the algebraic representation of epipolar geometry, it describes how two images are related. This matrix relates points in an uncalibrated camera setting. The fundamental matrix F satisfies the condition that for any pair of corresponding points $x \iff x'$ in two images I and I':

$$x'^T F x = 0$$

where $x, x' \in \mathbb{P}^2$. Based on this definition, we can define an algorithm that uses this as a basis. Given a set of correspondences that have been previously identified by matching feature points between images, that we will denote as $p \iff p'$, a homogeneous system of linear equations can be formulated to compute the fundamental matrix. Setting the last coordinate of p and p' to zero we have that:

$$p'^T F p = \begin{bmatrix} u' & v' & 1 \end{bmatrix} \begin{bmatrix} F_{11} & F_{12} & F_{13} \\ F_{21} & F_{22} & F_{23} \\ F_{31} & F_{32} & F_{33} \end{bmatrix} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = 0$$

We can rearrange this equation by isolating the unknown entries of f, so it becomes a multiplication of a $1x9$ vector computed from the points coordinates and a $9x1$ vector corresponding to the unknown fundamental matrix:

$$w_i f = \begin{bmatrix} uu' & vu' & u' & uv' & vv' & v' & u & v & 1 \end{bmatrix} \begin{bmatrix} F_{11} \\ F_{12} \\ F_{13} \\ F_{21} \\ F_{22} \\ F_{23} \\ F_{31} \\ F_{32} \\ F_{33} \end{bmatrix} = 0$$

From this system of equations, we can see that we need 8 point correspondences, where each point correspondence $p_i \iff p'_i$ will give rise to one different w_i. We only need 8 points as F is defined up to

scale. We can see in this system of equations that $F_{33} = 0$. By stacking the row vectors w_i, we create a matrix W which can be used to find a solution of f, corresponding to the nullspace of W. As in the DLT algorithm for homography estimation, we can use more points so that the solution is less sensitive to noise. When using more than 8 points, if the rank of W is exactly 8, then the solution is unique (defined up to a scaling factor). If there is noise in the correspondences, the rank of W could be greater than 8. As we typically have noise in the points, the nullspace will be empty and instead, we can set this problem as a minimization problem:

$$\begin{aligned} \min_{f} \quad & ||Wf||_2 \\ \text{subject to} \quad & ||f|| = 1 \end{aligned} \tag{1}$$

Implementation. The 8 point algorithm is implemented in the function *fundamental_matrix*. In this function, we start by normalizing the points, as in the case of the DLT algorithm for homography estimation, the algorithm is not independent of the reference coordinate system of the points and this produces a greater variance in the solution under noisy points. This arises from the elements of W which have very different scale, there are entries of the order of 10^0 and others like uu' which can be of the order of 10^6. As in the DLT algorithm for homography estimation, we normalize the points so that they are centered at the origin of the coordinate frame and their average distance to the origin is $\sqrt{2}$. Therefore, we create two normalizing homographies H and H', for the set of points p and p', respectively, of the form:

$$H = \begin{bmatrix} s & 0 & -sc_x \\ 0 & s & -sc_y \\ 0 & 0 & 1 \end{bmatrix}$$

where the mean of the set of points x is $c = [c_x, c_y]^T$ and s is the average distance to the centroid. This computation is defined in the function *normalization_H*. Then, with the normalized points we create the matrix W to estimate the fundamental matrix as in Equation 1. We find the matrix $\hat{F}$ by solving the previous system of linear equations, which will not be in general of rank 2 due to noise in the points. We can enforce this property by finding the closest rank 2 matrix to $\hat{F}$ in Frobenius norm, which is obtained by setting the last eigenvalue to 0. By setting the last eigenvalue to 0 in the eigenvalue matrix D of the SVD of $\hat{F} = UDV^T$ we obtain D_{rank2}. We can obtain the fundamental matrix of rank 2 as $F_{rank2} = UD_{rank2}V^T$. The last step is to denormalize F_{rank2}, as it has been estimated with normalized points. The denormalized fundamental matrix becomes $F = H'^T F_{rank2} H$. We can easily see that this is the appropiate denormalization as:

$$p'^T F p = (p'H')^T F_{rank2}(Hp) = \tilde{p'}^T F_{rank2} \tilde{p} = 0$$

where $\tilde{p'}$ and $\tilde{p}$ denote the normalized points p' and p, respectively.

Comparison with the groundtruth fundamental matrix. As we have defined the function that allows us to estimate fundamental matrix, we want to verify the correct implementation. We can do so by comparing the error given by comparison of the known fundamental groundtruth matrix with the results of our solution.

As we know fundamental matrix is defined by equation 2:

$$F = K'^{-T}[T'_x]RK^{-1} \tag{2}$$

Given the parameters of the cameras, we can calculate the groundtruth fundamental matrix. We have given T' and R in the example but we do not have intrinsic camera parameters K to calculate F. However since we know that the camera matrix is defined as:

$$P = K\left[R|t\right] \tag{3}$$

Given P1 equal to:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

We can deduce that K must be identity matrix. The same follows for $P2$ given the equation 3 for camera parameters, hence K' also must be identity. That leads to the conclusion that fundamental matrix ground truth can be calculated by:

$$F = [T'_x]R \tag{4}$$

This gives us the groundtruth fundamental matrix equal to:

$$F_{gt} = \begin{bmatrix} 0.097 & 0.365 & -0.188 \\ -0.365 & 0.097 & 0.566 \\ 0.035 & -0.596 & 0 \end{bmatrix}$$

And from the normalized 8 point algorithm, we get the estimated matrix equal to:

$$F_{est} = \begin{bmatrix} 0.097 & 0.365 & -0.188 \\ -0.365 & 0.097 & 0.566 \\ 0.035 & -0.596 & 0 \end{bmatrix}$$

with difference defined by the norm equal to

$$2.05^{-14}$$

which proves that the algorithm works as expected for this toy example.

2.2 Robust estimation

To get a robust estimate of the fundamental matrix we will use RANSAC. The algorithm described in subsection 2.1 is sensible to outliers in the correspondences, a situation that is common as we are matching computed keypoints. We will extend the algorithm to both be robust to these outliers and also get the set of inliers corresponding to the estimated matrix.

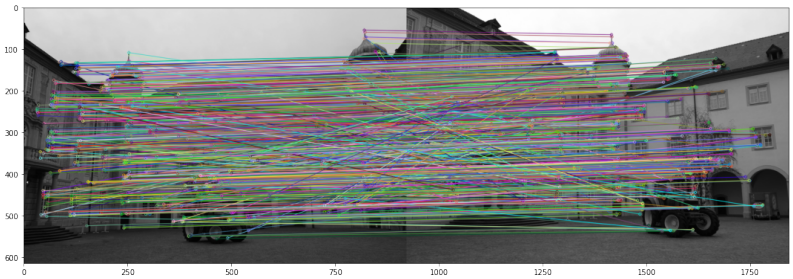
To do this, we will pick 8 samples randomly from our set of points. This is the required number of points to produce an estimate of the fundamental matrix. We will use this estimate to find the points that are a good fit to this matrix (the inliers), and when we get a certain number of inliers or we surpass the maximum number of iterations we will stop the algorithm, picking the matrix that had the maximum number of inliers.

To choose which points are inliers or outliers, we will use the Sampson distance, which gives a first-order approximation to the geometric distance. Given pairs of correspondences $x \leftrightarrow x'$ and an estimate of a fundamental matrix F, we can compute the Sampson distance as in Equation 5. Note that we do not require any subsidiary variables unlike the geometric distance in last lab.

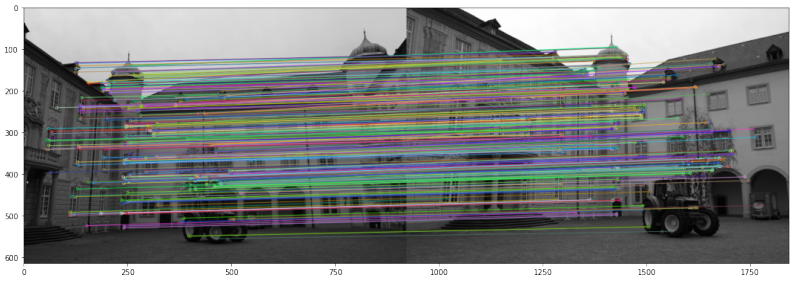
$$\frac{(x'^T F x)^2}{(Fx)_1^2 + (Fx)_2^2 + (F^T x')_1^2 + (F^T x')_2^2} \tag{5}$$

where $(\cdot)_i$ is the ith component of the vector. We can see how minimizing this distance also minimizes the distance between the epipolar line of a point and its correspondence. To classify the points as inliers or not, we have to compute the distance in Equation 5 for each one of the points, and use a threshold t: the points with a smaller distance than t will be considered inliers. We have implemented this distance making use of *numpy*'s vectorized operations, decreasing the execution time from the order of miliseconds when using a *for* loop to the microseconds.

To stop the algorithm, we either achieve the maximum number of iterations set by the user, or use an estimation of the number of iterations until finding at least one set of points with all inliers. This last number of iterations can be dynamically computed while we iterate, and it derives as follows:



(a) All matches found.



(b) Inlier matches after estimating F with RANSAC.

Fig. 1: Using RANSAC to filter outliers of keypoint matches between views related by F.

- f_{inlier}^s is the probability of picking s consecutive inliers, where f_{inlier} is the fraction of inliers with respect of the total points.
- $(1 - f_{\text{inlier}}^s)^N$ is the probability of picking N samples which contain outliers.
- Thus, $1 - (1 - f_{\text{inlier}}^s)^N = p$ is the probability that at least one of these N samples does not contain outliers.

With the use of logarithms, we can get the desired number of iterations:

$$N = \frac{\log(1-p)}{\log(1 - f_{\text{inlier}}^s)} \tag{6}$$

In our case, $s = 8$, and we will use $p = 0.99$. As we iterate, we compute the fraction of inliers f_{inlier}, and if Equation 6 gives us a number of iterations smaller than the ones we have already performed, it means that with probability 99% we have already seen the largest set of inliers and we can stop the algorithm.

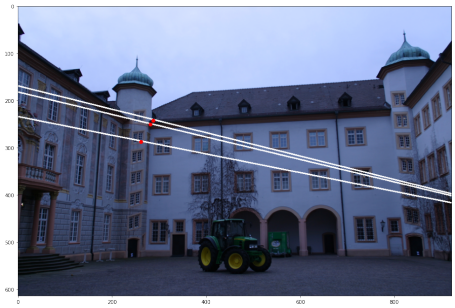


Fig. 2: Epipolar lines Fx, and the corresponding points x'

3 Application: Photo-sequencing

This week's application is Photo Sequencing. That is, temporally ordering a set of still images taken asynchronously by a set of uncalibrated cameras. In the original paper, two images taken from approximately the same position are needed. Since we don't have them, we will manually pick a dynamic point corresponding to a point in a van and the projection of its 3D trajectory in the reference image. This second point is already given, but we need to find a point in the van. To do so, we look through all the keypoints from the reference image and find those that are closest to the given trajectory point. Among these keypoints, we select the one that is more centered in the van and has better matchings with the other two images. Its index is 305, so `idx1=305`.

Once we have this index, we can automatically identify the corresponding point of `idx1` in images 2 and 3 by looking for the match that has this index as query descriptor index and taking its train descriptor index. These indices will correspond to the points $p2$ and $p3$.

Then, to compute the projection of the van trajectory in image 1, ℓ_t, we just have to compute the line that joins the van point $p1$ and the given trajectory point $p1_2 = (334, 293)$. We do this by computing their cross product:

$$\ell_t = p1 \times p1_2$$

Now, to compute the projection of the 3D position of the van in the time instances corresponding to images 2 and 3 (points $p2$ and $p3$), we have to compute their epipolar lines:

$$\ell_2 = F_{12}^T \cdot p2 \qquad \ell_3 = F_{13}^T \cdot p3$$

where F_{12} and F_{13} are the fundamental matrices between images 1 and 2, and images 1 and 3, respectively. The projection of the points $p2$ and $p3$ will be the intersection of their epipolar lines with the trajectory line that we previously found:

$$p2_p = \ell_t \times \ell_2 \qquad p3_p = \ell_t \times \ell_3$$

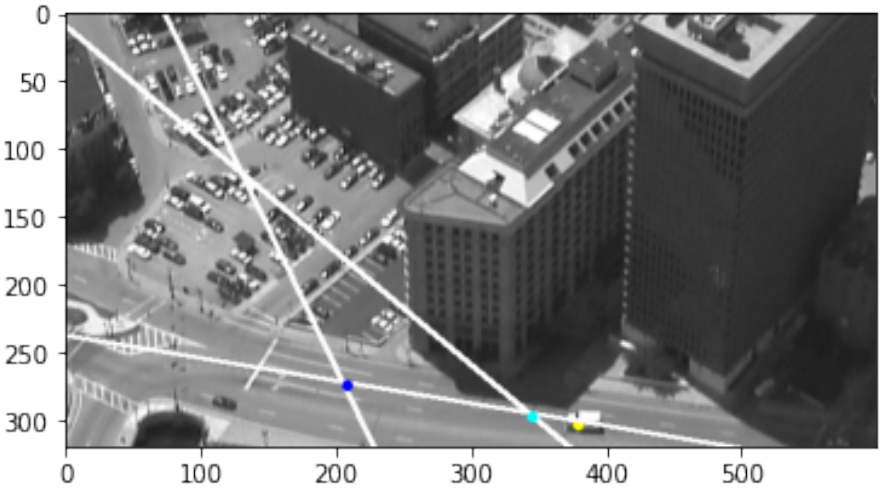


Fig. 3: Photo Sequencing results over the reference image: trajectory line, epipolar lines, van point ($p1$ in yellow) and intersection points ($p2_p$ in cyan and $p3_p$ in blue).

The results can be found in Figure 3. According to these results, the temporal order between the images is: Image 1, Image 2 and Image 3, since this is the order of the points in the direction of the trajectory line. However, in this figure we can see that the point $p3_p$ (in blue) is further than it should be if we look at the position of the van in Image 3. This can be explain if we take a look at the matches of the van point from Image 1 in images 2 and 3. As we can see in Figure 4, the match is correct between images 1 and 2; but the van is incorrectly identify in Image 3, so we are not really projecting the point of the van when computing

the epipolar line of $p3$ and its intersection with the trajectory line (because $p3$ doesn't correspond to the van).

(a) Match of the van point from Image 1 in Image 2.

(b) Match of the van point from Image 1 in Image 3.

Fig. 4: Corresponding points of the van point from Image 1 in images 2 and 3.

4 Conclusions

In this week's assignment, we have explored the 8 point algorithm for fundamental matrix estimation. We have seen that the more points we use for estimating the fundamental matrix, the more accurate the estimation will be due to being more robust to noise. After the fundamental matrix is estimated with a robust approach to outliers, it can be used to filter outliers in the point correspondences. In the photo-sequencing application, we have seen that it is crucial to accurately identify the corresponding points between images of the tracked object in order to correctly perform photo sequencing.