Chapter 10 – Fundamental Matrix Estimation

Also, Chapter 10.1 (Bundle adjustment)

Author: Gianmarco Scarano

gianmarcoscarano@gmail.com

1. Fundamental Matrix F

It is needed to map one 3D point from one view to another view.

This mapping is given by the following equation:

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

Prior to computing the Fundamental Matrix, we know it is a 3x3 singular matrix with rank 2 and 7 DOF (it means that we only need 7 points correspondences).

In order to capture the fact that the F matrix is singular, we can use the direct simple method which uses 8 points correspondences.

EXAM QUESTION: What's the minimum number of points required to estimate the F matrix? We can answer "8 point, with the direct simple method" or "In principle 7 points should be enough but we prefer the 8-point algorithm".

1.1 The 8-point algorithm

Here, each point correspondence can be expressed as a linear equation as follows:

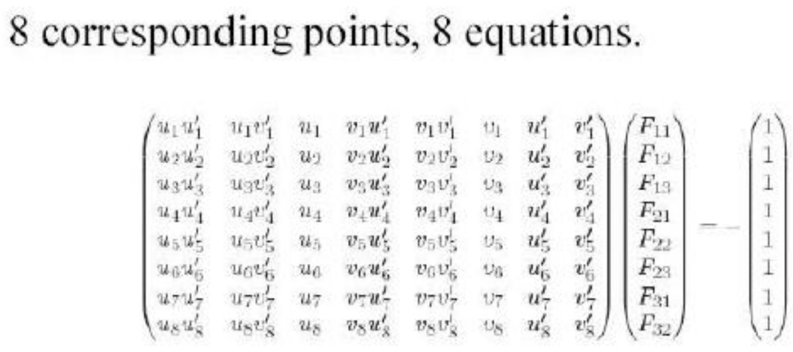
$$[u \quad v \quad 1] \begin{bmatrix} F_{11} & F_{12} & F_{13} \\ F_{21} & F_{22} & F_{23} \\ F_{31} & F_{32} & F_{33} \end{bmatrix} \begin{bmatrix} u' \\ v' \\ 1 \end{bmatrix} = 0$$

The [u v 1] vector is like the [x,y] vector. We add the 1 since we are in homogeneous coordinates.

$$[uu' \quad uv' \quad u \quad u'v \quad vv' \quad v \quad u' \quad v' \quad 1] \begin{bmatrix} F_{11} \\ F_{12} \\ F_{13} \\ F_{21} \\ F_{22} \\ F_{23} \\ F_{31} \\ F_{32} \\ F_{33} \end{bmatrix} = 0$$

We multiply both vectors with the F matrix (which is unknown) and by isolating the F matrix itself, we can end up in something like this on the left side, which is a linear equation (FOR ONE POINT ONLY).

Now, since we have again the ambiguity scaling factor, we can fix the F_{33} value to 1 and build a system of equations for all the 8 points.


8 corresponding points, 8 equations.

$$\begin{pmatrix} u_1u'_1 & u_1v'_1 & u_1 & v_1u'_1 & v_1v'_1 & v_1 & u'_1 & v'_1 \\ u_2u'_2 & u_2v'_2 & u_2 & v_2u'_2 & v_2v'_2 & v_2 & u'_2 & v'_2 \\ u_3u'_3 & u_3v'_3 & u_3 & v_3u'_3 & v_3v'_3 & v_3 & u'_3 & v'_3 \\ u_4u'_4 & u_4v'_4 & u_4 & v_4u'_4 & v_4v'_4 & v_4 & u'_4 & v'_4 \\ u_5u'_5 & u_5v'_5 & u_5 & v_5u'_5 & v_5v'_5 & v_5 & u'_5 & v'_5 \\ u_6u'_6 & u_6v'_6 & u_6 & v_6u'_6 & v_6v'_6 & v_6 & u'_6 & v'_6 \\ u_7u'_7 & u_7v'_7 & u_7 & v_7u'_7 & v_7v'_7 & v_7 & u'_7 & v'_7 \\ u_8u'_8 & u_8v'_8 & u_8 & v_8u'_8 & v_8v'_8 & v_8 & u'_8 & v'_8 \end{pmatrix} \begin{pmatrix} F_{11} \\ F_{12} \\ F_{13} \\ F_{21} \\ F_{22} \\ F_{23} \\ F_{31} \\ F_{32} \end{pmatrix} = - \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}$$

Note that the right part has a "-" sign since we moved the F_{33} value (which was 1) to the other side of the equation.

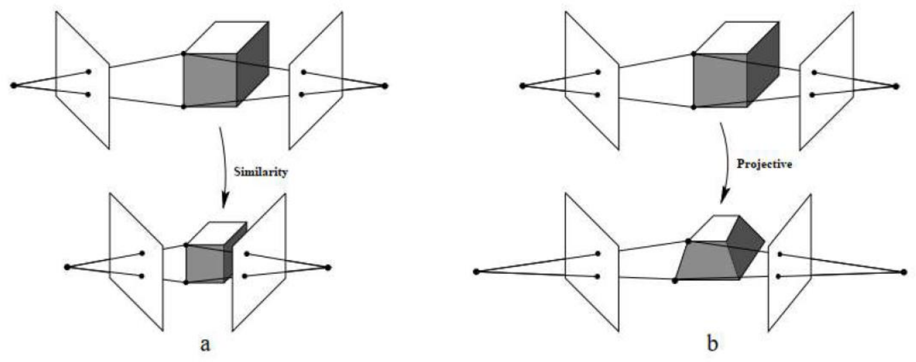
Now this is a linear system $Af = b$, but there is an issue. Often whenever we are getting the points, they do not always correspond perfectly so this $F_{33} = 1$ method isn't working properly since we are introducing some sort of noise which will not let us find a proper solution.

A fix for that is to build $Af = 0$ and solving this by minimizing $\|Af\|$ (subject to the constraint $\|f\| = 1$ in order to not let the solution collapse to 0 completely) through the Singular Value Decomposition (SVD) of the A matrix.

This solution will give us an approximation of the F matrix, which will be NON-SINGULAR, that's why we introduce the singularity constraint and get the correct estimate of F (F').

<u>N.B.</u> Always normalize the points coordinates between $[-1; +1]$ and use RANSAC to discard outliers, in order to avoid the A matrix to be ill-conditioned (since they could've been set between 0 and >1000).

1.2 3D Reconstruction



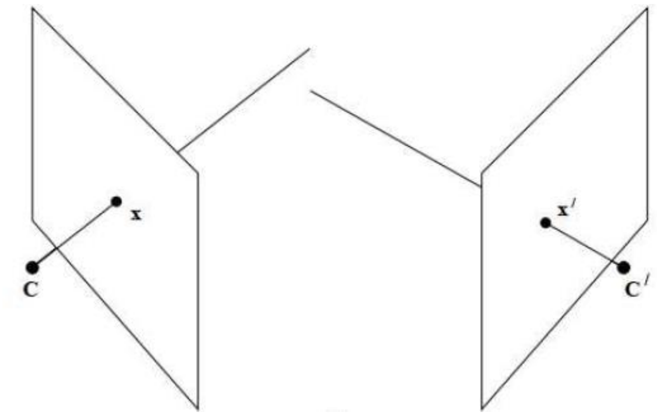
The following step after the F matrix, is to define a couple of P and P' matrices and setting P as a canonical camera (Chapter 8, 3.2).

But this introduces a reconstruction ambiguity, since those 2 matrices are defined up to a projective transformation.

So, starting from the 2D point, given F, P and P', I want to calculate the 3D point but since we have this projective ambiguity in the F matrix and P matrices, the 3D reconstruction ambiguity comes in.

We can end up fixing this with an affine transformation through calibrated cameras, which will preserve the angles.

1.2.1 The main problem



The problem is that if we have some correspondence points and we calculate the F matrix, since we have noisy measurements (because it's not a perfect correspondence), our estimation F' contains some errors and so X and x' do not satisfy the epipolar constraint $x'^T Fx = 0$ (meaning that the back-projected rays do not intersect as per the image below). We can fix this by minimizing this distance in the 2D image space (in order to avoid the problem of uncalibrated cameras in 3D projective space).

1.3 Minimizing the distance through Linear Triangulation

One of the Linear Triangulation algorithms is the Homogeneous method, which starts with rewriting

$$x = PX <=> Cross\ product\ of\ x\ and\ PX = 0.$$

This results into something like this:

$$\begin{aligned} x(\mathbf{p}^{3\mathsf{T}}\mathbf{X}) - (\mathbf{p}^{1\mathsf{T}}\mathbf{X}) &= 0 \\ y(\mathbf{p}^{3\mathsf{T}}\mathbf{X}) - (\mathbf{p}^{2\mathsf{T}}\mathbf{X}) &= 0 \\ x(\mathbf{p}^{2\mathsf{T}}\mathbf{X}) - y(\mathbf{p}^{1\mathsf{T}}\mathbf{X}) &= 0 \end{aligned}$$

The unknowns here are the X coordinates, but these equations are also linear in the components of X, so we can re-arrange these equations in order to retrieve and A matrix, as in the picture on the right ---->

$$\mathtt{A} = \begin{bmatrix} x\mathbf{p}^{3\mathsf{T}} - \mathbf{p}^{1\mathsf{T}} \\ y\mathbf{p}^{3\mathsf{T}} - \mathbf{p}^{2\mathsf{T}} \\ x'\mathbf{p}'^{3\mathsf{T}} - \mathbf{p}'^{1\mathsf{T}} \\ y'\mathbf{p}'^{3\mathsf{T}} - \mathbf{p}'^{2\mathsf{T}} \end{bmatrix}$$

We can now solve this equation $AX = 0$, by minimizing the $||AX||$ norm through the SVD of the A matrix, but the problem is that this homogeneous method is not projective/affine invariant. We can apply it but the result will be different if we apply a projective transformation or vice versa (in practice this algorithm works perfectly with an uncalibrated camera, but could be acceptable also to calibrated ones).

We have another algorithm (inhomogeneous method), where instead of working with the $X = (x, y, z)^T$ coordinates, we are working with the homogeneous coordinates $X = (x, y, z, 1)^T$ and so the linear triangulation method becomes inhomogeneous. It's very equal to the previous method, but in this case it's invariant to affine transformation (very good for calibrated cameras).

1.4 Geometric Error

The idea here is to minimize a distance function in the 2D Image Space, as we said before:

$$\hat{x} = P\hat{X}\ and\ \hat{x}' = P'\hat{X}$$

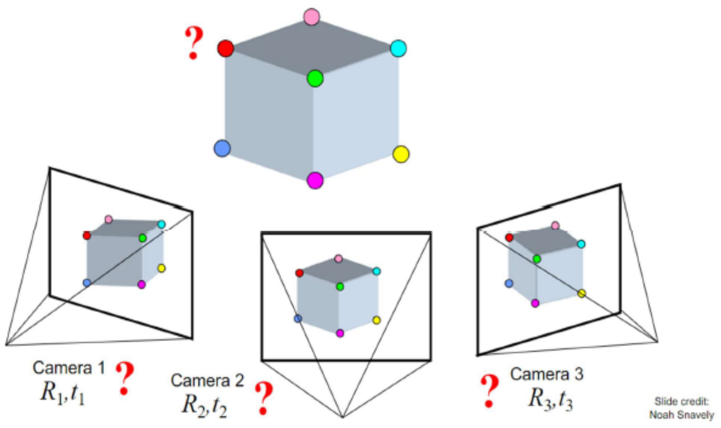
The $\hat{x}$ is the projection of the 3D point into the 3D space.

If we are able to reconstruct the 3D point without any error, $\hat{x} = x$ and $\hat{x}' = x'$, but in general if the 3D point is unknown and we go back to the 2D space, in order to find the best 3D point I have to minimize these distances below (since we are in the 2D space now):

$$C(x, x') = d(x, \hat{x})^2 + d(x', \hat{x}')^2$$

Subjected to the usual epipolar constraint that $\hat{x}'^T F\hat{x} = 0$

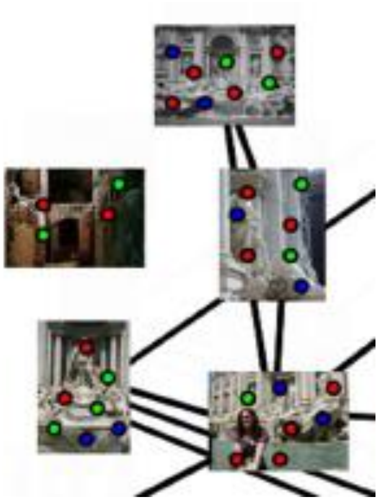
1.5 Structure from motion



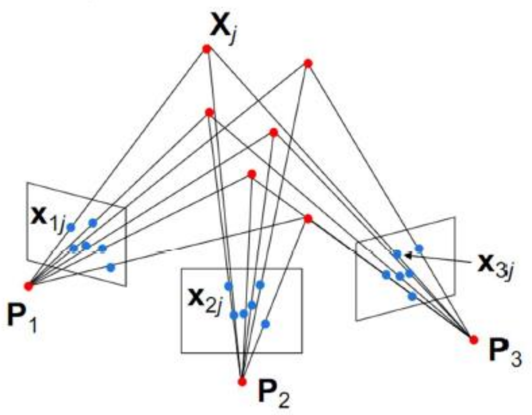
Often, in practical applications, we have a set of cameras which are moving inside the scene and we would now like to estimate (at the same time) the structure of the scene and estimate the position of each camera.

This task is called "Structure from motion".

Looking at those images on the left, we want to find the rotation and the translation parameters of the camera and the 3D point coordinates.



In the SFM pipeline, we want to reconstruct the object starting from random images as we see on the left (each one is an independent image). We can think about this through matching features, SIFT features and RANSAC which will help us in finding correspondence between the points (F matrix between each pair).



But things could get complicated, in fact, instead of calculating separately each F matrix for each pair of the images and given m images (cameras), n fixed 3D-points, we can calculate a correspondence $\boldsymbol{X_j}$ between each pair of projection matrices.

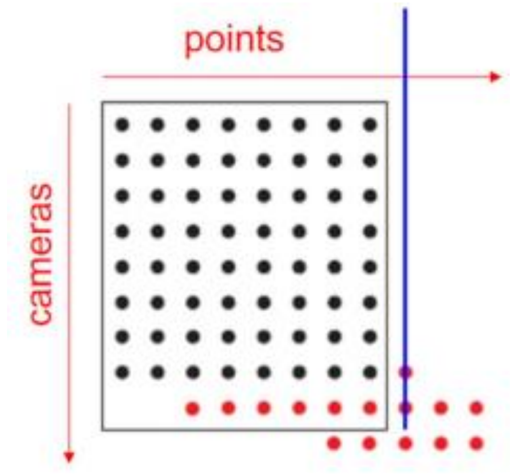
We can solve this through:

$$2mn \geq 11m + 3n - 15$$

Where m is the cameras, n is the number of 3D points.

$2mn$ is the number of correspondence points because we can relate each point to each other, as in the figure above. From a camera perspective, we have 11 unknowns (due to the P matrix). $3n$ since each 3D point has 3 coordinates, so the number of unknowns from the 3D point of view is $3n$. We subtract 15 due to the projective ambiguity. For 2 cameras, at least 7 points are needed. This is the **PROJECTIVE STRUCTURE FROM MOTION.**

An interesting case is the one where we have multiple cameras and each of them is capturing a subset of the points. In practice, this is called **Incremental structure from motion.**

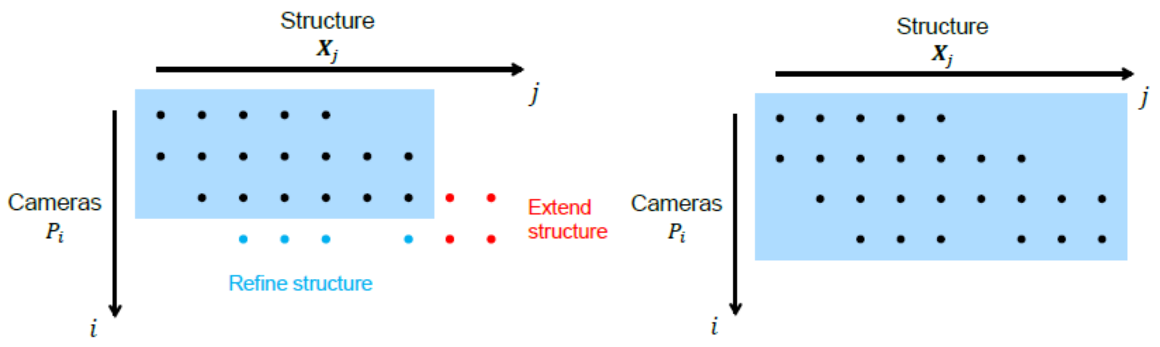


The main idea here is that I have an object and while I'm rotating around it, I'm taking pictures. It means that each picture that I take is a new camera. In practice, I add a row on the left for each picture that I'm taking.

But, because I'm moving around the object, I'm also capturing new 3D points and so I'm also adding columns.

In order to solve this task, for each additional view I add a projection matrix of the new camera using all the known 3D points that I know from the previous cameras. Along with this, I also update the 3D points based on the new camera information. So, it's some sort of an iterative way.

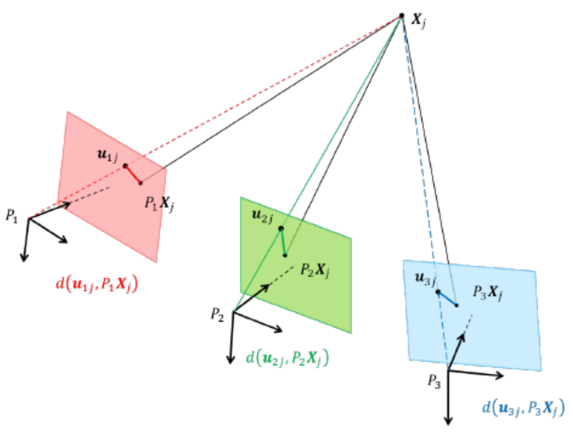
Of course, we can see that the red dots, are related to previous cameras so we can better estimate the relating points by adding new cameras. This can be seen better on the image below here:



1.5.1 Bundle adjustment algorithm

This is an algorithm that comes handy when we are dealing with the algorithm we have above.

The main idea is really about the minimization of the sum of squared reprojection errors (since if we have a very big system of equations, it could take a lot of time for an even fast computer to calculate the 3D reconstruction).



So, I take the 3D points -> I want to map to the 2D images -> I want to minimize the distances, as well as camera parameters and skew parameters (adding more unknowns).

In order to do this, we need initial estimates for all these parameters, such as 3 (for the 3D point), 12 for the camera and various intrinsic parameters (Focal length etc.).

We can reduce the calculation by also reducing the number of points/view or repeating multiple times the bundle adjustment on several different subsets.

We can also work with sparse bundle adjustments, which uses the Jacobian matrices of the cost function, determining (through the iterative minimization method) a vector Δ of changes which is a vector of changes to be made in the parameter vector. This helps us with finding the solution of extremely large structure from motion problems.