16.485: Visual Navigation for Autonomous Vehicles (VNAV) **Fall 2021**

Lecture 14: 2-view Geometry

Lecturer: Luca Carlone *Scribes: -*

This lecture discusses:

- basic principles of 2-view geometry for calibrated cameras (Epipolar constraint, Essential matrix),
- how to compute the essential matrix from pixel correspondences in 2 camera images,
- how to estimate the relative pose (up to scale) between the two cameras from the Essential matrix.

The presentation mostly follows Chapter 5 in [1, Sections 5.1-5.3].

14.1 Epipolar constraint and Essential matrix

From Lecture 12, we know how to compute keypoint correspondences in two images using feature tracking or descriptor-based feature matching. In other words, given a pixel $\boldsymbol{x}_1$ in image $\mathcal{I}_1$, we are able to compute the corresponding pixel $\boldsymbol{x}_2$ in image $\mathcal{I}_2$ (assuming the two images are picturing the same scene). Note that "corresponding pixels" refers to the fact that the two pixels picture the same 3D point.

In this lecture our goal is to compute the geometry of the two cameras taking images $\mathcal{I}_1$ and $\mathcal{I}_2$ (i.e., the relative pose between the cameras) given a number of pixel correspondences. Towards this goal, we take two main assumptions:

- the pixel correspondences are correct, i.e., for every correspondence $(\boldsymbol{x}_1, \boldsymbol{x}_2)$, the two pixels *do* represent the same 3D point. We will relax this assumption during the next lecture, since in practice many of the correspondences may be wrong. We also assume that the 3D point does not move.
- the cameras are *calibrated*, i.e., the calibration matrices:

$$\boldsymbol{K}_1 = \begin{bmatrix} s_{x_1} f_1 & s_{\theta_1} f_1 & o_{x_1} \\ 0 & s_{y_1} f_1 & o_{y_1} \\ 0 & 0 & 1 \end{bmatrix} \qquad \boldsymbol{K}_2 = \begin{bmatrix} s_{x_2} f_2 & s_{\theta_2} f_2 & o_{x_2} \\ 0 & s_{y_2} f_2 & o_{y_2} \\ 0 & 0 & 1 \end{bmatrix} \tag{14.1}$$

  of each camera is known. This is an acceptable assumption in robotics, where we can typically calibrate the cameras on our robots and compute $\boldsymbol{K}_1$ and $\boldsymbol{K}_2$ before deployment (note: for a mobile robot $\boldsymbol{K}_1 = \boldsymbol{K}_2$, i.e., both images are collected by the same camera at different time instants).

The perspective projection of the 3D point $\boldsymbol{p}^{\mathrm{w}}$ to the two cameras (Fig. 14.1) can be written as:

$$p_z^{\mathrm{c}_1}\ \tilde{\boldsymbol{x}}_1 = \boldsymbol{K}_1 \left[\boldsymbol{R}_{\mathrm{w}}^{\mathrm{c}_1}\ \boldsymbol{t}_{\mathrm{w}}^{\mathrm{c}_1}\right] \tilde{\boldsymbol{p}}^{\mathrm{w}} \qquad p_z^{\mathrm{c}_2}\ \tilde{\boldsymbol{x}}_2 = \boldsymbol{K}_2 \left[\boldsymbol{R}_{\mathrm{w}}^{\mathrm{c}_2}\ \boldsymbol{t}_{\mathrm{w}}^{\mathrm{c}_2}\right] \tilde{\boldsymbol{p}}^{\mathrm{w}} \tag{14.2}$$

where $(\boldsymbol{R}_{\mathrm{w}}^{\mathrm{c}_i}\ \boldsymbol{t}_{\mathrm{w}}^{\mathrm{c}_i})$ is the (inverse of) the pose of the camera i in the world frame, $p_z^{\mathrm{c}_i}$ is the depth of the point with respect to camera i, and $\tilde{\boldsymbol{p}}^{\mathrm{w}}$ contains the point coordinates with respect to the world frame (note: homogeneous coordinates).

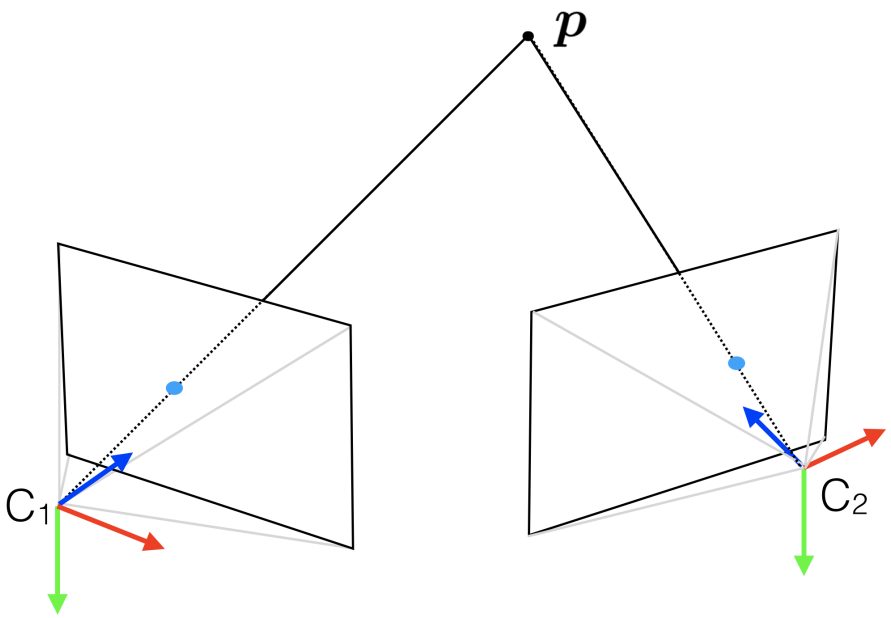

Figure 14.1: 2-view geometry.

Since the point is unknown anyway and we only attempt to compute the relative pose between the two cameras, we may simply assume that c_1 is the w frame, which allows simplifying the expressions above as:

$$d_1\tilde{\boldsymbol{x}}_1 = \boldsymbol{K}_1[\mathbf{I}_3\ \mathbf{0}_3]\tilde{\boldsymbol{p}}^{c_1} = \boldsymbol{K}_1\boldsymbol{p}^{c_1} \qquad d_2\tilde{\boldsymbol{x}}_2 = \boldsymbol{K}_2[\boldsymbol{R}_{c_1}^{c_2}\ \boldsymbol{t}_{c_1}^{c_2}]\tilde{\boldsymbol{p}}^{c_1} \tag{14.3}$$

where to simplify the notation we also defined $d_1 = p_z^{c_1}$ and $d_2 = p_z^{c_2}$.

Since the calibration matrices are known, we can pre-multiply both members of the equation on the left by $\boldsymbol{K}_1^{-1}$ and both members of the equation on the right by $\boldsymbol{K}_2^{-1}$:

$$d_1\tilde{\boldsymbol{y}}_1 = \boldsymbol{p}^{c_1} \qquad d_2\tilde{\boldsymbol{y}}_2 = \boldsymbol{R}_{c_1}^{c_2}\boldsymbol{p}^{c_1} + \boldsymbol{t}_{c_1}^{c_2} \tag{14.4}$$

where $\tilde{\boldsymbol{y}}_1 = \boldsymbol{K}_1^{-1}\tilde{\boldsymbol{x}}_1$ and $\tilde{\boldsymbol{y}}_2 = \boldsymbol{K}_2^{-1}\tilde{\boldsymbol{x}}_2$ (both still expressed in homogeneous coordinates). $\tilde{\boldsymbol{y}}_1$ and $\tilde{\boldsymbol{y}}_2$ are often called "calibrated" pixel coordinates.

Substituting $\boldsymbol{p}^{c_1}$ from the expression on the left to the right one:

$$d_2\tilde{\boldsymbol{y}}_2 = d_1\boldsymbol{R}_{c_1}^{c_2}\tilde{\boldsymbol{y}}_1 + \boldsymbol{t}_{c_1}^{c_2} \tag{14.5}$$

To simplify notation further, we drop the super- and subscripts for the rotation and translation and write $\boldsymbol{t}$ (instead of $\boldsymbol{t}_{c_1}^{c_2}$) and $\boldsymbol{R}$ (instead of $\boldsymbol{R}_{c_1}^{c_2}$). This should not cause confusion since these are the only translation and rotation we are going to deal with in this lecture. Therefore, (14.5) becomes:

$$d_2\tilde{\boldsymbol{y}}_2 = d_1\boldsymbol{R}\tilde{\boldsymbol{y}}_1 + \boldsymbol{t} \tag{14.6}$$

Premultiplying both members by $[\boldsymbol{t}]_\times$:

$$d_2[\boldsymbol{t}]_\times\tilde{\boldsymbol{y}}_2 = d_1[\boldsymbol{t}]_\times\boldsymbol{R}\tilde{\boldsymbol{y}}_1 \tag{14.7}$$

where we noticed that $[\boldsymbol{t}]_\times \boldsymbol{t} = \boldsymbol{0}_3$. Pre-multiplying both members by $\tilde{\boldsymbol{y}}_2^\mathsf{T}$:

$$d_2 \tilde{\boldsymbol{y}}_2^\mathsf{T} [\boldsymbol{t}]_\times \tilde{\boldsymbol{y}}_2 = d_1 \tilde{\boldsymbol{y}}_2^\mathsf{T} [\boldsymbol{t}]_\times \boldsymbol{R} \tilde{\boldsymbol{y}}_1 \tag{14.8}$$

However, $\tilde{\boldsymbol{y}}_2^\mathsf{T} [\boldsymbol{t}]_\times \tilde{\boldsymbol{y}}_2 = \boldsymbol{0}_3$ (since $[\boldsymbol{t}]_\times \tilde{\boldsymbol{y}}_2$ is orthogonal to $\tilde{\boldsymbol{y}}_2$). Therefore:

$$d_1 \tilde{\boldsymbol{y}}_2^\mathsf{T} [\boldsymbol{t}]_\times \boldsymbol{R} \tilde{\boldsymbol{y}}_1 = 0 \tag{14.9}$$

Since d_1 is non-zero, this leads to the *Epipolar constraint*:

$$\boxed{\tilde{\boldsymbol{y}}_2^\mathsf{T} \ [\boldsymbol{t}]_\times \boldsymbol{R} \ \tilde{\boldsymbol{y}}_1 = 0} \tag{14.10}$$

which relates corresponding pixels in two images. The matrix

$$\boxed{\boldsymbol{E} = [\boldsymbol{t}]_\times \boldsymbol{R}} \tag{14.11}$$

is known as the *Essential matrix*.

Definitions:

- epipolar plane: plane passing through the optical centers and the point $\boldsymbol{p}$
- epipoles: intersection between the segment connecting the optical centers and the image planes
- epipolar line: line corresponding to the set of pixels $\tilde{\boldsymbol{y}}_2$ in the second image that satisfy the epipolar constraint for a given pixel $\tilde{\boldsymbol{y}}_1$ in the first image.

Geometric interpretation. The epipolar constraint can be written as (see slide 12):

$$(\boldsymbol{t}_{\mathrm{c}_1}^{\mathrm{c}_2} \times \tilde{\boldsymbol{y}}_2) \perp (\boldsymbol{R}_{\mathrm{c}_1}^{\mathrm{c}_2} \tilde{\boldsymbol{y}}_1) \iff (\boldsymbol{t}_{\mathrm{c}_1}^{\mathrm{c}_2} \times \tilde{\boldsymbol{y}}_2)^\mathsf{T} \boldsymbol{R}_{\mathrm{c}_1}^{\mathrm{c}_2} \tilde{\boldsymbol{y}}_1 = 0 \iff ([\boldsymbol{t}_{\mathrm{c}_1}^{\mathrm{c}_2}]_\times \tilde{\boldsymbol{y}}_2)^\mathsf{T} \boldsymbol{R}_{\mathrm{c}_1}^{\mathrm{c}_2} \tilde{\boldsymbol{y}}_1 = 0 \iff \tilde{\boldsymbol{y}}_2^\mathsf{T} [\boldsymbol{t}_{\mathrm{c}_1}^{\mathrm{c}_2}]_\times \boldsymbol{R}_{\mathrm{c}_1}^{\mathrm{c}_2} \tilde{\boldsymbol{y}}_1 = 0$$

Stereo example (slide 14):

$$\tilde{\boldsymbol{y}}_2^\mathsf{T} [\boldsymbol{t}]_\times \boldsymbol{R} \tilde{\boldsymbol{y}}_1 = 0 \iff \tilde{\boldsymbol{y}}_2^\mathsf{T} \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & b \\ 0 & -b & 0 \end{bmatrix} \tilde{\boldsymbol{y}}_1 = 0 \iff \tilde{\boldsymbol{y}}_2^\mathsf{T} \begin{bmatrix} 0 \\ b \\ -bv_1 \end{bmatrix} = 0 \iff bv_2 - bv_1 = 0 \iff v_2 = v_1$$

Properties of the essential matrix:

- The epipolar constraint does not constrain the scale of the translation.
- A matrix is an essential matrix *if and only* if it has singular values $\{\sigma, \sigma, 0\}$ (in particular $\sigma = \|\boldsymbol{t}\|$), see Theorem 5.5 in [1].

  Proof. We only prove that the largest eigenvalue of $\boldsymbol{E} = [\boldsymbol{t}]_\times \boldsymbol{R}$ is $\lambda_{\max}(\boldsymbol{E}) = \|\boldsymbol{t}\|^2$ and the smallest eigenvalue is zero. A complete (but more involved) proof can be found in [1, Thm 5.5].

$$\lambda_{\max}(\boldsymbol{E}) = \max_{\|\boldsymbol{d}\|=1} \|\boldsymbol{E}\boldsymbol{d}\|^2 = \max_{\|\boldsymbol{d}\|=1} \boldsymbol{d}^\mathsf{T} \boldsymbol{E}^\mathsf{T} \boldsymbol{E} \boldsymbol{d} = \max_{\|\boldsymbol{d}\|=1} \boldsymbol{d}^\mathsf{T} \boldsymbol{R}^\mathsf{T} [\boldsymbol{t}]_\times^\mathsf{T} [\boldsymbol{t}]_\times \boldsymbol{R} \boldsymbol{d} \tag{14.12}$$

$$= \max_{\|\boldsymbol{d}\|=1} \boldsymbol{d}^\mathsf{T} [\boldsymbol{t}]_\times^\mathsf{T} [\boldsymbol{t}]_\times \boldsymbol{d} = \max_{\|\boldsymbol{d}\|=1} \|[\boldsymbol{t}]_\times \boldsymbol{d}\|^2 = \max_{\|\boldsymbol{d}\|=1} \|\boldsymbol{t} \times \boldsymbol{d}\|^2 = \|\boldsymbol{t}\|^2 \tag{14.13}$$

$$\lambda_{\min}(\boldsymbol{E}) = \min_{\|\boldsymbol{d}\|=1} \|\boldsymbol{E}\boldsymbol{d}\|^2 = \min_{\|\boldsymbol{d}\|=1} \|\boldsymbol{t} \times \boldsymbol{d}\|^2 = 0 \tag{14.14}$$

□

- The space of the essential matrices is called the *Essential space* $\mathcal{S}_E$ (i.e., the space of 3×3 matrices that can be written as $[\boldsymbol{t}]_\times \boldsymbol{R}$ for some $\boldsymbol{R} \in \mathrm{SO}(3)$ and $\boldsymbol{t} \in \mathbb{R}^3$). The projection of a matrix $\boldsymbol{M}$ onto the Essential space can be computed as prescribed in [1, Thm 5.9]:

$$\underset{\boldsymbol{E} \in \mathcal{S}_E}{\arg\min} \|\boldsymbol{E} - \boldsymbol{M}\|_F^2 = \boldsymbol{U} \begin{bmatrix} \frac{\lambda_1+\lambda_2}{2} & 0 & 0 \\ 0 & \frac{\lambda_1+\lambda_2}{2} & 0 \\ 0 & 0 & 0 \end{bmatrix} \boldsymbol{V}^\mathsf{T} \tag{14.15}$$

  where $\boldsymbol{M} = \boldsymbol{U}\mathrm{diag}\,(\lambda_1, \lambda_2, \lambda_3)\,\boldsymbol{V}^\mathsf{T}$ is a singular value decomposition of $\boldsymbol{M}$.

14.2 How to estimate the Camera Poses from Correspondences?

In this section we address the following problem: given N (calibrated) pixel correspondences, compute the relative pose (up to scale) between the cameras. This is typically done in 2 steps, discussed below:

- use the pixel correspondences and the epipolar constraint to estimate the essential matrix $\boldsymbol{E}$
- retrieve the relative pose $(\boldsymbol{R}, \boldsymbol{t})$ between the cameras from the essential matrix $\boldsymbol{E}$

14.2.1 Compute the Essential Matrix from Pixel Correspondences

Assume that we are given N (calibrated) pixel correspondences $(\tilde{\boldsymbol{y}}_{1,k}, \tilde{\boldsymbol{y}}_{2,k})$ for $k = 1, \dots, N$. As mentioned at the beginning of this document, we assume that there is no outlier (i.e., we do not have wrong correspondences). Each of these correspondences need to satisfy the epipolar constraint (14.10):

$$\tilde{\boldsymbol{y}}_{2,k}^\mathsf{T}\, \boldsymbol{E}\, \tilde{\boldsymbol{y}}_{1,k} = 0 \qquad k = 1, \dots, N \tag{14.16}$$

Noticing that $(\tilde{\boldsymbol{y}}_{1,k}, \tilde{\boldsymbol{y}}_{2,k})$ are known pixel values, we realize that these are simply linear equalities. Note that the essential matrix can be only computed up to scale since we can multiply (14.17) by an arbitrary constant without altering the equality. Recalling that $\boldsymbol{E} = [\boldsymbol{t}]_\times \boldsymbol{R}$ this means that we can apply an arbitrary scaling to the vector $\boldsymbol{t}$ without altering the epipolar constraint. This is consistent with the geometric intuition that from a set of camera images we are not able to resolve the scale of the scene, i.e., we are not able to understand if we are observing a small scene from close distance of a large scene from a far distance.

Therefore, it is customary to assume $\|\boldsymbol{t}\| = 1$, which means we only try to estimate the direction of the translation rather than it's scale.

Eight-point algorithm. In absence of noise, we can compute the essential matrix by solving a linear system. Rearranging the entries of $\boldsymbol{E}$ in a vector $\boldsymbol{e} \in \mathbb{R}^9$, the set of linear equations (14.17) can be written as:

$$\boldsymbol{a}_k^\mathsf{T} \boldsymbol{e} = 0 \qquad k = 1, \dots, N \tag{14.17}$$

where $\boldsymbol{a}_k$ are known vectors whose entries are only function of the pixel correspondences $(\tilde{\boldsymbol{y}}_{1,k}, \tilde{\boldsymbol{y}}_{2,k})$.

Stacking the vectors $\boldsymbol{a}_k^\mathsf{T}$ as rows of a matrix $\boldsymbol{A}$, the linear equations

$$\boldsymbol{A}\boldsymbol{e} = 0 \tag{14.18}$$

For this linear system to admit a unique solution, $\boldsymbol{A} \in \mathbb{R}^{N \times 9}$ should have rank 8, therefore we need $N = 8$ point correspondences to compute the essential matrix using the linear system (14.18).

By solving the linear system (14.18) and re-arranging the entries of $\boldsymbol{e}$ into a 3×3 matrix, we obtain the desired essential matrix.

Note that since $\boldsymbol{A}\boldsymbol{e} = -\boldsymbol{A}\boldsymbol{e} = 0$ both $\boldsymbol{E}$ and $-\boldsymbol{E}$ are valid solutions to the linear system (14.18), so we need to consider both as potential essential matrices (we resolve this ambiguity in Section 14.2.2).

Noisy pixel measurements. Since the pixels measurements are typically affected by noise, the solution of the linear system may not be an essential matrix. Therefore, it is common to project the solution onto the essential space using (14.15).

How many correspondences do we really need to estimate $\boldsymbol{E}$? The essential matrix $\boldsymbol{E} = [\boldsymbol{t}]_\times \boldsymbol{R}$ is fully defined by a unit vector $\boldsymbol{t}$ (recall that we imposed $\|\boldsymbol{t}\| = 1$) and a rotation $\boldsymbol{R}$. Therefore, it only has 5 degrees of freedom. Since each correspondence adds a single linear equation, we conclude that we need at least 5 points to estimate the essential matrix.

The eight-point algorithm uses more since it does not leverage the structure of the essential matrix (i.e., we first estimate a generic 3×3 matrix and then we project to the essential space).

Related work indeed provides 7-point, 6-point, and 5-point algorithms. The 5-point algorithm, developed by Nister in [2], is a *minimal* solver.

14.2.2 Retrieve Pose (up to scale) from the Essential Matrix

Theorem 1 (Pose recovery from essential matrix, Thm 5.7 in [1])**.** *There exist exactly two relative poses $(\boldsymbol{R}, \boldsymbol{t})$ with $\boldsymbol{R} \in \mathrm{SO}(3)$ and $\boldsymbol{t} \in \mathbb{R}^3$ corresponding to a nonzero essential matrix $\boldsymbol{E}$ (i.e., such that $\boldsymbol{E} = [\boldsymbol{t}]_\times \boldsymbol{R}$):*

$$\boldsymbol{t}_1 = \boldsymbol{U}\boldsymbol{R}_z(+\pi/2)\boldsymbol{\Sigma}\boldsymbol{U}^\mathsf{T} \qquad \boldsymbol{R}_1 = \boldsymbol{U}\boldsymbol{R}_z(+\pi/2)\boldsymbol{V}^\mathsf{T} \tag{14.19}$$

$$\boldsymbol{t}_2 = \boldsymbol{U}\boldsymbol{R}_z(-\pi/2)\boldsymbol{\Sigma}\boldsymbol{U}^\mathsf{T} \qquad \boldsymbol{R}_2 = \boldsymbol{U}\boldsymbol{R}_z(-\pi/2)\boldsymbol{V}^\mathsf{T} \tag{14.20}$$

where $\boldsymbol{E} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\mathsf{T}$ is the singular value decomposition of the matrix $\boldsymbol{E}$, and $\boldsymbol{R}_z(+\pi/2)$ is an elementary rotation around the z-axis of an angle $\pi/2$.

Using the matrices $\boldsymbol{E}$ and $-\boldsymbol{E}$ we computed in the previous section, we can use Theorem 1 to compute the corresponding pose estimates. Since each essential matrix leads to 2 potential poses, we end up with 4 alternatives. Out of these 4 potential poses, we can find the correct one as follows:

- estimate the position of the 3D points producing the correspondences: all points must satisfy (14.5):

  $$d_2 \tilde{\boldsymbol{y}}_2 = d_1 \boldsymbol{R} \tilde{\boldsymbol{y}}_1 + \boldsymbol{t}$$

  from which we can compute d_1 and d_2. Note that due to the scale ambiguity (we assumed $\|\boldsymbol{t}\| = 1$) d_1 and d_2 may be a "scaled" version of the true distance from the cameras to the point.
- select the pose for which the reconstructed 3D points are in front of both cameras (the so-called *cheirality* constraint).

References

[1] Y. Ma, S. Soatto, J. Kosecka, and S.S. Sastry. *An Invitation to 3-D Vision*. Springer, 2004.

[2] D. Nistér. An efficient solution to the five-point relative pose problem. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2003.