TTK4255: Robotic Vision

Homework 8: Two-view geometry

Originally created by Simen Haugo, and modified by Mau Hing Yip.

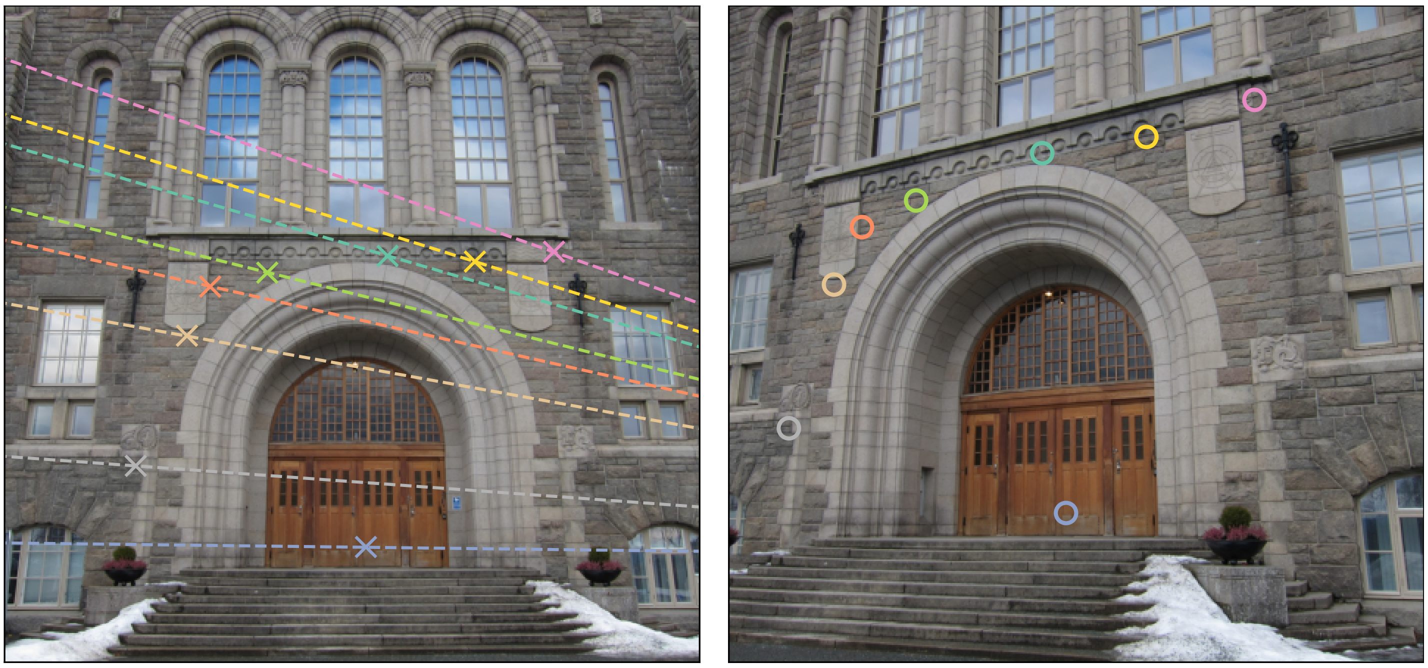

Figure 1: Visualization of the epipolar constraint. For any point in one image, the corresponding point in the other image lies on the associated epipolar line. Here the right image's camera is in front of the left image's camera. The epipolar lines intersect at the projection of the right camera's origin into the left image (the left epipole). The epipoles are not necessarily within the image bounds.

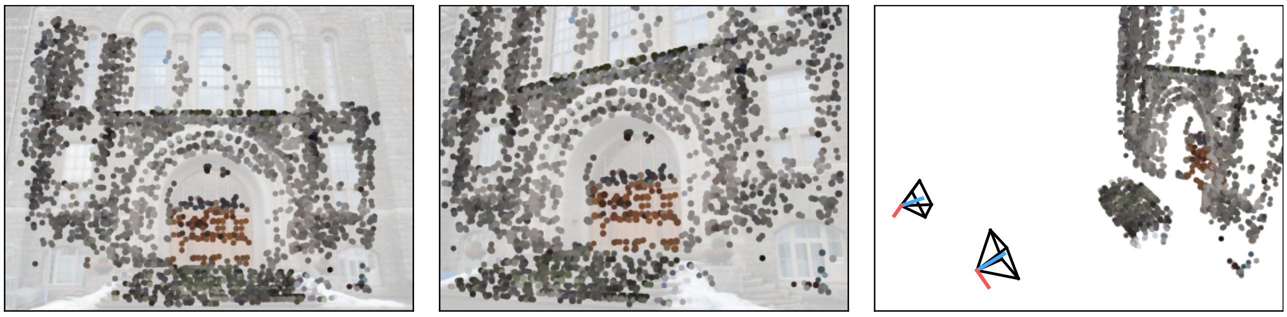

Figure 2: 3D reconstruction obtained with the methods in this assignment, shown from the two original viewpoints and a novel viewpoint.

Instructions

First make sure to read `about.pdf` in the "Course work" page on Blackboard. To get your assignment approved, you need to complete any 60%. Upload the requested answers and figures as a single PDF. You may collaborate with other students and submit the same report, but you still need to upload individually on Blackboard. Please write your collaborators' names on your report's front page. If you want detailed feedback, please indicate so on the front page.

About the assignment

From the projection of a 3D point in a single image we can only infer that the 3D point must lie somewhere along the associated back-projection ray. With its projection in a second image, we can estimate the point's 3D coordinates via triangulation, if we know the rigid transformation between the camera frames of the two images (also known as the *relative pose* or *camera motion*).

Surprisingly, we can estimate (up to a scaling factor) both the associated 3D point coordinates and the relative pose from a set of 2D–2D point correspondences relating a single pair of images. This is provably so by the existence of algorithms that can produce an exact solution for the relative pose (absent noise), without knowledge of the 3D point coordinates, such as Longuet-Higgins' "8-point algorithm" and Nistér's "5-point algorithm". The 3D point coordinates can then be estimated via triangulation.

These algorithms require point correspondences between the images. The problem of finding these is not covered by this assignment. Nonetheless, this step is likely to introduce a significant number of "bad correspondences", i.e. correspondences that are better left out of the estimation process. Most reconstruction algorithms are therefore used with a robust estimation strategy, such as RANSAC.

In this assignment, you will explore the epipolar constraint for perspective cameras, the 8-point relative pose algorithm, and RANSAC. At the end you will have a two-view 3D reconstruction pipeline that can handle outliers, and which can provide a reasonable initialization for an iterative optimal method (i.e. bundle adjustment).

Recommended literature

For the concepts needed to solve this assignment, any of the following can be helpful:

- Hartley and Zisserman:
  - §9: Epipolar geometry and the fundamental matrix
  - §10: 3D reconstruction of cameras and structure (overview of 3D reconstruction)
  - §11: Computation of the fundamental matrix (including 8-point algorithm and RANSAC)
  - §12: Structure computation (triangulation)
- Szeliski:
  - §8.1.4 and §8.5: RANSAC
  - §11.2.4: Triangulation
  - §11.3.1: Eight, seven, and five-point algorithms
- Förstner and Wrobel:
  - §4.7.7: RANSAC
  - §13: Geometry and orientation of the image pair

Beyond the assignment

From here, you can explore further topics, perhaps as part of your semester project:

- Implement (sparse) bundle adjustment.
- Collect your own point correspondences using e.g. RootSIFT, or a "learned" method.
- Compare modern RANSAC variants.
- Visual localization: Use the 3D points and associated descriptors to estimate the camera pose in new images.
- Incremental SfM: Incorporate point correspondences with additional images, e.g. using a strategy similar to COLMAP.

See L5: Optimal methods, L8: Initialization-free methods, L9: Robust estimation, and L10: Feature matching, for references regarding these topics, including links to implementations.

Keep in mind that the semester project is optional.

An archive containing additional images of the same scene, with the same camera, is provided on Blackboard ("data_ext.zip").

Matrix form of the cross product operator

It may be useful to know that the cross product between two vectors can be written as a matrix-vector product. We define this as an operator $[\mathbf{X}]_\times$, which converts $\mathbf{X}$ into a 3×3 matrix,

$$[\mathbf{X}]_\times = \begin{bmatrix} 0 & -Z & Y \\ Z & 0 & -X \\ -Y & X & 0 \end{bmatrix}. \tag{1}$$

This matrix has the property that multiplying it with another vector gives the cross product between $\mathbf{X}$ and this vector, e.g.

$$[\mathbf{X}]_\times \mathbf{X}' = \mathbf{X} \times \mathbf{X}'.$$

Homogeneous representation of 2D lines

The tasks in Part 1 require some knowledge of homogeneous representations of lines. Recall that the equation for a line can be written as $x \cos\theta + y \sin\theta = \rho$. This equation can equivalently be written using the dot product between two 3-vectors, requiring it to be zero:

$$\begin{bmatrix} x & y & 1 \end{bmatrix} \begin{bmatrix} \cos\theta \\ \sin\theta \\ -\rho \end{bmatrix} = 0. \tag{2}$$

As you will prove in Part 1, we can substitute either vector in this equation with any non-zero scalar multiple of it, and the resulting equation would still define the same line in Euclidean space. This motivates the definition of the homogeneous representation of a 2D line as a vector $\tilde{\boldsymbol{l}} = (a, b, c)$, with the corresponding line equation

$$\tilde{\mathbf{x}}^\mathsf{T} \tilde{\boldsymbol{l}} = a\tilde{x} + b\tilde{y} + c\tilde{z} = 0. \tag{3}$$

$\boxed{\text{Part 1}}$ The epipolar constraint (20%)

Consider a set of 2D–2D correspondences $\{\mathbf{u}_{1,j} \leftrightarrow \mathbf{u}_{2,j}\}_{j=1...n}$, related by being perspective projections from two viewpoints of a rigid set of 3D points:

$$\tilde{\mathbf{u}}_{1,j} = \mathbf{K}_1 \begin{bmatrix} \mathbf{I}_{3\times3} & \mathbf{0} \end{bmatrix} \mathbf{T}^1_{\text{w}} \tilde{\mathbf{X}}^{\text{w}}_j, \tag{4}$$

$$\tilde{\mathbf{u}}_{2,j} = \mathbf{K}_2 \begin{bmatrix} \mathbf{I}_{3\times3} & \mathbf{0} \end{bmatrix} \mathbf{T}^2_{\text{w}} \tilde{\mathbf{X}}^{\text{w}}_j. \tag{5}$$

For notational simplicity,

$$\mathbf{T}^2_1 = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix}.$$

The “8-point algorithm” can estimate the relative pose: the transformation between the camera frames (e.g. $\mathbf{R}$ and $\mathbf{t}$) from at least 8 such correspondences. Its basis is the epipolar constraint, which for a perspective camera model states that there exists a 3×3 matrix $\mathbf{F}$ such that

$$\tilde{\mathbf{u}}^{\mathsf{T}}_{2,j} \mathbf{F} \tilde{\mathbf{u}}_{1,j} = 0, \tag{6}$$

for all $j = 1...n$. The matrix $\mathbf{F}$ is called the fundamental matrix, and (as you will show in this part) is related to the relative pose and the camera intrinsics by

$$\mathbf{F} = \mathbf{K}_2^{-\mathsf{T}} [\mathbf{t}]_\times \mathbf{R} \mathbf{K}_1^{-1}. \tag{7}$$

Each correspondence gives one epipolar constraint (which is a scalar equation), and all constraints involve the same fundamental matrix. These constraints thereby provide a path to estimate the relative pose, without the need to jointly estimate the 3D point coordinates.

The goal of this part is to give you a hopefully satisfying understanding of where the epipolar constraint comes from, and how you can interpret it. The remaining parts use a different formulation of this constraint, which holds not only for perspective camera models, but for any central camera model for which the back-projection is defined. The interpretation in terms of epipolar lines, derived in the following tasks, is not applicable to this more general formulation, but a similar interpretation can be derived in terms of epipolar curves.

Task 1.1: (4%) The tasks in this part require some knowledge of homogeneous representations of lines. First, read the introduction on the previous page. Let $\tilde{\boldsymbol{l}} = (a, b, c)$ be the homogeneous representation of a 2D line.

(a) Prove that any non-zero scalar multiple of $\tilde{\boldsymbol{l}}$ represents the same line.

(b) Show how you can normalize $\tilde{\boldsymbol{l}}$ into the form $(\cos\theta, \sin\theta, -\rho)$, where θ and ρ are the line parameters in a normal form, with the line equation $x\cos\theta + y\sin\theta = \rho$.

Task 1.2: (4%) Besides letting us write the line equation concisely with the dot product, the use of homogeneous representations lets us construct the line connecting two points with the cross product. Show that for any pair of points $\mathbf{x}_a$ and $\mathbf{x}_b$, the line passing through the points can be represented by

$$\tilde{\boldsymbol{l}} = \tilde{\mathbf{x}}_a \times \tilde{\mathbf{x}}_b. \tag{8}$$

Task 1.3: (6%) Consider a single correspondence $\mathbf{u}_1 \leftrightarrow \mathbf{u}_2$ (omitting the subscript j for simplicity). Assuming noise-free correspondences, the common 3D point is located where the back-projection rays intersect, which is some unknown distance along either back-projection ray.

An arbitrary point along the back-projection ray of $\mathbf{u}_1$ has coordinates in camera frame 1 equal to $\mathbf{X}^1(\lambda) = \lambda \mathbf{B}_1^1$, for any $\lambda > 0$, and where $\mathbf{B}_1^1 = \mathbf{K}_1^{-1}\tilde{\mathbf{u}}_1$. Combining this with the rigidity assumption, this point's coordinates in camera frame 2 are then

$$\begin{aligned}\mathbf{X}^2(\lambda) &= \mathbf{R}\mathbf{X}^1(\lambda) + \mathbf{t} && (9)\\ &= \lambda \mathbf{R}\mathbf{K}_1^{-1}\tilde{\mathbf{u}}_1 + \mathbf{t}. && (10)\end{aligned}$$

The image coordinates $\mathbf{u}'$ in image 2 of this point are given by

$$\begin{aligned}\tilde{\mathbf{u}}'(\lambda) &= \mathbf{K}_2\mathbf{X}^2(\lambda) && (11)\\ &= \mathbf{K}_2\left(\lambda \mathbf{R}\mathbf{K}_1^{-1}\tilde{\mathbf{u}}_1 + \mathbf{t}\right). && (12)\end{aligned}$$

Note that $\mathbf{u}'(\lambda) = \mathbf{u}_2$ only for one specific λ. Find the limit value of the inhomogeneous $\mathbf{u}'(\lambda)$ when $\lambda \to 0$ and when $\lambda \to \infty$. Give an interpretation of what both limit values represent, possibly using the figure on the front page.

Task 1.4: (6%) Because the perspective camera model preserves straight lines, and because a back-projection ray defines a (half-)line, the values for $\mathbf{u}'$ obtained by sweeping λ from 0 to ∞ forms a line in image 2. Find an expression for this line's homogeneous representation and relate your answer to the epipolar constraint and the fundamental matrix $\mathbf{F}$.

Tip: Define $\tilde{\mathbf{x}}'(\lambda) := \mathbf{K}_2^{-1}\tilde{\mathbf{u}}'(\lambda)$, and first find the line $\tilde{\boldsymbol{l}}$ passing through $\tilde{\mathbf{x}}'(0)$ and $\tilde{\mathbf{x}}'(\infty)$. The line passing through $\tilde{\mathbf{u}}'(0)$ and $\tilde{\mathbf{u}}'(\infty)$ is then given by $\mathbf{K}_2^{-\top}\tilde{\boldsymbol{l}}$.

In the remaining parts we omit the subscript j counting the index in the set of correspondences, and instead present equations for just a single correspondence $\mathbf{u}_1 \leftrightarrow \mathbf{u}_2$ among the n correspondences.

$\boxed{\text{Part 2}}$ The 8-point algorithm (20%)

For the perspective camera model, we have that $\tilde{\mathbf{u}}_i = \mathbf{K}_i\mathbf{B}_i^i$, where $\mathbf{B}_i^i$ is a back-projection vector associated with $\mathbf{u}_i$. Inserting this into the epipolar constraint, we can express the constraint in terms of the back-projection vectors instead of image coordinates:

$$
\begin{aligned}
\tilde{\mathbf{u}}_2^\mathsf{T}\mathbf{F}\tilde{\mathbf{u}}_1 &= 0, & (13)\\
\Leftrightarrow (\mathbf{K}_2\mathbf{B}_2^2)^\mathsf{T}\mathbf{F}(\mathbf{K}_1\mathbf{B}_1^1) &= 0, & (14)\\
\Leftrightarrow (\mathbf{B}_2^2)^\mathsf{T}(\mathbf{K}_2^\mathsf{T}\mathbf{F}\mathbf{K}_1)\mathbf{B}_1^1 &= 0, & (15)\\
\Leftrightarrow (\mathbf{B}_2^2)^\mathsf{T}\mathbf{E}\mathbf{B}_1^1 &= 0. & (16)
\end{aligned}
$$

Note that the constraint holds if either back-projection vector is multiplied by a scalar; the right-hand side is still equal to zero. Thus, it is not necessary to ensure that these are of unit length. The matrix $\mathbf{K}_2^\mathsf{T}\mathbf{F}\mathbf{K}_1 = \mathbf{E}$ is called the essential matrix, and is related to the relative pose by

$$\mathbf{E} = [\mathbf{t}]_\times\mathbf{R}. \tag{17}$$

The constraint in Eq. (16), with the same definition of $\mathbf{E}$, holds for any central camera model for which back-projection is defined. In contrast, the constraint in Eq. (13) involving image coordinates is specific to straight-line-preserving camera models, and thus the perspective camera model (although generalizations have been proposed). Both are called the "epipolar constraint", but this one is more general. It does however assume that the intrinsics are known so as to be able to compute the back-projection vectors from the observed image coordinates.

The 8-point algorithm, described in the following, was originally proposed to estimate the essential matrix $\mathbf{E}$, but the same algorithm can in fact be used to estimate the fundamental matrix $\mathbf{F}$.

An algorithm to estimate $\mathbf{E}$ arises naturally by observing that the epipolar constraint is *linear* in the entries of $\mathbf{E}$. Therefore, if we label the entries of $\mathbf{E}$ as

$$\mathbf{E} = \begin{bmatrix} E_{11} & E_{12} & E_{13} \\ E_{21} & E_{22} & E_{23} \\ E_{31} & E_{32} & E_{33} \end{bmatrix}, \tag{18}$$

and we reshape the entries into a vector $\mathbf{e}$,

$$\mathbf{e} = \begin{bmatrix} E_{11} & E_{12} & E_{13} & E_{21} & E_{22} & E_{23} & E_{31} & E_{32} & E_{33} \end{bmatrix}^\mathsf{T}, \tag{19}$$

then the epipolar constraint for a single point correspondence can be written as

$$\begin{bmatrix} B_2B_1 & B_2C_1 & B_2D_1 & C_2B_1 & C_2C_1 & C_2D_1 & D_2B_1 & D_2C_1 & D_2D_1 \end{bmatrix}\mathbf{e} = 0, \tag{20}$$

where we have named the entries of $\mathbf{B}_i^i = (B_i, C_i, D_i)$, omitting the superscript for simplicity.

Each correspondence gives one linear equation. Stacking the equations from n correspondences gives a linear system $\mathbf{Ae} = \mathbf{0}$, where $\mathbf{A}$ is an $n \times 9$ matrix. Because the right-hand side is all zero, any scalar multiple of a solution to the linear system is also a solution. This reflects our inability to recover the absolute scale of the camera translation, and therefore also the absolute scale of the scene.

$\mathbf{A}$ must have a one-dimensional null-space for there to be a unique (up to scale) non-trivial solution. This can be achieved with at least eight correspondences. As usual, if the system is over-determined ($n > 8$), there may not be an exact solution. As in Homework 7, a solution minimizing the algebraic error subject to a scale constraint can be obtained from the singular value decomposition of $\mathbf{A}$.

The 8-point algorithm takes its name from the minimum number of correspondences required for $\mathbf{Ae} = \mathbf{0}$ to have a well-determined solution, but it can be used with more than eight correspondences to increase the accuracy, as you will do here. Note also that $\mathbf{E}$ can be estimated with fewer than eight correspondences (at least five), but that requires a different algorithm.

Task 2.1: (20%) Implement the 8-point algorithm in the provided stub function `estimate_E`.

The provided `main` script loads an image pair captured with the same camera, and a set of good point correspondences between the images. The images have been corrected for lens distortion so as to satisfy a perspective camera model, with the provided intrinsic matrix. Modify the script to estimate the essential matrix. Convert the estimated essential matrix into a fundamental matrix and uncomment the line calling the `draw_correspondences` function. This should generate a figure of the resulting epipolar lines in both images, for a random subset of the points. Include this figure in your report and comment on your results; are they what you expect?

Note (not required to complete the task): An essential matrix is not an arbitrary 3×3 matrix; indeed the existence of Nistér's five-point algorithm implies that the relationship that is represented by an essential matrix has only five degrees of freedom. Hence, there must be additional constraints between the nine entries of $\mathbf{E}$. These constraints are that one of the singular values of $\mathbf{E}$ is zero and the remaining two are equal. Nowhere in the algorithm did we impose these constraints. As such, there is no guarantee that we get a valid essential matrix. A commonly suggested solution (c.f. Hartley and Zisserman §11.7.3, and Förstner and Wrobel §13.3.2.3) is to compute the SVD of the estimated $\mathbf{E} = \mathbf{UDV}^\mathsf{T}$, and replace the estimated $\mathbf{E}$ by the matrix $\mathbf{U}\mathtt{diag}(1, 1, 0)\mathbf{V}^\mathsf{T}$. A slightly different correction can be done if the algorithm is used to estimate the fundamental matrix (c.f. Hartley and Zisserman §11.1.1, and Förstner and Wrobel §13.3.2.1).

$\boxed{\text{Part 3}}$ Triangulation (40%)

The essential matrix $\mathbf{E}$ can be decomposed into a relative pose that is consistent with the epipolar constraints (in fact, there are four possible decompositions; see e.g. Szeliski §11.3.1 and Task 3.2). The magnitude of the translation will be ambiguous if $\mathbf{E}$ is estimated as in the previous part. This is not a limitation of the decomposition step, but is instead because the scale of $\mathbf{E}$ itself cannot be uniquely determined from 2D–2D correspondences (why?).

Given a relative pose, the 3D coordinates of the common scene point of any 2D–2D correspondence can be estimated via triangulation. With noise-free correspondences, this is simply the intersection of the two back-projection rays. In practice, these rays will not intersect and we instead seek an estimate that is optimal by some criterion. Such an estimate can be obtained by iteratively minimizing weighted reprojection errors, with respect to both the relative pose and the 3D point coordinates; i.e. bundle adjustment. However, this requires an initial guess.

A linear algorithm for triangulation, which does not require an initial guess, can be derived as follows. Consider a single 2D–2D correspondence $\mathbf{u}_1 \leftrightarrow \mathbf{u}_2$ satisfying

$$\begin{aligned}\tilde{\mathbf{u}}_1 &= \mathbf{K}_1 \begin{bmatrix}\mathbf{I}_{3\times 3} & \mathbf{0}\end{bmatrix} \mathbf{T}^1_{\mathrm{w}} \tilde{\mathbf{X}}^{\mathrm{w}} = \mathbf{P}_1 \tilde{\mathbf{X}}^{\mathrm{w}},\\ \tilde{\mathbf{u}}_2 &= \mathbf{K}_2 \begin{bmatrix}\mathbf{I}_{3\times 3} & \mathbf{0}\end{bmatrix} \mathbf{T}^2_{\mathrm{w}} \tilde{\mathbf{X}}^{\mathrm{w}} = \mathbf{P}_2 \tilde{\mathbf{X}}^{\mathrm{w}},\end{aligned}$$

where $\mathbf{X}^{\mathrm{w}}$ are the unknown coordinates, and $\mathbf{P}_i = \mathbf{K}_i[\mathbf{I}_{3\times 3}\ \ \mathbf{0}]\mathbf{T}^i_{\mathrm{w}}$ are 3×4 "projection matrices". The image coordinates that we measure are the dehomogenized coordinates,

$$u_i = \frac{\mathbf{p}_{i,1}\tilde{\mathbf{X}}^{\mathrm{w}}}{\mathbf{p}_{i,3}\tilde{\mathbf{X}}^{\mathrm{w}}} \quad \text{and} \quad v_i = \frac{\mathbf{p}_{i,2}\tilde{\mathbf{X}}^{\mathrm{w}}}{\mathbf{p}_{i,3}\tilde{\mathbf{X}}^{\mathrm{w}}}, \tag{21}$$

where $\mathbf{p}_{i,1}, \mathbf{p}_{i,2}, \mathbf{p}_{i,3}$ are the three rows of $\mathbf{P}_i$, each being a 4D row vector. This gives a total of four non-linear equations. Multiplying each equation by its denominator and rearranging, we can write these as a linear system,

$$\mathbf{A}\tilde{\mathbf{X}}^{\mathrm{w}} = \begin{bmatrix} u_1\mathbf{p}_{1,3} - \mathbf{p}_{1,1} \\ v_1\mathbf{p}_{1,3} - \mathbf{p}_{1,2} \\ u_2\mathbf{p}_{2,3} - \mathbf{p}_{2,1} \\ v_2\mathbf{p}_{2,3} - \mathbf{p}_{2,2} \end{bmatrix} \begin{bmatrix} \tilde{X}^{\mathrm{w}} \\ \tilde{Y}^{\mathrm{w}} \\ \tilde{Z}^{\mathrm{w}} \\ \tilde{W}^{\mathrm{w}} \end{bmatrix} = 0, \tag{22}$$

for which there is an exact, unique (up to scale), non-trivial solution $\tilde{\mathbf{X}}^{\mathrm{w}}$, that can be obtained via the SVD of $\mathbf{A}$. This gives a homogeneous representation of the point. The Cartesian coordinates can be recovered by dehomogenizing.

Task 3.1: (10%) Points approaching an infinite distance away from the camera origin are better represented in homogeneous coordinates with $\tilde{W} \to 0$. This avoids issues that would otherwise occur when storing and performing floating point arithmetic on large numbers. However, is the $\tilde{W}$ component of the SVD solution necessarily close to zero if the point happens to be far away?

Task 3.2: (30%) Implement `triangulate_many`. The function should take all the correspondences, the two projection matrices, and return the dehomogenized 3D point coordinates. You can test your implementation by running the `test_triangulate` script. Uncomment the code that plots the 3D point cloud and include a figure from some viewpoint.

An implementation of the decomposition algorithm described in Szeliski §11.3.1 is in the hand-out code (`decompose_E`). Use it to extract the potential relative poses from $\mathbf{E}$. As explained in the book, there are always four valid decompositions of a given essential matrix. The correct one can be identified by requiring that the triangulated points that result from it are in front of **both** cameras.

There is no guarantee that the $\tilde{W}$ component of a homogeneous coordinate vector is positive, so it is not guaranteed that $\tilde{Z} > 0$ implies $Z > 0$. This may be relevant when testing if the points are in front of the cameras.

Without further constraints, we cannot estimate the rigid transformations between the camera frames and the world frame. Instead, the world frame is simply undefined. We can arbitrarily define it to coincide with the camera frame of image 1. The projection matrices are then $\mathbf{P}_1 = \mathbf{K}_1[\mathbf{I}_{3\times3} \;\; \mathbf{0}]$ and $\mathbf{P}_2 = \mathbf{K}_2[\mathbf{R} \;\; \mathbf{t}]$, where $[\mathbf{R} \;\; \mathbf{t}]$ is one of the relative poses extracted from $\mathbf{E}$.

The document continues on the next page $\rightarrow$

Part 4 Robust estimation with RANSAC (20%)

Point correspondences often contain outliers — misinformative measurements that should be excluded. RANSAC is a general strategy that can be used to simultaneously estimate the essential matrix and estimate its inliers (given an error threshold that presumably separates inliers from outliers):

1. In each iteration of a loop ($=$ a single "trial"), randomly select a sample of m correspondences.
   1.1. Estimate the essential matrix using only the m correspondences.
   1.2. Compute a vector of errors for the full set of correspondences.
   1.3. Count the number of errors within a specified threshold (the inlier count).
2. Return the essential matrix and the associated inlier set that had the highest inlier count.

The essential matrix is typically re-estimated using the full inlier set at the end. If the 8-point algorithm is used, $m = 8$ is a natural choice for the sample size in the loop.

For this part you should use the outlier-containing correspondences in `task4matches.txt`.

Task 4.1: (5%) The errors should quantify how well the correspondences fit an essential matrix. One error definition uses the signed distance between each point and its epipolar line, in the two images,

$$e_2 = \frac{\tilde{\mathbf{u}}_2^\top \mathbf{F} \tilde{\mathbf{u}}_1}{\tilde{w}_1 \tilde{w}_2 \sqrt{a_1^2 + b_1^2}} \quad \text{and} \quad e_1 = \frac{\tilde{\mathbf{u}}_1^\top \mathbf{F}^\top \tilde{\mathbf{u}}_2}{\tilde{w}_1 \tilde{w}_2 \sqrt{a_2^2 + b_2^2}}. \tag{23}$$

The numerators are actually identical. The denominator ensures that the distance is measured in pixel units in the respective image. Specifically (a_1, b_1) are the first two components of $\mathbf{F}\tilde{\mathbf{u}}_1$, and likewise for (a_2, b_2) and $\mathbf{F}^\top \tilde{\mathbf{u}}_2$. The two errors e_1 and e_2 can be averaged to get one error per correspondence. Note that this error can be negative, so you must compare the absolute value $|e_1 + e_2|/2$ against the error threshold. Implement `epipolar_distances` based on this definition. Include a histogram of the errors computed using the essential matrix from Part 2 and the correspondences in `task4matches.txt`.

Task 4.2: (5%) Implement RANSAC in `estimate_E_ransac`. Run 2000 iterations with an inlier threshold of 4 pixels. Modify your script to triangulate only the inliers of `task4matches.txt`. Include the generated figures and the size of the inlier set.

Task 4.3: (5%) You may get a few points that appear to not be attached to any real surface, and should clearly not be part of the reconstruction. Why do these arise and how can you get rid of them? (This question may be trickier than you think.)

Task 4.4: (5%) Eq. (8.30) in Szeliski §8.1.4 estimates the required number of trials, given a desired probability of success (the *confidence*) and the inlier fraction. The inlier fraction here is 0.50. Try running RANSAC for the number of trials produced by the formula with a confidence of 0.99. Try this a few times. Do you expect it to find the largest inlier set 99% of the time? Why/why not?

(In practice we don't know the inlier fraction. This data was generated by artifically adding outliers, so we therefore know the inlier fraction.)