TTK4255: Robotic Vision

Homework 4: Linear algorithms

Instructions

For general information about the assignments, including grading critera and how to get help, consult the assignments.pdf document on BB. To get your assignment approved, you only need to complete 60% (weighting is next to each task). Upload the requested answers and figures as a single PDF. You don't need to submit your code. You may use any convenient tool to create your report.

About the assignment

Several estimation algorithms in computer vision are called "linear algorithms". These are typically computationally cheap and can provide an estimate of the quantity of interest without an initial guess. On the other hand, the estimate is usually not optimal in a geometrically meaningful sense, and may therefore be suboptimal in the presence of noise. Because of this, linear algorithms are most often used to obtain a rough initial estimate, possibly in combination with RANSAC, which is then further refined by non-linear optimization of a geometric and/or probabilistic objective function.

In this assignment you will learn how to derive linear algorithms using the direct linear transform, and implement an algorithm to estimate the pose of a planar object from 2D-3D point correspondences. As a simple test case, you will apply the algorithm to a paper sheet with fiducial markers. These markers have known positions on the paper and are designed to be easily detected and uniquely identified, which simplifies the problem of establishing correspondences.

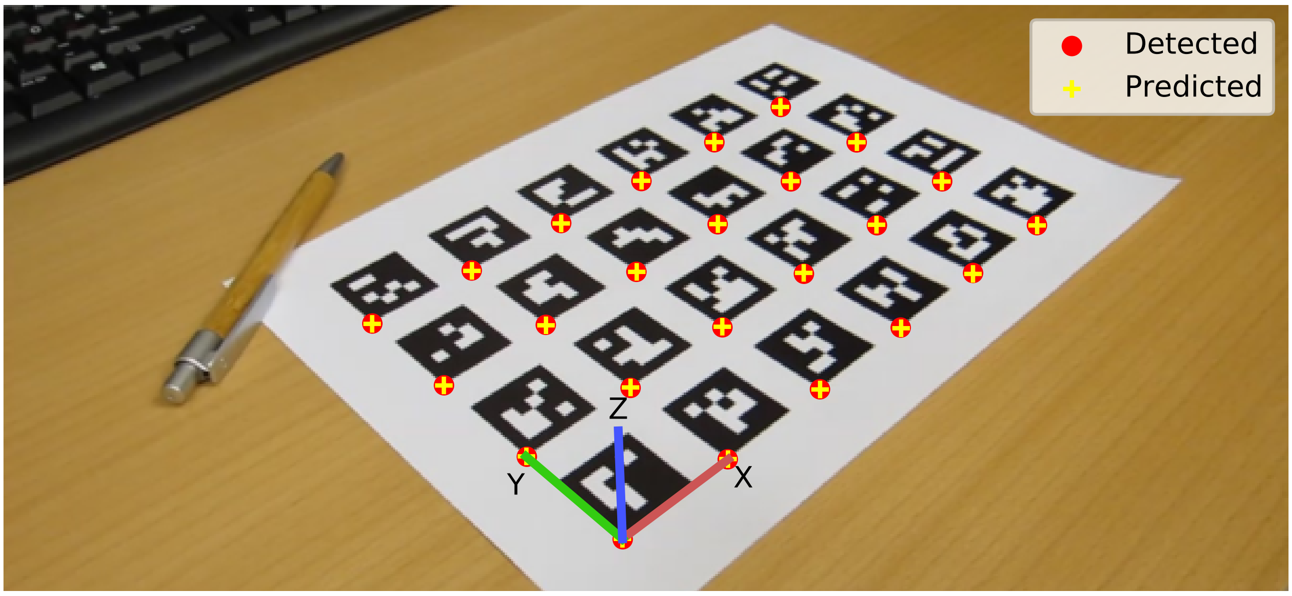

Figure 1: Output from the algorithm you will implement, showing the estimated object coordinate frame, along with the detected and predicted location of the markers.

Background

Linear algorithms appear several places in Szeliski (2010), including 6.1 (Feature-based alignment), 7.1 (Triangulation) and 7.2 (Two-frame structure from motion). The pose estimation algorithm in this assignment is not presented in Szeliski, however its derivation and implementation is similar to other linear algorithms in the book. For the curious, the algorithm here is based on the camera calibration paper by Zhengyou Zhang, which has had a heavy influence on the development of calibration software:

- Zhengyou Zhang. A Flexible New Technique for Camera Calibration. 2000. (link)

Linear algorithms are based on solving linear systems of equations. A good overview of this topic can be found in Appendix 5 (Least-squares minimization) of Multiple View Geometry by Hartley and Zisserman, which is available on Blackboard.

The first step of the pose estimation algorithm is to estimate a homography between the object plane and the image. Homographies (or projective transformations) are briefly introduced in Szeliski 2.1.2. A more detailed treatment can be found in Hartley and Zisserman, but is not necessary for completing this assignment. In summary, a homography is a mapping between two (homogeneous) 2D coordinates:

$$\tilde{x}' = \mathbf{H}\tilde{x} \tag{1}$$

where $\mathbf{H}$ is an arbitrary 3×3 matrix. Besides the perspective image of a planar object, some other transformations that can be described by a homography is the mapping between two images of a planar scene and between two images taken by a rotating camera.

$\boxed{\text{Part 1}}$ Transformation between a plane and its image (10%)

Consider a perspective image of a planar object. Without loss of generality, let points on the object have coordinates of the form $\mathbf{X} = (X, Y, 0)$, arbitrarily placing the object plane at $Z = 0$. The pixel coordinates of a given point on the object are

$$u = c_x + s_x f X^c / Z^c \tag{2}$$
$$v = c_y + s_y f Y^c / Z^c \tag{3}$$

where

$$\begin{bmatrix} X^c \\ Y^c \\ Z^c \end{bmatrix} = \underbrace{\begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}}_{\mathbf{R}} \begin{bmatrix} X \\ Y \\ 0 \end{bmatrix} + \underbrace{\begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix}}_{\mathbf{t}} \tag{4}$$

is the point transformed from object to camera coordinates by $\mathbf{R}$ and $\mathbf{t}$—which we refer to as the "pose". For convenience, we define the *calibrated image coordinates* $\boldsymbol{x} = (x, y)$ as

$$x := (u - c_x)/s_x f \tag{5}$$
$$y := (v - c_y)/s_y f \tag{6}$$

or generally $\tilde{\boldsymbol{x}} = \mathbf{K}^{-1}\tilde{\mathbf{u}}$ for an arbitrary intrinsic matrix $\mathbf{K}$, which can be thought of as "camera-agnostic" coordinates. Combining the above, we find the following non-linear relationship between points on the object (X, Y) and calibrated image coordinates (x, y):

$$x = \frac{X^c}{Z^c} = \frac{r_{11}X + r_{12}Y + t_x}{r_{31}X + r_{32}Y + t_z}, \tag{7}$$
$$y = \frac{Y^c}{Z^c} = \frac{r_{21}X + r_{22}Y + t_y}{r_{31}X + r_{32}Y + t_z}. \tag{8}$$

If we use homogeneous coordinates, this relationship can be written in the linear form

$$\begin{bmatrix} \tilde{x} \\ \tilde{y} \\ \tilde{z} \end{bmatrix} = \mathbf{H} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} \quad \text{where} \quad \mathbf{H} = \begin{bmatrix} r_{11} & r_{12} & t_x \\ r_{21} & r_{22} & t_y \\ r_{31} & r_{32} & t_z \end{bmatrix} \tag{9}$$

with $\tilde{\boldsymbol{x}} = (\tilde{x}, \tilde{y}, \tilde{z})$ being the homogeneous form of (x, y).

Task 1.1: (5%) We say that a homography is "unique up to a scaling factor" because any non-zero scalar multiple of the homography matrix defines the same mapping between the 2D coordinates (after dehomogenization). Show that this is the case for the matrix $\mathbf{H}$ in Eq. (9).

Task 1.2: (5%) A general 3×3 homography has eight degrees of freedom (nine arbitrary entries minus scale ambiguity). However, explain why $\mathbf{H}$ as defined in Eq. (9) has fewer than eight degrees of freedom, i.e. why the entries of $\mathbf{H}$ are more restricted than in an arbitrary homography.

$\boxed{\text{Part 2}}$ The direct linear transform (35%)

The direct linear transform (DLT) is a general technique for transforming a set of non-linear equations into a mathematically equivalent *linear* system that can be solved more easily. Here we will use the DLT to estimate $\mathbf{H}$ from a set of correspondences $(x_i, y_i) \leftrightarrow (X_i, Y_i)$. Note that although $\mathbf{H}$ as defined in Eq. (9) is more restricted, the algorithm here can be used to estimate an arbitrary homography.

The key idea of the DLT is to rearrange Eq. (7)-(8) by multiplying by the denominator on both sides, such that the pair of equations becomes

$$(r_{31}X_i + r_{32}Y_i + t_z)x_i = r_{11}X_i + r_{12}Y_i + t_x, \tag{10}$$

$$(r_{31}X_i + r_{32}Y_i + t_z)y_i = r_{21}X_i + r_{22}Y_i + t_y. \tag{11}$$

Notably, these equations are *linear* in the elements of $\mathbf{H}$. Hence, if we collect all the unknowns that we wish to estimate into a vector, e.g.

$$\mathbf{h} = \begin{bmatrix} r_{11} & r_{12} & t_x & r_{21} & r_{22} & t_y & r_{31} & r_{32} & t_z \end{bmatrix}^T, \tag{12}$$

then you may verify that Eq. (10)-(11) can be written as the linear system

$$\mathbf{A}_i\mathbf{h} = \mathbf{0} \tag{13}$$

where

$$\mathbf{A}_i = \begin{bmatrix} X_i & Y_i & 1 & 0 & 0 & 0 & -X_ix_i & -Y_ix_i & -x_i \\ 0 & 0 & 0 & X_i & Y_i & 1 & -X_iy_i & -Y_iy_i & -y_i \end{bmatrix}. \tag{14}$$

Each point correspondence gives rise to one pair of equations. By vertically stacking the equations from n correspondences, we obtain a linear system $\mathbf{Ah} = \mathbf{0}$, where $\mathbf{A}$ is a $2n \times 9$ matrix. This is called a *homogeneous* system. Unlike inhomogeneous systems (e.g. $\mathbf{Ah} = \mathbf{b}$ where $\mathbf{b} \neq \mathbf{0}$), homogeneous systems always have the trivial solution $\mathbf{h} = \mathbf{0}$, which is of no interest. In order to have a non-trivial solution, $\mathbf{A}$ must have a null-space of dimension one or higher. There will then be an infinite number of solutions given by linear combinations of the null-space vectors. If the null-space is 1-dimensional, then there is a non-trivial solution that is also unique up to a scaling factor. (This reflects the scale ambiguity you showed in Task 1.1, and will be addressed in Part 3.)

The null-space can be made 1-dimensional by stacking the equations from $n \geq 4$ correspondences. If $n = 4$, then there is always a unique (up to scale) exact non-trivial solution, as long as no three points on either side are collinear. If $n > 4$, then the system is over-determined, and will generally not have an exact solution apart from $\mathbf{h} = \mathbf{0}$ unless the point locations are free of noise. In absence of an exact solution, we normally seek the "best" solution in the least-squares sense

$$\min_{\mathbf{h}} ||\mathbf{Ah}||_2 \text{ subject to } ||\mathbf{h}||_2 = 1 \tag{15}$$

where we arbitrarily impose the scale constraint $||\mathbf{h}||_2 = 1$ to avoid the trivial solution. The least-squares solution $\mathbf{h}$ can be obtained from the singular value decomposition (SVD) of $\mathbf{A} = \mathbf{U\Sigma V}^T$ as the column of $\mathbf{V}$ corresponding to the smallest singular value. (A short proof of this is in section A5.3 of the Hartley and Zisserman chapter on Blackboard.)

The following data is included in this assignment:

- `K.txt`: Camera intrinsic matrix $\mathbf{K}$.
- `XY.txt`: Markers' paper coordinates (X, Y).
- `image0000.jpg...image0022.jpg`: Image sequence.
- `detections.txt`: Detected markers (one row per image). Each row contains 24 tuples of the form (d_i, u_i, v_i), where $d_i = 1$ if the i'th marker was detected and (u_i, v_i) are its detected pixel coordinates. Note that only one corner of each marker is used.

The `main.py/m` script has some helper code for loading the data and generating the requested figures. Stub functions are provided in equivalently-named files (Matlab) or in `common.py` (Python).

Task 2.1: (25%) Implement `estimate_H`. This should build the matrix $\mathbf{A}$, solve for the vector $\mathbf{h}$ and reshape the entries of $\mathbf{h}$ into the 3×3 matrix $\mathbf{H}$. Compute the predicted marker locations using $\mathbf{H}$ and run the `main` script, which should visualize the marker locations as in Fig. 1. (You will also get a 3D plot visualizing the camera and the object, but this will not work until Task 3.) Check that the predicted locations are close to the detected locations on all images and include the figure for image number 4.

Tip: Remember to convert the detected marker locations into calibrated image coordinates for use in `estimate_H`. These can be computed as $\tilde{\boldsymbol{x}}_i = \mathbf{K}^{-1}\tilde{\mathbf{u}}_i$, where $\tilde{\mathbf{u}}_i = (u_i, v_i, 1)$. After estimating $\mathbf{H}$, the predicted marker locations can be computed using Eq. (9), followed by conversion from calibrated image coordinates back to pixel coordinates: $\tilde{\mathbf{u}}_{i,\text{predicted}} = \mathbf{K}\tilde{\boldsymbol{x}}_{i,\text{predicted}}$. Note that either conversion produces a homogeneous 3-vector, which must be divided by the last component (and sliced) to obtain the actual 2D coordinates that we measure in the image.

Task 2.2: (10%) The distance between a marker's detected and predicted location is its *reprojection error*. It is measured in pixels and is defined as the Euclidean distance between pixel coordinates:

$$e_i = ||\mathbf{u}_i - \mathbf{u}_{i,\text{predicted}}||_2 = \sqrt{(\mathbf{u}_i - \mathbf{u}_{i,\text{predicted}})^T(\mathbf{u}_i - \mathbf{u}_{i,\text{predicted}})} \tag{16}$$

where $\mathbf{u}_i$ are the detected pixel coordinates of the i'th marker and $\mathbf{u}_{i,\text{predicted}}$ are computed as above. Modify your script to compute the average, minimum and maximum reprojection error over all markers in each image. Include the numbers for image number 4 in your writeup. The average reprojection error should be less than 1 pixel on this image.

Task 2.3: **(Optional self-study task - 0%)** The matrices $\mathbf{K}$ and $\mathbf{H}$ define a mapping from the object plane to the image. When this mapping is invertible, it is possible to go from pixel coordinates back to object coordinates. Use this to extract the texture of the object, i.e. the pattern of markers, into its own image. This is also called a "perspective-free" or "fronto-parallel" image and is highly useful for image processing. For example, parallel lines remain parallel, and circles and other shapes are preserved.

Part 3 Recover the pose (35%)

We can recover the pose ($\mathbf{R}$ and $\mathbf{t}$) of the planar object by observing from Eq. (9) that $\mathbf{H}$ contains the translation vector and two of the rotation matrix columns. The last column of the rotation matrix is not present, but if we know any two columns, the third can always be obtained by the cross product, e.g.

$$\mathbf{r}_3 = \mathbf{r}_1 \times \mathbf{r}_2 \tag{17}$$

where $\mathbf{r}_i = (r_{1i}, r_{2i}, r_{3i})$ is the i'th column of $\mathbf{R}$. However, recall that the solution from Part 2 is only unique up to a scaling factor. (We chose the scale $||\mathbf{h}||_2 = 1$ arbitrarily.) This means that the entries in the solved-for $\mathbf{H}$ are generally not equal to the rotation matrix and translation vector elements. Instead, there is an unknown scaling factor k such that

$$\mathbf{H} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} = k \begin{bmatrix} r_{11} & r_{12} & t_x \\ r_{21} & r_{22} & t_y \\ r_{31} & r_{32} & t_z \end{bmatrix}. \tag{18}$$

We can determine the scaling factor k by imposing the constraint that the columns of the rotation matrix should be of unit length. Hence,

$$\sqrt{h_{11}^2 + h_{21}^2 + h_{31}^2} = \sqrt{h_{12}^2 + h_{22}^2 + h_{32}^2} = |k|. \tag{19}$$

Because we can't recover the sign of k, there are two possible poses (corresponding to $+|k|$ or $-|k|$).

Task 3.1: (15%) Implement `decompose_H`. It should take in the result from `estimate_H` and return its two possible pose decompositions as 4×4 transformation matrices. If you run the `main` script, the generated figure should now draw the object coordinate frame into the image and visualize the camera and the object in 3D. Include the figure for both possible poses on image number 4.

Task 3.2: (10%) Only one of the poses is physically plausible in the sense that it places the object in front of the camera. Specifically, every detected marker should have a positive Z coordinate when transformed into the camera frame. Use this criterion to automatically choose the correct pose. Verify that the correct pose is chosen on all images, and include the figure for image number 4, 5 and 21.

Task 3.3: (10%) Due to noise, the matrix formed by $\mathbf{r}_1, \mathbf{r}_2$ and $\mathbf{r}_3$ may not exactly satisfy the properties of a rotation matrix. In Appendix C of his paper (see Blackboard), Zhang suggests to replace this matrix with the "closest" valid rotation matrix in the sense of the Frobenius norm. Read Appendix C and implement the function `closest_rotation_matrix`. Modify `decompose_H` using this function to return a valid rotation matrix for both poses.

Describe the properties of a rotation matrix, and suggest a way to numerically quantify how well the properties are satisfied by a given 3×3 matrix. Quantify how well the properties are satisfied by the rotation matrix of the chosen pose, with and without the above correction, on image number 4.

$\boxed{\text{Part 4}}$ Derive your own linear algorithm (20%)

Sometimes we have partial information about the pose, either through other sensors or by assumptions about the physically achievable motions of our robot. For example, aerial vehicles can often provide two rotational degrees of freedom from a gyroscope and accelerometer, and, if the ground is flat, one translational degree of freedom from a laser range finder. In such cases, we can derive specialized linear algorithms which may require fewer point correspondences than in the general case.

Task 4.1: (10%) Suppose you have a set of 2D-3D point correspondences $\mathbf{u}_i \leftrightarrow \mathbf{X}_i$ between an image and a general object (not necessarily planar), which satisfy the pinhole model:

$$\tilde{\mathbf{u}}_i = \mathbf{K}(\mathbf{R}\mathbf{X}_i + \mathbf{t}), i = 1...n. \tag{20}$$

Assume that $\mathbf{K}$ and $\mathbf{t}$ are both known and show (using the DLT) that n correspondences can be combined into a linear system of equations $\mathbf{A}\mathbf{m} = \mathbf{b}$, where both $\mathbf{A} \in \mathbb{R}^{2n\times 9}$ and $\mathbf{b} \in \mathbb{R}^{2n}$ are derived purely from known variables or constants, and $\mathbf{m}$ contains the unknown entries of $\mathbf{R}$.

Task 4.2: (5%) Is the system homogeneous or inhomogeneous?

Task 4.3: (5%) Suggest a way to solve the system for $\mathbf{m}$ and recover $\mathbf{R}$.