TTK4255: Robotic Vision

Homework 7: Linear algorithms

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

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

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 not necessarily optimal by your optimality criteria in the presence of uncertainty. In this assignment you will learn a strategy for deriving linear algorithms, and implement an algorithm to estimate the pose of a planar object.

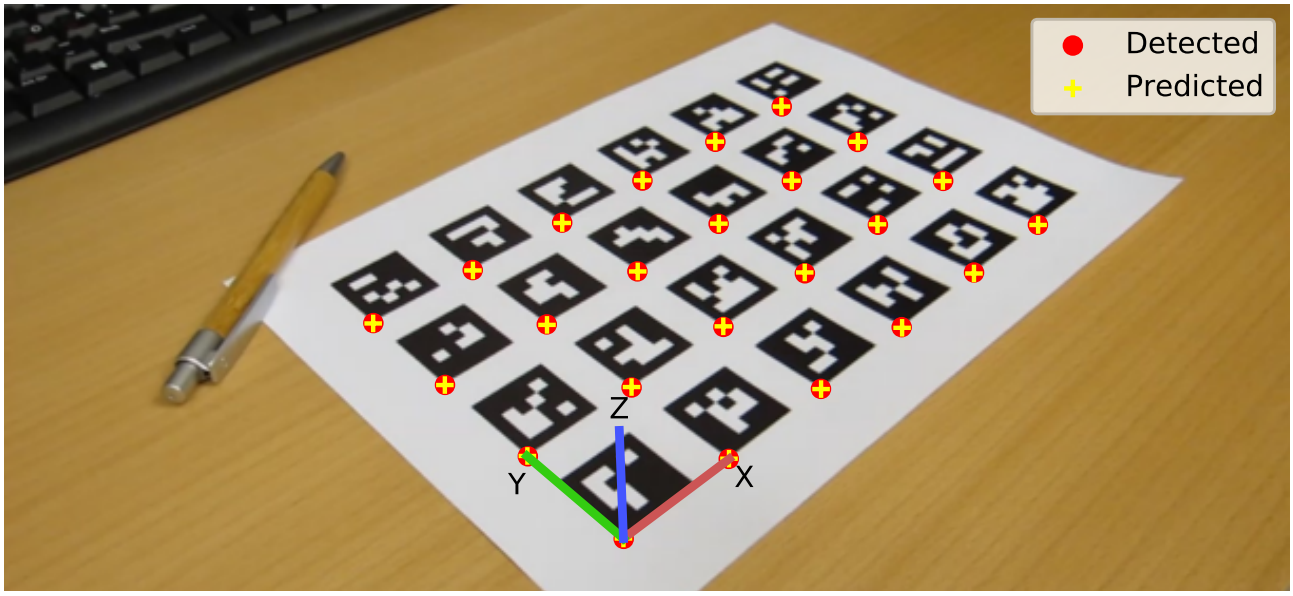

Figure 1: Output from the algorithm you will implement, showing the estimated object coordinate frame, along with the detected and predicted location of the markers.

Recommended literature

In Szeliski, linear algorithms appear in §11.2.1, §11.2.4, and §11.3.1. In Förstner and Wrobel, the general methodology is presented in §4.9, and several examples can be found throughout §10–§14 (search for "direct solution").

Linear algorithms are based on solving linear systems of equations. This topic is summarized in Hartley and Zisserman §A5, and Szeliski §A.2. We recommend that you read one of these if you are unfamiliar with this topic.

The homography (= *projective transformation* = *projective mapping* = *projectivity*) is introduced in Szeliski §2.1.1, and Förstner and Wrobel §6, and Hartley and Zisserman §2, but we provide a summary in the assignment text. Hartley and Zisserman §4 discusses the problem of estimating homographies in more detail, describing both a linear algorithm, and different cost functions for a maximum likelihood estimate.

As hinted at in Task 1.2 and Task 3.3, the linear algorithm in this assignment is related to homography estimation. For example, compare the algorithm in Part 2 with that in Hartley and Zisserman §4.1. The use of homographies to estimate the pose of planar objects is inspired by the well-known paper by Zhang:

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

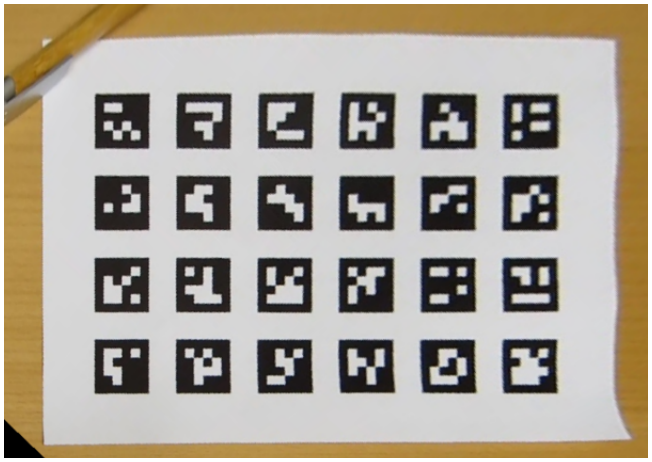

Figure 2: Homographies can be used to produce surface-aligned images of planar objects (i.e. “texture maps”), from images taken with a perspective camera from an arbitrary orientation and position.

Homographies

A homography refers to a class of mappings in 2D or in 3D. We may say “2D homography” or “3D homography”, when it is not clear from the context. A homography is the most general straight-line-preserving mapping of a space into itself. Unlike an affine mapping (which also preserves straight lines) it does not necessarily preserve parallelism between lines.

A 2D homography is a mapping between 2D coordinates $\mathbf{x}_1 \mapsto \mathbf{x}_2$. This mapping does not have a unique representation, but Hartley and Zisserman §2.3, Theorem 2.10 states that any 2D homography can be represented by a 3×3 invertible matrix $\mathbf{H}$ acting on the homogeneous 2D coordinates,

$$\tilde{\mathbf{x}}_2 = \mathbf{H}\tilde{\mathbf{x}}_1, \tag{1}$$

and vice versa. Thus a homography is simply an invertible linear transformation between homogeneous coordinates. However, keep in mind that homogeneous coordinates must be dehomogenized to obtain the Cartesian coordinates, which is a non-linear operation.

The mapping between a planar object and its perspective projection is described by a homography. The intrinsics do not need to be known; the homography can exactly describe the mapping from object coordinates to pixel coordinates regardless of the camera intrinsics and the rigid transformation between the object and camera frames. This can be used to create surface-aligned images, as shown in Fig. 2 (see also Example 2.12 in Hartley and Zisserman §2). If the intrinsics are known, as in this assignment, then the rigid transformation can be recovered from the homography.

Some other mappings that are described by a homography:

- The mapping between two images of a planar scene with a perspective camera, from an arbitrary viewpoint, and with arbitrary intrinsics (can be different for both images).
- The mapping between two images of a general scene with a perspective camera rotating (but not translating) between the images, and with arbitrary intrinsics (can be different for both images).

Part 1 Mapping between a plane and its image (10%)

Consider the perspective projection of a planar object. Without loss of generality, let points on the object have coordinates of the form $\mathbf{X} = (X, Y, 0)$. The corresponding image coordinates satisfy

$$u = c_x + f_x X^c/Z^c + s_\theta Y^c/Z^c, \tag{2}$$
$$v = c_y + f_y Y^c/Z^c, \tag{3}$$

or simply $\tilde{\mathbf{u}} = \mathbf{K}\mathbf{X}^c$, where $\mathbf{X}^c = \mathbf{R}\mathbf{X} + \mathbf{t}$ is the point's camera frame coordinates, written out as

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

You should verify for yourself that the ratios $(X^c/Z^c, Y^c/Z^c)$ are related to (X, Y) by

$$\frac{X^c}{Z^c} = \frac{r_{11}X + r_{12}Y + t_X}{r_{31}X + r_{32}Y + t_Z}, \tag{5}$$
$$\frac{Y^c}{Z^c} = \frac{r_{21}X + r_{22}Y + t_Y}{r_{31}X + r_{32}Y + t_Z}. \tag{6}$$

You should also verify for yourself that this relationship can equivalently be written as

$$\tilde{\mathbf{x}} = \mathbf{H}\begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} \quad \text{where} \quad \mathbf{H} = \begin{bmatrix} r_{11} & r_{12} & t_x \\ r_{21} & r_{22} & t_y \\ r_{31} & r_{32} & t_z \end{bmatrix}, \tag{7}$$

and we define $\mathbf{x} := (X^c/Z^c, Y^c/Z^c)$, and $\tilde{\mathbf{x}}$ is any homogeneous representation of $\mathbf{x}$.

Note that multiplying the right-hand vector by any non-zero scalar results in the same mapping from (X, Y) to (x, y). Hence, the right-hand vector could be replaced with any homogeneous representation of (X, Y). Note also that $\mathbf{x}$ can be computed from $\mathbf{u}$ as $\tilde{\mathbf{x}} = \mathbf{K}^{-1}\tilde{\mathbf{u}}$, which we use in the next part.

Task 1.1: (5%) A 2D homography is often represented by a 3×3 matrix. Any non-zero scalar multiple of this matrix is considered to represent the same homography, since it results in the same mapping between 2D coordinates (after dehomogenization). Show that this is also the case for the matrix $\mathbf{H}$ in Eq. (7) representing the mapping between (X, Y) and (x, y) in Eqs. (5)–(6).

Task 1.2: (5%) Eight parameters is sufficient to represent all different homography mappings[1]. However, the mapping between (X, Y) and (x, y) defined in Eqs. (5)–(6) require fewer than eight parameters. Why? (A consequence of this is that the mapping derived above is less general than a homography.)

Tip: How many parameters is sufficient to represent all 3D rotations?

[1] Synonymously, it is said that a homography has eight "degrees of freedom". See Hartley and Zisserman §2.3.

Part 2 A linear algorithm (35%)

A common strategy in deriving a linear algorithm is to cleverly parameterize the quantities of interest, and rearrange equations so as to be linear in these parameters, if possible. In this case, if we rearrange Eqs. (5)–(6) by multiplying by the denominator on both sides, then we get:

$$(r_{31}X_i + r_{32}Y_i + t_Z)x_i = r_{11}X_i + r_{12}Y_i + t_X, \tag{8}$$

$$(r_{31}X_i + r_{32}Y_i + t_Z)y_i = r_{21}X_i + r_{22}Y_i + t_Y. \tag{9}$$

Observe that these new equations are linear in the entries of $\mathbf{H}$, which are the unknowns that we want to estimate. To make this more clear, we can collect these entries into a vector, e.g.

$$\mathbf{h} = \begin{bmatrix} r_{11} & r_{12} & t_X & r_{21} & r_{22} & t_Y & r_{31} & r_{32} & t_Z \end{bmatrix}^\mathsf{T}, \tag{10}$$

and you may verify that Eqs. (8)–(9) can be written as the linear system $\mathbf{A}_i\mathbf{h} = \mathbf{0}$ where

$$\mathbf{A}_i = \begin{bmatrix} X_i & Y_i & 1 & 0 & 0 & 0 & -X_ix_i & -Y_ix_i & -x_i \\ 0 & 0 & 0 & X_i & Y_i & 1 & -X_iy_i & -Y_iy_i & -y_i \end{bmatrix}. \tag{11}$$

Each point correspondence gives one pair of equations. By stacking the equations from n correspondences, we get a system of the form $\mathbf{A}\mathbf{h} = \mathbf{0}$, where $\mathbf{A}$ is a $2n \times 9$ matrix.

Linear systems where the right-hand side is all zero are called *homogeneous systems*. Unlike *inhomogeneous systems*, homogeneous systems always have the trivial solution $\mathbf{h} = \mathbf{0}$, which is of no interest. A non-trivial solution exists if $\mathbf{A}$ has a null space of dimension one or more. Then there is an infinite number of solutions given by linear combinations of the basis vectors spanning the null space. If the null space is 1-dimensional, then there is a non-trivial solution that is unique up to a scaling factor. This scale ambiguity is unavoidable for homogeneous systems. (This reflects the scale ambiguity you showed in Task 1.1, and will be addressed in Part 3.)

For this particular system, if $n = 4$, then there is a unique (up to scale) 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 not have a non-trivial solution unless the correspondences are noise-free. Then we should ideally optimize some meaningful optimality criterion, e.g. maximum likelihood, but this may require an initial guess and iterative optimization. To avoid this, we can seek the solution that minimizes the algebraic error (Hartley and Zisserman §4.2.1):

$$\min_{\mathbf{h}} ||\mathbf{A}\mathbf{h}||_2 \quad \text{subject to} \quad ||\mathbf{h}||_2 = 1, \tag{12}$$

where we impose the constraint $||\mathbf{h}||_2 = 1$ to avoid the trivial solution (arbitrarily choosing the number 1; it could be anything but 0). The solution to this constrained least squares problem can be obtained from the singular value decomposition (SVD) of $\mathbf{A} = \mathbf{U}\mathbf{D}\mathbf{V}^\mathsf{T}$ as the column of $\mathbf{V}$ corresponding to the smallest singular value. (A short proof of this is in §A5.3 of Hartley and Zisserman.)

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`, which builds the matrix $\mathbf{A}$, solves for the vector $\mathbf{h}$ and reshapes 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 Part 3.) Check that the predicted locations are close to the detected locations on all images and include the figure for image number 4.

Tip: Convert the detected marker locations from pixel coordinates into (x, y) as defined in Part 1, for use in `estimate_H`. After estimating $\mathbf{H}$, the predicted marker locations can be computed using Eq. (7), followed by a conversion back to inhomogeneous pixel coordinates.

Task 2.2: (10%) Modify your script to compute the average, minimum and maximum reprojection error over all markers in each image. Recall that the reprojection error of a single correspondence is

$$e_i = ||\mathbf{u}_i - \hat{\mathbf{u}}_i||_2 = \sqrt{(\mathbf{u}_i - \hat{\mathbf{u}}_i)^\mathsf{T}(\mathbf{u}_i - \hat{\mathbf{u}}_i)} \tag{13}$$

where $\mathbf{u}_i$ are the i'th marker's detected pixel coordinates and $\hat{\mathbf{u}}_i$ are the predicted pixel coordinates computed in the above task. You can use `vecnorm` (Matlab) or `np.linalg.norm` (Python) to compute the reprojection error of multiple points without an explicit for-loop. Include the numbers for image number 4 in your writeup. The average reprojection error should be less than 1 pixel.

Part 3 Recover the pose (35%)

We can recover the pose ($\mathbf{R}$ and $\mathbf{t}$) of the planar object by observing from Eq. (7) 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

$$\mathbf{r}_3 = \mathbf{r}_1 \times \mathbf{r}_2, \tag{14}$$

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 matrix $\mathbf{H}$ as it is computed in Part 2 will generally not be equal to the entries of $\mathbf{R}$ and $\mathbf{t}$ as in Eq. (7). Instead, there is an unknown scaling factor k such that

$$\mathbf{H} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} = k \begin{bmatrix} r_{11} & r_{12} & t_x \\ r_{21} & r_{22} & t_y \\ r_{31} & r_{32} & t_z \end{bmatrix}. \tag{15}$$

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

$$\sqrt{h_{11}^2 + h_{21}^2 + h_{31}^2} = \sqrt{h_{12}^2 + h_{22}^2 + h_{32}^2} = |k|. \tag{16}$$

We can't recover the sign of k, so there remains two possible poses (corresponding to $+|k|$ or $-|k|$).

Task 3.1: (15%) Implement `decompose_H`, converting a given matrix $\mathbf{H}$ into its two possible poses. Rerun the `main` script and verify that the figure draws the object frame into the image and visualizes the camera and the object in 3D. Include the figure for both possible poses on image number 4.

Task 3.2: (5%) 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: (15%) Due to "noise", the matrix formed by $\mathbf{r}_1$, $\mathbf{r}_2$, $\mathbf{r}_3$ may not exactly satisfy all the properties of a rotation matrix (although the columns will be of unit length after the above). Indeed, nowhere in the linear algorithm do we constrain the relevant entries of $\mathbf{h}$ to satisfy all the properties of a rotation matrix. (What then is the algorithm actually estimating? Can you relate this to Task 1.2?)

Zhang suggests to find a valid rotation matrix that is "closest" to this matrix in the sense of the Frobenius norm. Implement `closest_rotation_matrix` following this approach (see Appendix C of Zhang's paper). Modify `decompose_H` 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.

Part 4 Derive your own linear algorithm (20%)

Sometimes we have partial information about the pose through other sensors or from assumptions about physically achievable motions. 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. Another example is motion on an approximately planar surface, which is described by one rotational degree of freedom, and two translational degrees of freedom. 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 point correspondences $\mathbf{u}_i \leftrightarrow \mathbf{X}_i$ between an image and a general object (not necessarily planar), which satisfy a perspective camera model:

$$\tilde{\mathbf{u}}_i = \mathbf{K}(\mathbf{R}\mathbf{X}_i + \mathbf{t}), i = 1...n. \tag{17}$$

Assume that $\mathbf{K}$ and $\mathbf{t}$ are both known and show that the equations provided by n correspondences can be rearranged and combined into a linear system $\mathbf{A}\mathbf{m} = \mathbf{b}$, 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%) Suggest a way to solve the system for $\mathbf{m}$ and recover $\mathbf{R}$.

Task 4.3: (5%) The above equations could alternatively be written using homogeneous coordinates also for the 3D points, $\tilde{\mathbf{X}}_i = (\tilde{X}_i, \tilde{Y}_i, \tilde{Z}_i, \tilde{W}_i)$:

$$\tilde{\mathbf{u}}_i = \mathbf{K}\begin{bmatrix}\mathbf{R} & \mathbf{t}\end{bmatrix}\tilde{\mathbf{X}}_i, i = 1...n, \tag{18}$$

and the algorithm could be rederived such that the components of the homogeneous vector $\tilde{\mathbf{X}}_i$ appear in the linear system, rather than the components of the inhomogeneous vector.

(a) How can you interpret 3D points for which $\tilde{W}_i = 0$?

(b) How can you interpret the remaining components $\tilde{X}_i, \tilde{Y}_i, \tilde{Z}_i$ for such points?

(c) How can you interpret the component $\tilde{W}$ in general (when it is not equal to zero)?

(d) What potential advantages does the implementation with homogeneous coordinates give?

(e) If instead $\mathbf{t}$ is unknown (and $\mathbf{R}$ is known or unknown), can you estimate $\mathbf{t}$ if $\tilde{W}_i = 0$ for all the 3D points?