3D measurements from camera images

3D Computer Vision

Ferdi van der Heijden University of Twente - RAM 11/12/2016

Contents

3D measurements from camera images

3D Computer Vision

1 Introduction

Human beings are quite capable of localizing 3D landmarks, and of finding their own position relative to those landmarks. In our common world, we can easily detect objects and localize ourselves with reference to these objects. We can estimate our own motion, and also the motion of other objects and subjects surrounding us. Much of this ability is due to our visual system.

Computerization of these tasks, based on camera images, or on video, appears to be nontrivial. There are several complicating factors:

- Images are nonlinear and incomplete measurements of a 3D world:
  - Only a 2D projection of the 3D world is measured.
  - The perspective projection is nonlinear.
  - Objects may be occluded by other objects.
- Images are indirect measurements with many unknown factors:
  - Radiometric quantities (shades of colors) are measured instead of geometric quantities.
  - Light-material interaction is complicated with many unknown factors.
  - The type of light sources (diffuse/specular) and their positions are often unknown.
- Motion involves differentiation of positions. This makes it highly sensitive to noise.

The consequence of these complicating factors is that there doesn't exist a single generic theory for gathering information from camera images. Instead, there is a hodgepodge of topics, models, techniques, and algorithms. Each application takes those parts from the hodgepodge that seem to fit

However, there are a few basic questions that come back in a diversity of applications. These basic questions are:

1. Landmark localization:
   How to localize static points (landmarks) in a 3D space that are observed by one or more cameras? Here, we assume that the position/orientation of the cameras are known.
2. Camera pose estimation 1:
   How to estimate the 3D position/orientation of a camera relative to 3D landmarks that are seen by this camera? We assume that the position of the 3D landmarks are known.
3. Camera pose estimation 2:
   How to estimate the 3D position/orientation of a camera relative to another camera when 3D landmarks are seen by this camera? We assume here that the positions of the 3D landmarks are not known.

This syllabus addresses these 3 problems. As such, it is a preparation for, for instance, visual navigation. Knowledge of 'camera models', and of 'stereo: from pixels to 3D surface meshes' is assumed.

2 3D landmarks from multiple images

This section addresses the case that we have static points, i.e. landmarks, which are observed in an image sequence. The image sequence is, for instance, a video obtained from a moving camera. The sequence could also be obtained from a multiple camera system, where each camera acquired an image. Conceptually, these two possibilities are equivalent.

Assumptions:

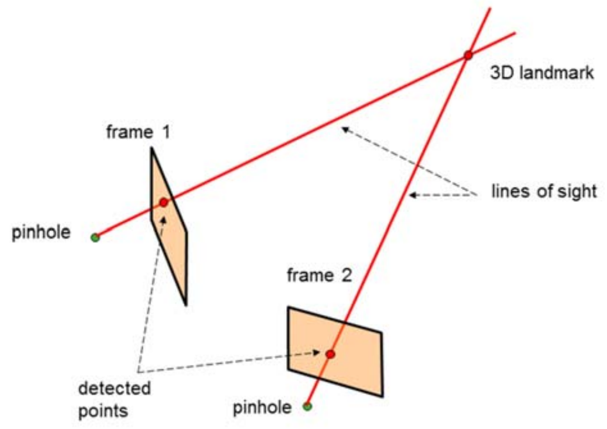
- The cameras are fully calibrated:
  - The intrinsic calibration parameters are known.
  - The poses (position/orientation) of the cameras are known with respect to some reference coordinate system.

The goal is to have an estimate of the 3D coordinates of the landmarks expressed in some reference coordinate system.

We first assume that we have only two images from the sequence. This problem looks like the problem of the estimation of 3D patches from two stereo images. However, in the current case, we are only interested in the positions of a few landmarks, which can be considered as 3D points. This is called *sparse stereo* which is in contrast of the *dense stereo* that was discussed in the syllabus 'Stereo: from pixels to 3D meshes'. In the current case, we only deal with a few 3D points with possibly more than two images. The whole machinery of stereo rectification and stereo matching is therefore not computational efficient.

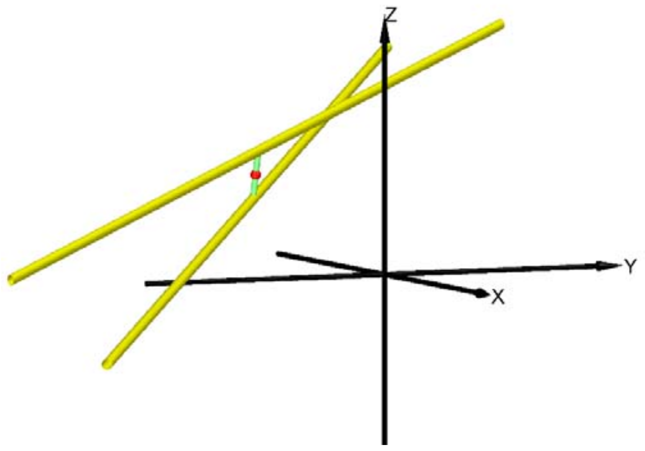
It suffices to restrict the discussion to just a single landmark as other landmarks can be processed in the same way. First, we will assume that the landmark is only visible in two images. Later, we will remove this restriction. If more images are involved, the accuracy of the estimation of the landmark can hopefully be improved.

An overview of the situation is shown below. The calibration matrices of camera 1 and camera 2 are supposed to be known, and are denoted by $\mathbf{K}_1$ and $\mathbf{K}_2$. The poses of these cameras are also known and given by the rotation matrices ${}^0\mathbf{R}_1$ and ${}^0\mathbf{R}_2$, and translation vectors ${}^0\mathbf{t}_1$ and ${}^0\mathbf{t}_2$, respectively. The prefix ^{0} indicates the reference coordinate system[1]; this is coordinate frame 0.



A landmark is detected in frame 1 and frame 2 at pixel locations ${}^1\mathbf{p}$ and ${}^2\mathbf{p}$, respectively. Knowledge of the camera calibration matrices suffices to construct lines of sight. Lines of sight are rays that are back-projected from the pinholes across the detected points into 3D space. The intersection of the two lines of sight should be the 3D position of the point (landmark) that created the two detected points.

The problem with this approach is that due to measurement errors and calibration errors the two lines do not intersect, even if these errors are small. A heuristic approach to solve this problem is to construct a line segment that connects the first line of sight with the second one. The starting point and end point of the line segment are then shifted along the two line of sights such that its length is minimized. This occurs when the line segment is orthogonal to both lines of sight. Finally, the solution is found at the midpoint of the line segment. This is demonstrated in the figure below.



2.1 Least Squares Error estimation

LSE estimation is an alternative for the midpoint algorithm. LSE estimation is as follows. Suppose that we have an estimate ${}^0\hat{\mathbf{X}}$ of the 3D position of the landmark. We then can 'predict' the positions of the detected points in the images:

$$\begin{aligned} {}^1\underline{\hat{\mathbf{p}}} &= \mathbf{K}_1\left({}^1\mathbf{R}_0\,{}^0\hat{\mathbf{X}} + {}^1\mathbf{t}_0\right) \\ {}^2\underline{\hat{\mathbf{p}}} &= \mathbf{K}_2\left({}^2\mathbf{R}_0\,{}^0\hat{\mathbf{X}} + {}^2\mathbf{t}_0\right) \end{aligned} \tag{1}$$

The results are in homogeneous coordinates, but can easily be converted to inhomogenous coordinates ${}^1\hat{\mathbf{p}}$ and ${}^2\hat{\mathbf{p}}$. These are the so-called *reprojected* points. Finally, we can calculate the *mean squared reprojection error* as:

$$SE = \frac{1}{2}\left(\left\|{}^1\mathbf{p} - {}^1\hat{\mathbf{p}}\right\|^2 + \left\|{}^2\mathbf{p} - {}^2\hat{\mathbf{p}}\right\|^2\right) \tag{2}$$

The LSE estimate is the estimate ${}^0\hat{\mathbf{X}}$ that minimizes this SE. Note that the procedure can easily be generalized to having more than two frames. The Matlab function `triangulate` (two images) and `triangulateMultiview` provide implementations of similar algorithms.

2.2 Minimum Mean Square Error estimation

From parameter estimation theory, yet another estimator is the MMSE estimator. This estimator takes a simple form if the observation model is linear, and the noise model is additive and Gaussian. Let us hypothesize that the model can be written in the form:

[1] The word 'frame' is ambiguous. In video processing, it means just an image from the sequence. In robotics, the word means: 'a coordinate system that is attached to some object'. In this syllabus, the context should make clear which connotation is meant.

$$\mathbf{z} = \mathbf{H}\,{}^0\mathbf{X} + \mathbf{n} \qquad (3)$$

The usual connotation of this model is as follows:

- ${}^0\mathbf{X}$ is the unknown parameter vector; in our case, the 3D position of the landmark. It is represented in reference coordinates: frame 0.
- $\mathbf{H}$ is the measurement matrix that describes the linear behavior of a sensor system.
- $\mathbf{n}$ is a noise vector that disturbs the sensor output. $\mathbf{n}$ is supposed to be Gaussian, zero mean and with covariance matrix[2] $\mathbf{C}_\mathbf{n}$.
- $\mathbf{z}$ is the observed measurement vector.

In MMSE estimation, one assumes that there is prior knowledge about the parameter of interest. This prior knowledge is quantified by means of an *expectation* ${}^0\hat{\mathbf{X}}_{prior}$ and a prior covariance matrix $\mathbf{C}_{prior}$. This matrix quantifies the prior uncertainty that we have about the position. In case that we have no prior knowledge at all, then we simply assume that $\mathbf{C}_{prior}$ is very large, i.e.:

$$\mathbf{C}_{prior} = \sigma^2_{prior}\mathbf{I} \quad \text{with} \quad \sigma_{prior} >> \text{depth range} \qquad (4)$$

This makes sure that the variances of the elements of the parameter vector is huge, and it indicates that the knowledge about the expectation is immensely weak.

The MMSE estimation proceeds as follows:

1. innovation matrix	$\mathbf{S} = \mathbf{H}\mathbf{C}_{prior}\mathbf{H}^T + \mathbf{C}_n$
2. Kalman gain matrix	$\mathbf{K} = \mathbf{C}_{prior}\mathbf{H}^T\mathbf{S}^{-1}$
3. posterior estimate	${}^0\hat{\mathbf{X}}_{post} = {}^0\hat{\mathbf{X}}_{prior} + \mathbf{K}\left(\mathbf{z} - \mathbf{H}\,{}^0\hat{\mathbf{X}}_{prior}\right)$
4. posterior covariance matrix	$\mathbf{C}_{post} = \mathbf{C}_{prior} - \mathbf{K}\mathbf{S}\mathbf{K}^T$

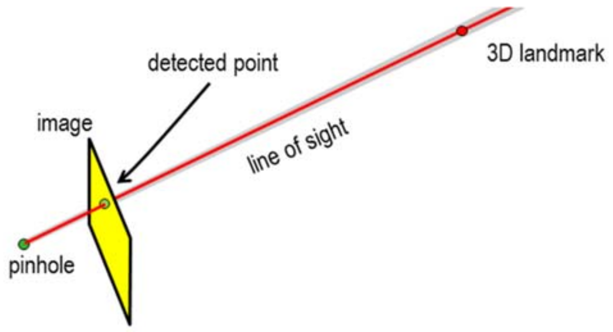
(5)

This procedure yields an estimate ${}^0\hat{\mathbf{X}}_{post}$ in which prior knowledge and sensorial information is used in a well-balanced fashion. It also provides a measure of accuracy, i.e. the covariance matrix $\mathbf{C}_{post}$ which quantifies the uncertainty.

2.2.1 MMSE with a single camera

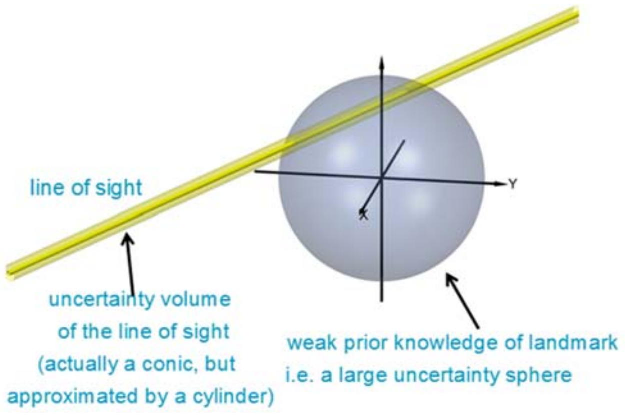
MMSE estimation works nicely, but its solution as stated in (5) is only valid if the model is linear; that is, according to (3). We have to modify the perspective projection equation in such a way that the linear model of (3) still holds true. To keep it simple, we first confine the discussion to only 1 camera; see below. The model that relates the observation ${}^1\underline{\mathbf{p}}$, i.e. the pixel position of the landmark, to ${}^0\mathbf{X}$, i.e. the 3D position of the landmark, is:

$${}^1\underline{\mathbf{p}} = \mathbf{K}_1\left({}^1\mathbf{R}_0\,{}^0\mathbf{X} + {}^1\mathbf{t}_0\right) \qquad (6)$$



The situation is illustrated above. Uncertainties can be visualized as volumes. This is presented below. The prior knowledge of the 3D position is a point ${}^0\hat{\mathbf{X}}_{prior}$ that is surrounded by a larger sphere representing $\mathbf{C}_{prior}$. The knowledge of the observation ${}^1\underline{\mathbf{p}}$ is a ray that is surrounded by a cylinder, which reflects the uncertainty in the localization of ${}^1\underline{\mathbf{p}}$ in the image.

The goal is to mold eq (6) such that it get the same structure as (3). To do so, we write $\mathbf{K}_1 = \left[\mathbf{k}_1^\mathrm{T} \quad \mathbf{k}_2^\mathrm{T} \quad \mathbf{k}_3^\mathrm{T}\right]^\mathrm{T}$ where $\mathbf{k}_n^\mathrm{T}$ is the n-th row from the matrix $\mathbf{K}_1$. We also substitute ${}^1\underline{\mathbf{p}} = \left[\alpha\,{}^1p_1 \quad \alpha\,{}^1p_2 \quad \alpha\right]^\mathrm{T}$. This yields:


line of sight
Y
X
uncertainty volume of the line of sight (actually a conic, but approximated by a cylinder)
weak prior knowledge of landmark i.e. a large uncertainty sphere

$$\begin{bmatrix} \alpha\,{}^1p_1 \\ \alpha\,{}^1p_2 \\ \alpha \end{bmatrix} = \begin{bmatrix} \mathbf{k}_1^\mathrm{T} \\ \mathbf{k}_2^\mathrm{T} \\ \mathbf{k}_3^\mathrm{T} \end{bmatrix}\left({}^1\mathbf{R}_0\,{}^0\mathbf{X} + {}^1\mathbf{t}_0\right)$$

[2] A covariance matrix is a symmetric matrix. The diagonal contains the variances of the noise elements. The off-diagonal elements are the covariances between different noise elements.

The last row reads as $\alpha = \mathbf{k}_3^T \left({}^1\mathbf{R}_0 {}^0\mathbf{X} + {}^1\mathbf{t}_0 \right)$. Substitution in the other two rows yields:

$$\begin{aligned}\mathbf{k}_3^T \left({}^1\mathbf{R}_0 {}^0\mathbf{X} + {}^1\mathbf{t}_0 \right) {}^1p_1 &= \mathbf{k}_1^T \left({}^1\mathbf{R}_0 {}^0\mathbf{X} + {}^1\mathbf{t}_0 \right) \\ \mathbf{k}_3^T \left({}^1\mathbf{R}_0 {}^0\mathbf{X} + {}^1\mathbf{t}_0 \right) {}^1p_2 &= \mathbf{k}_2^T \left({}^1\mathbf{R}_0 {}^0\mathbf{X} + {}^1\mathbf{t}_0 \right)\end{aligned}$$

After rearrangement:

$$\begin{aligned}\left({}^1p_1\mathbf{k}_3^T - \mathbf{k}_1^T \right) {}^1\mathbf{t}_0 &= \left(\mathbf{k}_1^T - {}^1p_1\mathbf{k}_3^T \right) {}^1\mathbf{R}_0 {}^0\mathbf{X} \\ \left({}^1p_2\mathbf{k}_3^T - \mathbf{k}_2^T \right) {}^1\mathbf{t}_0 &= \left(\mathbf{k}_2^T - {}^1p_2\mathbf{k}_3^T \right) {}^1\mathbf{R}_0 {}^0\mathbf{X}\end{aligned} \tag{7}$$

${}^1\mathbf{p}$ is the exact location of the imaged point. However, what we have is a *detected* location ${}^1\hat{\mathbf{p}}$. There might be a localization error $\boldsymbol{\varepsilon}$. We take this into account by substituting ${}^1p_1 = {}^1\hat{p}_1 + \varepsilon_1$ and ${}^1p_2 = {}^1\hat{p}_2 + \varepsilon_2$:

$$\begin{aligned}\left(({}^1\hat{p}_1 + \varepsilon_1)\mathbf{k}_3^T - \mathbf{k}_1^T \right) {}^1\mathbf{t}_0 &= \left(\mathbf{k}_1^T - ({}^1\hat{p}_1 + \varepsilon_1)\mathbf{k}_3^T \right) {}^1\mathbf{R}_0 {}^0\mathbf{X} \\ \left(({}^1\hat{p}_2 + \varepsilon_2)\mathbf{k}_3^T - \mathbf{k}_2^T \right) {}^1\mathbf{t}_0 &= \left(\mathbf{k}_2^T - ({}^1p_2 + \varepsilon_2)\mathbf{k}_3^T \right) {}^1\mathbf{R}_0 {}^0\mathbf{X}\end{aligned} \tag{8}$$

Rewritten into:

$$\begin{aligned}\left({}^1\hat{p}_1\mathbf{k}_3^T - \mathbf{k}_1^T \right) {}^1\mathbf{t}_0 &= \left(\mathbf{k}_1^T - {}^1\hat{p}_1\mathbf{k}_3^T \right) {}^1\mathbf{R}_0 {}^0\mathbf{X} - \varepsilon_1\mathbf{k}_3^T \left({}^1\mathbf{R}_0 {}^0\mathbf{X} + {}^1\mathbf{t}_0 \right) \\ \left({}^1\hat{p}_2\mathbf{k}_3^T - \mathbf{k}_2^T \right) {}^1\mathbf{t}_0 &= \left(\mathbf{k}_2^T - {}^1p_2\mathbf{k}_3^T \right) {}^1\mathbf{R}_0 {}^0\mathbf{X} - \varepsilon_2\mathbf{k}_3^T \left({}^1\mathbf{R}_0 {}^0\mathbf{X} + {}^1\mathbf{t}_0 \right)\end{aligned} \tag{9}$$

The point $\mathbf{X}$, represented in camera coordinates, is ${}^1\mathbf{X} = {}^1\mathbf{R}_0 {}^0\mathbf{X} + {}^0\mathbf{t}_1$. Note that $\mathbf{k}_3^T = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}$. Therefore, the term $\mathbf{k}_3^T \left({}^1\mathbf{R}_0 {}^0\mathbf{X} + {}^1\mathbf{t}_0 \right) = \mathbf{k}_3^T \, {}^1\mathbf{X}$ is just the z-component of ${}^1\mathbf{X}$, say ${}^1Z$. This is the depth op the landmark relative to the camera. With that, (9) simplifies to:

$$\begin{aligned}\left({}^1\hat{p}_1\mathbf{k}_3^T - \mathbf{k}_1^T \right) {}^1\mathbf{t}_0 &= \left(\mathbf{k}_1^T - {}^1\hat{p}_1\mathbf{k}_3^T \right) {}^1\mathbf{R}_0 {}^0\mathbf{X} - \varepsilon_1 {}^1Z \\ \left({}^1\hat{p}_2\mathbf{k}_3^T - \mathbf{k}_2^T \right) {}^1\mathbf{t}_0 &= \left(\mathbf{k}_2^T - {}^1p_2\mathbf{k}_3^T \right) {}^1\mathbf{R}_0 {}^0\mathbf{X} - \varepsilon_2 {}^1Z\end{aligned} \tag{10}$$

A further simplification is introduced by defining the following vectors and matrix:

$$\mathbf{z}_1 \overset{def}{=} \begin{bmatrix} {}^1\hat{p}_1\mathbf{k}_3^T - \mathbf{k}_1^T \\ {}^1\hat{p}_2\mathbf{k}_3^T - \mathbf{k}_2^T \end{bmatrix} {}^1\mathbf{t}_0$$

$$\mathbf{H}_1 \overset{def}{=} \begin{bmatrix} \mathbf{k}_1^T - {}^1\hat{p}_1\mathbf{k}_3^T \\ \mathbf{k}_2^T - {}^1p_2\mathbf{k}_3^T \end{bmatrix} {}^1\mathbf{R}_0 \tag{11}$$

$$\mathbf{n}_1 \overset{def}{=} \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \end{bmatrix} {}^1Z$$

So that:

$$\mathbf{z}_1 = \mathbf{H}_1 {}^0\mathbf{X} + \mathbf{n}_1 \tag{12}$$

This is exactly the form of (3). Hence, the solution of (5) is almost within reach. The only obstacle is that the covariance matrix $\mathbf{C}_{\mathbf{n}_1}$ is not known yet.

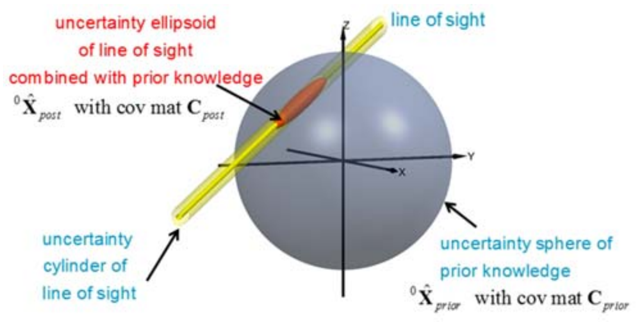
The 2x2 covariance matrix $\mathbf{C}_{\mathbf{n}_1}$ represents the uncertainty that is due to the localization error in the image. Suppose that the uncertainty in the pixel localization is σ pixels in all directions. A reasonable assumption is that σ is 1 pixel. The variances of ε_1 and ε_2 will be σ^2. According to (11), the propagation of the localization errors in the image towards the 3D position of the point is linear with the depth ${}^1Z$. Correspondingly, the covariance matrix of $\mathbf{n}_1$ becomes:

$$\mathbf{C}_{\mathbf{n}_1} = {}^1Z^2 \sigma^2 \mathbf{I} \tag{13}$$

A complication is that ${}^1Z$, being the depth wrt the camera, is not priory known. Thus, the solution (5) cannot be applied directly. We tackle this problem in a three stage approach. In the first stage, we use a wild guess of ${}^1Z$. The second and third stages are when a second line of sight is available. We are then able to estimate the depths to the two cameras, and substitute these in (13).

An illustration is shown below. The prior knowledge, is represented by a sphere. Since the prior uncertainty is assumed to be large, the radius σ_{prior} is large. The line of sight is surrounded by a cylindrical tube representing the uncertainty, The radius of the tube is ${}^1Z\sigma$. Actually, the tube is a conic with linearly increasing radius. In a first approximation, it is assumed that ${}^1Z\sigma$ is nearly constant within the sphere. This first approximation might be inaccurate, but when a second line of sight becomes available, a more accurate approximate will be obtained.

The estimate, resulting from (5) is the posterior knowledge represented by ${}^0\hat{\mathbf{X}}_{post}$ and ${}^0\mathbf{C}_{post}$, graphically illustrated by an ellipsoid approximating the intersection of the tube and the sphere. Thus, the larger the radius of the sphere, the larger the principal axis of the ellipsoid will be. Intuitively, this is correct since only one camera is given, so that we don't have depth information except for the prior knowledge.


uncertainty ellipsoid
of line of sight
combined with prior knowledge
$^0\hat{\mathbf{X}}_{post}$ with cov mat $\mathbf{C}_{post}$
line of sight
uncertainty
cylinder of
line of sight
uncertainty sphere of
prior knowledge
$^0\hat{\mathbf{X}}_{prior}$ with cov mat $\mathbf{C}_{prior}$

2.2.2 MMSE with more than one images

We can now easily proceed with the knowledge from a second image coming from a second camera (or the next frame in a video). In this second image we determine the location of the landmark, i.e. $^2\hat{\mathbf{p}} = {}^2\mathbf{p} + \boldsymbol{\varepsilon}$. In accordance with (11) we define:

$$\mathbf{z}_2 \overset{def}{=} \begin{bmatrix} {}^2\hat{p}_1 \mathbf{k}_3^T - \mathbf{k}_1^T \\ {}^2\hat{p}_2 \mathbf{k}_3^T - \mathbf{k}_2^T \end{bmatrix} {}^2\mathbf{t}_0$$

$$\mathbf{H}_2 \overset{def}{=} \begin{bmatrix} \mathbf{k}_1^T - {}^2\hat{p}_1 \mathbf{k}_3^T \\ \mathbf{k}_2^T - {}^2p_2 \mathbf{k}_3^T \end{bmatrix} {}^2\mathbf{R}_0 \qquad (14)$$

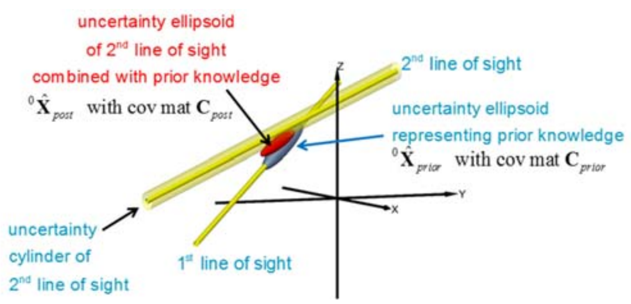
$$\mathbf{n}_2 \overset{def}{=} \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \end{bmatrix} {}^2Z$$

So that:

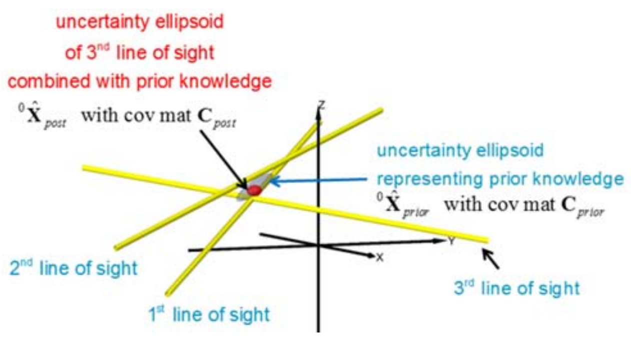
$$\mathbf{z}_2 = \mathbf{H}_2 {}^0\mathbf{X} + \mathbf{n}_2 \qquad (15)$$

The previous posterior knowledge, which we obtained after processing the first image, is prior knowledge in the second image. That is: $^0\hat{\mathbf{X}}_{prior} := {}^0\hat{\mathbf{X}}_{post}$ and $\mathbf{C}_{prior} := \mathbf{C}_{post}$. Application of (5) provides us the posterior knowledge based on the second image. At first, we still use some wild guesses for the depths ^1Z and ^2Z. This results in an updated estimate $^0\hat{\mathbf{X}}$ which is usually already close to the real solution. Hence, we can repeat (5) again, but this time with the newly obtained estimate of the depth ^1Z and ^2Z.

An illustration is shown below. The estimate from the first image (grey ellipsoid) is nicely combined with the knowledge coming from the second image, resulting in the red ellipsoid. The shape of this ellipsoid represents the uncertainty. Due to the unfavorable inclination angle of the two lines, the uncertainty of the estimate is in one direction quite large. Across the lines, the uncertainty is quite small.


uncertainty ellipsoid
of 2nd line of sight
combined with prior knowledge
$^0\hat{\mathbf{X}}_{post}$ with cov mat $\mathbf{C}_{post}$
2nd line of sight
uncertainty ellipsoid
representing prior knowledge
$^0\hat{\mathbf{X}}_{prior}$ with cov mat $\mathbf{C}_{prior}$
uncertainty
cylinder of
2nd line of sight
1st line of sight

The process of updating knowledge by means of additional lines of sight, when more images become successively available, is an iterative repetition of (14), (15), and (5). This will increase the accuracy as time proceeds. An update is advantageous, especially when the inclination direction of the line of sight is across the one of previous lines. An illustration is provided below


uncertainty ellipsoid
of 3rd line of sight
combined with prior knowledge
$^0\hat{\mathbf{X}}_{post}$ with cov mat $\mathbf{C}_{post}$
uncertainty ellipsoid
representing prior knowledge
$^0\hat{\mathbf{X}}_{prior}$ with cov mat $\mathbf{C}_{prior}$
2nd line of sight
1st line of sight
3rd line of sight

2.2.3 Matlab implementation

A Matlab implementation of such an iteration is provided in the function `ut_3Dlandmarks_from_2Dpoints`. Visualizations of the covariance matrices in 3D, i.e. the ellipsoids that represent the uncertainties are generated with the function `ut_plotcov3`.

3 Camera pose estimation from known landmarks

The situation here is as follows: Given:

- The calibration matrix $\mathbf{K}$ of a camera.
- A static scene with landmarks of which the 3D positions, $^0\mathbf{X}_n$ with $n = 1, \cdots, N$ and expressed in a reference coordinate frame 0, are known.
- These landmarks as detected and localized in the image at the pixel coordinates (column and row subscripts) $\mathbf{p}_n$.

The question is:

- What is the pose, i.e. position $^0\mathbf{t}_c$ and orientation $^0\mathbf{R}_c$, of the camera expressed in reference coordinates?

3.1 Using the projection matrix

Actually, this approach is already described in the syllabus on camera models. It is part of the camera calibration procedure. See syllabus “Camera models”, Section 5.

In short, the key to the solution is the equation of the perspective projection that is linear in the homogeneous representation:

$$\underline{\mathbf{p}}_n = \mathbf{M}\,{}^0\underline{\mathbf{X}}_n \quad \text{with} \quad \mathbf{M} = \mathbf{K}\begin{bmatrix} {}^c\mathbf{R}_0 & {}^c\mathbf{t}_0 \end{bmatrix} \tag{16}$$

where $\underline{\mathbf{p}}_n$ and ${}^0\underline{\mathbf{X}}_n$ are the homogenized versions of $\mathbf{p}_n$ and ${}^0\mathbf{X}_n$. The matrix $\mathbf{M}$ is a 3×4 matrix which can be estimated from the N equations stated in (16). For details, see Section 5.1 in “Camera Models.

Once an estimate $\hat{\mathbf{M}}$ of $\mathbf{M}$ is available, the pose follows from the definition of $\mathbf{M}$ in (16):

$$\begin{bmatrix} {}^0\hat{\mathbf{R}}_c & {}^0\hat{\mathbf{t}}_c \end{bmatrix} = \mathbf{K}^{-1}\hat{\mathbf{M}} \tag{17}$$

To get the pose expressed in world coordinates:

$${}^0\hat{\mathbf{R}}_c = {}^c\hat{\mathbf{R}}_0^{\mathrm{T}} \quad \text{and} \quad {}^0\hat{\mathbf{t}}_c = -{}^0\hat{\mathbf{R}}_c\,{}^c\hat{\mathbf{t}}_0 \tag{18}$$

Notes:

- As explained in Section 5.1 of “Camera Models”, the estimate of $\mathbf{M}$ is based on a SVD of a $2N\times 12$ matrix that is formed from the landmark positions and the corresponding pixel positions. There are certain conditions that have to be met for an ambiguous solution. For instance, we need at least 6 landmarks, and the landmarks are not allowed to lie in just one single plane.
- A necessary condition for any rotation matrix is that $\mathbf{R}\,\mathbf{R}^T = \mathbf{I}$. That is, the matrix has to be orthonormal. Because both $\hat{\mathbf{M}}$ and $\mathbf{K}$ are estimates, this condition for ${}^0\hat{\mathbf{R}}_c$ is usually only approximately met. An improvement is to enforce the orthogonality. This is accomplished by means of a SVD applied to ${}^0\hat{\mathbf{R}}_c$, and by setting the singular vales to one.

3.2 Optimal methods

Since pose has six degrees of freedom, 3 for position and 3 for orientation, one can imagine that estimation with less than 6 landmarks is possible as each landmark gives two equations. Optimal solutions that need less than 6 landmarks are obtained by defining an optimality criterion, which depends on the pose, and then by applying an optimization procedure to find the pose that optimizes the criterion. The optimization cannot be done analytically. Numerical optimization is needed.

3.2.1 Minimization of the reprojection errors

The classical method is to define the RMS (root mean square) of the reprojection errors as the optimality criterion that should be minimized. This criterion depends on the 6 parameters.

The orientation of the camera is described by three Euler angles: $\boldsymbol{\alpha} = \{\psi \quad \theta \quad \varphi\}$. See the syllabus “Math for Computer Vision and Navigation”; Section D.3. A rotation matrix ${}^c\mathbf{R}_0$ is constructed by three successive rotations ψ, θ, and φ around the axes of the coordinate system. Hence, we can write ${}^c\mathbf{R}_0(\boldsymbol{\alpha})$. With that, a rigid transform applied to the reference coordinates is described by:

$${}^c\mathbf{R}_0(\boldsymbol{\alpha})\,{}^0\mathbf{X}_n + {}^c\mathbf{t}_0 \tag{19}$$

With this transform, the reprojected points are $\hat{\underline{\mathbf{p}}}_n = \mathbf{K}\begin{bmatrix} {}^c\mathbf{R}_0(\boldsymbol{\alpha}) & {}^c\mathbf{t}_0 \end{bmatrix}{}^0\underline{\mathbf{X}}_n$. After inhomogenizing, this becomes $\hat{\mathbf{p}}_n$. The RMS of the reprojection errors is:

$$J(\boldsymbol{\alpha}, {}^c\mathbf{t}_0) = \sqrt{\frac{1}{N}\sum_{n=1}^{N}\left\|\mathbf{p}_n - \hat{\mathbf{p}}_n\right\|^2} \tag{20}$$

Numerical minimization of $J(\boldsymbol{\alpha}, {}^c\mathbf{t}_0)$ by varying $\boldsymbol{\alpha}$ and ${}^c\mathbf{t}_0$ provides the solution. The most popular minimization algorithm is Levenberg-Marquardt.

3.2.2 Orthogonal iteration algorithm

The numerical minimization of the reprojection error is not always successful. The iterative procedures do not always converge. An improvement is the so-called orthogonal iteration algorithm. Unlike, the reprojection error, the criterion here is not in the image plane, but rather in the 3D space. This is accomplished as follows.

Suppose that ${}^c\mathbf{X}_n$ are the 3D landmarks expressed in camera coordinates. The observed projected points, in homogeneous pixel coordinates, are $\underline{\mathbf{p}}_n$. The same points, can also be represented in non-homogenous 3D camera coordinates:

$${}^c\mathbf{x}_n = \mathbf{K}^{-1}\underline{\mathbf{p}}_n \tag{21}$$

Since ${}^c\mathbf{x}_n$ is a projection of ${}^c\mathbf{X}_n$, it must be lying on the ray from ${}^c\mathbf{X}_n$ to the origin. Therefore, ${}^c\mathbf{x}_n = \gamma\,{}^c\mathbf{X}_n$ with γ some positive real number. Consequently, ${}^c\mathbf{x}_n^{\mathrm{T}}\,{}^c\mathbf{X}_n = \gamma\left\|{}^c\mathbf{X}_n\right\|^2$ and $\left\|{}^c\mathbf{x}_n\right\|^2 = \gamma^2\left\|{}^c\mathbf{X}_n\right\|^2$. Therefore:

$$ {}^{c}\mathbf{X}_n = \frac{{}^{c}\mathbf{x}_n {}^{c}\mathbf{x}_n^T}{\left\| {}^{c}\mathbf{x}_n \right\|^2} {}^{c}\mathbf{X}_n \tag{22} $$

or:

$$ {}^{c}\mathbf{X}_n = \mathbf{F}_n {}^{c}\mathbf{X}_n \text{ with } \mathbf{F}_n = \frac{{}^{c}\mathbf{x}_n {}^{c}\mathbf{x}_n^T}{\left\| {}^{c}\mathbf{x}_n \right\|^2} \tag{23} $$

At the same time, we also have the rigid transform:

$$ {}^{c}\mathbf{X}_n = {}^{c}\mathbf{R}_0 \, {}^{0}\mathbf{X}_n + {}^{c}\mathbf{t}_0 \tag{24} $$

Substitution of (24) in (23) and subtraction yields:

$$ \begin{aligned} & \mathbf{F}_n \left({}^{c}\mathbf{R}_0 \, {}^{0}\mathbf{X}_n + {}^{c}\mathbf{t}_0 \right) - {}^{c}\mathbf{R}_0 \, {}^{0}\mathbf{X}_n - {}^{c}\mathbf{t}_0 = \mathbf{0} \\ \text{or} \quad & \left(\mathbf{F}_n - \mathbf{I} \right) \left({}^{c}\mathbf{R}_0 \, {}^{0}\mathbf{X}_n + {}^{c}\mathbf{t}_0 \right) = \mathbf{0} \end{aligned} \tag{25} $$

We then seek ${}^{c}\mathbf{R}_0$ and ${}^{c}\mathbf{t}_0$ to minimize:

$$ J({}^{c}\mathbf{R}_0, {}^{c}\mathbf{t}_0) = \sum_{n=1}^{N} \left\| \left(\mathbf{F}_n - \mathbf{I} \right) \left({}^{c}\mathbf{R}_0 \, {}^{0}\mathbf{X}_n + {}^{c}\mathbf{t}_0 \right) \right\|^2 \tag{26} $$

The minimization can be done iteratively and with use of Kabsch algorithm. Convergence is guaranteed. Details are given by Lu et al .[1]

A matlab implementation of this procedure is provided in the function `ut_ortho_iteration`.

4 Camera pose estimation from unknown landmarks

The problem addressed here is as follows: Given:

- The calibration matrix $\mathbf{K}$ of a camera.
- A static scene with an unknown number of visible landmarks of which the 3D positions, are unknown.
- The camera is moving.
- An image sequence (video) of that scene from the camera.

Questions are:

- What are the linear (translational) velocity and angular (rotational) velocity of the camera as time proceeds?
- What is the trajectory (3D path) of the camera?

Once the velocities are found, the trajectory of camera can be constructed by integrating the velocities. The principle is comparable with electro-mechanical odometry. An example of the latter is a bicycle computer. The sensor uses pulse frequency modulation to measure the turn rate of the wheel. By counting the number of pulses the travelled distance is obtained. This counting is analogous to integration.

The simplest example of visual odometry is the optical mouse. The sensor measures optical flow. The output of the optical mouse is 2D rather than the 3D problem that is addressed in the current section.

Since the acquisition of images is a time-discrete process, we can only find the velocities approximately. In fact, we will find a change of pose between one frame and the next one. Hence, visual odometry boils down to finding the displacement (translation) vector, and the rotation matrix that describe the change of pose. With that, 3D visual odometry is not only useful for estimating the velocity of a moving camera. It is also useful for stereo-vision in case no information is available about the extrinsic calibration parameters of two cameras.

The steps to go for finding the change of pose are as follows:

1. Detect a number of corresponding points in the two consecutive frames using optical flow or key point detection. See Sections 3 and 4 of syllabus 3.
2. Use these points to estimate the fundamental matrix. This is accomplished by the so-called *eight-point algorithm*.
3. Use the calibration matrix and the fundamental matrix to get the essential matrix.
4. The rotation matrix of the two view points of the camera, and the direction of the displacement, i.e. the base line vector, is encoded in the essential matrix.

Unfortunately, this does not give us the magnitude of the translation vector, only the direction. This problem cannot be overcome unless some reference distance in the scene is available.

4.1 Eight-point algorithm

Let us indicate a given frame from the image sequence by 1, and the next frame by 2. Then, the problem is to get the displacement vector ${}^{1}\mathbf{t}_2$ and the rotation matrix ${}^{1}\mathbf{R}_2$ from the two frames. For that purpose, we assume that a number of N key points, ${}^{1}\mathbf{p}_n$ or in homogeneous coordinates ${}^{1}\underline{\mathbf{p}}_n$, have been found in the first frame, and corresponding key points, ${}^{2}\mathbf{p}_n$ or ${}^{2}\underline{\mathbf{p}}_n$ in the second frame. These could be obtained by optical flow algorithms or key point detection and matching. The eight point algorithm is able to estimate the fundamental matrix $\mathbf{F}$ from these corresponding points.

As described in syllabus 2: '3D Surface Reconstruction', Section 2.2, the fundamental matrix is a 3×3 matrix that relates the corresponding points in the two frames:

$$ {}^2\underline{\mathbf{p}}_n^T \, \mathbf{F} \, {}^1\underline{\mathbf{p}}_n = 0 \tag{27}$$

Expanding this equation by writing:

$$ {}^1\underline{\mathbf{p}}_n = \begin{bmatrix} {}^1c_n \\ {}^1r_n \\ 1 \end{bmatrix} \qquad {}^2\underline{\mathbf{p}}_n = \begin{bmatrix} {}^2c_n \\ {}^2r_n \\ 1 \end{bmatrix} \qquad \mathbf{F} = \begin{bmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{bmatrix} $$

yields:

$$ {}^1c_n\,{}^2c_n f_{11} + {}^1c_n\,{}^2r_n f_{12} + {}^1c_n f_{13} + {}^1r_n\,{}^2c_n f_{12} + {}^1r_n\,{}^2r_n f_{22} + \\ + {}^2r_n f_{23} + {}^1c_n f_{31} + {}^2r_n f_{31} + f_{33} = 0 \tag{28}$$

This is recognized as a linear equation in the elements f_{ij}. By reshaping the elements into a 9D vector, i.e.:

$$ \mathbf{f} = \begin{bmatrix} f_{11} & f_{12} & f_{13} & f_{21} & f_{22} & f_{23} & f_{31} & f_{32} & f_{33} \end{bmatrix}^T $$

the following vector matrix equation is constructed:

$$ [{}^1c_n\,{}^2c_n \quad {}^1c_n\,{}^2r_n \quad {}^1c_n \quad {}^1r_n\,{}^2c_n \quad {}^1r_n\,{}^2r_n \quad {}^2r_n \quad {}^1c_n \quad {}^2r_n \quad 1]\mathbf{f} = 0 \tag{29}$$

For N corresponding points, we have:

$\mathbf{G}\mathbf{f} = \mathbf{0}$ f is the eigenvector of TGG with smallest eigenvalue.(30)

where $\mathbf{G}$ is a $N\times 9$ matrix constituting the N equations of (29). To solve (30) we need a constraint like $\|\mathbf{f}\| = 1$ to prevent the trivial solution $\mathbf{f} = \mathbf{0}$. Note that $\mathbf{F}$ is invariant under a scaling factor. Note that if $\mathbf{F}$ is a solution, then any $\alpha\mathbf{F}$ with $\alpha \neq 0$ is also a valid solution. Singular value decomposition provides the solution for the problem. $\mathbf{f}$ is the eigenvector of $\mathbf{G}^T\mathbf{G}$ with the smallest eigenvalue. See Section B.5 in the syllabus "Math for computer vision and navigation". Note that in principle 8 corresponding points suffices to find this eigenvector. Hence, the name *eight-point* algorithm. However, the more corresponding points we have, the more accurate the estimate of $\mathbf{F}$ will be. Also, an even distribution of points over the image plane contributes to higher accuracy.

A complication is that the matrix $\mathbf{G}^T\mathbf{G}$ is badly conditioned. That is, the 2nd smallest eigenvalue is often many orders of magnitude smaller than the largest one, and is often not much larger than the smallest eigenvalue. This is due to the fact that products of pixel coordinates in the elements of $\mathbf{G}$ have often an order of magnitude of, say, 10^6, so that in the squared version $\mathbf{G}^T\mathbf{G}$ this becomes 10^{12} which is very large compared with 1 which also occurs in one of the elements of $\mathbf{G}^T\mathbf{G}$. One way to prevent this is to normalize the coordinates first such that the average coordinate becomes zero, and the sample variance of the set becomes 1. Afterwards the estimated fundamental matrix must be de-normalized.

A further improvement of the estimated $\mathbf{F}$ can be obtained by using the fact that $\mathbf{F}$ must have rank 2. That means that if in a SVD of $\mathbf{F}$, i.e.:

$$ \mathbf{F} = \mathbf{U}\,\mathrm{diag}(s_1, s_2, s_3)\mathbf{V}^T \text{ with } \mathrm{diag}(s_1, s_2, s_3) = \begin{bmatrix} s_1 & 0 & 0 \\ 0 & s_2 & 0 \\ 0 & 0 & s_3 \end{bmatrix} \tag{31}$$

the singular values are sorted, i.e., $|s_1| \geq |s_2| \geq |s_3|$, then s_3 should be zero. Hence, an improved estimate is:

Key point matching and optical flow (32)

Usually, the set of corresponding points contains some outliers, e.g. mismatches and/or erroneous optical flow results. For that reason, the eight-point algorithm must be made robust. This is often done with the so-called RANSAC algorithm. See Section 7 in the syllabus "Key point matching and optical flow".

4.2 Getting the essential matrix

Once the fundamental matrix $\mathbf{F}$ is available, it easy to get the essential matrix. See equation (17) in syllabus "Stereo: from pixels to 3D surface mesh":

$$ \mathbf{E} = \mathbf{K}^T\mathbf{F}\mathbf{K} \tag{33}$$

Here another improvement can be brought in. The essential matrix has rank 2, but also has the property that the two singular values should be the same. So, if the SVD of the estimated $\mathbf{E}$ is $\mathbf{E} = \mathbf{U}\,\mathrm{diag}(s_1, s_2, 0)\mathbf{V}^T$, then an improvement is obtained by substituting:

$$ \mathbf{E} := \mathbf{U}\,\mathrm{diag}(1,1,0)\mathbf{V}^T \tag{34}$$

We have used here the property that the essential matrix is defined up to scale.

4.3 Estimation of the rotation and displacement

According to equation (12) in syllabus “Stereo: from pixels to 3D surface mesh“, the essential matrix encodes the desired rotation matrix ${}^{2}\mathbf{R}_{1}$ and displacement vecor ${}^{1}\mathbf{t}_{2}$ as follows :

$$\mathbf{E} = {}^{2}\mathbf{R}_{1}\mathbf{T} \quad \text{with} \quad \mathbf{T} = \begin{bmatrix} 0 & -{}^{1}t_z & {}^{1}t_y \\ {}^{1}t_z & 0 & -{}^{1}t_x \\ -{}^{1}t_y & {}^{1}t_x & 0 \end{bmatrix} \quad \text{and} \quad {}^{1}\mathbf{t}_{2} = \begin{bmatrix} {}^{1}t_x \\ {}^{1}t_y \\ {}^{1}t_z \end{bmatrix}$$

We need to solve the inverse problem: how to recover them from $\mathbf{E}$. The answer is again provided by singular value decomposition. For that purpose, let us rewrite (34) into:

$$\mathbf{E} = \begin{bmatrix}\mathbf{u}_1 & \mathbf{u}_2 & \mathbf{u}_3\end{bmatrix} \operatorname{diag}(1,1,0) \begin{bmatrix}\mathbf{v}_1 & \mathbf{v}_2 & \mathbf{v}_3\end{bmatrix}^T \qquad (35)$$

where the sets $\mathbf{u}_j$ are orthogonal unit vectors. And so is the set $\mathbf{v}_j$. Note that according to (35). Thus, $\mathbf{E}\mathbf{v}_3 = \mathbf{0}$. Thus $\mathbf{v}_3$ spans the null space of $\mathbf{E}$.

Finding the displacement vector

The matrix $\mathbf{T}$ is defined as the cross product operator, so that for any $\mathbf{x}$, we have: $\mathbf{T}\mathbf{x} = {}^{1}\mathbf{t}_2 \times \mathbf{x}$. It follows:

$$\mathbf{E}\,{}^{1}\mathbf{t}_2 = {}^{2}\mathbf{R}_1\mathbf{T}\,{}^{1}\mathbf{t}_2 = {}^{2}\mathbf{R}_1\,{}^{1}\mathbf{t}_2 \times {}^{1}\mathbf{t}_2 = \mathbf{0} \qquad (36)$$

Apparently, the vector ${}^{1}\mathbf{t}_2$ is in the null space of $\mathbf{E}$. Comparing this with (35), we conclude:

$${}^{1}\mathbf{t}_2 = \alpha\mathbf{v}_3 \quad \text{with} \quad \alpha \neq 0 \qquad (37)$$

Corollary 1:

The displacement ${}^{1}\mathbf{t}_2$ points either in the direction of the unit vector $\mathbf{v}_3$, or in the opposite direction $-\mathbf{v}_3$.

Finding the rotation matrix

By rotating the coordinate system, we immediately see that the following general property of the cross product between two arbitrary vectors $\mathbf{a}$ and $\mathbf{b}$ holds:

$$(\mathbf{R}\mathbf{a}) \times (\mathbf{R}\mathbf{b}) = \mathbf{R}(\mathbf{a} \times \mathbf{b}) \qquad (38)$$

where $\mathbf{R}$ is any rotation matrix. We use this property to derive the rotation matrix ${}^{2}\mathbf{R}_1$ from the essential matrix. Point of departure are the following equations:

$$\begin{aligned} \mathbf{E}\mathbf{x} &= {}^{2}\mathbf{R}_1(\alpha\mathbf{v}_3 \times \mathbf{x}) \\ &= \begin{bmatrix}\mathbf{u}_1 & \mathbf{u}_2 & \mathbf{u}_3\end{bmatrix}\begin{bmatrix}\mathbf{v}_1 & \mathbf{v}_2 & \mathbf{0}\end{bmatrix}^T \mathbf{x} \end{aligned} \qquad (39)$$

Here, $\mathbf{x}$ is an arbitrary vector. The SVD $\mathbf{E} = \mathbf{U}\operatorname{diag}(1,1,0)\mathbf{V}^T$ has been used, implying that $\mathbf{V}^T$ is a rotation matrix. Also (37) has been used. Without loss of generality, we suppose that ${}^{2}\mathbf{R}_1$ can be written as the product ${}^{2}\mathbf{R}_1 = \mathbf{U}\mathbf{W}^T$ in which $\mathbf{W}^T = \begin{bmatrix}\mathbf{w}_1 & \mathbf{w}_2 & \mathbf{w}_3\end{bmatrix}^T$ is another rotation matrix. Combining (38) with (39), we arrive at:

$$\begin{aligned} \mathbf{E}\mathbf{x} &= \alpha\mathbf{U}\left(\mathbf{W}^T\mathbf{v}_3 \times \mathbf{W}^T\mathbf{x}\right) \\ &= \alpha\mathbf{U}\left(\begin{bmatrix}\mathbf{w}_1^T\mathbf{v}_3 & \mathbf{w}_2^T\mathbf{v}_3 & \mathbf{w}_3^T\mathbf{v}_3\end{bmatrix}^T \times \mathbf{W}^T\mathbf{x}\right) \end{aligned} \qquad (40)$$

This expression simplifies considerably if we suppose that $\mathbf{w}_3 = \mathbf{v}_3$, so that $\mathbf{w}_1^T\mathbf{v}_3 = 0$ and $\mathbf{w}_2^T\mathbf{v}_3 = 0$ as $\mathbf{W}$ is a rotation matrix. Substitution yields:

$$\mathbf{E}\mathbf{x} = \alpha\mathbf{U}\left(\begin{bmatrix}0 & 0 & 1\end{bmatrix}^T \times \mathbf{W}^T\mathbf{x}\right) \qquad (41)$$

Further expansion of $\mathbf{W}^T\mathbf{x}$ and using the properties of the cross product, i.e. equation (38) of the syllabus “Math for Computer Vision and Navigation”, gives:

$$\begin{aligned} \mathbf{E}\mathbf{x} &= \alpha\mathbf{U}\left(\begin{bmatrix}0 & 0 & 1\end{bmatrix}^T \times \begin{bmatrix}\mathbf{w}_1^T\mathbf{x} & \mathbf{w}_2^T\mathbf{x} & \mathbf{w}_3^T\mathbf{x}\end{bmatrix}^T\right) \\ &= \alpha\mathbf{U}\begin{bmatrix}-\mathbf{w}_2^T\mathbf{x} & \mathbf{w}_1^T\mathbf{x} & \mathbf{0}\end{bmatrix}^T \\ &= \alpha\mathbf{U}\begin{bmatrix}-\mathbf{w}_2^T & \mathbf{w}_1^T & \mathbf{0}\end{bmatrix}^T \mathbf{x} \end{aligned} \qquad (42)$$

Comparing this with (39), we conclude that a valid solution is $\mathbf{w}_2 = -\mathbf{v}_1$ and $\mathbf{w}_2 = \mathbf{v}_1$, so that the solution becomes:

$${}^{2}\mathbf{R}_1 = \mathbf{U}\begin{bmatrix}-\mathbf{v}_2 & \mathbf{v}_1 & \mathbf{v}_3\end{bmatrix}^T$$

Unfortunately, this solution is not unique as we can change the signs of $\mathbf{w}_1$ and $\mathbf{w}_2$ and still meet all conditions. We therefore have the following corollary:

Corollary 2:

The rotation matrix ${}^{1}\mathbf{R}_2$ has two possible solutions:

$${}^{2}\mathbf{R}_1 = \mathbf{U}\begin{bmatrix}-\mathbf{v}_2 & \mathbf{v}_1 & \mathbf{v}_3\end{bmatrix}^T \qquad (43)$$

$${}^{2}\mathbf{R}_1 = \mathbf{U}\begin{bmatrix}\mathbf{v}_2 & -\mathbf{v}_1 & \mathbf{v}_3\end{bmatrix}^T \qquad (44)$$

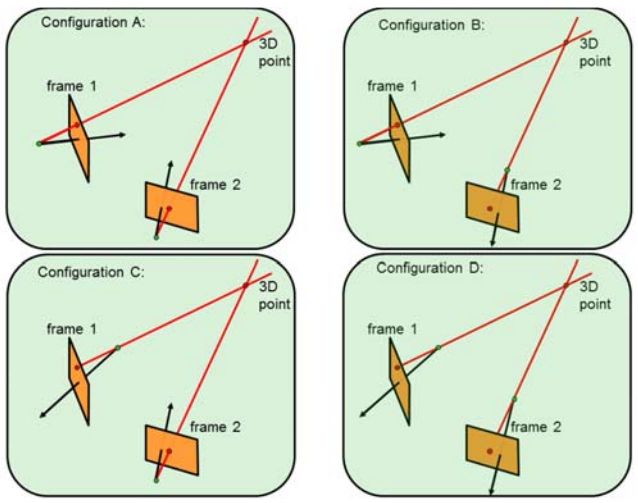
Resolving the ambiguities

The pose of the 2nd camera frame with respect to the 1st camera frame, knowing 8 or more corresponding points, can be estimated to a certain extent:

- Only the direction of the displacement vector between the two cameras, i.e. the baseline vector, can be recovered. There is a twofold ambiguity with respect to this direction as the displacement can also be in opposite position.

- The orientation of one camera with respect to the other camera can be estimated, but here too there is a twofold ambiguity since the direction of the optical axis can be in opposite direction.

Together, there is a four fold ambiguity. This ambiguity stems from the fact that our perspective camera model does not prescribe that imaged 3D points are lying in front of the cameras or at the back of the cameras. This geometrical interpretation of the four possible solutions is shown below. The ambiguity can be solved by involving the reconstruction of 3D points. In the figure above, only configuration A is ok. We have to select the configuration of which all reconstructed 3D points are lying in front of the cameras. This resolves the ambiguity of the rotation matrix, and it resolves the direction of the displacement, but the magnitude of the displacement is still unknown.



Appendices

1 References

[1] C. P. Lu, G. D. Hager, and E. Mjolsness, "Fast and globally convergent pose estimation from video images," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 22, pp. 610-622, 2000.