Task 1: Calibration of RGB-D Sensor and Thermo Camera

M. Pecka, V. Kubelka, K. Zimmermann and M. Reinstein

February 27, 2014

1 Task 1 assignment

In this task you will be given images captured by a Microsoft-Kinect-like sensor (see Figure 1, 1st and 2nd image) and by a thermo-camera (see Figure 1, 3rd image). Your main task is to find transformation matrix between the RGB image pixel coordinates and corresponding pixel coordinates in the thermo-camera image (for those pixels for which the transformation exists). This procedure is known as calibration and an example result is depicted in Figure 2.

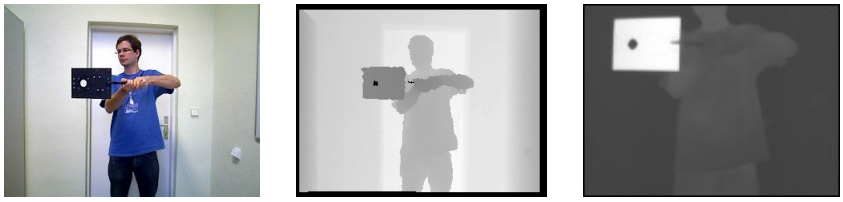

Figure 1: Three types of images provided in the *rosbag* file: a standard color image (left), a depth image (middle) and a thermo-cam image (right). The assistant is holding a hot metal sheet, that can be easily identified in all three image types.

1.1 Motivation

Since the thermo-camera and the depth sensor are available, it is desirable to enrich the information the standard color image provides by temperature and distance of the observed objects. This information can be later used for smarter detection of interesting objects in the camera image. However, images provided by the sensors are not the same size nor aligned since the sensors are mounted next to each other so they observe the scene from slightly different angles. That is the motivation to find a transformation that binds pixels of the images together.

Figure 2: An example of the calibration result. Pixels corresponding to the thermo-camera field-of-view are colored according to their temperature (blue is cold, red is hot).

1.2 System description

There are two sensors at your disposal for Task 1. The first sensor is the *ASUS Xtion PRO LIVE* providing standard color images (320x240 pixels) as well as depth images of the same size (a combination of these two images is denoted as RGB-D). These two images are already calibrated for you – there is both the RGB and depth information at all pixel coordinates. The depth information is expressed in *millimeters*, zero depth indicates no depth information at that particular pixel.

The second sensor is a thermo-camera, namely *thermoIMAGER TIM 160* from MICRO-EPSILON. This camera captures infrared images (160x120 pixels), where the value of each pixel is the temperature of the corresponding surface observed by the camera (approximate temperature in our case, emissivity is not taken into account).

These two sensors are attached to a common holder (see Figure 3), yet the exact configuration is not known (by the configuration, we mean rotation and translation of the camera optical centers). The only known parameter is the RGB-D camera *calibration matrix* K; refer to Section 1.4. This parameter is sufficient – combined with the depth information – to project each pixel of the RGB-D camera to corresponding 3D space coordinates. That is, for each color-depth image pair, you are able to get $320 \times 240 = 76800$ $X_{rgbd} = [x, y, z]^T$ points that create a colored *point-cloud*. The images are recorded using ROS, therefore the output provided to you is a *BAG* file containing all the images as messages published at appropriate topics. The BAG file is included in the provided ROS package: `data/task1.bag`. The topics necessary for calibration are

- `/openni_camera/depth`: depth images
- `/openni_camera/rgb/compressed`: JPEG compressed color images
- `/openni_camera/camera_info`: metadata containing the `K` matrix

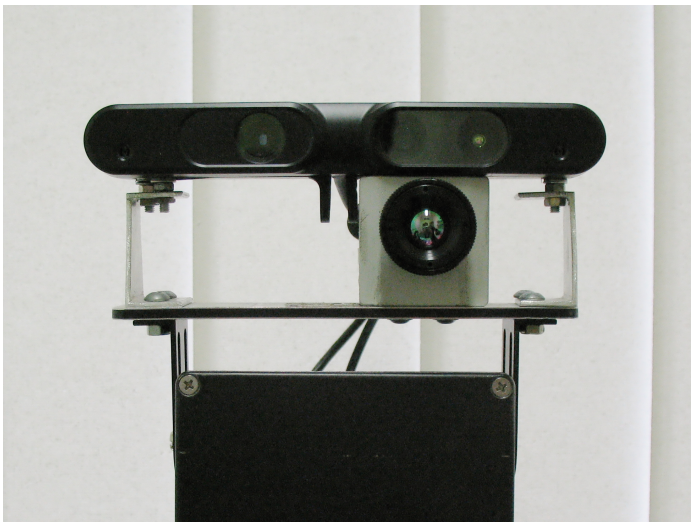

Figure 3: The two sensors involved: *ASUS Xtion PRO LIVE* (top), *thermoIMAGER TIM 160* (bottom).

- `/thermo_camera/ftemp`: the thermo-camera images

Feel free to call `rostopic list -v` to get a detailed list of all topics available and `rostopic info TOPIC_NAME` to get more information about the chosen topic. Of course the relevant topics will only be published when the BAG file is played – you may find the `-l` option of `rosbag play` helpful when examining the topics.

TECHNICAL NOTE: Due to technical reasons (buffering at the low-level drivers), the images are not synchronized, even though time stamps of the corresponding ROS messages may indicate the opposite. It is necessary to consider this fact during calibration procedure and select only the images captured while the assistant is not moving.

1.3 Calibration dataset

To perform the calibration it is necessary to identify correspondences between the camera images. You are provided with a *BAG* file containing a calibration dataset, see Figure 1. There are three images that depict an assistant holding a hot metal sheet. The metal sheet is held at one spot for several seconds and then moved to another one, this procedure is repeated several times so the view field of the thermo-camera is roughly covered while the sheet is held at several distances from the camera. This ensures that there are enough correspondences – the corners of the metal sheet, for example – to successfully perform the calibration.

1.4 Pinhole camera model

During the second IRO lecture, a general model of 3D to 2D was presented. The convention adopted by ROS (and actually OpenCV http://opencv.org/) differs in coordinate frame orientation, as described in Figure 4. It demonstrates the

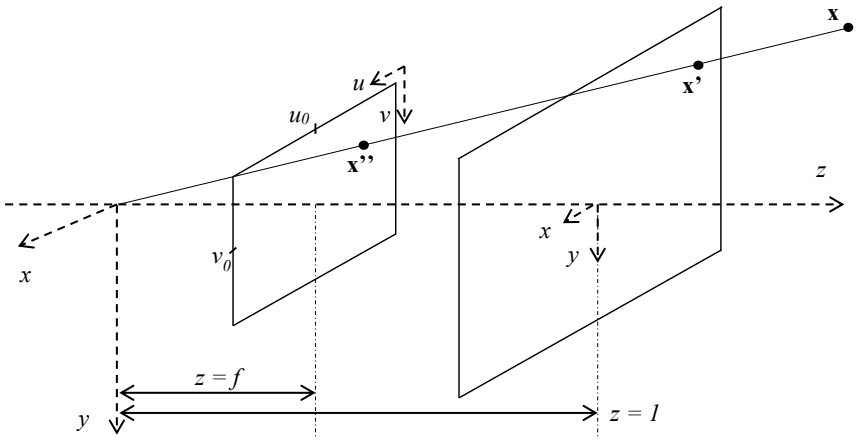

Figure 4: Projection from a 3D scene to an image plane by the ROS (OpenCV) conventions.

usage of camera calibration matrix `K`: a 3D point $\mathbf{x} = [x, y, z]^T$ is projected onto a plane where $z = 1$:

$$\mathbf{x}' = \begin{bmatrix} x/z \\ y/z \\ 1 \end{bmatrix} \tag{1}$$

to obtain $\mathbf{x}'$. Left-multiplication by the camera calibration matrix `K` will project $\mathbf{x}'$ to the image plane

$$\mathtt{K}\mathbf{x}' = \begin{bmatrix} fm_u & 0 & u_0 \\ 0 & fm_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \mathbf{x}' = \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{x}'' \\ 1 \end{bmatrix} \tag{2}$$

where the $[u, v]$ coordinates are expressed in pixels and m_u,m_v stand for pixels per distance unit (e.g. px/m) in the image plane (sensor); u_0,v_0 are image coordinates of the center of the camera plane and f is the focal length of the camera.

This approach can be generalized using homogeneous coordinates (see lecture

[2]) to

$$\begin{bmatrix} u \\ v \\ w \end{bmatrix} = \begin{bmatrix} fm_u & 0 & u_0 & 0 \\ 0 & fm_v & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ q \end{bmatrix} = \mathtt{K}[\mathtt{I}|\mathbf{0}] \begin{bmatrix} x \\ y \\ z \\ q \end{bmatrix} \tag{3}$$

where u, v, w are homogeneous coordinates in the image plane, $\mathtt{I}$ is an identity matrix and $\mathbf{0}$ is a zero vector.

It is important not to forget to correctly interpret the result, since the real image coordinates are $[\frac{u}{w}, \frac{v}{w}]^T$. Inversely, if we are given the image coordinates and we want to express them in homogeneous coordinates, we just set $w = 1$, which results in homogeneous point $[u, v, 1]^T$.

The homogeneous coordinates $[x, y, z, q]^T$ relate to a 3D point expressed in the coordinate frame of the camera, as shown in Figure 4. Since they are homogeneous, the same rule applies as in the previous case, if we construct the vector from known values, we set $q = 1$. To convert the homogeneous vector to a 3D vector, we simply multiply it by $\frac{1}{q}$.

Note that further on, we will denote points in homogeneous coordinates by upper case letters. So e.g. a 3D point $\mathbf{x}$ corresponds to point $\mathbf{X}$ in homogeneous coordinates.

In the case a 3D point is not expressed in the coordinate frame of the camera (that is our case with the RGB-D sensor points we want to project to the thermo-camera image) it needs to be transformed to the desired frame first:

$$\mathbf{x}_{thermo} = \mathtt{R}\mathbf{x}_{rgb} + \mathbf{t} \tag{4}$$

where $\mathtt{R}$ and $\mathbf{t}$ are appropriate rotation and translation respectively. It is handy to chain the transformation and rotation operations together, yielding the *camera projection matrix* $\mathtt{P}$:

$$\mathtt{P}\mathbf{X}_{rgb} = \mathtt{K}\,[\mathtt{R}|\mathbf{t}]\,\mathbf{X}_{rgb} = \mathtt{K}\,[\mathtt{R}|\mathbf{t}] \begin{bmatrix} x \\ y \\ z \\ q \end{bmatrix}_{rgb} = \begin{bmatrix} u \\ v \\ w \end{bmatrix}_{thermo} = \mathbf{X}_{thermo} \tag{5}$$

The projection matrix projects 3D points expressed in the *rgb* frame to the image plane of the thermo-camera. The next section explains a way how it can be obtained from known correspondences.

1.5 Camera projection matrix P retrieval

As was explained in the previous section, the projection matrix $\mathtt{P}$ operates on homogeneous coordinates and projects a 3D point to the image plane as $\mathbf{X}' = \mathtt{P}\mathbf{X}$, $\mathbf{X}' = [u, v, w]^T$, $\mathbf{X} = [x, y, z, q]^T$. Our task – given known correspondences $\mathbf{X}_i, \mathbf{X}'_i$ – is to estimate this matrix. Note that while substituting for $\mathbf{X}'_i$ and $\mathbf{X}_i$ by the actual measured values, we have to set w and q equal to 1 to comply the definition of the homogeneous coordinates. We will follow approach presented in

[1]: since both $\mathbf{X}_i, \mathbf{X}'_i$ are expressed in homogeneous coordinates, their magnitude may differ based on scale, so we cannot use the relation $\mathbf{X}'_i = \mathtt{P}\mathbf{X}_i$ directly. Nevertheless, the directions of our given point $\mathbf{X}'_i$ and the projected point $\mathtt{P}\mathbf{X}_i$ will be the same. A good test whether two vectors are parallel or not is the cross product, which must result in a zero vector if the two operands are parallel:

$$\mathbf{X}'_i \times \mathtt{P}\mathbf{X}_i = \mathbf{0} \tag{6}$$

This leads to linear equations for $\mathtt{P}$ to be derived. We can rewrite $\mathtt{P}\mathbf{X}_i$ to a more usable form if we denote j-th row of matrix $\mathtt{P}$ by $\mathbf{p}^{jT}$:

$$\mathtt{P}\mathbf{X}_i = \begin{bmatrix} \mathbf{p}^{1T}\mathbf{X}_i \\ \mathbf{p}^{2T}\mathbf{X}_i \\ \mathbf{p}^{3T}\mathbf{X}_i \end{bmatrix} \tag{7}$$

Since $\mathbf{X}'_i = [u_i, v_i, w_i]^T$, we can explicitly express the cross product:

$$\mathbf{X}'_i \times \mathtt{P}\mathbf{X}_i = \begin{bmatrix} v_i\mathbf{p}^{3T}\mathbf{X}_i - w_i\mathbf{p}^{2T}\mathbf{X}_i \\ w_i\mathbf{p}^{1T}\mathbf{X}_i - u_i\mathbf{p}^{3T}\mathbf{X}_i \\ u_i\mathbf{p}^{2T}\mathbf{X}_i - v_i\mathbf{p}^{1T}\mathbf{X}_i \end{bmatrix} = \mathbf{0} \tag{8}$$

The order of the variables can be swapped using transposition as $\mathbf{p}^{jT}\mathbf{X}_i = \mathbf{X}_i^T\mathbf{p}^j$, so the previous equation can be rewritten to:

$$\begin{bmatrix} \mathbf{0}^T & -w_i\mathbf{X}_i^T & v_i\mathbf{X}_i^T \\ w_i\mathbf{X}_i^T & \mathbf{0}^T & -u_i\mathbf{X}_i^T \\ -v_i\mathbf{X}_i^T & u_i\mathbf{X}_i^T & \mathbf{0}^T \end{bmatrix} \begin{bmatrix} \mathbf{p}^1 \\ \mathbf{p}^2 \\ \mathbf{p}^3 \end{bmatrix} = \mathbf{0} \tag{9}$$

where $[\mathbf{p}^{1T}, \mathbf{p}^{2T}, \mathbf{p}^{3T}]^T$ is a 12-element vector corresponding to the original $\mathtt{P}$ matrix:

$$\mathtt{P} = \begin{bmatrix} p_1 & p_2 & p_3 & p_4 \\ p_5 & p_6 & p_7 & p_8 \\ p_9 & p_{10} & p_{11} & p_{12} \end{bmatrix} \tag{10}$$

Note that for solving this equation, only the first and second rows are necessary since the third one is a linear combination of the two previous ones. Therefore, the final equation can be reduced to this form:

$$\begin{bmatrix} \mathbf{0}^T & -w_i\mathbf{X}_i^T & v_i\mathbf{X}_i^T \\ w_i\mathbf{X}_i^T & \mathbf{0}^T & -u_i\mathbf{X}_i^T \end{bmatrix} \begin{bmatrix} \mathbf{p}^1 \\ \mathbf{p}^2 \\ \mathbf{p}^3 \end{bmatrix} = \mathbf{0} \tag{11}$$

This equation is in form $\mathtt{A}_i\mathbf{p} = \mathbf{0}$. By substituting for corresponding 3D and image coordinates to $\mathtt{A}_i$, one correspondence pair $\mathbf{X}_i, \mathbf{X}'_i$ is utilized to obtain 2 equations. Since there are 12 unknown members of $\mathtt{P}$, at least 6 correspondence pairs are needed to solve the problem. The equations can be stacked to form one equation system

$$\mathtt{A}\mathbf{p} = \begin{bmatrix} \mathtt{A}_1 \\ \mathtt{A}_2 \\ \mathtt{A}_3 \\ \vdots \end{bmatrix} \mathbf{p} = \mathbf{0} \tag{12}$$

To make to computation more robust against measurement noise, more than 6 correspondences are preferred, yet we obtain an overdetermined system of equations. The next section discusses one way to solve such a system.

1.6 Solution of the overdetermined homogeneous linear system

We search for a non-trivial solution $\mathbf{p} \in \mathbb{R}^n$ of the overdetermined homogeneous linear system

$$\mathtt{A}\mathbf{p} = \mathbf{0},$$

where *non-trivial* means $\mathbf{p} \neq 0$ and *overdetermined* means that there are more independent equations than unknowns (i.e. dim rng($\mathtt{A}$) $\geq n$). To avoid the trivial solution we constrain the solution on the unit sphere, i.e. $\|\mathbf{p}\| = 1$. Of course, there is no exact non-trivial solution of such overdetermined system, therefore approximate solution minimizing a suitable cost function is sought. Ideally we would like to minimize geometric distance, however minimization of such cost functions often leads to iterative algorithms. The simplest solution is to minimize algebraic distance $\|\mathtt{A}\mathbf{p}\|$ which leads to the following constrained least-squares problem

$$\mathbf{p}^* = \arg\min_{\mathbf{p}} \|\mathtt{A}\mathbf{p}\|, \text{ subject to } \|\mathbf{p}\| = 1. \tag{13}$$

This problem has closed-form solution, which is equal to the eigen-vector of $\mathtt{A}^\top\mathtt{A}$ with the smallest corresponding eigen-value.

Unfortunatelly, the minimization of the algebraic distance is strongly depedent on the origin and scale of the Euclidean coordinate system in which the correspondence pairs $\mathbf{x} = [x\ y\ z]^\top \in \mathbb{R}^3$ and $\mathbf{x}' = [u\ v]^\top \in \mathbb{R}^2$ (corresponding to homogeneous coordinates X and X') are expressed. To supress such undesirable property, we *normalize* coordinate system by transforming points $\mathbf{x}$ to a new set of points $\mathbf{y}$ such that the centroid of the points $\mathbf{y}$ is the coordinate origin and the average distance from the origin is $\sqrt{3}$ (then, the "average" point's distance is the same as the distance of point [1,1,1]). For example, points $\mathbf{y}_i$ are computed from points $\mathbf{x}_i$ as follows:

$$\mathbf{y}_i = \sqrt{3}\Big(\frac{\mathbf{x}_i - \overline{\mathbf{x}}}{\sigma}\Big), \tag{14}$$

where $\overline{\mathbf{x}} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{x}_i$ is the centroid and $\sigma = \frac{1}{N}\sum_{i=1}^{N}\|\mathbf{x}_i - \overline{\mathbf{x}}\|$ is the average distance. Similarly, points $\mathbf{y}'_i$ are computed from points $\mathbf{x}'_i$ as $\mathbf{y}'_i = \sqrt{2}\Big(\frac{\mathbf{x}'_i - \overline{\mathbf{x}'}}{\sigma'}\Big)$. Therefore, it is higly desirable to build matrix $\mathtt{A}$ from normalized points $\mathbf{y}_i$ and $\mathbf{y}'_i$ and then compute the eigen-vectors of that "normalized" matrix.

For the sake of completeness, derivation of the solution of Problem (13) is provided in the rest of this section. To solve the constrained least squares problem, we first introduce Lagrange function

$$\begin{aligned} L(\mathbf{p}, \lambda) &= \|\mathtt{A}\mathbf{p}\| + \lambda(1 - \|\mathbf{p}\|) = & (15)\\ &= \mathbf{p}^\top\mathtt{A}^\top\mathtt{A}\mathbf{p} + \lambda(1 - \mathbf{p}^\top\mathbf{p}). & (16) \end{aligned}$$

and search for its critical points (i.e. points in which local extrema can be achieved) by equaling its derivatives to zero

$$\frac{\partial L(\mathbf{p},\lambda)}{\partial \mathbf{p}} = 2\mathtt{A}^\top\mathtt{A}\mathbf{p} - 2\lambda\mathbf{p} = \mathbf{0} \quad (17)$$

$$\frac{\partial L(\mathbf{p},\lambda)}{\partial \lambda} = 1 - \mathbf{p}^\top\mathbf{p} = \mathbf{0}. \quad (18)$$

Equation (17) is simply rearranged as the characteristic equation of $\mathtt{A}^\top\mathtt{A}$

$$(\mathtt{A}^\top\mathtt{A} - \lambda\mathtt{I})\mathbf{p} = \mathbf{0}, \quad (19)$$

therefore every eigen-vector $\mathbf{p}$ of $\mathtt{A}^\top\mathtt{A}$ with corresponding eigen-values λ satisfy this equation and the one which yields the smallest cost value $\|\mathtt{A}\mathbf{p}\|$ of Problem (13) has to be chosen. Using Equation (19) and the contraint (18), the following simiplification is derived

$$\|\mathtt{A}\mathbf{p}\| = \mathbf{p}^\top\mathtt{A}^\top\mathtt{A}\mathbf{p} = \mathbf{p}^\top\lambda\mathbf{p} = \lambda\mathbf{p}^\top\mathbf{p} = \lambda\|\mathbf{p}\| = \lambda.$$

It reveals that the solution of Problem (13) is the eigen-vector of $\mathtt{A}^\top\mathtt{A}$ with the smallest eigen-value.

In `numpy`, you can find the eigen-values and eigen-vectors using the function `eigenvalues,eigenvectors=numpy.linalg.eig(A)`. Please note that the eigen-values are not ordered. To order both the values and the vectors, you can use `idxs = eigenvalues.argsort()` to get the order of indices needed to sort either `eigenvalues` or the *columns* of `eigenvectors`.

2 Task to be solved

The following steps have to taken to successfully accomplish the calibration. Your procedure has to be completely documented in the submitted report. Pay attention especially to the experimental evaluation of your results:

1. Obtain a sufficient number of suitable RGB-D and thermo camera image triplets
   - There is a ROS node prepared for this purpose: `rosrun task1 image_recorder.py _num_frames:=???`.
   - It requires the `task1.bag` rosbag file being played manually (but first run the node!).
   - You need to find out by yourself what number to use instead of the `???` in the `rosrun` command parameter. `rosbaginfo` may be useful here. The parameter was introduced because there is no way to tell the node the playback has stopped.
   - Read through the code and fill missing parts of the code according to the code function and purpose. All parts of the code needing your attention are labelled by `# IMPLEMENT` comment.

- The output of this node should be four files containing all the images necessary: `data/thermo_images.mat`, `data/rgb_images.mat`, `data/depth_images.mat`, `data/kinect_K.mat`
- The output files will be utilized by another node that is prepared as well, but can be also opened and inspected using Mathworks MATLAB.

2. Manually mark correspondences between the RGB-D and thermo camera images
   - There is ROS node prepared to help you select and mark correspondences between the color images and the thermo-camera images. The node is run by `rosrun task1 create_correspondences.py` .
   - The code provided to you is incomplete on purpose. Your task is to understand the code and fill the missing parts.
   - Modify the code so it utilizes only those images where the hot sheet does not move to make sure that the correspondences are valid. The reason is relatively slow sampling rate of the cameras compared to the velocity of the moving metal sheet.
3. Evaluate 3D coordinates of points corresponding to the marked RGB-D camera image pixels
   - For this evaluation use `create_correspondences.py` as well
   - Theoretical background can be found in Section 1.4, the equation (2) describes an inverted problem, it determines image coordinates $\mathbf{x}''$ when the 3D point $\mathbf{x}'$ is known. In our case, we need to express $\mathbf{x}'$ when $\mathbf{x}''$ is provided. Modify the equation appropriately.
   - The values stored in the depth image are expressed in millimeters, the value corresponds to the z part of the 3D point. Use this value to obtain the final 3D point $\mathbf{x}$ from $\mathbf{x}'$ – see equation (1).
   - Document your solution in the report.
4. Normalize both the 3D coordinates and the 2D thermo camera pixel coordinates to increase numerical stability of the following step
   - The normalization step for 3D points is described by equation (14), equation for normalizing 2D points is provided in the section as well.
   - The normalization takes place in the `compute_camera_pose.py` executable.
5. Find the thermo-camera *projection matrix* P using the normalized 3D-2D correspondences
   - The code skeleton is prepared in `compute_camera_pose.py` .

6. Verify the obtained matrix `P` by re-projecting the 3D points to the 2D coordinates and by evaluating reprojection error

   - Reprojection error stands for the distance in the 2D image between the manually marked point and the resulting point obtained by applying the projection matrix `P` on the corresponding 3D point.
   - Project all the manually marked 3D points using your projection matrix `P` and compare the results to the manually marked 2D points in the thermo-cam image – evaluate the reprojection errors.
   - To visually inspect the effect of matrix `P` to the projection, you can utilize the prepared skeleton in file `reprojection_viewer.py` .
   - Propose a way to statistically assess your results and document them in your report.
   - Please note: reports submitted without the evaluation of results using the reprojection error will be marked as incomplete and cannot be accepted. After revising and resubmitting such report, penalty for late submission will be applied if the resubmission is done late. The standard time it takes to evaluate a report is one week and you do not have a right to demand us do it faster.

2.1 Format of the submitted data

In the Upload system you are expected to upload both your codes and the report. Together with your codes include an ASCII text file containing the unnormalized matrix `P`. The numbers have to be in normal (e.g. not scientific or exponential) format and they should print with 5 decimal digits. The matrix elements should be separated by either spaces or tabs. Use Linux line endings. For example:

```
1.00000  0.00000  0.00000  0.00000
0.00000  1.00000  0.00000  0.00000
0.00000  0.00000  1.00000  0.00000
```

References

[1] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second edition, 2004.

[2] Vaclav Hlavac. Digital image, basic concepts. http://cmp.felk.cvut.cz/ hlavac/TeachPresEn/17CompVision3D/11Geom1camera.pdf.