Computer Vision Group
TUM Department of Informatics

Exercise Sheet 1

Topic: Image Formation, Extended Kalman Filter

Exercise 1.1: Image Formation

In this exercise, you will implement a basic image projection using the pinhole camera model and image undistortion in Matlab.

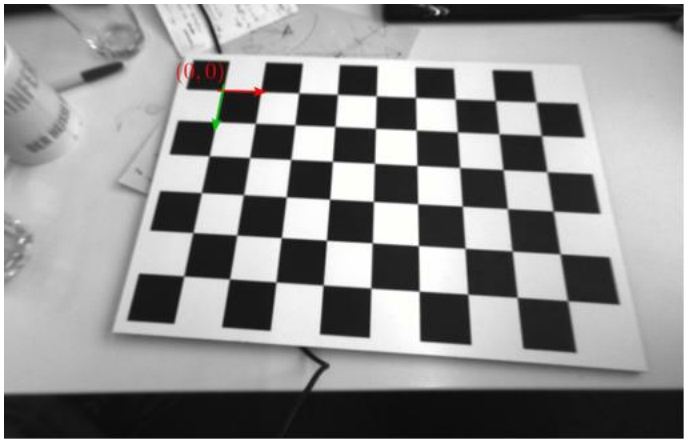

Figure 1: Checkerboard coordinate frame.

a) Download the image data for this part of the exercise from the course webpage. The archive contains two images **`sample_undistorted.png`** and **`sample_distorted.png`**. Read the sample images (**`imread`**) and convert them to grayscale using **`rgb2gray`**.

b) Start with the undistorted sample image and draw all the corner points on the checkerboard pattern by projecting their 3D coordinates into the image. For this, we define a coordinate frame on the checkerboard with the origin in the upper left inner corner of the pattern (see Fig. 1).
The x-axis is parallel to the longer side of the pattern with increasing values towards the right in the sample image.
The y-axis points downwards along the shorter side of the pattern in the image. The length between the corners on the checkerboard is 0.04 m.
The transformation from checkerboard frame to camera frame is given by
$\omega_1 = -0.372483192214$, $\omega_2 = 0.0397022486165$, $\omega_3 = 0.0650393402332$,
$t_1 = -0.107035863625$, $t_2 = -0.147065242923$, $t_3 = 0.398512498053$,
where $(\omega_1, \omega_2, \omega_3)$ defines the rotation in the axis-angle representation and (t_1, t_2, t_3) the translation vector in meters.
The camera intrinsics for the undistorted image are specified by the focal lengths
$f_x = 420.506712$, $f_y = 420.610940$
and the principal point
$c_x = 355.2082980$, $c_y = 250.3367870$.

For projecting the points on the checkerboard, first create the set of 3D points in the checkerboard frame. You can use the function **`meshgrid`**. Transform the points into the camera frame and project them to pixel coordinates on the image plane according to the pinhole camera model by using the given camera intrinsics.
Overlay the projected points with the image.

c) Now repeat the projection and visualization of the checkerboard corners from the previous step in the distorted image. To this end, you need to apply the following distortion model to the normalized image coordinates $\bar{\boldsymbol{y}}$ before mapping them to pixel coordinates using the camera matrix $\bar{\boldsymbol{y}}_p = \boldsymbol{C}\bar{\boldsymbol{y}}_d$:

$$\boldsymbol{y}_d = (1 + k_1 r^2 + k_2 r^4)\boldsymbol{y}, \quad r := \|\boldsymbol{y}\|_2.$$

The distortion parameters are $k_1 = -0.296609$ and $k_2 = 0.080818$.

d) Determine the horizontal and vertical field of view of the camera, measured horizontally and vertically across the image center in the distorted and the undistorted case.

Hint: Use the following iterative approach to determine undistorted from distorted image coordinates:

function Undistort($\boldsymbol{y}_d$)
  $t \leftarrow 0$
  $y_t \leftarrow \boldsymbol{y}_d$
  repeat
    $r = \|y_t\|_2$
    $r_d = (1 + k_1 r^2 + k_2 r^4)$
    $\boldsymbol{y}_{t+1} = \frac{1}{r_d}\boldsymbol{y}_d$
    $t \leftarrow t + 1$
  until $\|\boldsymbol{y}_t - \boldsymbol{y}_{t-1}\|_2 < threshold$
  return $\boldsymbol{y}_t$
end function

Exercise 1.2: Extended Kalman Filter

In this exercise, you will implement a robot localization algorithm based on the Extended Kalman Filter. We assume the robot moves in the 2D plane, for example, a wheeled robot with differential drive that moves on the floor inside a building. This means the robot state $\boldsymbol{x}_t = (x_t, y_t, \theta_t)^T$ is 3-dimensional and composed of the 2-dimensional position x_t, y_t in the plane and the robot's heading θ_t. We model the robot motion with an odometry-based motion model in this exercise, i.e. the state transition model is

$$\boldsymbol{x}_t = g(\boldsymbol{x}_{t-1}, \boldsymbol{u}_t) + \boldsymbol{\epsilon}_t := \boldsymbol{x}_{t-1} + \begin{pmatrix} u_{tr}\cos(\theta_{t-1} + u_{r1}) \\ u_{tr}\sin(\theta_{t-1} + u_{r1}) \\ u_{r1} + u_{r2} \end{pmatrix} + \boldsymbol{\epsilon}_t, \quad \boldsymbol{\epsilon}_t \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{\Sigma}_{dt}).$$

The action $\boldsymbol{u}_t = (u_{tr}, u_{r1}, u_{r2})$ is given by translational u_{tr} and rotational (u_{r1}, u_{r2}) motion measurements obtained from wheel odometry. For the noise covariance of the state-transitions, we assume

$$\boldsymbol{\Sigma}_{dt} = \begin{pmatrix} 0.1 & 0 & 0 \\ 0 & 0.1 & 0 \\ 0 & 0 & 0.01 \end{pmatrix}.$$

The robot measures the range r and bearing ϕ to 2D landmark points $\boldsymbol{l}_j = (l_{j,x}, l_{j,y})$ in the environment in the horizontal plane. It measures multiple landmarks in a time step for which we assume the association $c_{t,i} = j$ of measurements $\boldsymbol{z}_{t,i} = \left(r_{t,i}, \phi_{t,i}\right)^T$ to landmarks j known. The observation model is

$$\boldsymbol{z}_{t,i} \;=\; h\left(\boldsymbol{x}_t, c_{t,i}\right) + \boldsymbol{\delta}_{t,i} := \begin{pmatrix} \left\| (x_t, y_t)^T - \left(l_{j,x}, l_{j,y}\right)^T \right\|_2 \\ \operatorname{atan2}\left(l_{j,y} - y_t, l_{j,x} - x_t\right) - \theta_t \end{pmatrix} + \boldsymbol{\delta}_{t,i} \quad \boldsymbol{\delta}_{t,i} = \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_{mt,i}).$$

For the observation noise of an individual landmark measurement, we assume

$$\boldsymbol{\Sigma}_{mt,i} = \begin{pmatrix} 0.1 & 0 \\ 0 & 0.1 \end{pmatrix}.$$

The complete observation model in each time step, $\boldsymbol{z}_t = h(\boldsymbol{x}_t) + \boldsymbol{\delta}_t$ with $\boldsymbol{\delta}_t = \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_{mt})$ stacks the M measurements in a single vector $\boldsymbol{z}_t = \left(\boldsymbol{z}_{t,0}^T, \ldots, \boldsymbol{z}_{t,M-1}^T\right)^T \in \mathbb{R}^{2M}$. Analogously, we write $h(\boldsymbol{x}_t) := \left(h\left(\boldsymbol{x}_t, c_{t,0}\right)^T, \ldots, h\left(\boldsymbol{x}_t, c_{t,M-1}\right)^T\right)^T$. The covariance $\boldsymbol{\Sigma}_{mt}$ is formed from the individual measurement covariances,

$$\boldsymbol{\Sigma}_{mt} = \begin{pmatrix} \boldsymbol{\Sigma}_{mt,0} & \mathbf{0} & \ldots & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\Sigma}_{mt,1} & \ddots & \vdots \\ \vdots & \ddots & \ddots & \mathbf{0} \\ \mathbf{0} & \ldots & \mathbf{0} & \boldsymbol{\Sigma}_{mt,M-1} \end{pmatrix}.$$

a) Determine the analytic Jacobians of the state-transition function $g(\boldsymbol{x}, \boldsymbol{u}_t)$ and the observation function $h(\boldsymbol{x})$ for the robot pose $\boldsymbol{x}$.

b) Obtain the code sample and data for this part of the exercise from the course webpage. The archive contains three folders: data, matlab, plots. Implement EKF prediction and correction to localize the robot by finalizing the code in files **`prediction_step.m`** and **`correction_step.m`**.

c) Run your code and report the final robot pose estimate after processing the whole dataset. Visualize your final result using the provided plotting functions.