Homework Assignment 3

The homework assignment should be solved *individually*. However, it is allowed to ask questions to fellow students. All solutions should be clearly written and well motivated. Regarding the language, you may use Swedish or English.

The assignment is due on Wednesday, February 11, 2009. Please email your solutions to schon@isy.liu.se.

1 Reconstruction

This section contains some exercises on reconstruction taken from Ma et al. (2006). However, they are repeated below for convenience.

1. *Essential Matrix for planar motion*
   Suppose we know that the camera moves on a plane, say the xy plane (Note that moves here implies that we only allow rotations in the xy-plane). Show that:

   (a) The essential matrix $E = \widehat{T}R$ is of the special form

   $$E = \begin{pmatrix} 0 & 0 & a \\ 0 & 0 & b \\ c & d & 0 \end{pmatrix}, \qquad a, b, c, d \in \mathbb{R}. \tag{1}$$

   (b) Without using the SVD-based decomposition introduced in this chapter (5), find a solution to (R, T) in terms of a, b, c, d.

2. *Homography with respect to the second camera frame*
   In the chapter, we have learned that for a transformation $X_2 = RX_1 + T$ on a plane $N^T X_1 = 1$ (expressed in the first camera frame), we have a homography $H = R + TN^T$ such that $x_2 \sim Hx_1$ relates the two images of the plane.

   (a) Now switch roles of the first and the second camera frames and show that the new homography matrix becomes

   $$\widetilde{H} = \left(R^T + \frac{-R^T T}{1 + N^T R^T T} N^T R^T \right). \tag{2}$$

   (b) What is the relationship between H and $\widetilde{H}$? Provide a formal proof for your answer. Explain why this should be expected.

3. *Homography*
   Under a homography $H \in \mathbb{R}^{3\times 3}$ from $\mathbb{R}^2$ to $\mathbb{R}^2$, a standard unit square with the homogeneous coordinates for the four corners

   $$(0, 0, 1), \quad (1, 0, 1), \quad (1, 1, 1), \quad (0, 1, 1)$$

   is mapped to

   $$(6, 5, 1), \quad (4, 3, 1), \quad (6, 4.5, 1), \quad (10, 8, 1),$$

   respectively. Determine the matrix H with its last entry H_{33} normalized to 1.

2 Sensor Fusion for Position and Orientation Estimation of a UAV

The is implementation oriented assignment, where you will solve a dynamic vision problem. The task is to estimate the position and orientation of a UAV using measurements from a camera, an accelerometer, a gyroscope and a barometer. Hence, this is very much a sensor fusion problem. Note that we have cast the problem as a visual odometry problem and not as a SLAM problem.

2.1 Background

The UAV we are using is in this assignment is the Yamaha RMAX helicopter, shown in Figure 1, used in the MOVIII[1] project. Your goal is to

Figure 1: *The Yamaha RMAX helicopter used in the collecting the data. The on-board system is equipped with an IMU (accelerometers and gyroscopes), an air-pressure sensor (barometer) and a downward looking camera. The on-board GPS receiver will be used for evaluation only.*

compute estimates of the position and orientation of the UAV, using the available measurements. In order to do this there is MATLAB code and a data-set available from the course web site. Before giving the explicit assignments are given, we provide some useful background.

2.2 Coordinate Frames

The relevant coordinate frames are listed below and illustrated in Figure 2

[1]MOVIII is a Strategic Research Center, funded by SSF, the Swedish Foundation for Strategic Research, for more information about the project we refer to the project web site `moviii.liu.se`.

- **World** (w): The camera pose is estimated with respect to this coordinate frame, which is fixed to the environment. The 3D landmark positions are assumed to be static in this frame. The initial helicopter position is, without loss of generality, taken as the origin of this coordinate frame.
- **Camera** (c): The camera frame is attached to the UAV and hence moves with the UAV. Its origin is located in the optical center of the camera, with the z-axis pointing along the optical axis. More specifically, the x^c-axis is pointing in the direction of the nose and the z^c-axis is pointing downward. This implies that the y^c-axis is pointing to the right side of the UAV body.
- **Body** (b): This is the coordinate frame of the IMU and it moves with the UAV and hence it is rigidly connected to the c frame. All the inertial measurements are resolved in this coordinate frame.

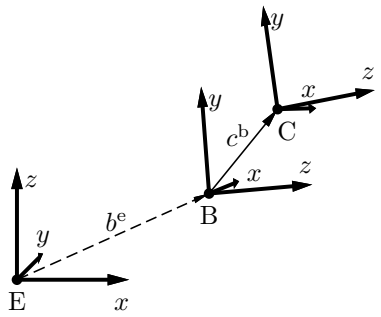

Figure 2: *Illustration of the coordinate frames. The sensor unit consists of an IMU (b frame) and a camera (c frame) that are rigidly connected. The rigid connection is illustrated using a solid line, whereas the dashed line indicates that this vector changes over time as the sensor is moved.*

2.3 Model

The state vector is given by

$$x_t = \begin{pmatrix} x_t^{\mathrm{v}} \\ m_t \end{pmatrix} \tag{3}$$

where x_t^{v} denotes the vehicle state and m_t denotes the map state, that is the positions of the landmarks. More specifically,

$$x_t^{\mathrm{v}} = \begin{pmatrix} p_t^T & v_t^T & q_t^T \end{pmatrix}^T, \tag{4a}$$

$$m_t = \begin{pmatrix} m_{1,t}^T & m_{2,t}^T & \dots & m_{M_t,t}^T \end{pmatrix}^T, \tag{4b}$$

where p_t denotes the position, v_t denotes the velocity, q_t denotes the orientation, $m_{i,t}$ denotes the 3D position of landmark number i at time t and M_t denotes the number of landmarks present at time t. For the sake

of brevity we have suppressed which coordinate frame the variables are expressed in, given here,

$$p_t = p_t^e, \tag{5a}$$
$$v_t = v_t^e, \tag{5b}$$
$$q_t = q_t^{be}, \tag{5c}$$
$$m_t = m_t^e. \tag{5d}$$

The dynamic model describes how the UAV and the map evolve over time and it is given below.

$$\begin{pmatrix} p_{t+1} \\ v_{t+1} \\ q_{t+1} \end{pmatrix} = \begin{pmatrix} I & TI & 0 \\ 0 & I & 0 \\ 0 & 0 & d_t \end{pmatrix} \begin{pmatrix} p_t \\ v_t \\ q_t \end{pmatrix} + \begin{pmatrix} \frac{T^2}{2} I \\ TI \\ I \end{pmatrix} u_{a,t} + \begin{pmatrix} \frac{T^3}{6} & 0 \\ \frac{T^2}{2} & 0 \\ TI & 0 \\ 0 & \frac{T}{2}\widetilde{S}(q_t) \end{pmatrix} \begin{pmatrix} w_{a,t} \\ w_{\omega,t} \end{pmatrix} \tag{6a}$$

$$m_{j,t+1} = m_{j,t}, \qquad j = 1, \ldots, M_t, \tag{6b}$$

where

$$w_{a,t} \sim \mathcal{N}(0, Q_a), \tag{6c}$$
$$w_{\omega,t} \sim \mathcal{N}(0, Q_\omega), \tag{6d}$$
$$d_t = I_4 + \frac{T}{2} S(u_{\omega,t}), \tag{6e}$$
$$u_{\omega,t} = \begin{pmatrix} \omega_{x,t} \\ \omega_{y,t} \\ \omega_{z,t} \end{pmatrix} \tag{6f}$$
$$S(u_{\omega,t}) = \begin{pmatrix} 0 & -\omega_{x,t} & -\omega_{y,t} & -\omega_{z,t} \\ \omega_{x,t} & 0 & \omega_{z,t} & -\omega_{y,t} \\ \omega_{y,t} & -\omega_{z,t} & 0 & \omega_{x,t} \\ \omega_{z,t} & \omega_{y,t} & -\omega_{x,t} & 0 \end{pmatrix}. \tag{6g}$$

The measurement equation for the barometer is given by

$$y_{\mathrm{b},t} = \begin{pmatrix} 0 & 0 & 1 & 0_{1\times 7} \end{pmatrix} x_t + e_t, \qquad e_t \sim \mathcal{N}(0, R_\mathrm{b}). \tag{7}$$

2.4 Data

The data contains the following data structures

- **cam**: Intrinsic camera parameters.
- **geometry**: Some constants related to the geometry of the hardware.
- **info**: Information about the current data.
- **meas**: Measurements (in SI units) from the accelerometers (`ya`), the gyroscopes (`yw`), the barometer (`yhp`, note that this is given as deviation from the starting point) and the camera (`yim`).
- **ref**: The reference position from the GPS.

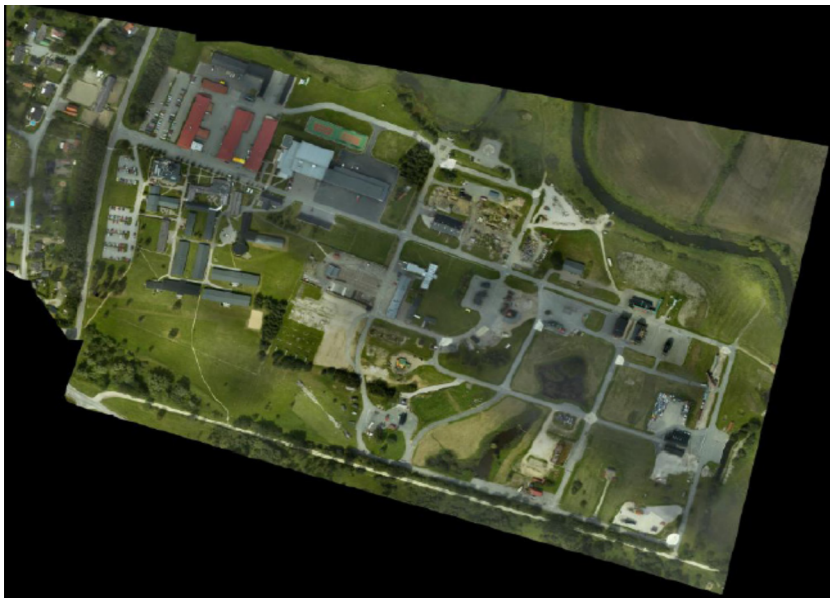

Figure 3: *Orthophoto of the flight test area in Revinge (southern part of Sweden).*

The data was collected during flights in Revinge, which is a flight test area in southern Sweden, see Figure 3 for an orthophoto of the area. For a more detailed background on the UAV and the hardware we refer to Conte (2007).

2.5 Assignments

In assignment $4-6$ below you are asked to write some code missing in the code available from the course web site. Note that the function skeletons are available, meaning that the interfaces are already specified. Provide your answers in terms of explanations and equations as usual, but also in terms of the actual MATLAB code that you have implemented. The main script is called `experiment`.

4. *Predict the landmarks*
   Write a function

   ```
   yhatf = predictLandmarks(x, par)
   ```

   that takes as inputs the current state `x` and the `par` struct and computes predictions for the landmark positions in the image. The predictions should be expressed in normalized image coordinates

$$p_n = \begin{pmatrix} x_n \\ y_n \end{pmatrix} = \mathcal{P}_n(P_c) \tag{8a}$$

where

$$P_c = \mathcal{R}(P_w). \tag{8b}$$

In other words, the predictions that you generate stems from a normalized camera. *Debug hint: Use the intrinsic parameters to compute the corresponding pixel positions and plot them in the image. They should of course be quite close to the actual detections.*

5. *Perform state update using vision measurements*
Perform measurement updates using the information available in the vision measurements produced by the data association. Note that these measurements will contain outliers due to erroneous associations. This implies that you will also have to implement some form of outlier detection. Provide a good explanation of the implemented outlier rejection. The function to be implemented is interfaced as follows

```
[x, P, S] = visionStateUpdate(x, P, visInnovation(:),
                              par, visible)
```

Hint: You may use numerical methods in order to compute the gradients, the function Jnum is useful. Debug hint: Once this function has been correctly implemented it is possible to execute the entire code. Obviously there will not be any new landmarks initialized until the next assignment has been solved.

6. *Search for new landmarks and initialize them in the map*
Since the UAV is continuously moving, there is, at times, a need to search for new landmarks and to incorporate the found landmarks in the state vector.

```
[x, P, patches] = searchNewLandmarks(x, P, patches,
          yhatfp_noinit, I, visible, par)
```

When the above functionality has been implemented everything is in place and you can execute the entire solution.

Besides the standard explanation of your solution you are also required to deliver plots of the Root Mean Square Error (RMSE) for the horizontal position error, both the individual errors in the x and the y directions and the joint error. The RMSE e_t is here calculated according to

$$e_t = \|x_t^0 - \hat{x}_t\|_2, \qquad t = 1, \ldots, N, \tag{9}$$

where x_t^0 denotes the true state (in this case the closest we can come to the true position is provided by the GPS) and $\hat{x}_t$ denotes the estimate.

7. *Improvements*
Give at least 5 ways in which the implemented solution can be improved. Your suggestions should deal both with the underlying theory (e.g., the model used and the problem formulation) and with the actual implementation (e.g., structure).

8. *Reflection*
   Now that you have a working system it is time for some reflection. You do not have to provide any answers for the assignment, but if you want that is perfect! The least you should do is to compare the implementation that you have just finished with the general structure of a visual odometry system presented during lecture 3.

3 "Gold Star" Assignment

This assignment is not mandatory. However, if you are interested this is a good way of trying out the new inverse depth parameterization provided by Civera et al. (2008). The standard way of describing the landmark position is to use the minimal Euclidean parameterization in terms of the landmark x, y, z position in the world coordinate frame, as we have done above. This parameterization has several problems; for instance it typically requires delayed (Bryson and Sukkarieh, 2005) or complicated undelayed initialization (Davison et al., 2007). However, since we have access to a barometer in this application we could to a large extent avoid these problems altogether. The reason for the problems is that there is no way to provide a good uncertainty description of the fact that the depth (i.e., distance) to the landmark is unknown. An elegant approximation which acknowledges this fact is provided by the inverse depth parameterization introduced by Civera et al. (2008). This parameterization allows us to straightforwardly include the landmarks directly when they are first observed, i.e., undelayed. Furthermore, it allows us to make use of very distant landmarks without any problems. These landmarks are of no or little use for inferring the camera translation, but they are very useful when it comes to the camera orientation.

The main idea underlying the inverse depth parameterization is to acknowledge the fact that when a landmark is first observed we can draw a ray from the landmark $\boldsymbol{l}^w$ through the image plane at position l^n to the current position of

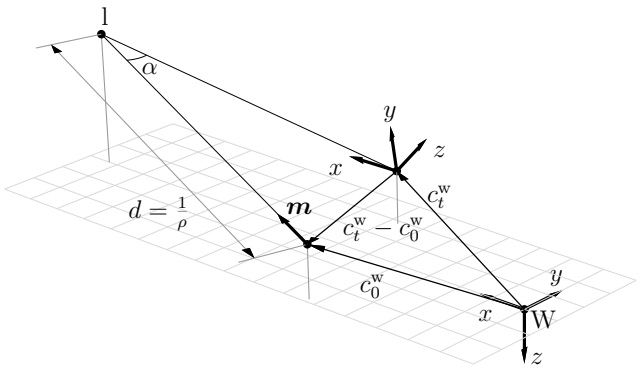

Figure 4: *The inverse depth parameterization used for the landmarks. The position of the landmark $\boldsymbol{l}$ is parameterized using the position of the camera the first time the feature was seen $\boldsymbol{c}_0^w$, the direction $\boldsymbol{m}(\phi^w, \theta^w)$ and the inverse depth ρ. This figure was made using Shapes (Tidefelt, 2009).*

the camera's optical center $\boldsymbol{c}_0^w$. This results in the following parameterization of the landmark

$$\boldsymbol{l}^w = \boldsymbol{c}_0^w + \frac{1}{\rho} \underbrace{\begin{pmatrix} \cos\phi^w \cos\theta^w \\ \cos\phi^w \sin\theta^w \\ \sin\phi^w \end{pmatrix}}_{\boldsymbol{m}(\phi^w, \theta^w)}, \tag{10}$$

where ρ denotes the depth to the feature and the direction to the landmark is described using spherical coordinates ϕ^w and θ^w, i.e., the azimuth and the elevation, respectively. This leads to the following state vector describing the position of a landmark

$$x^{\mathrm{l}} = \begin{pmatrix} (c_0^w)^T & \theta^w & \phi^w & \rho \end{pmatrix}^T \in \mathbb{R}^6. \tag{11}$$

For a graphical illustration of the parameterization (10), see Fig. 4.

The task is to replace the Euclidean (x, y, z) parameterization used in the present solution with the inverse depth parameterization briefly introduced above.

References

Bryson, M. and Sukkarieh, S. (2005). Bearings-only SLAM for an airborne vehicle. In *Proceedings of the Australasian Conference on Robotics and Automation*, Sydney, Australia.

Civera, J., Davison, A. J., and Montiel, J. M. M. (2008). Inverse depth parameterization for monocular SLAM. *IEEE Transactions on Robotics*, 24(5):932–945.

Conte, G. (2007). *Navigation Functionalities for an Autonomous UAV Helicopter*. Licentiate Thesis No 1307, Department of Computer and Information Science, Linköping University, Sweden.

Davison, A. J., Reid, I., Molton, N., and Strasse, O. (2007). MonoSLAM: Real-time single camera SLAM. *IEEE Transactions on Patterns Analysis and Machine Intelligence*, 29(6):1052–1067.

Ma, Y., Soatto, S., Kosecka, J., and Sastry, S. S. (2006). *An invitation to 3-D vision – from images to geometric models*. Interdisciplinary Applied Mathematics. Springer.

Tidefelt, H. (2009). The shapes language. lang-shapes.sourceforge.net/, Last accessed on February 8, 2009.