SLAM Summer School 2006

Practical 2: SLAM using Monocular Vision

Javier Civera, University of Zaragoza
Andrew J. Davison, Imperial College London
J.M.M Montiel, University of Zaragoza.
josemari@unizar.es, jcivera@unizar.es, ajd@doc.ic.ac.uk

1 Objectives

1. Understanding the characteristics of efficient (potentially real-time) SLAM using a monocular camera as the only sensor.
   (a) Map management.
   (b) Feature initialization.
   (c) Near and far features.
2. Understanding the inverse depth parametrization of map features in monocular SLAM.
3. Understanding the performance limits of a constant velocity motion model for a camera when no odometry is available.

2 Exercise 1. Feature selection and matching.

One of the characteristics of vision-based SLAM is that there is too much information in an image sequence for current computers to process in real-time. We therefore use heuristics to select which features to include in the map. The desirable properties of map features are:

1. Saliency: features have to be identified by distinct texture patches.
2. A minimum number (e.g. 14) should be visible in the image at all times — when this is not the case, new map features are initialized.
3. The features should be spread over the whole image.

The goal of this exercise is to manually initialize features in order to meet the above criteria, and to understand better how an automatic initialization algorithm should work. Run `mono_slam.m`. With the user interface, you can add features and perform step by step EKF SLAM:

1. In the first image add about ten salient features spread over the image. You can watch the movie `juslibol_SLAM.mpg` (using for instance `mpeg_play` on a Unix workstation) as an example of how to select suitable features (but you can of course select other ones). This movie shows the results of applying automatic feature selection.
2. As the camera moves, some features will leave the field of view, and you will have to add new ones in order to maintain around 14 visible map features.

3 Exercise 2. Near features and far features.

A camera is a bearing-only sensor. This means that the depth of a feature cannot be estimated from a single image measurement. The depth of the feature can be estimated only if the feature is observed from different points of view — and only if the camera translates enough to produce significant parallax. In particular, it may take a very long time to obtain accurate estimates of the depths of distant features, since they display very little or no parallax when the camera moves small distances.

The goal of this exercise is to observe the different evolution of depth estimates in the cases of near and distant features and the influence that this has on the camera location estimate.

1. Open the video `parallax.mpg`. Observe the different motions *in the image* of features at different depths. Open the video `noparallax.mpg` (taken from a camera which does not significantly translate) and observe that the image motion of features at different depths.

2. Now look at the video we are using for this practical (`juslibol.mpg`). Distinguish which parts of the scene in this video contain *low parallax motion*.

3. Run `mono_slam.m`.

   - Observe what happens to the features in the 3D map (initialization value and covariance and value and covariance after several frames). Red dots display the estimated values and red lines bound 95% probability regions denoting uncertainty.
   - The code singles out features #5 and #15 and displays their depth estimates and 95% probability regions: [lower limit, estimation, upper limit]. When clicking, make sure that feature #5 corresponds to a near one (for example, on the car) and feature #15 corresponds to a far one (for example, the tree appearing on the left).
   - Notice the difference between the evolution of the estimates of these near and distant features.
   - Observe the evolution of the camera location uncertainty (use the axes limit controls in the user interface).
   - Observe what happens to features and camera location uncertainties in the low parallax motion part of the image sequence discussed above in 2. Notice the difference between this part of the 3D map (constructed with low parallax information) and the high parallax parts.

4 Exercise 3. Inverse depth parameterization.

Initializing a feature in monocular SLAM is a challenging issue, because the depth uncertainty is not well modelled by a Gaussian. This problem is overcome using inverse depth instead of the classical XYZ representation.

The Matlab code of this practical uses the inverse depth parametrization of feature positions. The total state vector:

$$\mathbf{x} = \left(\mathbf{x}_v^\top, \mathbf{y}_1^\top, \mathbf{y}_2^\top, \dots \mathbf{y}_n^\top\right)^\top . \tag{1}$$

is composed of:

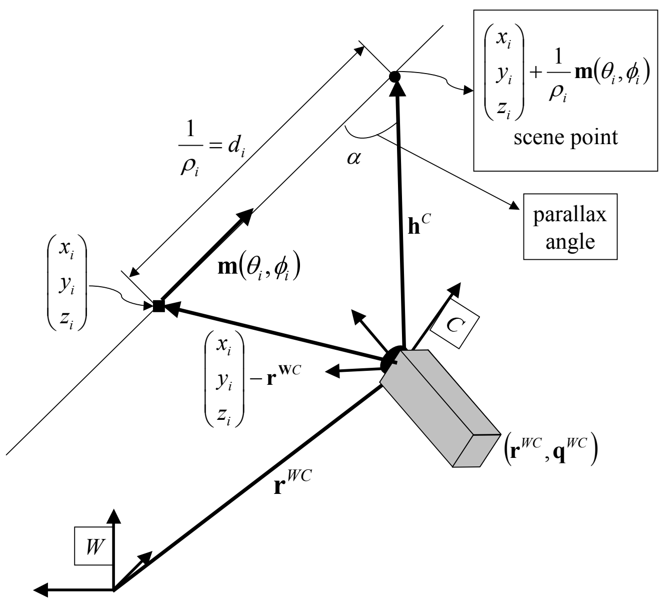
$\begin{pmatrix} x_i \\ y_i \\ z_i \end{pmatrix} + \frac{1}{\rho_i}\mathbf{m}(\theta_i,\phi_i)$
scene point
$\frac{1}{\rho_i} = d_i$
α
parallax angle
$\mathbf{h}^C$
$\mathbf{m}(\theta_i,\phi_i)$
$\begin{pmatrix} x_i \\ y_i \\ z_i \end{pmatrix}$
$\begin{pmatrix} x_i \\ y_i \\ z_i \end{pmatrix} - \mathbf{r}^{WC}$
C
$(\mathbf{r}^{WC},\mathbf{q}^{WC})$
$\mathbf{r}^{WC}$
W

Figure 1: Feature parameterization and measurement equation.

1. 13 components that correspond to the location, orientation, and velocity and angular velocity of the camera:

$$\mathbf{x}_v = \begin{pmatrix} \mathbf{r}^{WC} \\ \mathbf{q}^{WC} \\ \mathbf{v}^{W} \\ \omega^{W} \end{pmatrix}. \tag{2}$$

2. The rest of the components are features. Each feature is represented by 6 parameters; the position of the camera the first time the feature was seen x_i, y_i, z_i, a semi-infinite ray parametrized with azimuth-elevation angles (θ, ϕ), and the inverse depth ρ of the feature along the ray:

$$\mathbf{y}_i = \begin{pmatrix} x_i & y_i & z_i & \theta_i & \phi_i & \rho_i \end{pmatrix}^\top. \tag{3}$$

So the transformation from the inverse depth parametrization to a standard Euclidean system is:

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} x_i \\ y_i \\ z_i \end{pmatrix} + \frac{1}{\rho_i}\mathbf{m}(\theta_i, \phi_i). \tag{4}$$

where:

$$\mathbf{m} = \begin{pmatrix} \cos\phi_i \sin\theta_i & -\sin\phi_i & \cos\phi_i \cos\theta_i \end{pmatrix}^\top. \tag{5}$$

The goal of this exercise is to understand the inverse depth parametrization.

1. The code stores partial information about features #5 and #15 in the file `history.mat`:

   - `feature5History` is a 6 row matrix, each column containing the feature #5 location coded in inverse depth at step k.
   - `rhoHistory_5` is a row vector containing the inverse depth estimation history for feature 5.

- `rhoHistory_15` is a row vector containing the inverse depth estimation history for feature 15.
- `rhoStdHistory_5` is a row vector containing the inverse depth standard deviation history for feature 5.
- `rhoStdHistory_15` is a row vector containing the inverse depth standard deviation history for feature 15.

2. Compute the XYZ Euclidean location for feature #5 after processing all the images.
3. Do a graph with the value of the inverse depth and the 95% acceptance region history for both features #5 and #15. Use the matlab functions `open`, `figure`, `hold`, and `plot`. Comment on the difference between the two graphs.
4. After processing the whole sequence, what are the estimates and the uncertainty regions *expressed in depth* for both features?

   Think about a feature at infinity — what inverse depth would it have?

   In the graphs, are there regions where infinity is included within the uncertainty bounds as a possible depth for each feature? The inverse depth parametrization is particularly valuable in being able to represent the possibility of features at infinity.

5 Exercise 4. Constant velocity motion model (optional)

Monocular SLAM uses a camera as the unique sensor, without any odometry input. A constant velocity model is instead used to model approximately smooth motion of the camera. This model requires parameters to be set defining the camera frame rate, and the maximum expected angular and linear accelerations. These parameters together determine how much uncertainty is added to the camera position and orientation estimates during each motion prediction, and therefore also determines the size of the uncertainty-guided search regions used for feature matching. High expected accelerations or a low frame rate will lead to large search regions.

The goal of this exercise is to analyse the effect of changing the linear acceleration, the angular acceleration and frame rate parameters.

1. The initial linear and angular acceleration tunings are $6\frac{m}{s^2}$ and $6\frac{\text{rad}}{s^2}$.
2. Increase the angular acceleration only (for instance, double the value) and analyse the effect on the search regions. Find this parameter in the file `mono_slam.m` (its name is `sigma_alphaNoise`).
3. Increase the linear acceleration only (for instance, double the value), analyse the effect and compare what happens now. The name of this parameter is `sigma_aNoise`.
4. Reduce the frame rate of processing to see what happens when only:
   (a) 1 out of 2 images
   (b) 1 out of 4 images

are processed.

Clue: find the variable step and the code associated with this variable. You will also have to modify the variable `deltat` which sets the time between frames.

Does the processing time has increase or decrease as the result of processing less images? Can you explain this?

References

[1] Y. Bar-Shalom and T. E. Fortmann. *Tracking and Data Association*, volume 179 of *Mathematics in Science and Engineering*. Academic Press, INC., San Diego, 1988.

[2] A. Davison. Real-time simultaneous localization and mapping with a single camera. In *Proc. International Conference on Computer Vision*, 2003.

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

[4] J.M.M Montiel, Javier Civera, and Andrew J. Davison. Unified inverse depth parametrization for monocular slam. In *Robotics Science and Systems*, 2006.