Lab Course: Robot Vision WS 2012/2013

Philipp Heise, Brian Jensen, Sebastian Klose

Assignment 3 - Due: 03.12.2012

In this assignment sheet, you are going to work with stereo data. The goal is to create a motion estimation system–making use of calibrated stereo input–referred to as Visual Odometry. You will be using a class of techniques typically known as Structure from Motion (SfM) in the computer vision community or Visual Simultaneous Localization and Mapping (VSLAM) in the robotics community. You will be working with the kitti dataset files you already met in sheet 1. We are also providing some additional bag files with different trajectories and environments:

Dataset Link	Size
`http://www6.in.tum.de/~kloses/rvc/kitti_01.bag`	988MB
`http://www6.in.tum.de/~kloses/rvc/kitti_03.bag`	717MB
`http://www6.in.tum.de/~kloses/rvc/kitti_05.bag`	2.4GB
`http://www6.in.tum.de/~kloses/rvc/kitti_06.bag`	960MB
`http://www6.in.tum.de/~kloses/rvc/kitti_07.bag`	960MB
`http://www6.in.tum.de/~kloses/rvc/kitti_10.bag`	1.1GB

NOTE: All the Kiti datasets provided contain rectified images. If you use other datasets you will likely need to perform stereo rectification prior to any of the processing steps in this sheet.

Exercise 1 Sparse Stereo Matching

The first step in the your visual motion estimation process is to estimate the 3D position for a sparse set of robustly matched points between the left and right views of a calibrated stereo system.

1. *(Visual Odometry Package)*
   - Create a new package named `stereo_vo_<GroupName>` within your repository. You are going to reuse the feature descriptors from the last sheet, so you should make your new package depend on your package from assignment sheet 2.
2. *(Synchronized Stereo Subscription)* You will need to subscribe to two image topics at once, together with the associated `camera_info` topics. Moreover you need to ensure that the images synchronized and correspond to the same time step.
   - You can use `http://www.ros.org/wiki/message_filters` to achieve time synchronous subscription. (e.g. approximate or exact sync.).
   - There is also a special and useful class for stereo calibration data (`StereoCameraModel`) in the `image_geomerty` package: `http://www.ros.org/wiki/image_geometry`.
3. *(Feature Detection)*
   - Use your Harris corner detector as well as your feature descriptor implementation from the previous assignment sheets to extract features from both input images at each time step.

- All important parameters governing your implementation should again be available over dynamic_reconfigure.

4. *(Feature Matching and Disparity Computation)*
   - Match the extracted features from the left image to the right image from the same time step. You can start by using your general matching function from the previous sheet, with the additional constraint that feature matches must be in the same horizontal line in both images.
   - Disparity should be within a useful range. Range thresholds should be given over dynamic reconfigure as minimum depth and maximum depth in meters.
   - Experiment with different techniques to speed up and improve your implementation, for example:
     - Create a row lookup table for your features.
     - Preemptively exclude features matches that are outside of valid disparity range.
     - Perform consistency checking: match from left to right and from rigth to left, only accepting matches equal in both directions.
     - Soften the requirement that features matches must be in the same line.
     - Enforce the ordering constraint: matched features along one line in the left image must be in the same order along the corresponding line in the right image.

     Choose the parameters and implementation that deliver the best results.
   - Create 3D points from your matches using the left matching point's x, y and disparity value (You can use functions of the `StereoCameraModel`).

5. *(PointCloud publishing)*
   - Publish a `PointCloud2` message from your sparse set of matched feature points: `http://www.ros.org/doc/api/sensor_msgs/html/msg/PointCloud2.html`.
   - Make sure to set your frame ids correctly (left camera frame).

Exercise 2 Temporal Feature Matching

The next phase of your 3D motion estimation implementation involves taking the robustly matched 3D points gathered at each time step and finding correspondences between the left-right feature matches at the previous time step with left-right feature matches at the current one.

1. *(Windowed Matching)* A useful assumption for tracking features throughout images is that their positions will only change gradually. Therefore a typical approach is to constrain the matching between features to a certain window around a predicted (or the previous) position. You will implement a matching strategy, that finds the closest match for a given left-to-right feature correspondence from the last frame within a user specified radius. Experiement with different matching strategies, for example:
   - Use descriptors from left and right image while enforcing the epipolar constraint.
   - Employ circular matching: for a feature f_l^t (left image, time step t), find match in right image (f_r^t (see Exercise 1)). For f_r^t, find the matching feature in the current right image f_r^{t+1}. For f_r^{t+1} find the stereo correspondence in the left image f_l^{t+1}, for which you again search the correspondence in time in the previous left image f_l^t. For a correct match, you should end up with the same feature position f_l^t!

   At the end of this stage, you should have a set of 3D-3D correspondences.

2. *(Visualization of the 3D Matches)* To visualize your results, as in the last sheet, draw your matches side by side into an image augmented with the appropriate views. Depending on your matching strategy, either show all the four images or only the left or right ones and add lines between matching feature positions.

Exercise 3 3D Pose from Correspondences

1. *(3D pose estimation)* Use the method of **Umeyama** to compute the 3D pose between consecutive frames. A detailed explanation can be found in the original publication ”Least-squares estimation of transformation parameters between two point patterns” (`http://cis.jhu.edu/software/lddmm-similitude/umeyama.pdf`).
2. *(Robust estimation)* In order to cope with possible outliers–primarily due to incorrect 3D-3D correspondences– and prevent incorrect point cloud esimates a naive version of RANSAC should be implemented. A user specified number of iterations with random minimal subsets of the 3D-3D point correspondences are performed and in each iteration the set of all 3D-3D correspondences is partitioned into subsets consisting of inliers and outliers according to some distance metric. Correspondence distances below a user specified distance threshold are then considered inliers and all others outliers. After completion the largest subset of inliers is then used to calculate the final 3D pose estimate.
3. *(Pose Publishing)* Use the tf package to publish your current pose estimate.
4. *(Evaluation)* Compare you results with the ground-truth data provided in the bag files.
   - Generate error plots for the translation and rotation using rxplot.
   - For the rotational error the error quaternion should be used. For human readable output the quaternion should also be converted to Euler angles.

Exercise 4 rviz Visualization

Now that we have the motion estimate, it’s time to visualize some additional information in `rviz`. The following information should be visible in `rviz` when running:

- The trajectory of the computed poses from the previous exercise
- Lines from the current position, to the 3D points that are currently tracked
- The point-cloud from the first exercise

You can use rviz markers, the path-display or any other integrated rviz display. Whatever seems most convenient to you. Optionally you could additionally implement:

- A 3D-flow visualization. Therefore you probably need additional book-keeping about those features, that are tracked over a longer period of time, as the 3D flow vector between consecutive frames will probably be quite small, but you can have a try.
- A visualization of the confidence in your estimate of the 3D position. Draw for each 3D feature an ellipsoid representing the covariance of the estimated 3D position. To obtain the covariance for a 3D estimate you have to define an accuracy metric for your matching process and then calculate the respective variances along the axis using similar triangles.