Visual Odometry

Utkarsh Sinha
Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213
usinha@andrew.cmu.edu

Jai Prakash
Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213
jprakash@andrew.cmu.edu

Abstract

In this project, trying to localize the camera using visual odometry. The major component of the project is to generate keyframes according to pre-defined heuristics and triangulate the points to create 3D reconstruction of the scene. The intermediate frames can be found using Perspective-n-point algorithm. In addition, we also perform local bundle adjustment over last few frames so that the localization is locally consistent. We also plan to exploit the onboard inertial sensors to get prior for the localization.

1 Introduction

Augmented reality has been around for years, yet not all problems are solved in the domain. One of the challenges is precise localization of the device in world coordinates. Several augmented reality applications on smartphones are based on markers. One good example of marker-based AR is Vuforia. On the other hand, there are standalone devices like Hololens, which has number of sensors to understand the scene and localize the head mounted display in the scene.

In many such applications, understanding the scene is not important. Localizing the camera in the world is enough to solve certain problems. In this project we focus on localizing the phone camera using the camera and the inertial sensors.

2 Background

Visual SLAM vs. Visual Odometry: The focus in the visual SLAM techniques is both in reconstruting the scene and also localizing the camera in the scene. However, our main focus is just in localization of the camera. For the scope of the project, we focus only to be locally consistent. So, our system's camera position might drift over time - however, we are only interested in accurate localization in a short timespan. We do not explore ideas like loop closure in this project.

3 Our approach

Put an overall system diagram here.

3.1 Feature Extraction and Matching

We have experimented with KLT features, AKAZE features and ORB features. The KLT features can also be used for tracking the features in the subsequent frames using optical flow. For AKAZE[1] and ORB features, the correspondences are found using feature matching. Experimentally, we found that AKAZE was able to generate good matches and in greater number than the other two approaches for the datasets we were using.

30th Conference on Neural Information Processing Systems (NIPS 2016), Barcelona, Spain.

To remove outliers from the matches, we used three tests. The first is the **ratio test** which ensures that the feature is discriminative enough. This is accomplished by calculating the ratio of the distances between the best match and the second best match for a feature. A discriminative feature would have a high ratio and thus be a good feature to keep.

The second test is the **symmetry test** where we ensure that matches from image $\mathbf{I}_i$ to $\mathbf{I}_j$ also match in the reverse direction - that is from $\mathbf{I}_j$ to $\mathbf{I}_i$. The third test is the **epipolar test** where we discard matches that do not satisfy the epipolar geometry of the two frames used for 3D reconstruction.

Put an image of matches

3.2 Three dimensional reconstruction

Using the feature matching, we can triagulate the points. The fundamental matrix gives the relationship between the feature points. We use a RANSAC based 8-point algorithm to find the fundamental matrix.

Using this fundamental matrix, we can estimate the camera pose between the two images. However, this gives rise to four possible camera configurations (with W/W^T and $\pm t$). The correct location can be found using the camera configuration in which all the points are in front of both the cameras.

We use 3D reconstruction in two modes: one with stereo image pairs and one with a monocular images. Reconstruction in both modes was successful. However, as hypothesized, we noticed qualititively that the monocular 3D reconstruction had more drift compared to the stereo reconstruction.

We run 3D reconstruction only on keyframes. A new keyframe is created everytime the number of matches to the previous keyframe goes below a certain threshold. Currently, we use a threshold of 45%. This results in a stable reconstruction and allows for registering the different pointclouds together as well.

3.3 Point cloud registration

Something here

3.4 Camera pose recovery

Once the scene is reconstructed using the keyframes, the camera pose for normal frames can be recovered using Perspective-n-point (PnP) algorithms. By knowing the 3D points from the reconstruction, and its corresponding feature location in any image the camera pose can be recovered using PnP.

We chain these camera poses together to produce an initial guess of the camera trajectory. The trajectory from such chaining drifts as the sequence proceeds and errors accumulate. To account for these errors in the camera projection matrix (and also the point cloud reconstruction), we use bundle adjustment over the past m frames.

3.5 Bundle Adjustment

In visual odometry, the current camera pose is obtained by adding the last observed motion to the current detection change. This leads to a superlinear increase in pose error over time. In this section, we look at the techniques we intend to use to correct this pose drift.

One solution is to use bundle adjustment to impose geometrical constraints over multiple frames. The computational cost increases with the cube of the number of frames used for computation. Thus, we limit the number of frames to a small window from the previously captured frames. This approach is called local bundle adjustment.

We use a local bundle adjustment that affects the global point cloud and the recent m camera poses. We track visibility of 3D points in each frame and use that to construct a cost function.

$$\min_{\mathbf{M}_i, \mathbf{X}_j} \sum_i \sum_j ||\mathbf{x}_j^i - \mathbf{M}^i \mathbf{X}_j||_2^2$$

After accounting for visibility of keypoints, our optimization has about fifteen thousand variables. We use the dense schur method for optimization and use Ceres solver[4]. The optimization is able to reduce reprojection error across multiple frames as shown in **ADD FIGURE**.

4 Results

So far, we have been working with the datasets available online. The results are illustrated on the Middlebury Temple dataset [2]. We first find the feature correspondences and then remove the outliers using Epipolar constraints. The outliers can be found by using a threshold on distance from epipolar line and along the epipolar line.

We are able to recosntruct the temple structure using the two keyframes (handpicked for now). The results are shown in figure **??**. Once the reconstruction is done, we are able to recover the camera poses using PnP algorithm as illustrated in the figure **??**. We are using OpenCV for feature matching and visualization.

5 Future Work

Currently, we calculate the fundamental matrix using the Ransac version of the 8-point algorithm. However, it is possible to improve upon this by using the three-point algorithm. This requires integration with the IMU as an additional clue for estimating the fundamental matrix.

Another area to improve is selecting when to generate a new keyframe. We use the number of keyframe feature matches as the selection criteria. A more mathematically grounded technique would be to use the covariance of the various points to do this.

Our data structures used for storing point correspondences, the 3D point cloud, etc can be made much more memory and complexity efficient. While this wouldn't contribute to computer vision, it would allow us to run experiments faster and better tune the algorithm to be more generic.

Finally, we hope to model the point cloud as probability densities or planes. This would reduce the memory footprint and potentially improve the quality of pose estimation.

References

[1] J. Nuevo P. Alcantarilla and A. Bartoli. Fast explicit diffusion for accelerated features in nonlinear scale spaces. *BMVC*, 2013.

[2] Unknown. Temple dataset. `http://vision.middlebury.edu/mview/data/`, 2004.

[3] T. Schulze P. Tiefenbacher and G. Rigoll. Off-the-shelf sensor integration for mono-slam on smart devices. *CVPR*, 2015.

[4] Sameer Agarwal, Keir Mierle, and Others. Ceres solver. `http://ceres-solver.org`, 2012.