ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Monocular Visual Odometry

Vision Algorithms for Mobile Robotics

Dongho Kang, Jaeyoung Lim, and Jihwan Youn

January 8, 2016

Contents

Contents i

Instruction Manual iii
Environment . . . iii
Files . . . iv
Installation and Running . . . v

Monocular Visual Odometry vii
Initialization . . . viii
Continuous Operation . . . viii
Bundle Adjustment . . . ix
Loop Closure . . . ix

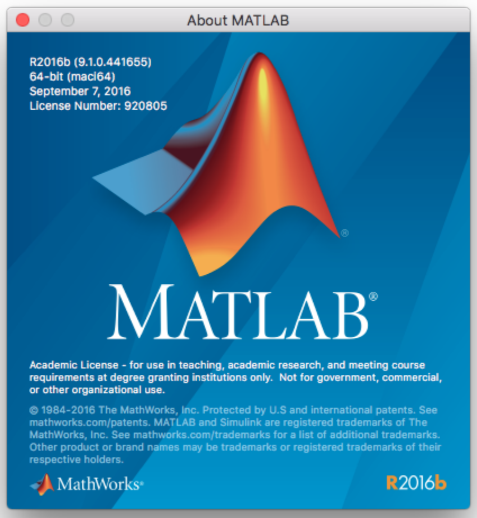
Experiment Results xi
KITTI . . . xi
Malaga . . . xi
Parking . . . xii
Custom Dataset (Seoul) . . . xii
Conclusions . . . xiv

Bibliography xv

Chapter 0

Instruction Manual

0.1 Environment



R2016b (9.1.0.441655)
64-bit (maci64)
September 7, 2016
License Number: 920805

Academic License - for use in teaching, academic research, and meeting course requirements at degree granting institutions only. Not for government, commercial, or other organizational use.

© 1984-2016 The MathWorks, Inc. Protected by U.S and international patents. See mathworks.com/patents. MATLAB and Simulink are registered trademarks of The MathWorks, Inc. See mathworks.com/trademarks for a list of additional trademarks. Other product or brand names may be trademarks or registered trademarks of their respective holders.

Figure 0.1: MATLAB Version Information (MATLAB for MAC OS X)

The code is written in Mathworks ©**MATLAB**. Due to the differences in several matrix operation, the code only works properly in **R2016b** version. Operating Systems of testing environments are **Linux Ubuntu 14.04 LTS** and **Apple Mac OS X El Capitan Version 10.11.6.**

We did not use any external libraries but only the modules developed during the exercises sessions and several MATLAB classes or functions which are allowed. (e.g. *vision.PointTracker*)

0.2 Files

Scripts

.m script files are under *src* directory.

- *main.m* A main script for visual odometry pipeline. It invokes VO functions such as *initializeVO, processFrame, bundleAdjustment* and *visualizeFrame.* Also, it includes several running options such as choosing data set among KITTI / Malaga / Parking / Seoul and turn on/off features etc.
- *create_bag.m* This script generates bag of visual words and image index for each data set which can be used for image retrieve test and debug. Since place recognition does not work properly in our pipe line, it is deprecated now.

Visual Odometry Functions

.m files below includes core logic of our VO pipeline. Please refer to **Monocular Visual Odometry** section for more details. Files are under *src/visual_odometry* directory.

- *initializeVO.m* Function for VO initialization operation.
- *processFrame.m* Function for VO continuous operation.
- *bundleAdjustment.m* Function for bundle adjustment.
- *getParams.m* This functions returns proper parameters for each data sets which were carefully chosen.
- Functions for Loop Closure (*./loopClosure_sub*)
- Other sub-functions for functions above (*./other_sub*)

Classes

Class files are written in Object Oriented Programming style for containing data such as state, landmarks, candidates and parameter(*Params_XXX.m* files) They also includes several methods to process these data. Files are under *src/class* directory.

Note: Please do not change value of parameter class properties. These values have been carefully chosen.

Package Functions

These .m files are developed during the exercise session. The files are under *src/exercise_packages* directory.

Files for Debugging

These .mat files were used for debugging. It includes exported bag of visual words, image index etc. The files are under *src/var_debug* directory.
Note: For test VO, you do not need to concern about these files.

Data Set Files

Data set files are images for VO pipeline. The files are under *data* directory. **For submitted version, we included whole dataset files in the directory.** Please refer to **Installation and Running** section.

- KITTI   Path of the data set is *data/kitti*
- Malaga   Path of the data set is *data/malaga-urban-dataset-extract-07*
- Parking   Path of the data set is *data/parking*
- Seoul   Data set which was created by us. It was generated from video recorded by iPhone 6 and processed (i.e. calibrated and rectified) using MATLAB Camera Calibration Toolbox for MATLAB by Caltech. Path of the data set is *data/iphone*

0.3 Installation and Running

We submit one .zip file which contains whole code and our own dataset. For a testing, extract the .zip file and please set dataset paths first, which defined in main script. Finally run *main.m* script. *main.m* automatically refer to other dependencies for running VO pipeline.

Remark, our VO pipeline only works in **MATLAB R2016b**

Note: Before run main script, **please check if you are in the *src* directory.** In only works properly when work directory is *src* directory.

Chapter 1

Monocular Visual Odometry

Our Visual Odometry pipeline(VO pipeline) consists of initialization, continuous operation, and bundle adjustment as shown in figure . A loop closure module was also implemented but excluded from the experimental result.

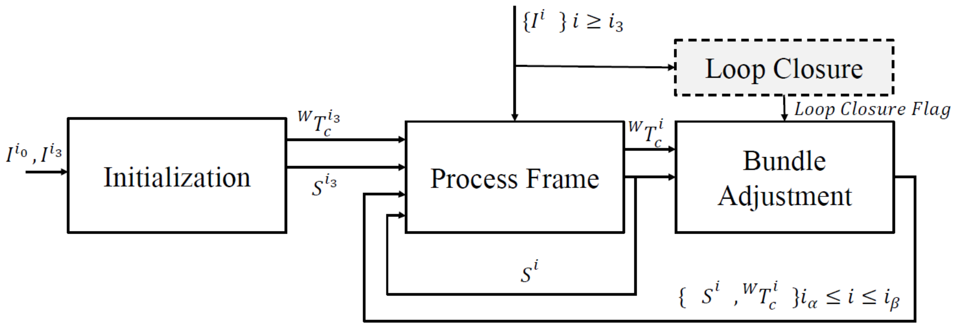


Figure 1.1: Overview of the visual odometry pipeline.

The initialization module outputs the initial pose and landmarks based on the manually selected two images, I_0 and I_1. The process frame uses the previously estimated pose and landmarks to continuously estimate the current pose. Additionally, VO pipeline performs window-based bundle adjustment to optimize the poses and the landmarks by minimizing the reprojection error in a nonlinear fashion. A loop closure feature was added by storing the landmark history. However, due to the poor robustness of place recognition function, the loop closure feature is not demonstrated properly in the demo videos.

1.1 Initialization

Our VO pipeline uses a **monocular standard camera** to estimate initial pose and landmarks. We carefully selected two image frames from each dataset which are sufficiently distant with each other to guarantee stable triangulation performance.

Initialization for Monocular VO

Firstly, Harris features are extracted and matched from two selected frames. Based on the matched corners, we apply 8 points RANSAC and obtain a fundamental matrix. The world coordinate frame is the first frame camera pose and the second frame camera pose is calculated by decomposing the fundamental matrix. Additionally, using the inliers from RANSAC, the VO pipepline triangulates new landmarks and removes the landmarks behind the camera.
We tested two different error distances for 8 point RANSAC. One is the distance to epipolar line and the other is Sampson error(first-order geometric error). Our VO pipeline uses the distance to epipolar line as both showed qualitatively similar results.

1.2 Continuous Operation

State Propagation

Our VO pipeline tracks keypoints from previous state using KLT tracker. To cope with the large displacement of keypoints between frames, we use carefully tuned number of pyramids.

Pose Estimation

The current camera pose is estimated based on the tracked keypoints and their corresponding landmarks. We apply P3P RANSAC to obtain inlier landmarks and initial pose estimation. A nonlinear optimization minimizing the reprojection error was implemented to refine the initial pose estimation. The number of tracked keypoints is constrained to less than a certain number to ensure enough keypoints to estimate the pose of the camera. This saves computational power while estimation.

New Landmarks Triangulation

Keypoints that are not triangulated are kept as **candidates** and tracked by KLT tracker. The total number of candidates are constrained by adding a limited number of new candidates in each frame. New candidates are selected among extracted corners that are far enough from existing candidates to avoid selecting certain corners multiple times. To check the triangulability of each candidates, we calculate the angle between bearing vectors of the original candidates and the tracked candidates. The angles between the bearing vectors are extremely small, therefore, arcsin was used instead of arctan (i.e. *atan2()* function in MATLAB) to take advantage of the numerical stability. Triangulable candidates are updated to new landmarks and used in the pose estimation.

1.3 Bundle Adjustment

A window based optimization strategy was implemented for a full bundle adjustment. The full bundle adjustment is to prevent drift in pose and landmarks during the continuous operation. The window based optimization strategy deploys a sliding window of N frames for minimizing reprojection error every M frames. Such approach was implemented to reduce the computational complexity while maintaining performance of bundle adjustment. The optimization window N is larger than the bundle adjustment period N so that $N - M$ states overlap with the previous bundle adjustment.
A new data structure, **bundle adjustment state,** was defined to construct hidden states and observations. After each bundle adjustment, the bundle adjustment state is reconstructed so that the landmark index can be labeled correctly.
Each dataset is implemented with different sizes of bundle adjustments which shows the best performance for the operating environment.

1.4 Loop Closure

We tried to implement **Loop Closure module using place recognition** as follows.

1. Build a **bag of visual words** and create **inverted image index.** Add images which are used for *initialiveVO* step frame to index.
2. For each step of continuous operation, **retrieve similar images with current image frame** from image index. Check whether retrieved images are from the same place with current place. (Loop detection)
3. If the places are same start loop closure, **optimize trajectory using bundle adjustment from the beginning to the end of the loop.** Use Harris match for matching landmarks.

The logic to identify the same place is follows.

- Check if index of retrieved image is distant enough from the current image.

- Check if the number of retrieved images are enough and their indices are sequential.

The Caltech training set[3] was used to construct the bag of words but the performance on place recognition on the KITTI or MALAGA dataset images were not sufficient to test the loop closure module. KITTI and MALAGA dataset images were also used to construct the bag of words but the performance of place recognition was not improved.

Chapter 2

Experiment Results

Our VO pipeline was tested on four different datasets to verify performance: KITTI [2], MALAGA [1], Parking and Seoul(Our own dataset).

2.1 KITTI

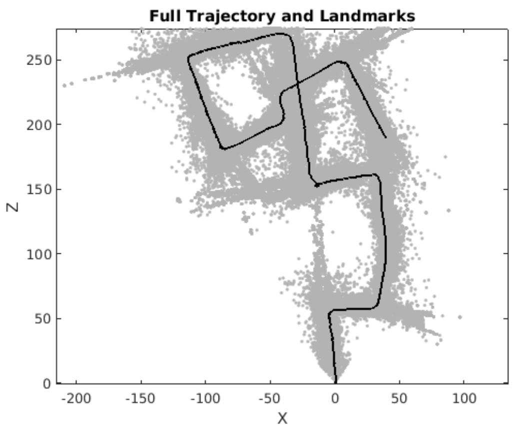


Figure 2.1: KITTI dataset full trajectory and landmarks

KITTI dataset shows significant scale drift without the bundle adjustment implementation. After bundle adjustment, the VO pipeline is robust enough to run to the end of the dataset.

2.2 Malaga

MALAGA dataset shows scale drift even after bundle adjustment. This is thought to be from the large variations of landmark distance from the camera.

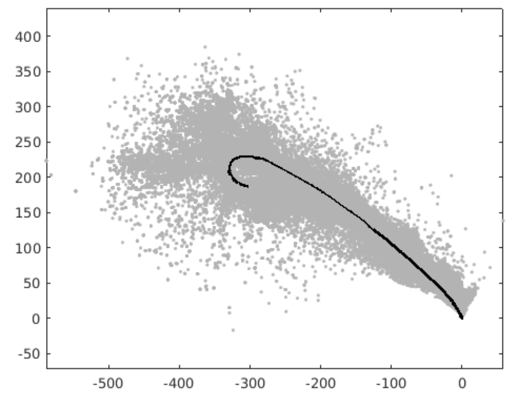


Figure 2.2: MALAGA dataset full trajectory and landmarks

2.3 Parking

Parking dataset shows robust performance even though the camera is facing the right angle to the direction of travel. Parking has an acculative error as the dataset has no loop even though running bundle adjustment.

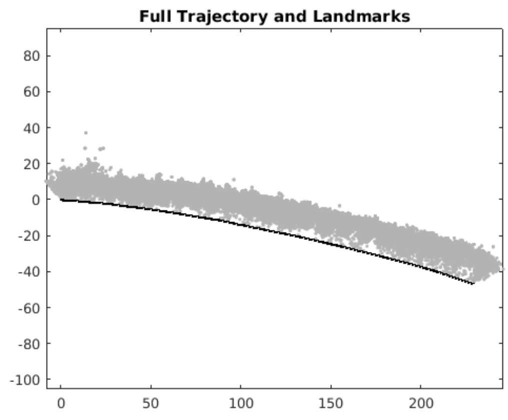


Figure 2.3: Parking dataset full trajectory and landmarks

2.4 Custom Dataset (Seoul)

A custom dataset was constructed to verify that the visual odometry pipeline is robust enough to work in various environments. The custom dataset is

based on 360 sequence of images taken from an iPhone 6s. While taking the images the camera had a fixed focal length and the images were down sampled by a factor of 0.2 for computational issues in calibrating the camera and rectifying images. Images were filmed in a narrow alley in Seoul, Republic of Korea in December 27th 2016.
The successful trajectory estimation of the VO pipeline in a narrow alley with large variations of landmark distance and illumination changes shows robust performance of the VO pipeline.

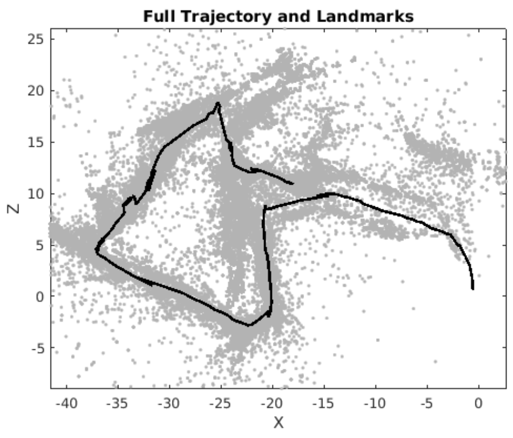


Figure 2.4: Seoul dataset full trajectory and landmarks

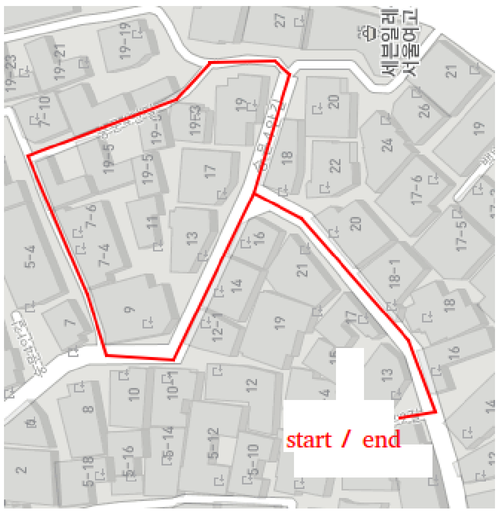


Figure 2.5: Estimated Ground truth of Seoul dataset from Daum Map

Video Links

- Youtube Playlist https://www.youtube.com/playlist?list=PL4osb-habf7AwyqOOP_enbcIkzncwvSIB
- KITTI: https://youtu.be/dS4KoISoAcE?list=PL4osb-habf7AwyqOOP_enbcIkzncwvSIB
- Malaga: https://youtu.be/3fxA7I47CuM?list=PL4osb-habf7AwyqOOP_enbcIkzncwvSIB
- Parking: https://youtu.be/WfOTkpfoyzY?list=PL4osb-habf7AwyqOOP_enbcIkzncwvSIB
- Seoul (our data set): https://youtu.be/Y01DwR3oK0w?list=PL4osb-habf7AwyqOOP_enbcIkzncwvSIB

2.5 Conclusions

We successfully implemented a fully working and robust monocular visual odometry pipeline as well as the additional features as the following.

- An appealing visualization was implemented including keypoint tracking information, camera heading and trajectory in each frame. A full landmark and trajectory visualization is also implemented.
- Many ideas for improving the performance of the VO pipeline.
- refined estimated pose by minimizing reprojection error.
- VO pipeline verification on a custom recorded dataset.
- Full bundle adjustment(motion and structure)
- Not fully working but implemented loop closure module using place recognition.

Bibliography

[1] José-Luis Blanco, Francisco-Angel Moreno, and Javier González-Jiménez. The málaga urban dataset: High-rate stereo and lidars in a realistic urban scenario. *International Journal of Robotics Research*, 33(2):207–214, 2014.

[2] Jannik Fritsch, Tobias Kuehnl, and Andreas Geiger. A new performance measure and evaluation benchmark for road detection algorithms. In *International Conference on Intelligent Transportation Systems (ITSC)*, 2013.

[3] P. Perona L. Fei-Fei, R. Fergus. Learning generative visual models from few training examples: an incremental bayesian approach tested on 101 object categories. *CVPR*, 2014.