Technische Universität München

Informatics Master's Program

Drone Photo-Goal Navigation with LSD-SLAM

Hidenobu Matsuki
Yunus Emre
Ahmad Tahir

Project Supervisors:
Vladyslav Usenko
Lukas von Stumberg

Version 1.0
March 12, 2017

Introduction

The aim of this project is to build a photo-goal navigation system for Micro Aerial Vehicle (MAV). Given a goal photo, the quadcopter has to navigate to the goal location where the target was taken. This system can be applied to autonomous packet delivery with a quadcopter. This report can be divided into three sections. First, we describe the overview and our approach. Then, we discuss about each steps and theoretical background. Next, we explain problem setting and result of our flight experiment. Finally, we discuss a future work of this project. Our platform is Parrot Bebop Drone, a quadcopter which weighs 400g and has fish eyes monocular camera. Our application is built using C++ on ROS and works with bebop_tum_ardrone and bebop_lsd_slam package. The bebop_lsd_slam does localization and mapping using LSD-SLAM [Engel 2014] algorithm with monocular camera. The bebop_tum-ardrone package gives the LSD-SLAM based state estimation and navigation to quadcopter. We run our system on a laptop and communicate with quadcopter via WiFi. Our system is fully automated, which means quadcopter is autonomously navigated once the system starts.

Contents

List of Figures

List of acronyms used

FAST	*Features from Accelerated Segment Test*
FLANN	*Fast Library for Approximate Nearest Neighbors*
IMU	*Inertial Measurement Unit*
LSD-SLAM	*Large Scale Direct-Simultaneous Localization And Mapping*
PnP	*Perspective n-Points*
SIFT	*Scale Invariant Feature Transform*

Chapter 1

Approach and implementation

The objective of this system is to navigate drone to the goal where the photo is taken, and our approach is divided into the following steps:

1. Take a goal photo
2. Explore the environment running LSD-SLAM
3. Image matching
4. Calculate goal position and navigate quadcopter.

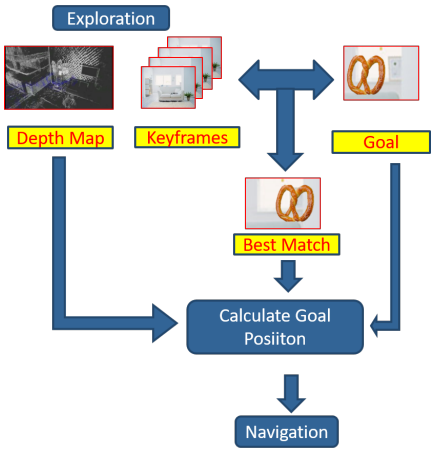

Figure 1.1: Overview of our approach.

1.1 Take a Goal Photo

First, we take a goal photo. We should keep camera horizontally because we cannot set pitch or roll angle value as a target command. Ideally the photo should be taken with smartphone camera, but we used drone camera before the experiment because we only have camera calibration data of drone camera.

1.2 Explore the environment running LSD-SLAM

Next, the quadcopter starts exploring the environment running LSD-SLAM. Then we can get recorded key frame and semi-dense depth map.

1.2.1 Exploration

We use fixed exploration command, which contains 360-degree rotation, up and down every 10 degree, 1m side movement every 90 degree. Since we use monocular camera, scale ambiguity often occurs after rotation. So, we added up and down command every 10 degree to improve the scale estimation. Also, we added 1m side movement to see the environments from different viewpoint and make our depth map more accurate.

1.2.2 LSD-SLAM

We use LSD-SLAM, an algorithm of vision based Simultaneous Localization and Mapping (SLAM). Even if robot does not have any prior knowledge of the environment, using camera image, SLAM estimates its position and create 3D depth map at the same time. There are several SLAM algorithms, and especially LSD-SLAM is based on direct method. Direct method does not extract feature point, but use an image directly and can create more dense 3D depth-map. Since we use monocular camera, we cannot avoid scale ambiguity but can estimate the value more accurately by using IMU and Extended Kalman Filter. LSD-SLAM calculates depth not for all frames, but for frames called keyframe.

The figure **1.2** shows the overview of LSD-SLAM. First, LSD-SLAM does tracking, which means direct image alignment between current keyframe and input video. Then LSD-SLAM estimate the depth. If input image is different enough from current keyframe, LSD-SLAM take it as a new keyframe. If not, input image is used to refine keyframe. Next, all keyframes are added to map and finally optimize the map with **Sim(3)** pose-graph optimization. Then we get semi-dense depth map of the environment.

1.3 Image matching

After enough keyframes that describe the environment are collected, the phase of matching is executed. The purpose of this step to find the key image that resembles the goal image the most. The best matching image is assumed to have the biggest amount of corresponding points with the goal image.

When finding the best match, all the keyframes are stored, registered and a knowledge base is created that is ready for lookup. Then the goal image is registered and queried through the database to find the best match.

The matching consists of 3 steps:

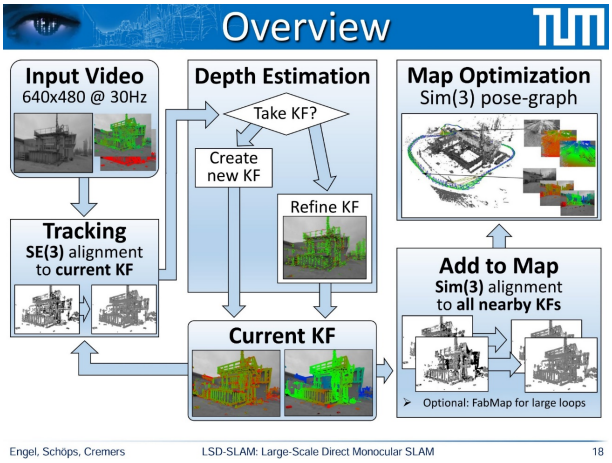

Figure 1.2: Overview of LSD-SLAM.

1. Detecting points of interest

2. Calculating the descriptors for those points

3. Finding matching points using the descriptors

In this application, the matching calculations are done via openCV detectors, descriptors and matchers.

1.3.1 Detecting points of interest

The step of detection is crucial because it reduces the images into a set of points that have distinguishing characteristics compared to their neighborhood [Tuytelaars 2008]. For this application, the FAST (Features from accelerated segment test) algorithm is used for detecting feature points. FAST algorithm is a computationally effective corner detection method and is fit for real time applications [Trajković 1998]. It is also quite effective for scenes with planar objects [Guerrero 2011]. This method proved to be the best performing detector in the test environment which included many planar objects with well-defined edges and corners such as the markers with two dimensional barcodes.

1.3.2 Calculating the descriptor

The next step is to generate comparable feature vectors for these feature points. These calculated values are called feature descriptors. SIFT (The Scale Invariant Feature Transform) algorithm is used to calculate descriptors for feature points in this project. SIFT is a scale invariant method developed by David Lowe. In SIFT, each 16x16 window is divided into 4x4 grids. Each cell has 8 orientation bits, yielding in 128-dimensional vector for each descriptor. Having scale invariant descriptors help to achieve a robust matching.

1.3.3 Finding matching points

After the descriptors are generated for each image, the set of descriptors are separated into two groups: training and query. Training descriptors are the descriptors of each key image and query descriptors are the descriptors of the query image. Train descriptors serves as a database where the provided query image is searched using the matching function.

During matching, each feature point from one image is compared to all other feature points in the compared image. This comparison is done via the descriptors. The distance between two descriptor vectors is a measure of resemblance between two points. Thus, the point pairs with minimum distances are considered matches. However, this approach yields as many matches as the minimum number of feature points among two images whereas only the matches between the corresponding points are desired. For this purpose, matches with the large distances between descriptors should be eliminated.

For assessing the matches, FlannBasedMatcher from OpenCV library is used. FLANN is a computationally efficient nearest neighbor search algorithm that can operate in high-dimensional spaces which is quite time-effective on big datasets [Muja 2014]. The train function of FlannBasedMatcher sets up the randomized k-d tree from the key images' descriptors. Then the query image is matched to the set of train descriptors, yielding in a single matching vector that contains the matching corresponding points from query image to each image in the train set.

FlannBasedMatcher generates different amount of matching points for each key image in the training set by eliminating many matched points that depending on the distance between the descriptors. Thus, the best matching image is assumed to be the image that has the most matching points in the resulting match vector.

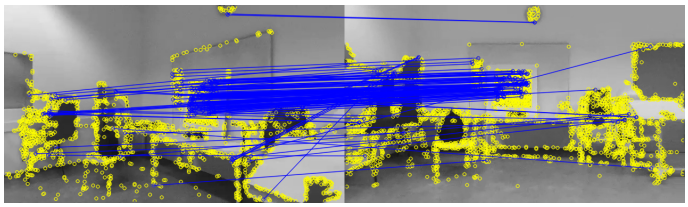

Figure 1.3: The result of the matching (Goal image on the left, best matching key image on the right).

As it is visible in the Figure **1.3**, marker board that has many edges and corners has a high amount of feature points and most of them are correctly matched.

Once the best matching image is determined, the corresponding points are accessible through the match vector. The corresponding points are assumed to be the projection of the same 3D point onto two different image planes. Considering the position of the 3D point and the position of the camera where the matched image is taken can be accessed through LSD-SLAM, all the required inputs for solving the perspective-n-point problem are ready.

Even though the descriptor distances of these corresponding points are sufficiently small,

it's still very likely for these set of points to have incorrect matches. Such points can be seen in Figure 1.3 as the points intersecting the parallel matching lines. These outliers are planned to be eliminated during the step of PnP solving with the use of RANSAC. In future improvements, distance vectors that contradicts with the general parallelism of the vectors or the vectors that intersect with most the distance vectors may be eliminated as a heuristic preprocessing before providing the points as inputs to the PnP solution.

1.4 Calculate goal position

At this point we have the best matched keyframe and the transformation (rotation and translation) to that keyframe from the reference point, which is computed from LSD-SLAM. The goal is to find the transformation from that reference point to the goal position. The only piece of puzzle missing is the transformation from the keyframe to the goal position. If we find this transformation from the keyframe to the goal, we can add this to the transformation to the transformation from the reference point to the best matched keyframe and we will get the desired transformation.

To find this transformation, let's consider our pinhole camera model. We know the 3D points

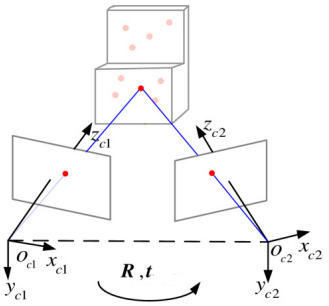

Figure 1.4: Pinhole camera model. Keyframe image plane and goal image plane, from left to right.

of an object from our keyframe and we also know the one-to-one correspondence of key points between keyframe and goal image. So, for each point in our keyframe, for which the corresponding point in the goal image in known, we have a projection equation like:

$$s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{1.1}$$

Where u and v are the points on the goal image, f_x, f_y, c_x and c_y are the drone camera parameters r_{11}, ... r_{33} is the rotation and t_1, t_2, t_3 is the translation and X, Y and Z represents the 3D point location. The transformation matrix is the unknown. We have this equation for each point correspondence. We can use this system of equation to find the unknown transformation by minimizing the projection error.

$$\sum_{n=1}^{N} \left(s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} - \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \right)^2 = 0 \tag{1.2}$$

This optimization is achieved in an iterative manner using Levenberg-Marquardt optimization technique. This implementation is available in openCV as SolvePnP. We used the RANSAC version of it to cater outliers, which can occur in our implementation.

The resulting transformation computed using SolvePnP is then added to transformation from reference point to the best matched keyframe and the resultant is the final transformation from the reference point to the goal image.

Chapter 2

Experiments, results and future work

2.1 Experiments and results

For the flight test, the robot explored the room for **150** seconds doing 360° rotation with limited translations. These transformations can be seen as green edges on the Figure **2.1**. Locations of the keyframes are shown with blue markers. As it is visible, the goal image (shown as the red viewpoint **G**) is taken from a point that is relatively far away from the exploration area compared to the translations during the exploration. Also, the image is taken from a completely different angle, resulting in a challenging task for the test.

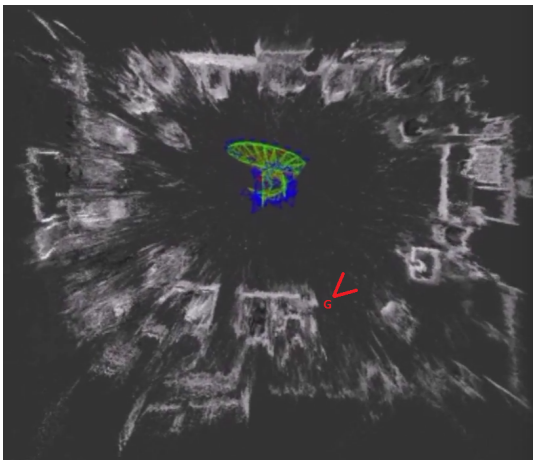

Figure 2.1: The map of the room after exploration.

Figure **2.2** shows the difference between the goal image and the image taken from the estimated position of the goal image. The resulting differences between the positions and the images can be qualitatively assessed as small enough considering many practical factors that are not considered such as the drifting of the drone.

Figure 2.2: The result of the flight test(Goal image on the right, image taken at the estimated goal position on the left).

2.2 Future work

Like all projects, we assumed some assumptions for this project as well. The most restrictive assumption was that the drone initially would be placed near the goal, so when the drone explores the environment it could find a good match between the explored environment and the goal. If the drone can't find a good match during the hard-coded automated exploring process, it can't reach its destination. The most obvious improvement it can have is the ability to explore the environment until it can't find a good match. Also, for now, the drone first fully explores the environment, getting keyframes, and it is not until it fully explores the environment that it finds a match with the keyframe and the goal photo. This restriction was imposed to find absolutely the best match between the keyframes and the goal. One improvement could be to formulate a method to find a good enough match between the keyframe and the goal, so that the drone could reach its goal as soon as the drone finds a good match, instead of first fully exploring the environment and then go the goal. This would require defining a threshold for the number of matching points between the keyframe and the goal.

We also restricted the environment to be on a small scale. One future goal can be broadening the scale of the environment. So, instead of trying to find the goal in nearby locations, the drone should be capable of explore the environment on its own, in a more intelligent and robust manner. This means fully utilizing the dense geometry of the environment. The goal would be to roam around all the available space and then find the keyframe which matches our goal the most. This would require hurdle avoidance, path planning. To make it even more functional, we could enable the drone to locate exits and find more entries into other rooms in an indoor environment. We could increase the scope of observable environment outdoors even more by using a GPS sensor. One application of this fully robust system would be to deliver parcels for any delivery system like Amazon or e-Bay, where the drone will have a GPS coordinates of the destination and an image of the location at the destination where the delivery parcel is to be placed, which in fact is the original idea behind the project.

Bibliography

[Engel 2014] J. Engel, T. Schöps et D. Cremers. *LSD-SLAM: Large-Scale Direct Monocular SLAM.* In European Conference on Computer Vision (ECCV), September 2014.

[Guerrero 2011] Maridalia Guerrero. *A comparative study of three image matcing algorithms: SIFT, SURF, and Fast.* 2011.

[Muja 2014] Marius Muja et David G Lowe. *Scalable nearest neighbor algorithms for high dimensional data.* IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 36, no. 11, pages 2227–2240, 2014.

[Trajković 1998] Miroslav Trajković et Mark Hedley. *Fast corner detection.* Image and vision computing, vol. 16, no. 2, pages 75–87, 1998.

[Tuytelaars 2008] Tinne Tuytelaars, Krystian Mikolajczyk*et al.* *Local invariant feature detectors: a survey.* Foundations and trends® in computer graphics and vision, vol. 3, no. 3, pages 177–280, 2008.