A Real-Time Visual Navigation System for Quadcopter based on LSD-SLAM Algorithm

Mr. M. Sudhakaran, M. E, (Ph.D.)
Associate Professor,
Department of Electrical and Electronics Engineering,
Ganadhipathy Tulsi's Jain Engineering college
Vellore-632 102.

S. Anandharaman
PG Student
Department of Electrical and Electronics Engineering
Ganadhipathy Tulsi's Jain Engineering college
Vellore-632 102.

***Abstract*—Visual odometry principles are rapidly demanding in navigation and mapping fields, which can be implemented with Unmanned Ariel Vehicles (UAV). Building a spatially consistent probabilistic model for a quadcopter for its navigation, 3D mapping and autonomy are the key purposes of this work. It also aims to put a spotlight on cost effective simulation solutions for the build and makes the drone autonomous by pose estimation computation. The environment mapping for the navigation in GPS-denied and clustered environments are carried out by traditional Simultaneous Localisation and Mapping (SLAM) technique with Large-Scale Direct Odometry Algorithm (LSD).**

Keywords—SLAM; Visual Navigation; Robotics; Visual Odometry

I. INTRODUCTION

In recent years, both remote controlled and autonomously flying Quadcopters have become an important tool not only in the military domain but also in civilian environments. They can also be used especially for observational and exploration purposes in indoor and outdoor environments, but also for data collection, object manipulation or simply as high-tech toys.

While the concept of an aircraft flying with four horizontally aligned rotors had already been proposed in 1922, this design quickly disappeared and was dominated by the much more common two-rotor helicopter. There are two main reasons for this development: While mechanically very simple, a quadcopter is inherently unstable and hence difficult to control - without the help of advanced electronic control systems and stabilizing routines, manual control turned out to be too complex. Furthermore, quadcopters are less energy-efficient than traditional helicopters.

With the growing importance of UAVs however, the quadcopter design has become more popular again. It is because that, mechanically quadcopters are simpler to control due to fixed thrusts. In addition to that, the four rotors can be enclosed by a frame, protecting them from the collisions and permitting the drone to flight indoors and in obstacle-dense environments. Finally, the use of four rotors allows each to have a smaller diameter, causing them to store less kinetic energy during flight and reducing the damage caused should the rotor hit an object, making quadcopters significantly safer to use close to the people.

In order to navigate, modern UAVs can rely on a wide range of sensors. In addition to an inertial measurement unit (IMU) measuring the aircraft's orientation and acceleration, a The GPS-based navigation system is often used to determine the UAV's absolute position - enabling it to autonomously navigate in a known environment, or simply to stay at its position without drifting away.

II. RELATED WORKS

In the literature, many authors presented autonomous methods of navigating the UAV by Visual Odometry methods. The mapping and pose estimation can be accomplished with both monocular and stereo cameras. Recently, M.Pizzoli et al 2014 [2] presented such navigational systems with the Monocular camera. But, using such monocular cameras for the odometry purposes will not sufficiently estimate the real scale of surrounding environments. So, stereo cameras can be used for the improvement in accuracy of generating the maps. But some works Changhong Fu et al 2015 [1] and 2014 [5] have cited that, there will be some problems in implementing the stereo camera for visual odometry.

These papers really an emphasis on SLAM methods, which should be more computationally feasible and easy as well. If we implement normal SLAM methods without sensor fusions as presented in [5], there will be some inaccuracy in computing the depth maps, which will lead to invalid pose estimation. On the other hand, J.Engel et al 2014 [3] proposed a Direct Monocular SLAM algorithm (feature-less) for mapping the environment under dense reconstruction. This can be used for the stereo cameras for correct rendering and computation of environment maps. Other works like J.Zhang et al 2014 [4] proposes the odometry systems with Direct SLAM along with LiDar based sensor devices, which produced promising results.

The [1] presents low-cost stereo odometry system based on the SGM and SLAM algorithms. But this algorithm really, peaks the processor load in the time of flight. It also has some liabilities as well. It produces an overhead on the processor, such that high-end heterogeneous cores are needed to carry out the odometry tasks. On focusing the cost and payload capabilities of these unmanned systems, VisLab 3dv [7]; Skybotic iv [8] and DLR device [9] presented a clean stereo guidance and mapping systems, which has a higher degree of accuracy and also lesser payload. But the liability lies on the cost of the system. These earlier efforts also tell us that there was a technical advantage of reducing the payload of the

quadcopter. It also has a disadvantage of using their proposed systems for dense visual reconstructions only. The basics of the SLAM, 3D mapping, and reconstruction of the environment was studied to the vast extent. They are dealt in-depth on various technical papers. The problem of inaccuracy while moving a monocular camera was fully covered in Matia Pizolli et al 2014 [2], which proposes a new framework called: REMODE. It also takes less footprint memory devices and fewer cores for its operation. The [5] presents the Fuzzy logic controller for the monocular visual odometry base on the Cross-entropy optimization for the Extended Kalman filter, used for the feature extraction. It also proposes a way to clear collision avoidance errors by using FLC optimization.

Although there are numerous types of reconstruction principles available for the mono camera, the Andreas et al 2011 [6], proposes a reconstruction method in real time for Stereo camera. It utilizes parse feature matching principle from the high definition stereo camera pair in real time, which can also be implemented on FPGA-based systems. The inferences obtained for stereo 3D reconstruction was only 3 to 4 fps using the StereoScan algorithm. A traditional Semi-Global Matching (SGM) based depth mapping system was designed by G.Camellini et al 2014 [7], which also describes the hardware implementation. They have implemented their system on Xilinx- Zynq SOCs for high performance. The accurate real-time SLAM in the Janosch et al 2014 [8], was a highly sophisticated real-time SLAM system, which uses IMUs and the same Zynq SOCs. Some authors also design Mobile Robots based on the SLAM and IMUs. The authors of [8] and [9] design these robots on FPGA platform. They invoke all the algorithms on Heterogeneous processors for the real-time data transfer to the base station system. The study of SGM was dealt in Simon Hermann 2012 [10], which proposes a new Fast SGM algorithm focusing on the Driver Assistance Systems. They benchmarked their results with the Trinocular camera based frames which they tell that it was 40% faster than novel SGM methods.

The basics of Kalman filter, Corner detection, Feature detection and Edge detection was studied for the use in SLAM algorithm. The Extended Kalman filter (EKF) was the majorly used filter in the algorithm. The Bernd Kitt et al 2010 [11] proposes a new faster and accurate filtering of the real time images with the help of EKF and Random Sample Consensus (RANSAC) based outlier rejection schemes. The corner detection algorithm was proposed in [14] and [12]. The famous Harris detector was used as a regular detector in all types of SLAM algorithms. The [12] proposes new binary strings for independent feature detection, called BRIEF. Thus, the literature study gives the clear outline of the existing system and various algorithms and techniques. Proper selection of the algorithm and the technique of control was applied onto the proposed system, aiming at the following points: Cost, Payload, Algorithm complexity, and Processor selection.

III. THE SLAM ALGORITHM

The most fundamental problems in robotics, the simultaneous localization and mapping problem. This problem is commonly abbreviated as SLAM and is also known as Concurrent Mapping and Localization, or CML. SLAM problems arise when the robot does not have access to a map of the environment; nor does it have access to its own poses. Instead, all it is given are measurements *z1: t* and controls *u1: t*. In SLAM, the robot acquires a map of its environment while simultaneously localizing itself relative to this map. SLAM is significantly more difficult than all robotics problems discussed thus far: It is more difficult than localization in that the map is unknown and has to be estimated along the way. It is more difficult than mapping with known poses since the poses are unknown and have to be estimated along the way.

A. Extended Kalman Filter SLAM

The EKF SLAM algorithm applies the EKF to online SLAM using maximum likelihood data association. In doing so, EKF SLAM is subject to a number of approximations and limiting assumptions:

1) Feature-based extraction. Maps, in the EKF, are composed of point landmarks. For computational reasons, the number of point landmarks is usually small (e.g., smaller than 1,000). Further, the EKF approach tends to work well the less ambiguous the landmarks are. For this reason, EKF SLAM requires significant engineering of feature detectors, sometimes using artificial beacons or landmarks as features.

2) Photometric noise. As any EKF algorithm, EKF SLAM makes a Gaussian noise assumption for the robot motion and the perception. The amount of uncertainty in the posterior must be relatively small since otherwise the linearization in EKFs tend to introduce intolerable errors.

3) Pose measurements. This algorithm can only process positive sightings of the sustained landmarks. It cannot process negative information that arises from the absence of sustained landmarks in the set of sensor measurements. This is a direct consequence of the Gaussian belief representation.

B. Large Scale Direct-SLAM (LSD-SLAM)

The LSD SLAM is the featureless SLAM, which operates on the direct point clouds of the image and allows the system to build large-scale, consistent maps of the environment, which it percepts. Along with highly accurate pose estimation based on direct image alignment, the 3D environment is reconstructed in real-time by using pose-graph of key frames, associated with semi-dense depth maps. These are obtained by filtering over a large number of pixel-wise point comparisons. The two key objectives fulfilled by using direct methods can be:

- A novel direct tracking method which operates on sim (3), thereby explicitly detecting scale-drift, and
- An elegant probabilistic solution to include the effect of noisy depth values into tracking. The resulting direct monocular SLAM system runs in real-time on a CPU.

1) Direct Odometry

Direct visual odometry (VO) methods circumvent the limitation of high CPU overhead by optimizing the geometry directly on the image intensities, which enables using all information in the image. In addition to higher accuracy and

robustness in particular with little keypoints at particular dense areas, this provides substantially more information about the geometry of the environment, which can be very valuable for drone navigation.

2) Outline of the Algorithm

The algorithm consists of three major components: tracking, depth map estimation and map optimization as visualized in Figure 1.

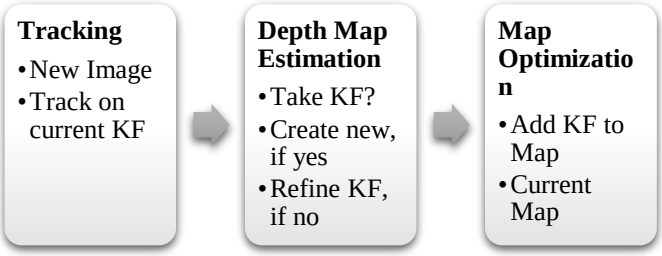

Fig. 1 Overview of LSD-SLAM

In brief, the algorithm undergoes three phases in its operation as follows:

- The **tracking** component continuously tracks new camera images. That is, it estimates their rigid body pose $\xi \epsilon$ se (3) with respect to the current key frame, using the pose of the previous frame as initialization.
- The **depth map estimation** component uses tracked frames to either refine or replace the current key frame. Depth is refined by filtering over many per-pixel, small-baseline stereo comparisons coupled with interleaved spatial regularization. If the camera has moved too far, a new key frame is initialized by projecting points from existing, close-by key frames into it.
- Once a **key frame** is replaced as tracking reference and hence its depth map will not be refined further, it is incorporated into the global map by the map optimization component. To detect loop closures and scale drift, a similarity transform $\xi \epsilon$ 2 se (3) to close-by existing key frames (including its direct predecessor) is estimated using scale-aware, direct sim(3)-image alignment.

3) Representation of the Map

The map is represented as a pose graph of key frames: Each key frame Ki consists of a camera image $I_i: \Omega_i \rightarrow \mathbb{R}$, an inverse depth map $D_i: \Omega_{Di} \rightarrow \mathbb{R}$ and the variance of the inverse depth $V_i: \Omega_{Di} \rightarrow \mathbb{R}$. Note that the depth map and variance are only defined for a subset of pixels $\Omega_{Di} \subset \Omega_i$, containing all image regions in the vicinity of sufficiently large intensity gradient, hence semi-dense. Edges ε_{ji} between key frames contain their relative alignment as similarity transform $\varepsilon_{ji} \epsilon \, \mathfrak{sim}(3)$, as well as the corresponding covariance matrix ε_{ji}.

4) Tracking new frames: direct se (3) image alignment

Starting from an existing key frame $K_i = (I_i; D_i; V_i)$, the relative 3D pose $\varepsilon_{ji} \epsilon \, \mathfrak{sim}(3)$ of a new image I_j is computed by minimizing the variance-normalized photometric error

$$E_p(\varepsilon_{ji}) = \sum_{p \in \Omega D_i} \left\| \frac{r_p^2 (p, \varepsilon_{ji})}{\sigma^2_{r_p(p,\varepsilon_{ji})}} \right\|_\delta$$

$$r_p(p, \varepsilon_{ji}) := I_i(p) - I_j(\omega(p, D_i(p), \varepsilon_{ji}))$$

$$\boldsymbol{\sigma^2_{r_p(p,\varepsilon_{ji})}} := 2\sigma_I^2 + \left(\frac{\partial r_p(p, \varepsilon_{ji})}{\partial D_i(p)} \right)^2 V_i(p)$$

Where $\| . \|_\delta$ is the Huber norm :

$$\| r^2 \|_\delta := \begin{cases} \frac{r^2}{2\delta} & \text{if } |r| \leq \delta \\ |r| - \frac{\delta}{2} & \text{otherwise} \end{cases} \qquad (4)$$

5) Keys and Depth maps

a) Key frame : If the camera moves too far away from the existing map, a new key frame is created from the most recent tracked image. The threshold accompanying weighted combination of relative distance and angle to the current key frame:

$$\text{dist}(\varepsilon_{ji}) := \varepsilon_{ji}^T \, W \, \varepsilon_{ji} \qquad (5)$$

where W is a diagonal matrix containing the weights. Note that, each key frame is scaled such that its mean inverse depth is one. This threshold is, therefore, relative to the current scale of the scene, and ensures sufficient possibilities for small baseline stereo comparisons.

b) Depth Map: Once a new frame is chosen to become a key frame, its depth map is initialized by projecting points from the previous key frame into it, followed by one iteration of spatial regularization and outlier removal. Afterward, the depth map is scaled to have a mean inverse depth of one this scaling factor is directly incorporated into the sim (3) camera pose. Finally, it replaces the previous key frame and is used for tracking subsequent new frames.

c) Refinement. Tracked frames that do not become a key frame are used to refine the current key frame: A high number of very efficient small baseline stereo comparisons is performed for image regions where the expected stereo accuracy is sufficiently large. The result is incorporated into the existing depth map, thereby refining it and potentially adding new pixels, this is done using the EKF filtering.

IV. THE SOFTWARE IMPLEMENTATION

The LSD SLAM is bound as a software for all its operations through ROS package. All the necessary nodes are built into ROS binary packages and cross compiled. ROS is an open source, node based software for robotics. The nodes are predefined in the c++ based codings, which also encapsulates all the other pipes and Process IDs into the same main level thread.

1) The software Architecture

For running the software, various new nodes are to be created as different real-time clients for core process. These are inter-related with each other and communicates within themselves by using the (3)

roscore main process. Figure 2 shows the architecture and relationship between these nodes: lsd_slam, usb_cam and ptam. The background services govern these nodes and their communication.

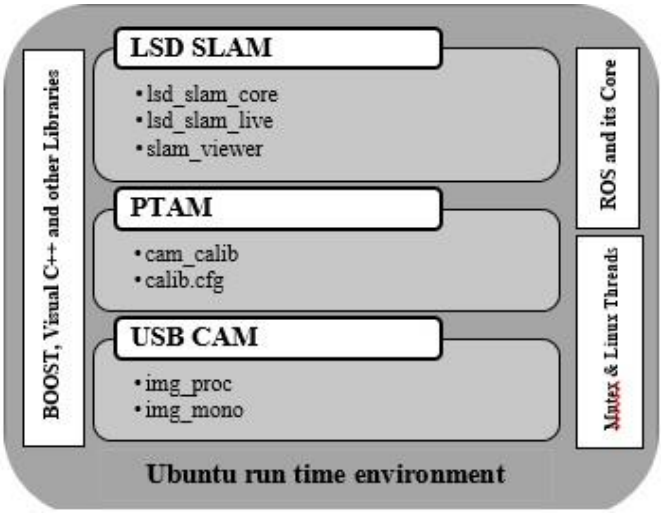

Fig. 2 The Software Architecture

2) Compilers and tools

This work is fully implemented via software by using Cross compilation of different compilers and language tools. The basic language used for the LSD SLAM process and its background processes like viewer, depth map estimator are fully written and compiled under C++. To bring more synchronization and integrity among all the nodes, the BOOST libraries are used. These are free, open source dynamically reconfigured subroutines written in C++. The tools are summarized as follows:

TABLE I. TABLE I COMPILER TOOLS USED

S. No.	Tool/compiler	Purpose
1.	Robotic Operating System	For running the LSD SLAM and its core services. To provide node communication and kernel based interaction with the algorithm.
2.	Open CV	To compile composite color geometry at the runtime, with both static and dynamic link libraries, which will be useful for I/O rendering with the algorithm.
3.	Open GL	To compile 3D geometry and visual c++ libraries for the purpose of core services of the ROS.
4.	Boost lib	It is a c++ library collection for implementing kernel services at user defined environment. With this libraries, mutex locks are implemented into the program.
5.	USB_CAM node	It is a package, which uses USB camera for giving inputs and outputs within the ROS.

V. SIMULATION RESULTS

The simulations done using the software tools as discussed earlier are presented here. Both SLAM algorithm and Calibration of live camera implementations are simulated. As SLAM is probabilistic model based system, the global coordinates and mapping coordinates are rendered simultaneously. The coherence between them should be acceptably accurate.

1) Results of LSD-SLAM Algorithm

The ROS is deployed with master-client node strategy in Ubuntu. The backend image rendering for the algorithm is provided by a camera calibration tool. Since the camera used for the algorithm's simulation must be synchronized with rendered images, this camera calibration tool gives real-time error correction and calibration. This is done by PTAM (Parallel tracking and mapping) camera calibrator tool. The output of this node is shown below.

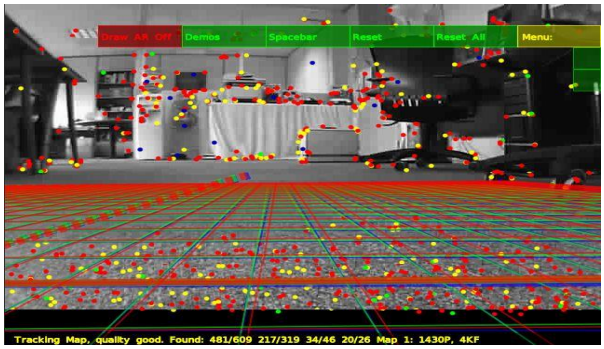

Fig. 3 Result of PTAM calibrator tool in ROS

Next, this calibrated VGA image sequences are streamed as input the core of the algorithm, i.e., SLAM node. Our algorithm must have two components viz., the DEBUG WINDOW depth image and 3D POINT CLOUD mapping for occupancy. These two windows provide us the correct mapping of the environment and depth image synchronization. The tracking of the global frame and current observance frame are also done by the algorithm.

The overall output of SLAM algorithm can be rendered with two types of inputs: Live SLAM and Dataset SLAM. The live SLAM is the real-time algorithm implementation by connecting a webcam with the system and to track the occupancy for odometry purposes. But on the other hand, the dataset SLAM gives an output of already recorded dataset video. The simulation was done for both types of algorithmic implementation. They are carried out on Pentium processor powered Laptop at No overclocked frequencies. This is an evident that the algorithm runs with no computational overhead. The figure below shows the output of dataset SLAM in the system:

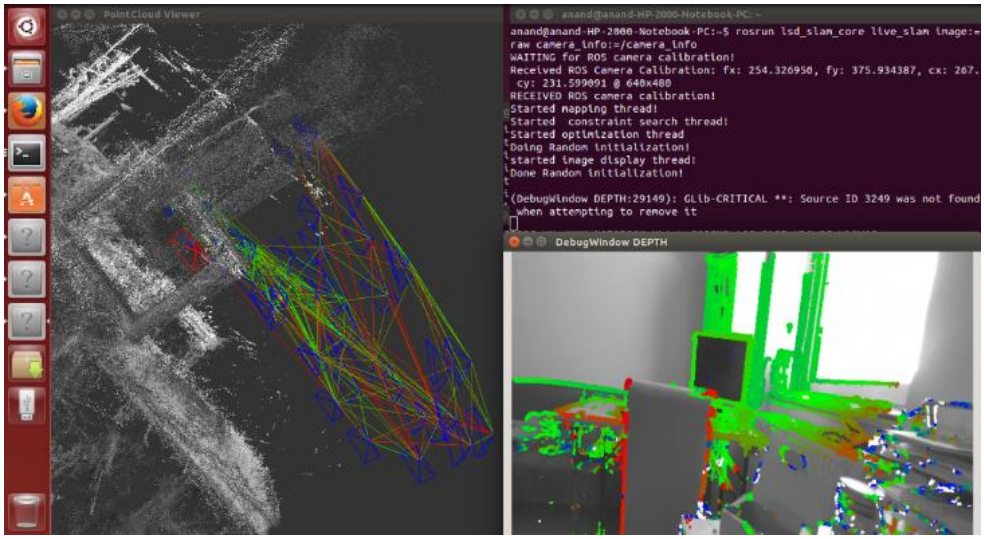

Fig. 4 Result of Dataset SLAM in ROS

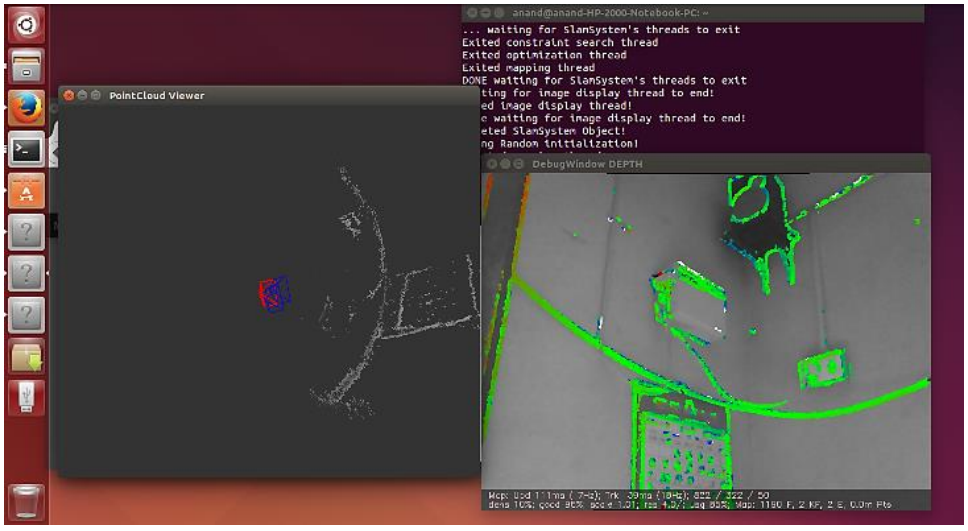

Fig. 5 Result of live SLAM in ROS

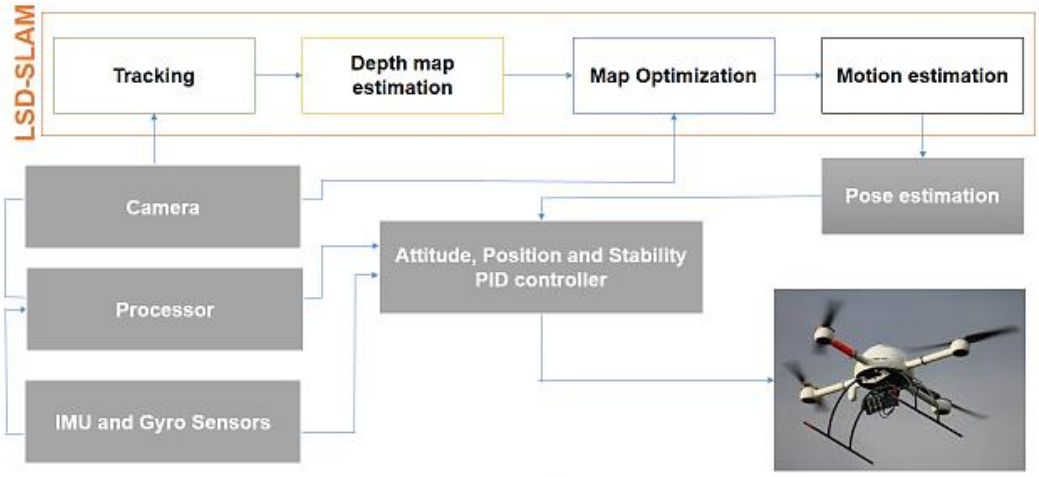

Fig 6 Block diagram of future implementation

VI. CONCLUSION AND FUTURE WORKS

The expected results for this work is tried to implement using the software and simulation. It is successfully done to bring the closeness of the objective. By using the proposed simulation, following points can be observed:

• The ground truth is improved by using the proposed algorithm. The plotting accuracy in terms of the number of point clouds is also increased[1].

• For large loop closure, the depth of the map is clearly closed with the short period of time, thanks to the Parallel Tracking and Mapping feature.

• The key frame observance and remapping in a case of tracking loss is highly accurate. When comparing to the feature and sensor based SLAM algorithms, this direct SLAM technique brings less computational overhead to the CPU as well as simulates faster[2].

However, there are several future enhancements that can be done. These include introduction of IMUs and Gyroscopes along with the Quadcopter to improve accuracy, implementing the direct SLAM technique with an additional camera for improved 3D loop closure as well as observance and implementing this software in real time with the help of low-cost Quadcopter arrangements. Figure 6 shows our future implementation with the sensors and drone.

REFERENCES

[1] Changhong Fu, A. Carrio, P. Campoy, "Efficient Visual Odometry and Mapping for Unmanned Aerial Vehicle Using ARM-based Stereo Vision Pre-Processing System" in International Conference on Unmanned Aircraft Systems (ICUAS) / 2015. pp. 957 – 968.

[2] M. Pizzoli, C. Forster, and D. Scaramuzza, “Remode: Probabilistic, monocular dense reconstruction in real time,” in Robotics and Automation (ICRA), 2014 IEEE International Conference on, pp. 2609–2616.

[3] J. Engel, T. Schps, and D. Cremers, “Lsd-slam: Large-scale direct monocular slam,” in Computer Vision ECCV 2014, ser. Lecture Notes in Computer Science, vol. 8690, 2014, pp. 834–849.

[4] J. Zhang, M. Kaess, and S. Singh, "Real-time Depth Enhanced Monocular Odometry," in Intelligent Robots and Systems (IROS), 2014 IEEE/RSJ International Conference on, 2014, pp. 4973–4980.

[5] C. Fu, M. Olivares-Mendez, R. Suarez-Fernandez, and P. Campoy, “Monocular Visual-Inertial SLAM-based Collision Avoidance Strategy for Fail-Safe UAV Using Fuzzy Logic Controllers,” Journal of Intelligent & Robotic Systems, vol. 73, no. 1-4, pp. 513–533.

[6] A. Geiger, J. Ziegler, and C. Stiller (2011), “Stereoscan: Dense 3d reconstruction in real-time,” in Intelligent Vehicles Symposium (IV), IEEE, 2014, pp. 963–968.

[7] G. Camellini, M. Felisa, P. Medici, P. Zani, F. Gregoretti, C. Passerone, and R. Passerone, “3DV - An embedded, dense stereovision-based depth mapping system,” in Intelligent Vehicles Symposium Proceedings, IEEE, 2014, pp. 1435–1440.

[8] J. Nikolic, J. Rehder, M. Burri, P. Gohl, S. Leutenegger, P. T. Furgale, and R. Siegwart, “A Synchronized Visual-Inertial Sensor System with FPGA Pre-Processing for Accurate Real-Time SLAM,” in Robotics and Automation (ICRA), 2014 IEEE International Conference on, 2014, pp. 431–437.

[9] K. Schmid and H. Hirschmuller, “Stereo vision and imu based realtime ego-motion and depth image computation on a handheld device,” in Robotics and Automation (ICRA), 2013 IEEE International Conference on, pp. 4671–4678.

[10] S. Hermann and R. Klette, “Evaluation of a new coarse-to-fine strategy for fast semi-global stereo matching,” in Advances in Image and Video Technology, ser. Lecture Notes in Computer Science, vol. 7087, 2012, pp. 395–406.

[11] B. Kitt, A. Geiger, and H. Lategahn, “Visual odometry based on stereo image sequences with ransac-based outlier rejection scheme,” in Intelligent Vehicles Symposium (IV), 2010 IEEE, pp. 486–492.

[12] E. Rosten and T. Drummond, “Machine Learning for High-speed Corner Detection,” in Computer Vision ECCV, 9th European Conference on, 2006, pp. 430–443.

[13] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, “BRIEF: Binary Robust Independent Elementary Features,” in 2010 11th European Conference on Computer Vision (ECCV), 2010 pp. 778–792.

[14] C. Harris and M. Stephens, “A combined corner and edge detector,” in Proc. of Fourth Alvey Vision Conference, 1988, pp. 147–151.

[15] A. Hornung, K. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, “Octomap: an efficient probabilistic 3d mapping framework based on octrees,” Autonomous Robots, vol. 34, no. 3, 2013, pp. 189–206.