



Reconstructing a 3D Room from a Kinect Carrying UAV

Eleni Mangina^(✉), Conor Gannon, and Evan O’Keeffe

School of Computer Science, University College Dublin, Dublin, Ireland
eleni.mangina@ucd.ie
<https://people.ucd.ie/eleni.mangina>

Abstract. This project focuses on the creation of a portable SLAM (Simultaneous Localisation and Mapping) system, which uses an Unmanned Aerial Vehicle (UAV) as the transportation medium. The main purpose of the system is to create a 3D map of the environment, while concurrently localizing itself within the map. The real world applications of this system concentrate on search and rescue scenarios. The system uses the Microsoft Kinect as its primary sensor. Within this project we utilized Visual SLAM, which is the process of using data from the Kinect sensor to calculate position. The algorithm looked at successive frames and depth estimates from the Kinect and then matched features across the images to calculate distance and stability. The work presented in this paper is approached from a practical point of view rather than purely theoretical basis. The end result is a physical prototype which is ready to be deployed in the field for further testing.

Keywords: UAVs · Simultaneous Localisation and Mapping

1 Introduction

The aim of the project was to create a prototype capable of mapping environments in 3D for use in search and rescue operations - focusing mainly on collapsed buildings. UAVs are more versatile than traditional ground robots and they are financially affordable. A GPS is not sufficient for navigation indoors due to range and reception issues. The UAV must rely on on-board sensors to accurately gauge its state and plan trajectories.

The release of the Microsoft Kinect attracted attention among the scientific community; a module (OpenNI) was made for ROS (Robotic Operating System) [14]. The Kinect is a motion sensing device with three sensors, with an infrared laser projector, a colour camera and an infrared camera. The infrared laser projector is a depth sensor; it projects the environment in 3D with any lighting condition. The cameras have 640×480 resolution running at 30 frames per second. This allows the data to be transmitted in real time. The Kinect can be stripped down to a satisfactory weight of 115 grams, an easy payload for the

UAV. The motion of image features from a camera can be used to create the trajectory for the UAV and also create a 3D model of the scene.

The method for calculating the trajectory uses feature matching between image frames, is Visual Odometry. If the matching is between a live map of the scene and the current image, it is Visual SLAM [18]. With Visual Slam drift does not occur. The only cost of creating and maintaining a live map is based on the computational power and the complexity needed. Visual SLAM allows a UAV in a new novel environment to gradually create an accurate map of the environment, while concurrently estimating its position within the map. Most of the current systems used for search and rescue operations are expensive and are not mass produced - making such systems financially inaccessible in impoverished regions. The idea of this project was to create a prototype that could be utilized and built in developing countries where emergency relief is slow or non-existent.

A set of underlying principles were set for the project - all of the software needed to be open source; the hardware needed to be robust and cost effective and the final prototype needed to be user friendly for non-technical operators. The objective was to create a system that would allow users to respond and react to emergency situations themselves, reducing reliance on external factors.

This paper presents the background work of the research, the design and implementation of this work. Finally the testing and evaluation of the proposed solution is presented along with the conclusions for further development.

2 Background Work

The background research focused mainly on two topics - Visual SLAM and Visual Odometry using a Kinect RGB-D camera. Although there are many different SLAM methods there is a constant principle of non existence of the environment map at the beginning. The UAV becomes the origin of the coordinate system and the measurement taken at this location is the initial measurements. Every measurement after this contains already known data and new novel data. The UAV can find an overlapping by comparing the current measurement with the data set and then calculate its new position. When the new measured data is inserted into the map, the environment can then be mapped incrementally. As GPS performs poorly within buildings, many indoor positioning methods have been developed but they have proved to be relatively expensive and require regular maintenance and calibration. The Kinect allows a novel SLAM (Simultaneous Localization And Mapping) method to be employed by providing colour images and dense depth maps. By combining the visual features and the scale information of the 3D depth sensing, a dense 3D environment can be represented. The research for this topic focused mainly on the evaluation of the RGB-D SLAM System [3] and the evaluation of SLAM Approaches for Microsoft Kinect [16].

In [3] the authors split the trajectory estimation into a front end and back end. The front end extracts spatial relations between individual observations and the back end optimizes the poses of these observations in a so-called pose graph and with respect to a non-linear error function. OpenCV [2] is used for

detection and matching of the different feature types: SURF (Speeded up Robust Feature) [1], ORB [15] and SIFT (Scale invariant feature transform) [11]. The authors evaluate the accuracy and processing time for the three feature descriptors. OpenCV is essentially a library of programming functions used for real time computer vision. ORB is a key point detector and feature descriptor. The SURF key point detector keeps the number of key points stable using a self-adjusting variable. The feature locations are then projected from the image to 3D using the depth measurement at the centre of the key point. “However, no visual feature provides perfect reliability with respect to repetition and false positives”, to solve the problem of noisy data the RANSAC [4] algorithm can be used. In terms of the back end, “The pairwise transformations between sensor poses, as computed by the front end, form the edges of a pose graph”. This does not form a globally consistent trajectory due to inaccuracies in the estimation. To correct this, the pose graph can be optimized using the g2o framework [10]. The g2o framework is an easily extensible graph optimizer that can be applied to a wide range of problems including several variants of SLAM and bundle adjustment. Many problems in computer graphics involve the minimization of a nonlinear error function that can be represented as a graph. In graph based SLAM the state variables are the positions of the drone in the environment or the location of the landmarks in the map that can be observed with the UAV’ sensors. The measurement relies on the relative location of the two state variables. If represented in a graph each node of the graph is a state variable that needs to be optimized and each edge between the variables is a pairwise observation of the nodes it connects. In summary, this approach extracts visual key points from the colour images and uses depth images to localize them in 3D. RANSAC is used to estimate the transformations between RGB-D frames and optimize the pose graph using non-linear optimization.

Corporate research in technologies, as described in [16], focuses on two different approaches, the first is based on visual key points the second one is based on point clouds. The following SLAM algorithms are evaluated in the paper: visual key points, hybrid and ICP. For visual key points, the key points have to be “detected and categorized”. For example, using SURF (Speeded up Robust Feature) [1] or SIFT (Scale-invariant feature transform) [11]. The key points between two pictures that were found using the SIFT and SURF detectors can be matched with the minimal Euclidean distance technique. The key point pairs are then used to calculate the position transformation using RANSAC [4]. The goal of the RANSAC algorithm is to find a suitable model that describes the position transformation best. The authors describe the RANSAC algorithm with 4 steps in an intuitive manner. First randomly choose similar key point pairs. Define a characteristic for the model. Apply the model to all of the key points of the first picture, the key point pairs fitting the model are inliers. If the new model is better than the current model, simply replace the current model with the new. The authors allude to the main problem for SLAM methods, as the “The errors in sensor measurement accumulates over time and results in a deviation that also increases over time”. To combat the errors, the TORO (Tree-based net-

work optimizer) optimization is used [8]. The point clouds approach uses ICP (Iterative Closest Point), point clouds are inputted to then calculate position transformations. The authors use a generalized ICP approach [17]. This approach takes two point clouds which are moderately overlapping or identical and then aligns them.

2.1 Project Background Research

Research was carried out to find the best solution for path planning using SLAM. One of the objectives for the project is to allow the UAV to autonomously plan the best route for navigation. Most SLAM algorithms represent the environment using a sparse set of features. This type of representation is not sufficient for collision free path planning; it lacks information about the obstacles in the surrounding environment. The most successful path planning methods are those based on randomized sampling. Samples are stochastically drawn in the configuration space. Then neighbouring collision free samples are connected via collision-free paths forming a road map. This road map is later used to connect any two given configurations. All paths in the configuration space are equally valid. The aim is to determine the shortest path between the given start and goal configurations.

Belief Road Maps (BRM) Algorithm: In BRM the edges defining the road map include information about the uncertainty change when traversing the corresponding edge. However, the BRM technique needs an existing model of the environment. [19] technical report examines Pose SLAM graphs and their use as belief road maps (BRMs). The BRM algorithm uses a known model of the environment. It then uses probabilistic sampling to generate a roadmap. The authors employ a technique in which builds the road map on line using the Pose SLAM algorithm. The result blends BRM and Pose SLAM, “method that devises optimal navigation strategies on-line by searching for the path with lowest accumulated uncertainty for the robot pose”. For Pose SLAM, only the trajectory is estimated and landmarks are used to provide relative constraints between robot poses.

Probabilistic Road Maps (PRM) Algorithm: The map for Pose SLAM only contains the collision free trajectories. Belief road maps were originally used as a variant of probabilistic road maps (PRMs). As with PRMs, BRMs are constructed by probabilistic sampling in configuration space of a given environment model. The authors argue that the set of poses defining the map in Pose SLAM can be used as the starting point for a BRM. The poses stored in the map by Pose SLAM during exploration are without obstacles, they were already traversed by the robot when the map was originally built. The result of the Pose SLAM algorithm is a directed graph, in which the nodes are poses or way points, and the edges are established through Odometry or sensor registration of the environment. The poses in the Pose SLAM map can be taken

as samples for a belief road map and used for planning. An advantage of using the Pose SLAM graph versus any other delayed-state SLAM graph is that Pose SLAM has its nodes equally distributed in the information space. This allows an estimation to be made taking into account the length of the path and the uncertainty in the poses. This is in contrast with existing approaches that either take into account the path length or the path uncertainty, and combinations of both methods is usually unreliable considering they are defined in different spaces and with different units.

RGB-D Mapping: In [5] the authors introduce the idea of “using a RGB-D camera to generate dense 3D models of indoor environments”. Sparse points are extracted from the RGB-D images and then matched using the RANSAC method. Iterative Closest Point matching estimates the best alignment between the frames. Alignment between successive frames can give an accurate estimation for tracking the UAV over small distances. But errors start to accumulate, most notably when the Drone is following a long path and it encounters a previously visited location. The error in frame alignment results in a map that has two representations of one location. This is the loop closure problem. The Visual Odometry approach suffers from this problem significantly - however there are many solutions to try and combat the problem. In [5] Loop closures are detected using a method of matching data frames with a collection of previously collected frames. Consistent alignments can be achieved in a multitude of ways such as sparse bundle adjustment. Further research on the loop closure problem was carried out and the research in [12] proposes a strong alternative approach of detection and correction at key frame-rate, in loops containing hundreds of key frames.

Visual SLAM and Visual Odometry: In [20] the idea of combining Visual SLAM and Visual Odometry is introduced. The Visual Odometry technique tracks hundreds of visual features per frame giving a clear and accurate estimation of relative camera motion, but drift occurs. The combination keeps the map minimal but uses more inter frame point matches leading to a more accurate pose estimate. The landmarks are not permanent therefore the computation overhead is still light. By examining the literature surrounding Visual Odometry and mapping using an RGB-D camera, there is a possibility of combining aspects of both for the project. [6] examines “the best practices in Visual Odometry using an RGB-D camera to enable the control of a micro air vehicle”. These methods have become somewhat redundant with the advent of cheap structured light RGB-D cameras. However, the RGB-D cameras have limited range because of their projectors. The position and velocity can be estimated with the sensor data; the 3D motion is estimated by comparing the relative motion of the Drone “from sensor frame to sensor frame”. A map can then be built with the positional data. In [6] the authors describe the algorithm they have developed for the Visual Odometry process using the following steps: image pre-processing,

feature extraction, initial rotation estimation, feature matching, inlier detection and motion estimation. There are many ways to approach Visual Odometry.

2.2 Project Definition

The background research has proved that 3D maps of indoor environments can be made with inexpensive cameras such as the Kinect and autonomous flight for UAVs in indoor environments is also possible with current Visual Odometry methods. The literature proves that it is possible to create a system that is, “able to plan complex 3D paths in cluttered environments while retaining a high degree of situational awareness”. However, there are still many problems that need to be addressed.

- Current mapping techniques only use “two consecutive frames to estimate the motion of the camera”.
- The loop closures detected by frame to frame for visual feature matching, is not sufficient for the creation of large accurate maps.
- Most Visual Odometry techniques do not perform well in environments with few visual features.
- The Kinect performs better in small cluttered scenarios, as the Kinect’s range is not sufficient for large expansive outdoor environments.

This work is inspired by research carried out at the University of St Andrews [9], which uses the Kinect’s depth sensing capabilities to aid the blind navigate novel environments. An attempt was made to allow the UAV to fly autonomously by using the Kinect’s depth information alone. The UAV would essentially navigate its environment by taking turns of the least resistance. For example, if there was an object ahead it would rotate and turn; if there was no object ahead it would move forward - this process would allow the UAV to map the environment and navigate safely without colliding with objects. This form of navigation would require no operator; the UAV or multiple UAVs could be left to their own devices until they run out of power.

3 Design and Implementation

The aim of this project was to create a prototype capable of mapping environments in 3D for use in search and rescue operations – focusing mainly on collapsed buildings. Within these kind of Scenarios there is no GPS signal available for localization. From a practical point of view, operators of the prototype must be able to view the map that is being built and see the trajectory and location of the entity within the map. This will allow the operators to become spatially aware of the environment as they prepare to enter the building. A SLAM (Simultaneous Localisation and Mapping) approach was implemented. Mapping integrates all of the entities sensor information into a representation of the environment and localization is the process of estimating the pose of the

entity relative to the map. In essence SLAM is the process of building a map while at the same time localizing the entity within the map.

The prototype needed to be robust and easy to use for such environments - it would have to withstand hard handling in extreme environments and have the ability to be operated by non-technical personnel. From the onset it was clear that the prototype would have to be completely wireless. This led to many technical questions - for example, how to transmit the live data and how to power the system. To bring the prototype from a loose idea into a clear design and plan, the system was broken down into various parts. The high level design is shown in Fig. 1. Transforming the Microsoft Kinect from an entertainment device into a robotic sensor proved to be rewarding and challenging. The Kinect first had to be stripped to its most minimal state due to weight restrictions imposed by the UAV's maximum payload. The goal was to remove as much hardware and housing as possible, as long as it did not interfere with the functionality of the two on board cameras (IR CMOS and Colour CMOS) and IR projector.

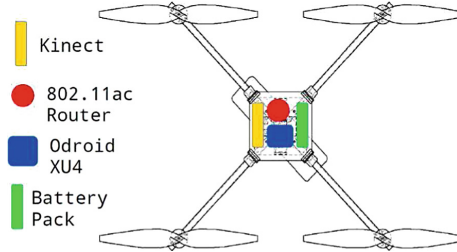


Fig. 1. High level of proposed design

As the prototype needed to be completely wireless, every component mounted on the drone had to be battery powered. The first approach taken was to build a battery box containing 8AA battery's, which gives 12 v. This solution did not align with some of the goals that were set for building the prototype - normal and rechargeable AA batteries are expensive and deteriorate quickly. Even though the experiment failed, it was concluded that the Kinect can run on a portable battery source effectively. A 12 v 6800 mAh DC rechargeable lithium ion battery was used instead. To ensure steady and correct power regulation, a RioRand adjustable boost power supply voltage converter with a digital voltmeter was used. The battery may not output 12 v 100% of the time and the RioRand acts as a booster between the Kinect and the battery source.

To utilize the data produced by the Kinect, the libfreenect driver was used, as shown in Fig. 2. The libfreenect driver, which is open source, allows access to the raw data streams produced by the Kinect- for example the depth stream, IR stream and RGB stream. OpenNI is another competing open source library, which is based on the original and official prime sense code, as shown in Fig. 3. Due to its commercial inclinations we did not utilise it for this project. The

Kinect is attached to the Odroid XU4 via USB. The Odroid runs Ubuntu 14.04 arm edition - tests were carried out with Ubuntu Mate and Arch Linux but were not compatible with ROS Indigo. ROS was manually installed on the Odroid along with the libfreenect drivers. When the Kinect is powered and connected, the libfreenect driver is launched with the aid of ROS. The data produced by the Kinect is published to various topics - for example depth stream and IR stream. The ROS master runs on the Odroid and coordinates the system from here.

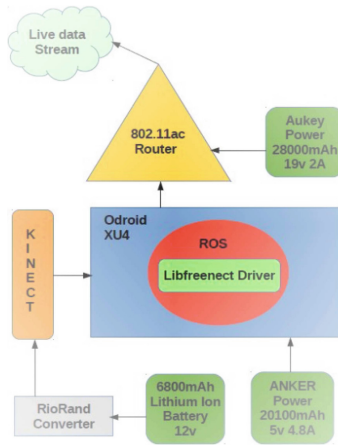


Fig. 2. Proposed solution for data transfer

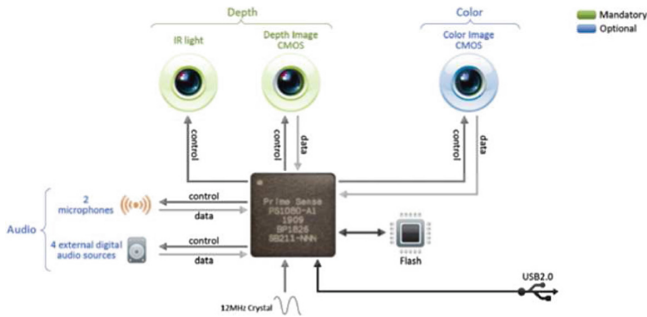


Fig. 3. PrimeSense data flow and control [13]

The Odroid is connected to a router via its Gigabit Ethernet port. ROS provides the infrastructure to create a network allowing the various nodes to contact and communicate with each other in a distributed manner. The first approach taken to power the Odroid used a 12 v lithium Ion battery and connected it to

a DROK step down voltage transformer, it takes an input of 12–24 v and converts it to 5 v using 4.8 amps. This is the exact voltage and amperage needed to power the Odroid. This solution worked, but it proved to be relatively delicate due to the soldering and wiring required to set it up. It would not be robust enough to survive a search and rescue environment. Instead, an Anker Power Core was used, as a portable power bank. The unit can be recharged and is very strong and robust. It proved to be a much better solution than the handmade DROK converter.

As previously mentioned it was not feasible to transmit the data from a wireless adapter connected to the Odroid. There proved to be too much latency for a SLAM solution to operate efficiently, the results were not good enough to benefit a search and rescue team on the ground. Instead the Odroid is connected to a wireless router via a gigabit Ethernet port. This gives the speed necessary for low levels of latency, the router can transmit the data much more efficiently and with higher power than any wireless adapter. The base station machine then connects to the wireless network and pulls the data stream for processing by one of the two SLAM algorithms. The wireless standard chosen was 802.11ac, this is the fastest standard and is roughly 3x times faster than 802.11n. It also has beamforming technology which is essentially a smart signal - meaning the signal is channelled in the direction of the connected devices rather than transmitting the signal in every direction. It operates on the 5Ghz band which means there is less interference. 802.11 ac is the only standard that produces satisfactory results - 802.11 a/b/g/n were tested as they are the most widely used standards. To give an idea of how much data is transmitted over the network various ROS bag files were recorded. The bag files were in the GB range for tests of less than one minute. The router is powered by an Aukey 28000 mAh multi voltage external battery. The battery can output 5,12 or 19 v at 2 amps - the battery outputs the power with a USB cable. In order to transfer the power to the router, a 3.5 mm outer DC connector was soldered to an existing USB power cable and then connected to the router.

The Robotic Operating System (ROS) coordinates the data transfer process between nodes. The wireless network is created by the on board router and the base station machine connects to the wireless network. A script is run on the base station machine and the Odroid. The script passes parameters to ROS, for example the IP address of the various devices and settings. For the libfreenect driver, parameters include the rate to throttle the images i.e. 5 Hz. On the base station one of the two algorithms packages, RGBDSLAM or RTAB-Map process the data and perform SLAM. The packages subscribe to the topics needed and the master coordinate the process, the processing is then done on the base station machine resulting in a 3D map of the environment and online SLAM.

3.1 Visual SLAM Algorithms

There are many Visual SLAM algorithms in existence, since the Kinect was being used it had to be narrowed down to algorithms that used RGB-D sensors.

RGBDSLAM: uses visual features such as SIFT (Scale Invariant Feature Transformation) and SURF (Speeded Up Robust Features) to match images and then uses RANSAC to estimate the 3D transformation between them. Objects in images contain defining points, SIFT extracts such points and turns them into a feature descriptor- the feature descriptor can then be used to identify the object in new novel images. SIFT allows recognition even if the objects are of different scale, orientation and illumination. SURF is based on SIFT and claims to be three times faster. However, these results could not be reproduced in the testing process. The deciding factor for which algorithm performed best was the ability to be used in synergy with the prototype in a real world situation. The RGBDSLAM algorithm proved to be ineffective for practical purposes. This will be explained in the testing and evaluation section.

RTAB-Map: Real Time Appearance-Based Mapping proved to be the best algorithm for practical use. It is based on an incremental appearance based loop closure detector which uses a bag of words technique to identify if an image has come from a novel location or previously visited one. Rviz and RtabmapViz are the two main visualization tools that can be used with the algorithm. As shown in the videos created, Rviz gives an excellent overview of the map building process but it is not practical for an operator to use. RtabmapViz shows the map being built in real time from the drones' perspective and also shows a clear trajectory. It also shows loop closure detection in action and other topics can be attached to the visualisation screen such as depth stream or IR stream. Rtabmapviz was modified to add a live RGBD stream in order to aid the operator navigate the transport medium - in this case a UAV.

4 Testing and Evaluation

The testing and evaluation was approached from a practical point of view. The hardware components were tested in terms of their ability to perform in harsh environments, charge capacity and general robustness. In terms of software the SLAM algorithms RGBDSLAM and RTAB-Map were tested and compared. The whole system was then tested in synergy to obtain a final result. Rather than comparing certain metrics against each other, the case of the two SLAM algorithms and their performance in a number of real world scenarios were compared -speed of map building; clarity of map building; how they react in low light environments with few features; how susceptible are they to rapid movements without losing their position within the map; level of detail within the map and practical details such as computational power needed to obtain accurate results. One had to take the position of an operator to test the system rather than approach it from a purely theoretical basis.

The first test consisted of finding the maximum run time of the system, the performance of the on-board power sources dictated this. There were three battery sources, one for each of the components in the system, all of which have different capacities and power requirements. The test consisted of letting the

system run until one of the power sources failed. Five tests were carried out and the mean result was 45 min before the system had power failure. In the future, a new power source will be needed to reduce the complexity.

Three volunteers with no technical ability were given the system and asked to use it following the instruction manual. A detailed instruction manual was created to show non-technical personnel how to use the system. The end goal was to refine the system to be a complete black box; the operator would only have to perform a limited number of tasks to enable full functionality. Two of the users had little difficulty getting full functionality but the third needed further assistance. In terms of robustness, the main discoveries were weaknesses connecting the components - more specifically soldering joins and crimps. A 3D design for a case is currently being created to house the components safely, when the design is complete it can be 3D printed.

In terms of the Visual SLAM algorithms, RGBDSLAM failed a number of tests relating to practical applications of the system in a real world scenario. An early decision was made that RTAB-Map would be used for the final system. RGBDSLAM does not build a clear concise map quickly and it takes a considerable amount of time to build anything that can be utilised by the operator. RGBDSLAM also required a powerful machine. Considering this system needs to be deployed and improved upon in the developing world, a machine of average capability needs to be sufficient to process the data. However, RGBDSLAM performs well when it is connected directly to a machine as opposed to transmitting the data produced over the wireless network. Most of the applications to date have used RGBDSLAM on a ground robot connected directly to a powerful machine, which processes the data on board. In the case of a UAV, it is possible to use powerful computers to do on board processing but this comes with a cost of capital and size - which did not align with this project.

RTAB-Map allows for modifications to be made in the case of remote mapping, for example the data stream can be compressed and the rate of transmission can be changed - 10 HZ worked best in this case. RGBDSLAM does not allow for compression, other measures were taken to increase performance including using a greyscale stream instead of colour. None of these worked in terms of increasing the performance of the real time map building process. It was concluded that RGBDSLAM was not useful for any real world application using the current system. The testing of the whole system consisted of using the system in a number of locations for a fixed period of time under certain conditions, as shown in Table 1. The system performed best in enclosed environments with low UV exposure, as UV washes out the infrared pattern created by the Kinect. The best result came from a 20-minute test in an abandoned apartment complex stairwell, as shown in Fig. 4. There was an ample amount of features at the location - the trajectory and map were extremely accurate from an operator's point of view.

The system does not perform well in large expanses with few features. A test was carried out for 20 min in a car park basement. The basement had few features and was expansive, which led to a poor result. Testing for a scenario in a disaster situation was difficult - a dark and cramped attic proved to be the

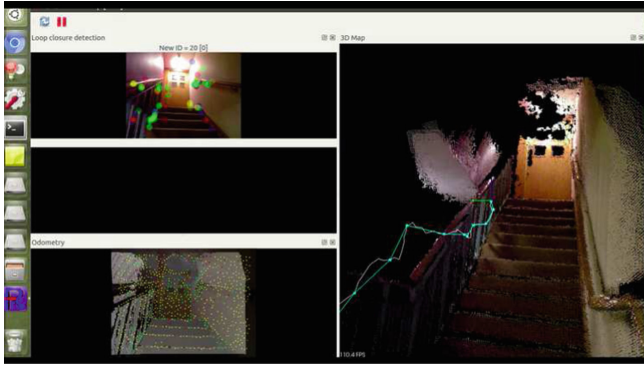


Fig. 4. 20-minute test in an abandoned apartment complex stairwell [7]

best location to do this. From an operators' point of view, the live stream was perfect and the map was coherent - artificial lighting had to be added to the environment to produce the results. This indicates that a high powered LED will most likely have to be added to the system to ensure accurate results for all scenarios.

Table 1. Testing results from a Kinect carrying UAV

Scenario	Accuracy	Speed	Latency	Map detail	Performance
Apartment complex	25 m	Real time	Unrecognizable	High	Very good
Underground car park	Inadequate	Very slow	Inadequate	Poor	Inadequate
Confined Attic	Inadequate	Very slow	Inadequate	Poor	Inadequate
Warehouse	Inadequate	Very slow	Inadequate	Poor	Inadequate
Office block	0.5 m	Real time	Unrecognizable	High	Very good

Tests were carried out with chaotic trajectories and speeds to gain an insight into how the system would react in such a scenario. As of now irregular movements sometimes cause the system to lose its position within the map, which requires the Odometry to be reset. Most of the time this reset causes the map to be destroyed, which means all previous data is lost. This is obviously an issue that needs to be resolved. At the moment possible solutions include adding a separate sensor so that the system does not rely on the Kinect's information alone. However, this may not be a major issue as the UAV's hovering abilities keep it stable, especially as it will be hovering at very low speeds so rapid irregular movements are unlikely. In the case of computational power needed for base station processing, RTAB-Map performs much better than RGBDSLAM but still requires significant power. After 20–25 min of use, the map building and localization process start to lag and eventually become inoperable. There is a strong possibility that modifying the system will solve this issue, for example increasing RAM allocations.

5 Conclusions and Future Work

Overall the project has been successful with a prototype built for a portable SLAM system. The prototype can create a 3D map of its environment, while simultaneously localizing itself within the map that is being constructed. The system is not just a theoretical concept; it is a physical object that can be tested in the real world. The system created can be replicated easily, produced for less than 200 dollars and can be used by non-technical personnel.

There is a considerable amount of capital being invested in the robotics industry. The systems that are produced are extremely expensive and not widely available. It became apparent that robotic systems for search and rescue operations would only be available to a select few due to high commercial costs. Most of the developing world does not have the financial means to avail of such systems. Our focus was to create an open source solution that can be applied to create cost effective systems easily to be built and replicated in the developing world where emergency relief is slow. If a region had several of these systems, the inhabitants could take the rescue into their own hands. There is still a considerable amount of work to do, new issues emerge every time the prototype is tested to its limit. The testing process needs to cover as many scenarios as possible. Currently the system works in limited environments and scenarios. The aim is to make the prototype user friendly, as it needs to operate as a black box. The current manual has many steps and requirements. This needs to be refined further and rewritten in a style of language that is accessible to all. The blueprint for building the system is also being refined - the hope is that the blueprint will allow and encourage enthusiasts to improve the prototype.

Future work will also include researching further the state of the art UAV technologies. This project focused mainly on the portable SLAM system that would be mounted on the UAV. More research needs to be carried out in terms of the practicalities of using different types of UAV models.

References

1. Bay, H., Ess, A., Tuytelaars, T., Van Gool, L.: Speeded-up robust features (SURF). *Comput. Vis. Image Underst.* **110**(3), pp. 346–359 (2008). <https://doi.org/10.1016/j.cviu.2007.09.014>
2. Bradski, G., Kaehler, A.: *Learning OpenCV: Computer Vision with the OpenCV Library*. O'Reilly Media, Sebastopol (2008)
3. Endres, F., Hess, J., Engelhard, N., Sturm, J., Cremers, D., Burgard, W.: An evaluation of the RGB-D SLAM system. In: *2012 IEEE International Conference on Robotics and Automation*, Saint Paul, MN, pp. 1691–1696 (2012). <https://doi.org/10.1109/ICRA.2012.6225199>
4. Fischler, M.A., Bolles, R.C.: Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM* **24**(6), 381–395 (1981). <https://doi.org/10.1145/358669.358692>
5. Henry, P., Krainin, M., Herbst, E., Ren, X., Fox, D.: RGB-D mapping: using kinect-style depth cameras for dense 3D modelling of indoor environments. *Int. J. Robot. Res.* **31**(5), 647–663 (2012). <https://doi.org/10.1177/0278364911434148>

6. Huang, A.S., et al.: Visual odometry and mapping for autonomous flight using an RGB-D camera. In: International Symposium of Robotics Research (ISRR) (2011). <https://doi.org/10.1177/0278364912455256>
7. Gannon, C.: Live test of portable SLAM prototype (2017). <https://www.youtube.com/channel/UCYDCmP7B87dvna9YHEMXT3A>
8. Kim, J.H., Matson, E.T., Myung, H., Xu, P.: Robot Intelligence Technology and Applications 2012, An Edition of the Presented Papers from the 1st International Conference on Robot Intelligence Technology and Applications. Springer, Heidelberg (2013). <https://doi.org/10.1007/978-3-642-37374-9>
9. Kristensson, O.: Supporting Blind Navigation using Depth Sensing and sonification. School of Computer Science, University of St Andrews. United Kingdom (2013). <https://doi.org/10.1145/2494091.2494173>
10. Kümmerle, R., Grisetti, G., Strasdat, H., Konolige, K., Burgard, W.: G2o: a general framework for graph optimization. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), pp. 3607–3613. <https://doi.org/10.1109/ICRA.2011.5979949>
11. Lowe, D.G.: Int. J. Comput. Vis. **60**, 91 (2004). <https://doi.org/10.1023/B:VISI.0000029664.99615.94>
12. Mur-Artal, R., Tardós, J.D.: Fast relocalisation and loop closing in keyframe-based SLAM. In: 2014 IEEE International Conference on Robotics and Automation (ICRA), Hong Kong, pp. 846–853 (2014). <https://doi.org/10.1109/ICRA.2014.6906953>
13. PrimeSense.org: Subsidiary of Apple Inc. (n.d). <http://www.PrimeSense.org/>
14. ROS.org: Powering the world’s robots (n.d.). <http://www.ros.org/>
15. Rublee, E., Rabaud, V., Konolige, K., Bradski, G.: ORB: an efficient alternative to SIFT or SURF. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 2564–2571 (2011). <https://doi.org/10.1109/ICCV.2011.6126544>
16. Schindhelm, C.K.: Evaluating SLAM approaches for microsoft kinect. In: ICWMC 2012: The Eighth International Conference on Wireless and Mobile Communications, Siemens AG - Corporate Research and Technologies, pp. 402–407 (2012). ISBN 978-1-61208-203-5. <https://www.thinkmind.org/download.php?articleid=icwmc.2012.17.40.20438>
17. Segal, A., Haehnel, D., Thrun, S.: Generalized-ICP. In: Proceedings of Robotics: Science and Systems (2009)
18. Taketomi, T., Uchiyama, H., Ikeda, S.: Visual SLAM algorithms: a survey from 2010 to 2016. IPSJ Trans. Comput. Vis. Appl. **9**, 16 (2017). <https://doi.org/10.1186/s41074-017-0027-2>
19. Valencia, R., Andrade-Cetto, J.: Path planning in belief space with pose SLAM. In: Mapping, Planning and Exploration with Pose SLAM. STAR, vol. 119, pp. 53–87. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-60603-3_4
20. Williams, B.P., Reid, I.D.: On combining visual SLAM and visual Odometry. In: Proceedings of International Conference on Robotics and Automation, ICRA 2010, pp. 3494–3500 (2010)