Intelligent Robotics

Assignment 2

By Joshua Foulkes

Contents

Introduction

The goal of this assignment is to review and compare two papers which investigate different approaches to a similar problem.

The papers chosen are *MonoSLAM: Real-Time Single Camera SLAM* [1] and *SLAM++ Simultaneous Localisation and Mapping at the Level of Objects* [2]. These papers investigate the use of a camera for vision-based SLAM. The aim of this report is to analyse the papers, to understand the problem these papers are focusing on and why is the research carried out important and necessary whilst evaluating the conclusions each paper came to. The contributions of each paper shall also be investigated. Additionally, this report compares both papers and see what possible improvements could be done.

➢ Task analysis

The task of both papers is to propose a method on how to do Simultaneous Localisation and Mapping (SLAM) using a visual sensor, in this case a camera. The use of a camera has many advantages being that “cameras are compact, accurate, non-invasive and well-understood—and today cheap and ubiquitous” [1]. A further reason for developing SLAM using vision systems is that “most advanced humanoids have vision systems” [1]. However, this approach comes with issues inherent to using a camera. Cameras are very rich in information but capture the world’s geometry only indirectly and it is difficult to turn sparse sets of features of an image into reliable long-term maps generated in real-time. This is also exacerbated due to the data-rates from cameras are higher than those from other sensors. The camera used in [2] is not a standard camera as the one used in [1]. It is a handheld-depth camera.

Analysis

➢ MonoSLAM

Aim

The first paper uses “standard PC and camera hardware” [1] to recover the 3D trajectory of a camera, moving rapidly though a previously unknown scene. Note: this paper would have used hardware from 2007 which has since greatly improved. The goal of this paper is highly focused on high frame-rate real-time performance (typically 30Hz), as well as “long-term repeatable localisation within restricted volumes” [1]. This is achieved by the algorithm used, being “optimised towards enabling localisation” [1]. The mapping part of the algorithm generates a sparse map of landmarks to be used. The landmark features are 11x11 pixel image patches.

Application

This algorithm has the aim to function in indoor rooms and to be used with a standard camera which “is carried by an unknown person, robot, or other moving body” [1]. A demonstration of the algorithm showed it being used for Augmented Reality (AR). For this demonstration, objects were added to a room virtually. To give off this illusion the objects had to move in the image as if they are anchored to the floor. This system could be adopted for people to see how furniture looks in their house before buying it. For this demonstration the camera was placed in a previously unknown room. This algorithm allowed for the objects to be placed and updated in real time at 30Hz. This is impressive as previous attempts to do this were offline or used fiducial targets. The demonstration also showed that the tracking persisted even through significant occlusion.

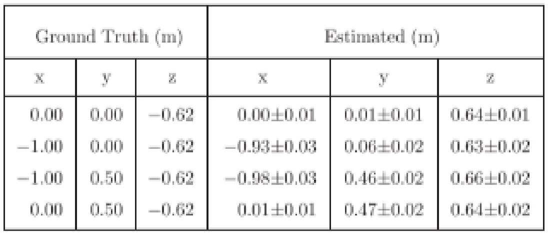
I believe this method of SLAM could be used when the fast localisation of the camera, with unpredictable motion using reasonably low computer specification, is more important compared to the creation of a detailed map of the environment. For example, in VR goggles, where speed is paramount.

Results

The result from this is a SLAM algorithm which can estimate the handheld camera’s motion in real-time from the live image stream.

This system doesn’t rely on odometry which could cause drift, which the paper [1] stated about some previous works, some of them used artificial floor markers and therefore this was said to be “not true SLAM”.

An additional experiment to the previous AR one was conducted to access the accuracy of the algorithm. The experiment involved a camera being moved around a rectangular track where it was facing a cluttered desktop table. A plumb line was added to obtain the ground truth. The ground truth result was then compared to the averaged estimated values from the *MonoSLAM* over several loops. The result is the table below:



Ground Truth (m)			Estimated (m)		
x	y	z	x	y	z
0.00	0.00	−0.62	0.00±0.01	0.01±0.01	0.64±0.01
−1.00	0.00	−0.62	−0.93±0.03	0.06±0.02	0.63±0.02
−1.00	0.50	−0.62	−0.98±0.03	0.46±0.02	0.66±0.02
0.00	0.50	−0.62	0.01±0.01	0.47±0.02	0.64±0.02

The table shows that this algorithm gives accurate result to a few centimetres. Note from the paper: the error for the second estimated x value displayed consistent large errors, which would persist through loops but would gradually reduce after each iteration.

The time required at each frame at 30Hz was also put into a table:

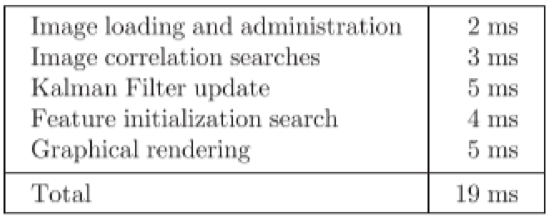


Image loading and administration	2 ms
Image correlation searches	3 ms
Kalman Filter update	5 ms
Feature initialization search	4 ms
Graphical rendering	5 ms
Total	19 ms

This indicates that performance at 30Hz is easily achieved.

Limitations

As previously stated, this algorithm is optimised towards localisation rather than mapping, therefore only creates "a sparse map of high-quality features".

Due to the way it is operated, the number of features is bounded to only 100 which means the system may not scale well to larger, more complex environments. To counter this the paper does state that 100 "well-chosen features" turn out to be sufficient with careful map management to span a room.

During the AR demonstration it was stated that the algorithm has some issues with rooms where no useful features could be detected within the field of view but could deal with rooms with only a few features. Good tracking recaptured when more features came back into view.

A very important assumption made is that the scene is rigid and that each landmark is a stationary world feature. The system needs high quality 3D models of repeatedly occurring objects in database before use.

➢ SLAM++

Aim

The second paper [2] presents "the major advantages of a new 'object oriented' 3D SLAM paradigm". This takes full advantage of prior knowledge present in some scenes such as "repeated, domain-specific object and structures". Due to the world having "intrinsic symmetry in the form of repetitive objects" the algorithm uses real-time 3D object recognition to produce maps directly at object orientated level. This means that the algorithm uses "prior knowledge of the object likely to be repetitively present" in an environment. These predictions are employed for camera tracking and the creation of a detailed maps.

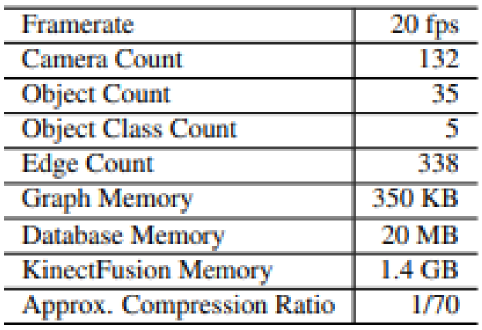
The camera used is a camera with Point Cloud capabilities.

Application

The major benefit of this algorithm is that it generates an object-level scene description, it also enables "vast compression of map storage compared to a reconstructive system with similar predictive power". This means it has great potential for "the use of domain-specific priors" such as office rooms like location as shown during the testing.

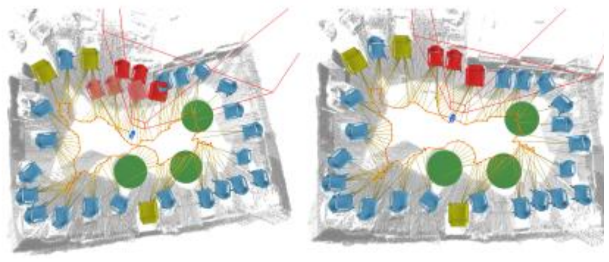
Results

The algorithm was run in a room with dimensions (10x6x3m) using a gaming laptop with unspecified specifications, but due to it being a gaming laptop it will have a reasonably powerful GPU. Memory footprint was compared to that of the KinectFusion [3].



Framerate	20 fps
Camera Count	132
Object Count	35
Object Class Count	5
Edge Count	338
Graph Memory	350 KB
Database Memory	20 MB
KinectFusion Memory	1.4 GB
Approx. Compression Ratio	1/70

The loop closing is also shown when tested within that room.



The left image shows open loop drift during exploration of a room. The right shows the "re-optimising the graph closes the loop and yields a more metric map".

Limitations

One of the key limitations is the processing power needed for object detection. In the paper it is stated that a gaming laptop is used which will have a powerful in-built GPU. This might limit its use as not all robots will have a GPU to run the algorithm.

One of assumptions made is that the objects are located on a common ground plane. This assumption to improve robustness. I believe this assumption would hold true for most situations where this algorithm is adapted for localisation. But this may affect some cases such as object on top of tables and could cause a mislabelling and effect the localisation.

Contributions

➢ MonoSLAM

The key contribution is to show that it is indeed possible to "achieve real-time localisation and mapping with a single freely moving camera as the only data source". Previous algorithms which attempted something similar such as Structure from Motion (which is an algorithm for generating 3D structures from 2D images are fundamentally offline in nature) [3]. From the paper it says it is able to do real-time localization and mapping at 30Hz, with a standard camera and a standard PC. Also, this is achieved where the motion model is unknown allowing for the camera to be operated by a human.

➢ SLAM++

Previous work, stated by the paper[2] focused on post-hoc labelling. This entails a depth camera scanning a scene and all the data being fused into a single large point cloud where the objects are them compared to off-line learnt models. This means that this previous work focused mainly on labelling rather than aiding mapping. The paper also states that several previous works had used object detection for SLAM, but never to the same scale or with the same details. These improvements are due to the algorithm using "real-time processing, full 3D operation, dense prediction and modern graph optimised back-end".

Comparison

Both methods are using a visual sensory input to perform SLAM. The distinction between each approach can be attributed to the different aim of each algorithm. For SLAM++[2] the aim was truer to the idea behind SLAM of localising an agent whilst also creating a map of the environment. This was taken further by having the map incredibly detailed, including objects that are present, but at the cost of requiring high computer specification. It also requires an object database to be made before operation. This algorithm doesn't not use a standard RGB camera but rather one with Point Cloud capabilities.

The MonoSLAM [1] algorithm focused on the localisation with high fps. What was achieved was the localisation of a camera which could be performed with high speed and accuracy using a standard camera in a room with no prior knowledge. Unfortunately, the map created is rather sparse, but this means it can be carried out by computers with much lower specifications.

Another large difference is that the MonoSLAM algorithm assumes that the world is fixed, therefore I do believe that this system has promise to be integrated into VR headsets where the user is in a small un-changing room. The comparison to the SLAM++ algorithm is that it can deal with objects being moved by stopping their use in the localisation to avoid "corrupting the rest of the graph" [2].

Possible Improvements

➢ MonoSLAM

One of the issues with this proposed algorithm is that it will not handle well when there are no features in the camera's view. A remedy to this could be the addition of a gyro and accelerometer which could give the acceleration and tilt of the camera, as a backup. The integration would be further facilitated due to the motion model used having acceleration and angular acceleration components. A gyro is added to the system when it is tested on the HRP-2 humanoid robot which is equipped with a 3-axis gyro. The gyro measurements are incorporated directly into the EKF.

An improvement that is proposed by the paper would be the use of a camera with a higher frame rate. The one used currently is 30Hz. The outcome of using a 60Hz camera is discussed. The doubling of framerate would "not imply a doubling of image processing effort" because "search regions would become correspond[ingly] smaller due to reduced motion uncertainty.", therefore would mean greatly improved localisation.

➢ SLAM++

The paper states that the approach is best suited for environments with many repeated, identical items. Therefore, one of the improvements stated by the paper would be the possibility for the algorithm to describe new object classes on the fly as they are "automatically segmented". So, a possible improvement to solve this is to take advantage of objects with "low dimensional shape variability".

Conclusion

In conclusion, both papers demonstrate a method for SLAM using vision technology. The first method of MonoSLAM focused mainly on localisation rather than mapping but was very efficient and used a standard camera and computer. Whilst the other method of SLAM++ used point cloud technology to not only localise the camera but also to create a very detailed map of the environment and the objects present within it, but at the cost of requiring higher processing requirements.

I would combine both algorithms running in parallel. This would allow for fast localisation using the MonoSLAM [1] algorithm which would run at much higher frame rates, whilst having the SLAM++[2] algorithm running much slower, mapping the environment more accurately as well as making the localisation more precise.

References

Davidson, A. J., Reid, I. D., Molton, N. D. & Stasse, O., 2007. *MonoSlam: Real-Time Single Camera SLAM,* s.l.: s.n.

Salas-Moreno, R. F. et al., 2013. *SLAM++: Simultaneous Localisation and Mapping at the Level of Objects,* s.l.: s.n.

Anon., n.d. *Structure from motion.* [Online]
Available at: https://en.wikipedia.org/wiki/Structure_from_motion