OpenFABMAP: An Open Source Toolbox for Appearance-based Loop Closure Detection

Arren Glover, *Student Member, IEEE*, William Maddern, *Student Member, IEEE*, Michael Warren, Stephanie Reid, Michael Milford, *Member, IEEE*, and Gordon Wyeth, *Member, IEEE*

***Abstract*— Appearance-based loop closure techniques, which leverage the high information content of visual images and can be used independently of pose, are now widely used in robotic applications. The current state-of-the-art in the field is Fast Appearance-Based Mapping (FAB-MAP) having been demonstrated in several seminal robotic mapping experiments. In this paper, we describe OpenFABMAP, a fully open source implementation of the original FAB-MAP algorithm. Beyond the benefits of full user access to the source code, OpenFABMAP provides a number of configurable options including rapid codebook training and interest point feature tuning. We demonstrate the performance of OpenFABMAP on a number of published datasets and demonstrate the advantages of quick algorithm customisation. We present results from OpenFABMAP's application in a highly varied range of robotics research scenarios.**

I. Introduction

Appearance-based localization techniques have advanced rapidly over the past decade. Approaches such as Fast Appearance-Based Mapping (FAB-MAP) have been demonstrated mapping large, challenging environments. Notable experimental successes include the mapping of a 1000 km journey through England [1] and a 142 km journey using stereo data and bundle adjustment [2]. However, with a few exceptions [3] most of the notable results using FAB-MAP have been achieved "in-house" by researchers associated with the original system.

To facilitate use by other research groups, the creators released a set of binaries in 2009 with two visual vocabularies, one for outdoor urban environments and one for indoor environments. However, FAB-MAP has not been fully "open sourced" as of this writing. Working only with binaries and the provided visual vocabularies imposes significant restrictions on the usability across a wide range of applications such as those detailed in this paper.

In this paper we describe OpenFABMAP, a new fully open source implementation of the FAB-MAP algorithm. The known implementation differences between OpenFABMAP and the FAB-MAP binaries are discussed, as well as the novelties of OpenFABMAP: the option to self-generate the training structures; providing multiple detection options; allowing location revisiting; and performing the likelihood calculation using a look-up table. We present results from OpenFABMAP's application to a number of varied research projects including ground-based mapping [3-5], aerial mapping and temporal (rather than spatial) mapping.

The paper proceeds as follows. In Section II we provide a brief general background on FAB-MAP and the advantages of open source software, and then describe the FAB-MAP system in more detail. The new open source system is described in Section III, with details on the additional options it provides users, such as generation of their own codebooks and tuning of interest point detectors to suit their application. Section IV presents the application of OpenFABMAP in a wide variety of new applications illustrating the potential use of an open source appearance-matching tool. The paper concludes in Section V with discussion and an outline of future work.

II. Background

A brief background on the evolution and use of FAB-MAP is given before a more detailed explanation of the algorithm is presented.

A. Appearance-based Loop Closure

A common problem in metric SLAM is the detection of loop closure events [6]. Recently more attention has been paid to using appearance-based approaches to inform potential loop closures. As the appearance of a scene is independent of the pose and error estimate, these methods can potentially provide a solution which allows loop closures to be quickly and robustly informed.

The current state-of-the-art in appearance-based loop closure detection is FAB-MAP, which is inspired by bag-of-words image recognition systems [7] and employs a strong Bayesian framework. Most recently the state-of-the-art in the field, FAB-MAP, has demonstrated loop closures in very large trajectories (over 1000km) with recall rates sufficient for coherent maps at high precision [1]. Various improvements and advancements on the original algorithm have been made to get to this point; FAB-MAP was introduced in smaller urban areas using mobile robots [6]. Algorithm speed-ups were introduced to perform mapping in constant time [8] and in larger datasets [1]. Geometry was introduced in the bag-of-words model to increase precision amongst perceptually similar but structurally different scenes [9]. Multiple sensor modes and pose estimations were

Manuscript received September 16, 2011. This work was supported in part by the Australian Research Council under a Discovery Project Grant DP0987078 and a Special Research Initiative on Thinking Systems TS0669699.

A. Glover, W. Maddern, M. Warren, S. Reid, M Milford and G. Wyeth are with the School of Electrical Eng. and Computer Science, Queensland University of Technology, Brisbane, Australia. (aj.glover@qut.edu.au).

then used to form more complete SLAM systems [10],[11].

FAB-MAP has shown state-of-the-art performance; however since its first publication in 2007, the most exemplary results have been mainly achieved by its creators. The authors have made FAB-MAP publically available in a binary or executable format which can be installed and run with ease on Windows and Linux systems. While some comparisons have been made to FAB-MAP using these binaries [12-14] as they cannot be recompiled from source they are inflexible for use in less than typical situations.

In recent times open sources projects have become more successful. Projects in robotics and computer vision such as OpenCV [15], the Robot Operating System (ROS) [16] and OpenSURF [17] are widely used throughout the robotics community. Open-sourcing software encourages use, through openly accessible code, and rapid development, as many parties are able to contribute to the project. The open source implementation introduced in this paper was first used in [3] and was consequentially published on-line under GNU-GPL license in June 2011. As at the time of this publication, it is the only fully functional open source version of the algorithm available.

B. The FAB-MAP Algorithm

Here we present the main algorithms that drive FAB-MAP. A more in depth description can be found in [6] and [8]. OpenFABMAP has been designed by implementing the content of the above publications and thus the core algorithms of both implementations use the same science.

1) Bag-of-Words

FAB-MAP matches the appearance between the current visual scene and a past location by representing the image with bag-of-words techniques [7]. An appearance-based feature detector is used to find interesting points in the image and represent the appearance of the local region using a 64 or 128 dimensional vector. The detection location is discarded and each feature is matched to a list of *a-priori* generated words dubbed a 'codebook'. The codebook is generated by clustering a large amount of features, extracted from a training dataset, to form a finite list (commonly thousands) of 'general' appearances often encountered in the environment. The observation, Z, of the image at time k is then reduced to a binary vector, indicating the codebook words which are present in the image.

$$Z_k = \{z_1,..., z_{|v|}\} \tag{1}$$

where the individual observation of codebook word i of codebook size $|v|$ is z_i.

2) Location Representation

Locations in space are represented as they are encountered by the probability the physical object e_i in the environment that produced the visual feature z_i is present at location L_k for each visual feature in the codebook.

$$\{p(e_1 = 1 \mid L_k),..., p(e_{|v|} = 1 \mid L_k)\} \tag{2}$$

where each individual observation probability is estimated by modelling the reliability of feature detection, $p(z_i|e_i)$, and the prior knowledge of how common a given feature is in the environment, $p(e_i)$:

$$p(e_i = 1 \mid L_k) = \frac{p(z_i \mid e_i = 1)p(e_i = 1)}{\sum_{s_e \in \{0,1\}} p(z_i \mid e_i = s_e)p(e_i = s_e)} \tag{3}$$

3) Data Association

The probability of matching to location L_i given all observations $\mathcal{Z}^k$, which includes the current observation Z_k, is calculated using recursive Bayes:

$$p(L_i \mid \mathcal{Z}^k) = \frac{p(Z_k \mid L_i, \mathcal{Z}^{k-1})p(L_i \mid \mathcal{Z}^{k-1})}{p(Z_k \mid \mathcal{Z}^{k-1})} \tag{4}$$

where the prior probability of a location $p(L_i|\mathcal{Z}^{k-1})$ is estimated using a naïve motion model which weights a neighbourhood of locations around previously recognized locations more highly. The probability is normalized over all observation likelihood calculations.

4) Estimation of the Full Probability Distribution

The accuracy of FAB-MAP is significantly improved over a naïve Bayesian approach by modelling the dependence of feature co-occurrence in the environment. For example, a single window frame will produce several different features and the high co-occurrence can be captured in a conditional probability. When matching observations to locations which have these features, the likelihood will be exponentially higher when all features are detected (the complete window), rather than only a partial set. The full distribution of feature co-occurrence is estimated using a Chow-Liu dependency tree which is calculated *a-priori*, off-line from training data. The Chow-Liu algorithm forms a minimum spanning tree which maximizes information entropy. The likelihood calculation is estimated as:

$$p(Z_k \mid L_i) \approx p(z_r \mid L_i) \prod_{q=1}^{|v|} p(z_q \mid z_{p_q}, L_i) \tag{5}$$

where the probability of observing z_q is dependent on the single parent observation z_p, and z_r is the root node of the Chow-Liu tree.

5) Location Generation

FAB-MAP generates a list of visited locations $\mathcal{Z}^{k-1}$ as new observations of the environment are made. The probability that the current observation comes from a previously unvisited location is also calculated, which promotes mapping rather than solely performing localization. The 'new location' probability is calculated as Equation 4 is normalized over all currently mapped locations as well as all possible unmapped locations. Since the latter cannot be directly evaluated, it is estimated by sampling a random selection of observations from training data:

$$\sum_{n \in M} p(Z_k \mid L_n)p(L_n \mid \mathcal{Z}^{k-1}) \approx p(L_{new} \mid \mathcal{Z}^{k-1}) \sum_{u=1}^{n_s} \frac{p(Z_k \mid L_u)}{n_s} \tag{6}$$

where L_{new} is the fixed prior probability of being at a new location, L_u is a sampled location and n_s is the total number of samples.

6) Fast Bail-out

The calculation time is $O(n)$ with the number of locations

that have been encountered. To reduce the calculation time when a large number of locations exist, the fast bail-out technique is employed. The likelihood calculation is performed for all locations in parallel, allowing unlikely locations to be discarded before the entire calculation is complete. The decision to discard a match occurs when the probability that the discarded location becoming a 'better' match than the current best location falls below a threshold, typically as small as 1×10^{-6}. The required margin between the two partial likelihood calculations is estimated using the Bennett Inequality which uses bounds on the maximum value and variance of the remaining calculations, see [8].

III. OpenFABMAP

The OpenFABMAP algorithm was based only on published FAB-MAP theory and inherently there will be implementation differences. This section reports some known added functionality available with OpenFABMAP, and the known differences which may affect performance. The original FAB-MAP algorithm is available in executable format (Linux, Mac and Windows) and codebooks are provided for both indoor and outdoor scenarios.

A. System Requirements

OpenFABMAP can be checked-out via SVN, or downloaded from the project home-page: http://code.google.com/p/openfabmap/. The core computer and vision functionality is driven using OpenCV [15] and other open source projects [17]. The implementation is written in C++ with a C-Make build environment compatible with Windows and Linux systems.

B. Additional Operation Modes

By being open source users can easily alter or improve the code to perform more specific functionality, or incorporate the algorithms into a larger system. The OpenFABMAP implementation also has additional operation modes not available in the binary version of FAB-MAP.

1) Codebook and Chow-Liu Tree Generation

The performance with which the algorithm will run is dependent on the training data used. If the training data does not represent the operation environment results are compromised. While the FAB-MAP binaries are locked to only 2 training data sets, OpenFABMAP comes equipped with the ability to generate Codebook and Chow-Liu trees to model any environment.

2) A Variety of Feature Detection Methods

The performance of a system is only as good as the input data is reliable. Different interest point detectors have different performance in different environments. OpenFABMAP allows users to choose and tune the detection algorithm, using the variety available in OpenCV, including SURF, STAR and MSER. In addition it is also possible to view the detection results to verify reasonable feature detection is occurring.

3) Location Revisiting

The available FAB-MAP binaries assume that the visual input is a collection of images, one for each physical location. Each sequential image is then added as a location in $\mathcal{Z}^{k-1}$ as it is encountered, even if a strong match was made to a previous location.

OpenFABMAP incorporates location revisiting: an image is only added to the list of locations if the maximal probability match was generated from an unmapped location, as per Equation 6. Alternatively a threshold can be set to add new frames at a fixed probability.

A significant advantage of revisiting is the ability to use high frame rate datasets. As sequential frames will look similar to the frames directly prior, a new location will not be added until a frame is encountered that looks significantly different. The sub-sampling that would otherwise be required to use such a dataset is automated within the system. It is more robust than a fixed sub-sample rate as the appearance itself directly informs the required sample rate.

When not using location revisiting (as in the original FAB-MAP binaries), the number of re-traversals of a given area is limited. When a new image is encountered the observation must be added as a new location, which will then be compared to all following observations. The more locations recorded that are observations of the same location, the harder it is to make a strong match, as many high likelihoods of matching result in lower probabilities after they are normalised.

4) Fast Look-up Likelihood

Given the representation of a location does not change after it is made (which is true even when using location revisiting), the portion of the likelihood calculation (Equation 5) considering the representation of L_i is only dependent on the observation z_i that was made at that location (Equation 3). The entire likelihood calculation then only depends on the three values: z_q, z_{pq} and z_i. The calculation of the location likelihood can be pre-computed for each combination of these variables, for each word in the codebook, given the codebook and Chow-Liu tree are fixed. Using a look-up table for this calculation has shown to increase calculation speed by an order of magnitude.

While this approach cannot be used for calculations involving the mean field approximation, or any other custom representation not directly formed from a single bag-of-words observation, the majority of calculations can take advantage of this speed boost.

C. Known Implementation Differences

The fast bail-out method as described in [8] uses a combination of bisection and Newton–Raphson methods, computed at intervals of 10 features, to numerically solve the Bennett Inequality. In contrast, OpenFABMAP uses only the bisection method calculated at intervals of 1 feature. While the OpenFABMAP fast bail-out implementation increases computation speed over the full calculation, it may not be as efficient as the original FAB-MAP fast bail-out implementation.

The OpenFABMAP implementation employs only a mean

field approximation to estimate the probability of the current observation coming from an unvisited location. The mean field (as described in [6]) replaces Equation 6, using:

$$\sum_{n \in \bar{M}} p(Z_k \mid L_n) p(L_n \mid Z^{k-1}) \approx p(Z_k \mid L_{avg}) p(L_{new} \mid Z^{k-1}) \quad (7)$$

to calculate the probability of an unknown location, where L_{avg} is a single location which represents the 'average' appearance of all locations.

The latest release of FAB-MAP, FAB-MAP 2.0, which boosts speed and precision when processing very large datasets, is not, as of yet, included in the current version of the OpenFABMAP toolbox.

IV. Applying OpenFABMAP

A range of different appearance-based mapping and loop closure scenarios are presented. The datasets vary in terms of the environment and data capturing methods. The exact use of the algorithms varies with application. OpenFABMAP has been used to quickly train codebook and Chow-Liu trees on specific environment types, providing good loop closure results in each unique situation. Each scenario has used OpenFABMAP to provide a research outcome, but for a different application.

It is important to note that, with the exception of the New College dataset, a comparison between the performance of the original FAB-MAP algorithm and OpenFABMAP on each dataset could not be made in a robust and well measured manner. The FAB-MAP binary does not permit testing under the same conditions as existed in the other applications of OpenFABMAP.

A. New College Dataset

OpenFABMAP was used to map the New College 1 dataset as used in [18], previously mapped by the original FAB-MAP algorithm. The dataset consisted of panoramic images (Fig. 1) collected on a mobile robot with a GPS ground truth.

OpenFABMAP was used as visual input for continuous appearance-based trajectory SLAM (CAT-SLAM) [4],[19] which demonstrated improved results over appearance-only SLAM by incorporating robust particle-based pose estimation. In this application the OpenFABMAP was altered to remove the fast bail-out approximation as a high accuracy of match probability was required over the full location distribution.

Fig. 1. An example image taken from the New College 1 dataset. The high resolution panoramic images provide high amounts of information.

Fig. 2 indicates the positive loop closures of OpenFABMAP at 100% precision. SLAM systems require 100% precision as any false positives result in catastrophic map failure. As perhaps one of the only metrics available to compare implementations, the reported recall at 100% precision on this dataset is used. Both OpenFABMAP [19] and FAB-MAP [18] report similar recall rates of slightly less than 10%. However, as analysis techniques may differ, this comparison may not be valid.

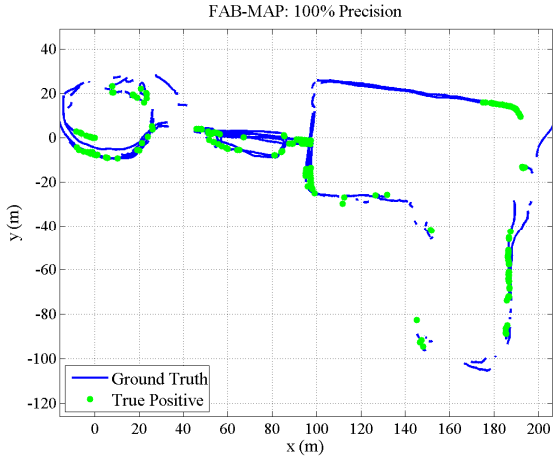


Fig. 2. The loop closures detected for OpenFABMAP on the New College 1 dataset at 100% precision [19]. The visual comparison of this image to [18] shows similar results.

B. St. Lucia Suburbs Dataset

The St. Lucia suburbs dataset was gathered using a low resolution, forward facing web-cam attached to a car which was driven through a series of streets to gather a total of 10 datasets at different times of day over the course of three weeks. GPS was used to evaluate mapping performance.

(a) (b) (c)

Fig. 1. A selection of scenes from the St. Lucia dataset. highlighting the particular challenge of mapping between different times of day. The same location at (a) 8:45am (b) 3:45pm and (c) 8:45am approximately 3 weeks later [3].

The data was somewhat different to that of the Oxford dataset as the appearance of a scene could change drastically over the course of a day as lighting conditions changed. The typical information provided by SURF extraction also changed due to the reduced field of view and image resolution.

In [3], OpenFABMAP was used to generate a probabilistic location likelihood which formed the front end to the biologically inspired pose filtering of the RAT-SLAM system. The amalgamation of the two algorithms provided a means to create coherent maps linking all 10 datasets (see Fig. 3), where the both algorithms failed when applied individually. While the underlying problem was similar to the Oxford dataset test, the vastly different environment and lower resolution images (generating different SURF descriptors) required new training data to effectively model the environment. At best, OpenFABMAP reported 16%

recall at 100% precision between 2 of the 10 datasets; values on par with typical FAB-MAP results.

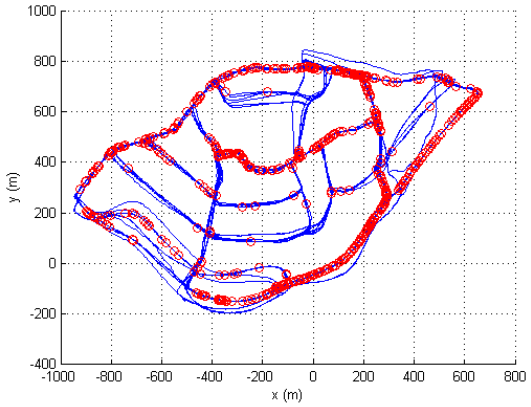
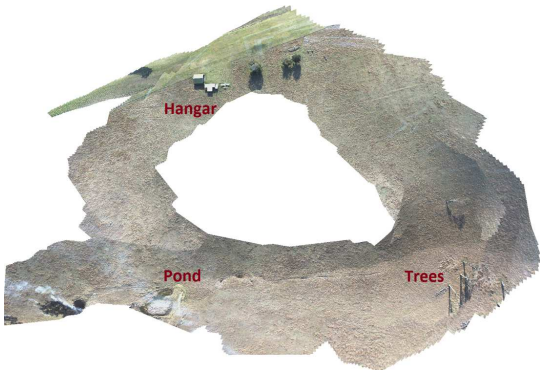
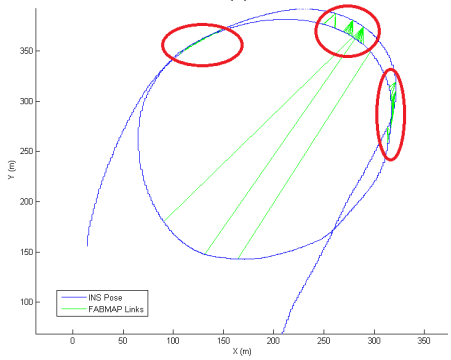


Fig. 3. The OpenFABMAP loop closures (red) when matching back to a map (blue) created using FAB-MAP/RatSLAM amalgamation on data taken three weeks earlier [3].

C. Aerial Mapping



(a)



(b)

Fig. 4. (a) The final structural map of the aerial dataset with reprojected texture and (b) the OpenFABMAP links greater than 0.99 plotted on the 6DOF UAV odometry ground truth. Despite constant appearance similarity OpenFABMAP rejected most of the incorrect matches. The few false positives were discarded in a further check for structural similarity.

OpenFABMAP was used within a vision-only geometric SLAM system for airborne vehicles. The dataset consisted of high resolution images of the ground plane taken from a low flying UAV following a path of over 1.7km. The complete system developed simultaneous estimates of pose and the reconstructed 3D scene using Structure from Motion and bundle-adjustment techniques. OpenFABMAP was used to select hypothesis 'loop-closure' locations in appearance space, before performing a 3-D geometry check. Successful loop closures were used to inform a pose-graph optimisation of the entire scene.

The dataset again provided unique challenges due to the environment. Typically, while flying over grassy fields, the environment consisted of many self-similar features with little variation in the scenery. Salient unique features existed only rarely in the dataset, as seen in Fig. 4. FAB-MAP is designed to handle these conditions given the appropriate training data. The performance of the algorithm was therefore highly dependent on the ability of OpenFABMAP to customize on new training data. It was also found that the use of a different feature detector (OpenCV's STAR detector) further increased loop-closure performance on this dataset.

D. Loop Closure in Time

OpenFABMAP has been used for appearance matching in new research addressing the problem of how robots can develop an understanding of time in order to inform spatial navigation. OpenFABMAP is used to create 'locations' that refer to a *time* rather than a place. The initial studies have investigated whether appearance can then be used to localise back to a previous time of day, as in Fig. 5.

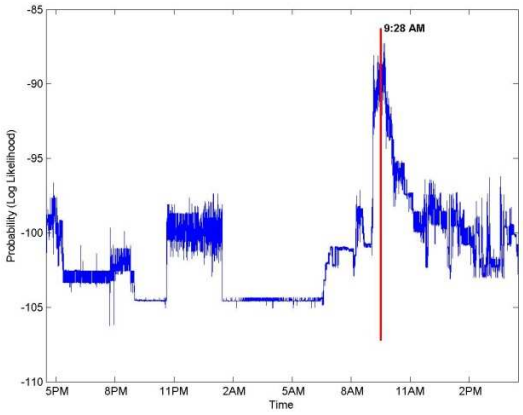


Fig. 5. The Log Likelihood PDF showing a strong match for the current frame (taken at around 9:30) to frames in the 24hr training dataset. Both the 'new location' and normalisation steps of FAB-MAP were not needed for this investigation.

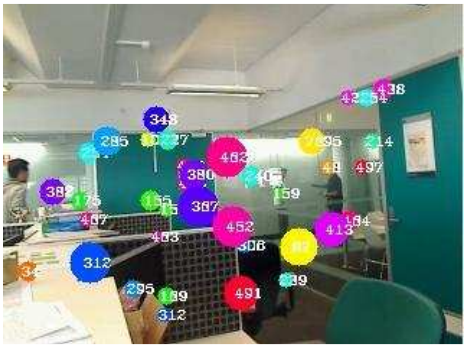


Fig. 6. The static camera field of view for the temporal loop closure experiments using OpenFABMAP.

OpenFABMAP provided a rapid and easily customizable tool with which to conduct these experiments. Particularly

important was the ability to train a codebook that suited the nature of the experiments – in this case the training datasets were taken entirely from one camera location (see Fig. 6), the very opposite of the normal training process for FAB-MAP.

E. Facilitating Comparison

OpenFABMAP has also been used to facilitate rapid comparison of novel localisation techniques to the current state-of-the-art of the FAB-MAP algorithm. In ongoing work by Milford et al [5], the focus is on mapping of relatively low quality camera footage from environments under significantly varying environmental conditions caused by day-night and seasonal cycles (Fig. 7). OpenFABMAP was used to rapidly train codebooks on data that was more representative of the environmental changes, and could then be used to validly compare performance with other mapping algorithms.

Fig. 7. Datasets with changes caused by seasons [5].

V. DISCUSSION

We described a wide variety of applications in which OpenFABMAP has been used, despite its relatively recent release. The ease of use and simple customisability of the complex algorithms makes this a useful tool for research in many applications which require robust visual image matching.

The utility of customisable calculation processes and output can be particularly seen in the temporal matching study. The appearance domain was interestingly different than typical robotics scenarios due to the gathering of the dataset on a stationary camera. As the appearance had low variance, a novel environment model was required. Also a 'new location' measure was not required as only localisation within the previous day was performed. The functionality could easily be achieved by compiling an executable in which the base function calls to the matching algorithms were customised as required.

The ability to incorporate FAB-MAP style matching in a larger system is easily done using OpenFABMAP as demonstrated in both the ground mapping scenarios and the aerial mapping. The software can also be used as a stand-alone tool for quick comparison with the state-of-the-art, in a SLAM domain as presented, or in many other situations.

Incomplete knowledge of both implementation and analysis methods, combined with the limited running modes of the original FAB-MAP binaries, made a direct performance comparison to OpenFABMAP invalid. However, the high-quality results obtained in the wide range of applications validates the toolbox as a very useful resource.

By presenting this paper to the robotics community we hope to encourage the use of FAB-MAP in many new and novel robotics applications. In addition, we hope to add to the growing amount of open source code available to further support state-of-the art research in robotics and other fields.

REFERENCES

[1] M. Cummins and P. Newman, "Appearance-only SLAM at large scale with FAB-MAP 2.0", *The International Journal of Robotics Research*, 2010

[2] G. Sibley, C. Mei, I. Reid and P. Newman, "Vast-scale outdoor navigation using adaptive relative bundle adjustment", *The International Journal of Robotics Research*, 29(8):958, 2010

[3] A. Glover, W. Maddern, M. Milford and G. Wyeth, "FAB-MAP+ RatSLAM: Appearance-based SLAM for Multiple Times of Day", *The International Conference on Robotics and Automation*, pages 3507-3512, Anchorage, Alaska, USA, 2010

[4] W. Maddern, M. Milford and G. Wyeth, "Continuous appearance-based trajectory SLAM", In *The International Conference on Robotics and Automation*, Shanghai, China, 2011

[5] M. Milford and G. Wyeth, "Visual Route-Based Navigation for Sunny Summer Days and Stormy Winter Nights", In *The International Conference on Robotics and Automation*, St. Paul, Minnesota, USA, 2012

[6] Mark Cummins and Paul Newman, "FAB-MAP: Probabilistic Localization and Mapping in the Space of Appearance", *International Journal of Robotics Research*, 27(6):647-665, 2008

[7] J. Sivic and A. Zisserman, "Video Google: a text retrieval approach to object matching in videos", *Proceedings of the Ninth IEEE International Conference on Computer Vision, 2003*, pages 1470-1477 vol.1472, 2003

[8] M. Cummins and P. Newman, "Accelerating FAB-MAP with concentration inequalities", *IEEE Transactions on Robotics*, 26(6):1042-1050, 2010

[9] R. Paul and P. Newman, "FAB-MAP 3D: Topological mapping with spatial and visual appearance", In *The International Conference on Robotics and Automation*, Anchorage, Alaska, USA, 2010

[10] I. Posner, M. Cummins and P. Newman, "A generative framework for fast urban labeling using spatial and temporal context", *Autonomous Robots*, 26(2):153-170, 2009

[11] C. Mei, G. Sibley, M. Cummins, P. Newman and I. Reid, "RSLAM: A system for large-scale mapping in constant-time using stereo", *International Journal of Computer Vision*, 2010

[12] A. Kawewong, N. Tongprasit, S. Tangruamsub and O. Hasegawa, "Online and Incremental Appearance-based SLAM in Highly Dynamic Environments", *The International Journal of Robotics Research*, 2010

[13] C. Cadena and J. Neira, "A Learning Algorithm for Place Recognition", In *The ICRA Workshop on Long-term Autonomy*, Shanghai, China, 2011

[14] B. Williams, M. Cummins, J. Neira, P. Newman, I. Reid and J. Tardós, "A comparison of loop closing techniques in monocular SLAM", *Robotics and Autonomous Systems*, 57(12):1188-1197, 2009

[15] G. Bradski and A. Kaehler, *Learning OpenCV: Computer vision with the OpenCV library*. O'Reilly Media, 2008

[16] M. Quigley, *et al.*, "ROS: an open-source Robot Operating System", In *The ICRA Workshop on Open Source Software in Roboticss* Kobe, Japan, 2009

[17] C. Evans, "Notes on the opensurf library", *University of Bristol, Tech. Rep. CSTR-09-001, January*, 2009

[18] P. Newman, *et al.*, "Navigating, recognizing and describing urban spaces with vision and lasers", *The International Journal of Robotics Research*, 28(11-12):1406, 2009

[19] W. Maddern, M. Milford and G. Wyeth, "CAT-SLAM: Probabilistic Localisation and Mapping using a Continuous Appearance-based Trajectory", *The International Journal of Robotics Research*, (under review)