ORB-SLAM: Tracking and Mapping Recognizable Features

Conference Paper · July 2014

CITATIONS
28

READS
6,409

2 authors:

Raul Mur-Artal
University of Zaragoza
7 PUBLICATIONS **2,568** CITATIONS
SEE PROFILE

Juan Domingo Tardos
University of Zaragoza
79 PUBLICATIONS **8,694** CITATIONS
SEE PROFILE

ORB-SLAM: Tracking and Mapping Recognizable Features

Raúl Mur-Artal and Juan D. Tardós
Instituto de Investigación en Ingeniería de Aragón
Universidad de Zaragoza, Spain
Email: {raulmur, tardos}@unizar.es

***Abstract*—Current visual SLAM systems build maps that are very good to track a camera during a session. However these maps are not designed to perform localisation in other sessions, with a different camera, or even the same camera, but observing the map from different viewpoints. This limitation comes from the lack of the map capacity to be recognized. In this paper we present a new keyframe-based SLAM system, which boosts the reusability of the maps, by directly mapping features that can be used for recognition. Our system performs relocalisation and loop closing with a high invariance to viewpoint in real time. Large scale is addressed building a covisibility graph that allows to perform most mapping and tracking operations locally. We also incorporate a pruning mechanism to reduce the redundancy of keyframes in the map. We believe that the capacity of our system to build reusable maps makes it a good choice for the community to investigate in long-term and recognition problems.**

I. Introduction

The problem of estimating the pose of a camera, and at the same time, a geometrical reconstruction of the world it is observing, is commonly known as monocular SLAM. The term *monocular* remarks the difference to the same problem using stereo or RGB-D cameras. Using just one camera is a more complex problem because the depth of observed features in the image is unknown and multi view geometry is necessary to solve the problem. Stereo and RGB-D cameras have a range in which they can recover the depth but in certain situations monocular techniques are still necessary.

Monocular SLAM has been developed for a long time, from the initial filtering approaches to the most modern keyframe-based SLAM systems. In a recent work, Strasdat et al. [10] concluded that keyframe-based techniques are more accurate per computational unit than filtering approaches. One of the most representative keyframe-based systems is Parallel Tracking and Mapping, PTAM [2], still considered by many as the gold standard in monocular SLAM. Nevertheless, despite the maturity of the problem, and the excellent performance of PTAM if we compare it to earlier solutions, monocular SLAM cannot still be considered solved. Maps are used today merely to compute the pose of the camera during a certain operation, but they are useless to localise a different camera or to relocalise the same from different viewpoints. The problem gets worse if the scenario is not static. To address this limitation we consider that the capacity of the maps to be recognized is crucial, and we find a big lack in these sense in PTAM-like systems.

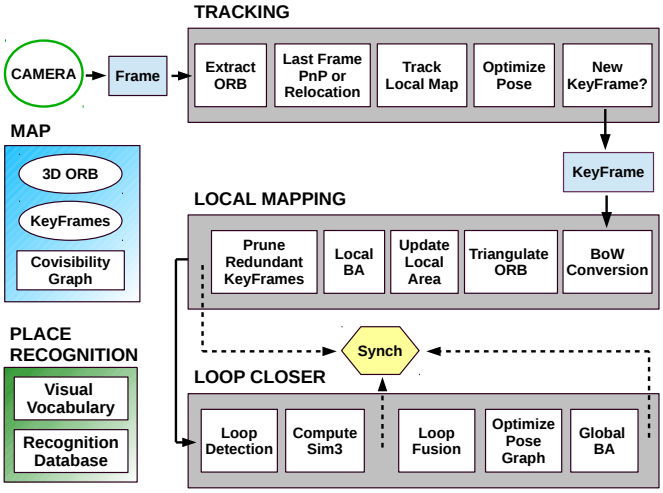


Fig. 1. System Overview

In our previous work [5] we integrated a very efficient place recognition system in PTAM to perform relocalisation and loop closing. That place recognition was based on bags of binary words [1] with ORB features [6], yielding a real time relocalisation system with a high invariance to viewpoint. We demonstrated experimentally the capacity of the system to reuse maps with different cameras and trajectories. It has to be noted that we needed to extract specific features (ORB) for recognition to get this capacity. In this work we want to go a step further and build a completely new SLAM system from scratch that uses the same features for tracking, mapping and recognition. We plan to release an open source implementation for ROS in a near future so that other researchers can use this SLAM system as base.

Similarly to Stradat et. al [9], our system makes use of covisibility information to operate at large scale. We also exploit the covisibility information between keyframes to boost the performance of our place recognizer, and we integrate a pruning algorithm to reduce the redundancy of keyframes in the map, so it is multiple-pass efficient. The system is composed of three main tasks: tracking, local mapping and loop closer, that run in parallel in a multi-core machine. Fig 1 shows a scheme of the different building blocks of the system, which we describe in the following sections.

II. The Map

In this section we present the information that our system stores from the world.

A. Map Points or 3D ORB

Map points form the structure of the 3D reconstruction of the world. Each map point corresponds to a textured planar patch in the world whose position has been triangulated from different views and refined by bundle adjustment. In this work, points correspond to ORB features in the images, so map points are triangulation of FAST corners. The patch size and orientation in the world depends on the scale level in which the FAST were detected and the orientation of the keypoints. Each map point has associated a patch normal $\mathbf{n}$, which is the mean vector of the viewing directions (the rays that join the point with the keyframe camera centers). A single map point may be associated with ORB features in several keyframes, and so there are several descriptors associated. In the absence of mismatches this variety of descriptors accounts for the different appearances of the map point depending on the viewpoint. To speed up data association each map point stores a representative descriptor D, whose distance is least with respect to all associated descriptors. In addition each map point stores the maximum d_{max} and minimum distance d_{min} at which it can be observed according to the scale and distance at which it has been observed.

B. Keyframes

Keyframes are snapshots that summarize the visual information of the real world. Each keyframe stores all the ORB features in the frame (associated or not to a map point), and the camera pose. In our framework, keyframes are not immutable, instead they can be discarded if an area is represented enough by other keyframes, as explained in section V-E.

C. Covisibility graph

Covisibility information is very useful in several tasks as determining local maps or to boost the performance of the place recognizer. This covisibility information is stored in the system as an undirected weighted graph, where each node is a keyframe and an edge between two keyframes exists if they see at least θ common map points. The lowest θ the most interconnected is the graph, and so more expensive to traverse. Strasdat et. al [9] proposed typical values for θ between 15 and 30. The weight of the edges is the number of common map points.

III. Place Recognition

The system integrates a place recognizer to detect loops and for relocalisation. This recognizer is based on our previous work [5], which was built on the very efficient bags of binary words place recognizer DBoW2 [1]. There are two main improvements in this work with respect to that previous work. First, the place recognizer is able to return multiple hypotheses in case there is not a clear match in the database. Second, we replace all temporal information by covisibility information. Next we describe the novelties with respect to our previous work

A. Recognition Database

The database contains an inverted index that stores for each visual word in the vocabulary in which keyframes it has been seen, and its weight in the keyframe. Therefore querying the database can be done very efficiently by checking only common visual words. New keyframes are inserted in the database by the loop closer task (see Fig. 1), after the local mapping has finished processing it.

When the relocalisation or the loop detector query the database, a similarity score is computed between all those keyframes that share visual words with the query image. Because there exists visual overlap between keyframes, there will not exist a unique keyframe with a high score, rather there will be several keyframes with high scores. In this work, as a novelty, we form covisibility groups which sum the scores of all those keyframes that are directly connected in the covisibility graph, being the keyframe match, the one with the highest individual score. In [5] and [1] images were grouped by closeness in time which would not sum the score from keyframes viewing the same place, but inserted at a different time. Instead of getting just the best match, we get all those matches whose scores are higher than the 90% of the best score.

B. Efficient and refined ORB matching

In relocalisation and loop detection, after querying the database, we will compute the camera pose and a similarity transformation respectively, which will also serve as geometrical validation. We need first to compute a set of ORB correspondences between two images. The brute force matching can be speeded up one order of magnitude, without reducing significantly the matching performance, if we compare only ORB that belong to the same node at the second level of the vocabulary tree [1]. This initial correspondences can be refined by an orientation consistency test [5].

C. Covisibility Consistency

Robustness in loop closure is essential as false loop closure can corrupt the map. Gálvez-López and Tardós [1] proposed a temporal consistency test, so that the system waits to obtain a number of consecutive temporally consistent matches to finally accept a loop closure. Here we propose instead a covisibility consistency test, so that two consecutive keyframe matches are considered consistent if there exist and edge in the covisibility graph linking them.

IV. Tracking

The map initialization is performed finding an homography (planar scenes) or an essential matrix from two near frames in an automatic process. Once an initial map exists, the tracking estimates the camera pose with every incoming frame. Next we describe each step of the tracking.

A. ORB extraction

The first step is to extract ORB features in the frame. At first a scale image pyramid is computed and FAST keypoints are detected at each level, then only a subset of these keypoints will be described. There are several parameters involved in this task as the number of levels in the pyramid, the scale factor between levels, the number of ORB to extract (and so keypoints to describe), the detector threshold and very importantly the rule to determine the subset of keypoints to be retained and described. These parameters have direct effect on the scale invariance of the features, the computational cost of other tasks (bag of words conversion, data association, etc) and the keypoint distribution over the image. We extract 1000 ORB at 8 scales with a scale factor of 1.2. To maximize the dispersion of keypoints in the image, we discretize the image in a grid and retain the same number of keypoints in each cell. If some cell has not enough keypoints the rest of cells are allowed to retain more keypoints.

B. Track from previous frame

If tracking was successful in the last frame, the tracking tries to get a first estimation of the camera pose from the last frame. Each ORB from the last frame, which has a map point associated, searches a match in the current frame in a small area around its previous position. In case of not finding enough correspondences, which can be caused by a fast movement of the camera, the matching process is repeated but with a bigger search area. The initial correspondences are refined by an orientation consistency test [5]. At this point we have a set of 3D to 2D correspondences and we can compute the camera pose solving a PnP problem [4] inside a RANSAC scheme.

C. Relocalisation: Track from scratch

If the tracking is lost, we convert the frame into bag of words and query the keyframe database for relocalisation candidates. There can be several candidates if there are several places with similar appearance to the current frame. For each candidate keyframe we compute ORB correspondences between the current frame and the ORB associated to map points in the keyframe, as explained in section III-B. At this point we have a set of 2D to 3D correspondences for each candidate keyframe. Now we perform alternatively RANSAC iterations with each candidate and try to compute a camera pose solving at each iteration a PnP problem. If we find a camera pose we can continue with the tracking process.

D. Track the Local Map

Now that we have an estimation of the camera pose in the world, we can project the map into the frame and search more map point correspondences. Instead of projecting all the map points in the map, as in PTAM [2], we follow a similar approach to Strasdat et. al [9] and project a local map. This local map contains the set of keyframes $\mathcal{K}_1$, that share map points with the current frame, and a set $\mathcal{K}_2$ with neighbors to the keyframes $\mathcal{K}_1$ in the covisibility graph. In very interconnected maps, the set $\mathcal{K}_2$ could be excessively big, so we limit the number of neighbors that we take from $\mathcal{K}_1$. The local map also has a reference keyframe $K_{\text{ref}} \in \mathcal{K}_1$ which shares most map points with current frame. Now all map points seen in the keyframes $\mathcal{K} = \mathcal{K}_1 \cup \mathcal{K}_2$ of the local map are searched in the frame as follows:

1) Compute the map point projection x in the current frame. Discard if it lays out of the image
2) Compute the angle between the viewing ray $\mathbf{v}$ and the map point normal $\mathbf{n}$. Discard if $\mathbf{v} \cdot \mathbf{n} < \cos(45^\circ)$.
3) Compute distance d from map point to camera center. Discard if it is out of the scale invariance region of the map point $d \notin [d_{\text{min}}, d_{\text{max}}]$.
4) Compute the scale in the frame by the ratio d/d_{min}.
5) Compare the representative descriptor D of the map point with the ORB features in the frame, at the predicted scale, and near x.
6) Associate the map point with the best match.

E. Final pose optimization

The final step in the tracking is to optimize the camera pose, using the initial estimation from IV-B or IV-C and all correspondences found between ORB features in the frame and local map points. The optimization minimizes the reprojection error maintaining fixed the map point positions. We use g2o [3] with the Levenberg-Marquadt algorithm robustified with the Huber cost function.

F. New keyframe decision

If tracking was succesful it is time to decide if we insert a new keyframe. To insert a new keyframe all the following conditions must be met:

1) More than 20 frames must have passed from the last relocalisation.
2) Current frame tracks at least 50 points.
3) Current frame tracks less than 90% map points than K_{ref}.
4) At least 10 frames have passed from last keyframe insertion and local mapping has finished processing the last keyframe. Or 20 frames have passed and a signal is sent to local mapping to finish local bundle adjustment.

It can be seen that we do not use any condition on the distance to other keyframes as in PTAM, here the criteria is more in the sense of visual change (condition 3), and trying to insert frames as soon as possible (condition 4). First condition ensures a good previous relocalisation and condition 2 a good tracking.

V. Local Mapping

The local mapping thread processes new keyframes, and updates and optimizes their local neighborhood. New keyframes inserted by the tracking are stored in a queue which ideally contains only the last inserted keyframe. When the local mapping takes the oldest keyframe in the queue it follows the next steps:

A. Bags of Words conversion

The first step is to transform the keyframe into its bag of words representation. This will return a bag of words vector which will be used later by the loop closer, but also will associate each ORB feature to the second node in the vocabulary tree which will be used for data association in next step.

B. Triangulate new map points

New map points are created by triangulating ORB in different keyframes. PTAM triangulates points only with the closest keyframe, which is the one with the lowest parallax, instead we triangulate points with the N neighbor keyframes in the covisibility graph that share most points. We need to bound the number of keyframes, because in highly interconnected maps, the computational cost would be prohibitive. For each free ORB in the current keyframe we search a match to other free feature in other keyframe just comparing descriptors. We only compare ORB that belong to the same node at the second level in the vocabulary and discard those matches that do not fulfill the epipolar constraint. Once a pair of ORB have been matched, they are triangulated. If the map point parallax is less than one degree it is discarded. Initially the map point is seen in two keyframes but it could exists in others, so the map point is projected in the rest of the keyframes predicting the scale as the tracking does, see section IV-D.

C. Update the local area

A map point is checked not to be wrong during the following three keyframes after creation and must fulfill the following two conditions not to be discarded:

1) The tracking must find the point in more than the 25% of the frames in which it is predicted to be found.
2) If more than one keyframe has passed from map point creation, it must be seen in at least three keyframes.

As there have been new measurements of points in keyframes, the edges in the covisibility graph of those keyframes must be recomputed. In addition we recompute the representative descriptor D for each point that have new measurements.

D. Local Bundle Adjustment

Similarly to PTAM [2], our local bundle adjustment optimizes the currently processed keyframe, all the keyframes connected to it in the covisibility graph, and all the map points seen by these keyframes. All other keyframes that see that points but are not connected to the currently processed keyframe are included in the optimization but remain fixed. To solve the non-linear optimization problem we use the Levenberg-Marquadt method implemented in g2o [3], and the Huber robust cost function. After the optimization we recompute the normal n, and scale invariance distances $d_{\min}$ and $d_{\max}$, for each map point.

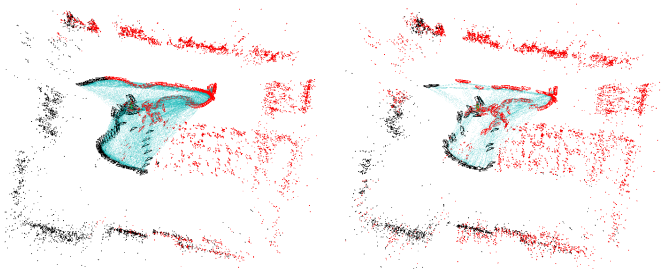


Fig. 2. Comparison of the reconstruction of a bedroom. Left: without pruning keyframes the map consist of 180 keyframes and 8258 points. Right: with the pruning procedure, the map contains 102 keyframes and 8269 points. The covisibility graph is shown in blue, and the local map in red.

E. Prune redundant keyframes

To reduce the redundant information of the keyframes in the map and do not allow the number of keyframes to grow unbounded, we discard all those keyframes whose 90% of the map points have been seen in at least other three keyframes in the same or finer scale. An example of this pruning procedure is shown in Fig. 2.

VI. Loop Closer

The loop closer thread is in charge of detecting loops in the map and if found it performs global optimizations that maintain the global consistency of the map. The loop closer takes the last processed keyframe K_i by the local mapping and perform several tasks, which we describe next.

A. Loop detection

At first we compute the similarity between the bag of words vector of K_i with all its neighbors in the covisibility graph and retain the lowest score $s_{\min}$. Then the loop closer queries the keyframe database and discard all those keyframes whose score is lower than $s_{\min}$. In addition all those keyframes directly connected to K_i are discarded from the results. To increase the robustness of the detector we must detect three consecutive loop candidates that are covisibility consistent, see section III-C. If one or more loop candidate keyframes have passed the covisibility consistency test, the next step is to compute a similarity transformation.

B. Compute the similarity transformation

In monocular SLAM there are seven degrees of freedom in which the map can drift, three translations, three rotations and a scale factor [8]. Therefore to close a loop we need to compute a similarity transformation from the current keyframe K_i to the loop keyframe K_l that informs us about the error accumulated in the loop. The computation of this similarity will serve also as geometrical validation of the loop.

As in the relocalisation procedure there can be several loop candidates if there are several places with similar appearance to the current keyframe. We first compute ORB correspondences between ORB associated to map points in the current keyframe and the loop keyframes, following the procedure explained in section III-B. At this point we have 3D to 3D

correspondences for each loop candidate. We alternatively perform RANSAC iterations with each candidate, trying to compute a similarity transformation. The first candidate for which we find a similarity $\mathbf{S}_{il}$ with enough inliers is accepted.

C. Synchronization with local mapping thread

Before correcting the loop, the loop closer sends a signal to the local mapping so that it stops when it finishes with its current operations. Once the local mapping has stopped the loop closer will proceed to correct the loop. This is necessary because both threads would modify keyframe poses and map points probably mixing results and corrupting the map.

D. Loop fusion

The first step in the loop correction is to fuse duplicated map points and insert new edges in the covisibility map that will attach the loop closure. At first the current keyframe pose T_{iw} is corrected with the similarity transformation $\mathbf{S}_{il}$ and this correction is propagated to all the neighbors of K_i, concatenating transformations, so that both sides of the loop get aligned. All map points seen by the loop keyframe and its neighbors are projected into K_i and its neighbors and search matches in a narrow area from the projection, as done in section IV-D. All those map points matched and those that were inliers in the computation of $\mathbf{S}_{il}$ are fused. All keyframes involved in the fusion will update its edges in the covisibility graph effectively creating edges that attach the loop closure.

E. Covisibility pose graph optimization

To close the loop and correct the error accumulated, we must perform a global optimization. However because the use of a robust cost function is necessary when performing bundle adjustment, to be robust against mismatches, we need first to compute an initial solution so that we facilitate the convergence of the optimization. This initial solution will consist in the result of a pose graph optimization over the covisibility graph, were the error in the loop closure edges will be distributed along the graph. The pose graph optimization is applied over similarity transformations to correct the scale drift [8]. This approach is more general than our previous work [5], were the pose graph was just a cycle formed linking each keyframe to the previous, but still the definition of the residuals and optimization procedure is the same and it is not repeated here.

F. Global Bundle Adjustment

The whole map is finally optimized using the seed of the pose graph optimization. The optimization is the same as for the local bundle adjustment of section V-D, but here all keyframes and map points are optimized. After the optimization the local mapping is released again.

VII. Experiments

We performed two experiments, which attempts to show different strengths of our system. The first is a hand-held camera sequence in a laboratory were there is a loop closure, strong occlusions and abrupt movements of the camera. The second is a section of the NewCollege sequence [7], which test our system in the large. Our system is integrated in ROS and sequences are processed simulating a camera at the same fps as they were recorded. The accompanying videos [1] show our system running in both sequences and PTAM in the lab sequence for comparison.

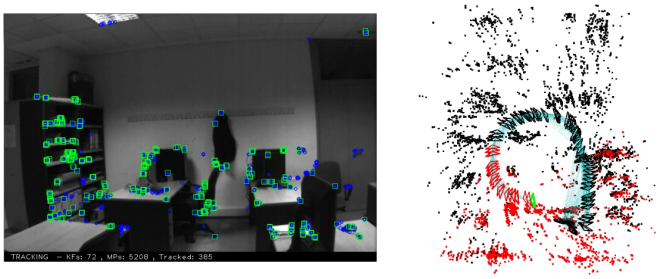


Fig. 3. Our system running in the lab sequence. Left: ORB extracted on the frame are shown in blue. Green squares are ORB matched with map points. Right: Map reconstruction (local map is shown in red and the current camera pose in green).

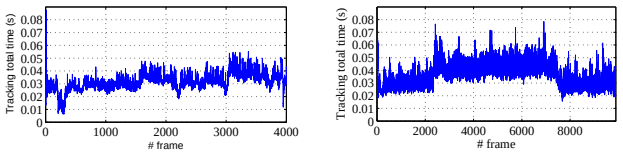
Fig. 4. Challenging relocalisations in the lab sequence. PnP inliers in blue.

A. Lab sequence

This sequence is composed of 752x480 resolution images at 30 fps. A view of our system running in the sequence is shown in Fig. 3. Some challenging relocalisation performed during the sequences are shown in 4. The total tracking time at each frame is shown in Fig. 5. It can be seen that the tracking spent less than 40ms in 90% of the frames. The tracking times increase slightly during the sequence, because the covisibility graph gets more interconnected and so the local map is larger. The loop detector spent 5ms to retrieve a loop candidate in the keyframe database that contains 86 keyframes, computing the similarity and fusing the loop took 173ms, and the pose graph optimization spent 187 ms to optimize a graph with 1692 edges. Finally the global bundle adjustment took 518ms.

For comparison we have also run PTAM in this sequence. First we must say that it took more than 20 trials to achieve a good initialization. PTAM lost the track in several parts of the sequence where our system does not, and relocalisation fails repeatedly because of its lack of viewpoint invariance.

[1] http://webdiis.unizar.es/~raulmur



(a) Lab (b) NewCollege

Fig. 5. Total tracking time per processed frame in each sequence

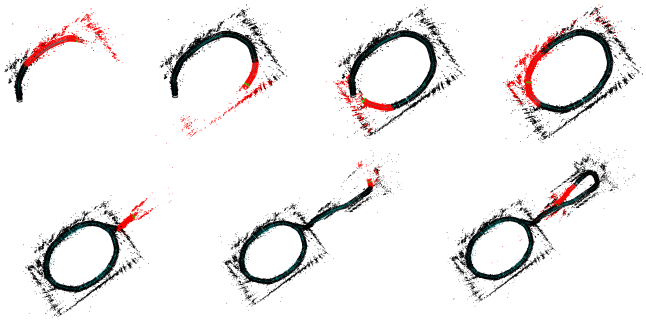


Fig. 6. NewCollege reconstruction at different times. A loop closure is detected between third and fourth images.

B. NewCollege

This is a stereo sequence from which we process only the left images, which have a resolution of 512x384 and were taken at 20 fps. The map reconstruction at different times is shown in Fig. 6, and the final reconstruction in Fig. 7, which contains 690 keyframes and 33,000 points. Fig. 5 shows the total tracking time spent per frame, where the 90% of the frames were processed in less than 50ms. In the central part of the sequence the robot is revisiting the big loop that can be seen in Fig. 6, where the local map is larger than in exploration areas, and therefore the tracking spends more time. The loop detector spent 7 ms to detect the loop in the keyframe database when it contained 280 keyframes. The loop fusion took 333ms and the pose graph optimization 2.07s to optimize a graph of 10831 edges. This makes clear that optimizing the whole covisibility graph becomes too expensive in large loops and so not all edges should be included in the optimization, such an strategy will be addressed in future work.

VIII. Conclusion

We have proposed a new visual SLAM system with the focus on building recognizable maps. Instead of performing SLAM over features which are only useful for tracking the current camera, we map features that can be used for place and object recognition. This effectively expands the reusability of the map. Compared to PTAM in small maps, our front-end is more expensive but still runs at frame-rate, with a vast improvement in camera relocation capabilities. The back-end is able to perform large scale mapping, including an efficient and reliable loop closing. We are working on the robustness of the map initialization, omitted here because lack of space, and the sparsification of the pose graph to be optimized by the loop closer. We plan to release the code so the community can benefit from our visual SLAM approach.

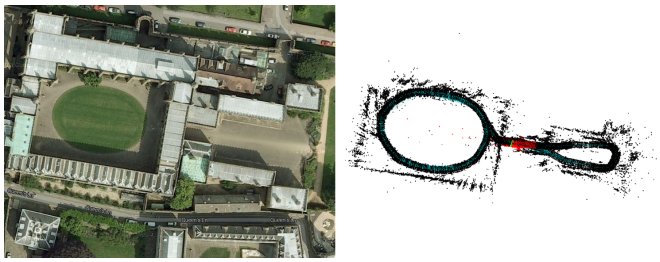


Fig. 7. Reconstruction of our system of NewCollege

Acknowledgments

This work was supported by Dirección General de Investigación of Spain under Project DPI2012-32168, and Gobierno de Aragón Scholarship B121/13.

References

[1] D. Gálvez-López and J. D. Tardós. Bags of binary words for fast place recognition in image sequences. *IEEE Transactions on Robotics*, 28(5):1188–1197, 2012.

[2] G. Klein and D. Murray. Parallel tracking and mapping for small AR workspaces. In *International Symposium on Mixed and Augmented Reality (ISMAR)*, 2007.

[3] R. Kuemmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard. g2o: A general framework for graph optimization. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2011.

[4] V. Lepetit, F. Moreno-Noguer, and P. Fua. EPnP: An accurate O(n) solution to the PnP problem. *International Journal of Computer Vision*, 81(2):155–166, 2009.

[5] R. Mur-Artal and J. D. Tardós. Fast relocalisation and loop closing in keyframe-based SLAM. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2014.

[6] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. ORB: an efficient alternative to SIFT or SURF. In *IEEE International Conference on Computer Vision (ICCV)*, 2011.

[7] M. Smith, I. Baldwin, W. Churchill, R. Paul, and P. Newman. The new college vision and laser data set. *The International Journal of Robotics Research*, 28(5):595–599, 2009.

[8] H. Strasdat, J. M. M. Montiel, and A. J. Davison. Scale drift-aware large scale monocular SLAM. In *Robotics: Science and Systems (RSS)*, 2010.

[9] H. Strasdat, A. J. Davison, J. M. M. Montiel, and K. Konolige. Double window optimisation for constant time visual SLAM. In *IEEE International Conference on Computer Vision (ICCV)*, 2011.

[10] H. Strasdat, J. M. M. Montiel, and A. J. Davison. Visual SLAM: Why filter? *Image and Vision Computing*, 30(2): 65–77, 2012.