Video-rate Localization in Multiple Maps for Wearable Augmented Reality

Robert Castle, Georg Klein, David W Murray
Department of Engineering Science, University of Oxford, Oxford OX1 3PJ, UK
[bob,gk,dwm]@robots.ox.ac.uk

Abstract

We show how a system for video-rate parallel camera tracking and 3D map-building can be readily extended to allow one or more cameras to work in several maps, separately or simultaneously. The ability to handle several thousand features per map at video-rate, and for the cameras to switch automatically between maps, allows spatially localized AR workcells to be constructed and used with very little intervention from the user of a wearable vision system. The user can explore an environment in a natural way, acquiring local maps in real-time. When revisiting those areas the camera will select the correct local map from store and continue tracking and structural acquisition, while the user views relevant AR constructs registered to that map.

1. Introduction

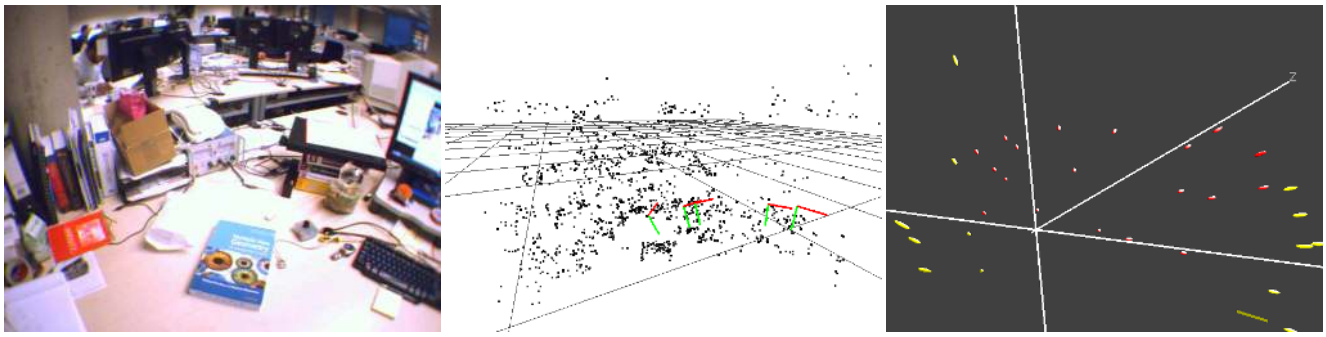
Many of the more advanced applications of wearable and hand-held cameras, particularly those involving augmented reality (AR), share characteristics with robot visual navigation. Principally, both require the acquisition and maintenance of an estimate of the camera's position and orientation relative to some geometric representation of its surroundings. The map of the surroundings may be pre-built, perhaps most simply using distinctive artificial fiducials or landmarks; or it may be a CAD model with complexity ranging from merely a single object, through the interior of a building, to a complete streetscape. Hand-crafting of maps has long been regarded as impracticable, and much effort in visual structure from motion (SfM) and robot simultaneous localization and mapping (SLAM) over the last few decades has been dedicated to acquiring structural models and maps automatically. While SfM has been rooted in the off-line optimal reconstructions of both minimalist and dense structure (*e.g*. [20, 6, 18]), SLAM's tradition is in recursive recovery of the scene on-line whilst respecting the correlations amongst and between camera and scene state entities (*e.g*. [22, 12, 23]).

There are however important differences between vision for wearables and for robot navigation, and we highlight three. The first is that a wearable camera is much more agile than a robot camera, which will cause image degradation through motion blur. The second is that odometry is also usually unavailable, and when combined with the first, it allows less confidence to be placed in priors. However good the model describing camera motion is, and for human motion they are usually far from good, it is important to accept the inevitability of the camera frequently becoming lost. Therefore, robust methods of recovering from such events must be incorporated into any system.

Rapid relocalization of a lost camera was demonstrated recently by Williams *et al*. [24] using a variant of Lepetit and Fua's feature description and matching using randomized trees [15]. It was demonstrated operating within the Extended Kalman Filter (EKF) monoSLAM algorithm of Davison *et al*. [3, 4], a method which has already proved a useful vehicle for AR using wearable vision [16, 2].

However, despite the added ability to relocalize, our experience is that the wearer or holder of the camera always feels constrained to move cautiously rather than naturally, a result of the EKF having to get every match correct at each frame, despite the shortcomings of the motion model. Furthermore, mitigating the loss of certainty in the motion model makes it all the more important to process everything at frame-rate. This shows up the EKF's relatively high computational cost per update, and unfavourable computational complexity which is quadratic in the number of features in the map. We find a usable maximum for 30 Hz operation is in the order of one hundred points, which means even the local environment around the user is sparsely populated with features.

In this paper we build on the work of Klein and Murray who demonstrated that by separating the task of maintaining the camera track at frame-rate from the less urgent task of optimally updating the 3D structure from measurements made in keyframes only, SfM methods could provide a viable alternative to conventional SLAM [9]. It was found that the method allowed for greater freedom of movement for the handler of such a camera and, perhaps more importantly, 30 Hz operation was maintained with several *thousand* point features (*e.g*. Figure 1). Here we show empir-



(a) (b) (c)

Figure 1. A desk-top scene (a), and a comparison of the 3D maps obtained after 20 seconds' acquisition at 30 Hz frame rate using (b) the parallel tracking and mapping method of [9] and (c) monoSLAM [4]. The far greater feature density in (b) is evident.

ically that such maps are sufficiently spatially sized, and sufficiently detailed, to represent the local visual environment around a wearable camera, and that they contain sufficient redundancy to cope with a degree of change over time. They allow the user to walk freely around an environment, acquiring local maps in real-time, and later, when revisiting those areas, to match to the local map in store and continue structural acquisition, while viewing relevant AR constructs registered to the map.

This paper points out too that decoupling tracking from mapping makes it quite straightforward to add further independent cameras which may add to a map.

From a broader contextual view, the results argue for the utility of *separate* local structural maps for wearable vision; a utility which originates in a third principal distinction between wearable and robot vision. It is that the human is considerably more intelligent than a robot camera platform, and can be relied upon to provide the spatial and navigational links between individual local maps at A and B, navigating between with minimal information and avoiding obstacles with a deftness that robots still cannot. Also, for a wearer, the areas of interest are *at* A and B, not on the route in between. Our concerns differ here from the approach in robotic SLAM of submapping (used as a method of taming complexity, *e.g.* [13, 7, 14]), in that we are not concerned with the detailed geometrical relationship between submaps (indeed, if at all in some applications). The very recent work of [11] has some similar aims to ours, though the approaches differ in detail.

Section 2 reviews the tracking and mapping algorithm which our work is based upon. Sections 3 and 4 explain the modifications required to allow multiple tracking cameras to work with multiple maps, and how we use them in practical applications. Section 5 describes the implementation of the system on a wearable vision platform built to support this work. Section 6 provides an experimental evaluation of the method. As these experiments are carried out on-line the accompanying video submission is an important adjunct to the section, and Section 7 lists our conclusions and provides more general discussion.

2. Tracking and mapping

We outline the tracking and mapping method of [9] before describing how it is used with multiple cameras and how maps can be matched and re-opened. The method is designed for scenes that are predominantly rigid and that can be effectively summarized using the order of 10^3 visual corner features. Thus office-sized rather than city-sized scenes are handled, a size which is highly compatible with many AR applications related to the workplace. The motivation for separating tracking from mapping comes from the observation that visual SLAM methods, both EKF- and FastSLAM-based, spend too great a time per frame referencing the 3D map and its covariance. The root of their difficulties is that they under exploit the epipolar geometry which allows camera motion to be found, without addressing the structure explicitly.

2.1. Tracking

Suppose that a 3D map has already been initialized. Tracking proceeds at each frame by first subsampling the new grey-level image into a four-level pyramid spanning three octaves, and computing corner features at each scale using the FAST-10 detector [21]. A prior pose for the camera is obtained using a model that assumes constant or decaying velocity in the presence or absence of measurements, respectively. Around 50 map points (which are stored with an appearance model from the keyframe in which they are first observed) are projected in the image at the coarsest scale, and those that are matched to the current image feature list are used to update the camera pose using robust minimization. Further iterations are made, pulling in more points as the scale moves from coarse to fine. Details of the matching and minimization objective function are in [9].

The fraction of features successfully matched is monitored as a measure of tracking quality. Two thresholds are

chosen. If the fraction falls below the higher, tracking continues as above, but the frames involved are prevented from becoming keyframes for the mapping process described below, as the image quality is probably poor. The fractions falling below the lower threshold indicates that tracking has failed terminally, and a method of relocalization is used to recover. The recovery method is also used in further contexts in the present work, and is discussed in Section 3.

2.2. Mapping

The mapping process runs separately from the tracking and uses measurements from *keyframes*, here defined as frames for which the camera pose is quite distinct from existing keyframes.

To initialize a map at the outset, a camera viewpoint is selected by the user and its image and feature list becomes the first keyframe. The camera is then moved[1] to a new position, with sufficient care to allow features to be tracked using the image alone. This image is captured as the second keyframe, and Faugeras and Lustman's method [5] is used to compute the new camera pose and to triangulate scene points. Often in our work those initial 3D features will lie on a desk-top or ground plane and, as a convenience, a dominant plane is found in the structure, and the map reoriented so that this defines the scene's $Z = 0$ surface. It is stressed that the plane plays no special rôle in the structure from motion calculations.

As the camera is moved further, additional images and 2D feature lists are designated as keyframes whenever (i) tracking is deemed good, (ii) the camera has translated by a minimum distance from *any* previous keyframe, and (iii) around 20 frames have passed since the previous keyframe. This last condition restricts the volume of processing.

Recall that the tracking process will already have detected corner features in the image pyramid and will have matched a subset of them to the projections of existing 3D map points. To add new points to the map, features are sought in the image which are unmatched, particularly salient, and which are distant in the image from any matched feature. To initialize the points in the map, a search is made in the spatially closest keyframe for image matches, and their 3D positions triangulated (the relative pose between the keyframes is known from the tracking process).

At this point, the putative 3D positions $\{\boldsymbol{p}_i\}, i = 1 \dots M$ of all map points and all keyframe camera poses except the first $\{\boldsymbol{\mu}_j\}$, $j = 2 \dots N$ are optimized in a bundle adjustment, using Levenberg-Marquardt (as described in Appendix 6.6 of [8]) to minimize a robust cost $C(\cdot)$ based on image errors $\boldsymbol{e}$

$$\{\hat{\boldsymbol{p}}\}, \{\hat{\boldsymbol{\mu}}\} = \arg\min_{\{\boldsymbol{p}\},\{\boldsymbol{\mu}\}} \sum\sum C(|\boldsymbol{e}_{ji}|/\sigma_{ji}, \sigma_T) \; .$$

[1]The motion must of course include translation, which is assumed to be some 100 mm in order to set a reasonable scale on the map.

Because this is a batch process and the data retained throughout, it is open to the map builder to delay performing the complete adjustment until time and processor loading permits. The load is monitored, and if keyframes are being added frequently, local bundle adjustments on a limited number of keyframes are performed. In recent work on visual odometry [17, 19], using *temporally* local bundle adjustments (a sliding window) has been found to be remarkably accurate even without a global adjustment. Here the locale is *spatial*, which will be partially correlated with time, but will include old keyframes if the camera revisits a location after a period of neglect. Here then, the "mortar" between the geometrical elements never "sets hard" as it does in the case of visual odometry. Details of the method, and an informative system diagram, are in [9].

Figure 1 contrasts the density of points in typical maps of the same scene obtained after the same time by frame-rate implementations of the tracking and mapping method used here and monoSLAM [4]. The increase in feature density, sometimes of two orders of magnitude, is apparent. Note too the dominant plane: this derives here from the desk-top, but the graphical representation extends without limit.

3. Incorporating multiple trackers and maps

Given sufficient processing hardware, it would be routine to replicate tracking and mapping processes so that multiple cameras build multiple maps in a one-to-one fashion. Much more interesting is that, because the tracking process (image capture, feature detection, and camera pose estimation) is largely independent of the map making process, it proves remarkably straightforward to allow multiple cameras to add information into a *single* map.

The question of when to insert keyframes is made particularly easy because the determination is made by *spatial* separation of poses, not temporal separation. Multiple trackers can offer frames to a single map making process, but it is the map maker that decides which to accept. Any additional (calibrated) views can be utilized at will, and multiple trackers can work on a map simultaneously or at different times. The only requirement is that subsequent trackers must register their camera poses to the map's established coordinate system. This registration is exactly that required to relocalize a lost camera during the normal run of single camera tracking, and indeed exactly that required to relocalize a camera in a map after a period of neglect. We describe the relocalizer below, but an important point here is that the registration is made using appearance not structure so that a camera can be localized without first building its own map.

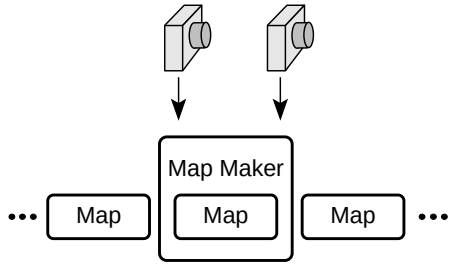


Figure 2. System configuration: two cameras feeding a map maker, which is used to build an array of maps.

3.1. Relocalization

A highly important component is the relocalizer, which becomes the key to automatically switching between maps. Thus answering not only "where is the camera in this map?", but also "in which map is the camera?". The original system [9] used the relocalizer developed earlier in the laboratory by Williams *et al*. [24]. This relocalizer was based on randomized ferns [15], and used them to describe the appearance of points in a monoSLAM map. This method, while fast and effective, is very memory intensive at about 1.3 MB per map point. While this is tolerable for a map with a couple of hundred features, it cannot be contemplated for several maps each holding several thousand features. However, unlike monoSLAM, in the tracking and mapping approach we have the benefit of keyframes. Klein and Murray [10] recently replaced the randomized fern based relocalizer with a fuzzy image one. This relocalizer exploits the relatively dense distribution of keyframes. Descriptors are made for each keyframe and, once relocalization has begun, also for the incoming camera images. The descriptors are created by subsampling the image eight-fold (in practice $640\times480 \Rightarrow 80\times60$), then applying a Gaussian blur with $\sigma = 5$. The keyframe descriptors are then compared to the current camera image descriptor to find the closest match, using zero mean sum of square differences:

$$D = \sum((I_{ki} - \mu_k) - (I_{ci} - \mu_c))^2 \ ,$$

where I_k and I_c are the intensity values of the i^{th} pixel for the keyframe and camera image descriptors respectively, and μ is the mean intensity value of each descriptor. Each calculation takes 0.016 ms on average. The keyframe that has the least difference with the camera image is accepted as a match, and the camera position is set to that of the keyframe. The rotation of the camera is estimated using a direct second order minimization [1] of the descriptors to minimize the sum of squared differences. Tracking is then resumed, and the system tries to track from that position. If the match is correct then the system continues tracking, otherwise tracking fails and the relocalizer is invoked again.

3.2. Map switching

Tracking is usually lost because the camera's viewing direction has changed suddenly. In a multiple map system it is uncertain whether the camera remains looking at an area belonging to the same map, or at one of another map, or indeed at an unmapped area. In the single map system, when a correlation score that was above a threshold was found, tracking was restarted from that putative pose and the structure and motion from the subsequent few images used to verify or reject the hypothesis. Rejection would lead to another search, but experimentation showed that the rejected fraction was low. With multiple maps (particularly those from similar scenes, as shown in the experiments) and no prior knowledge whether this is an intra- or inter-map failure, this approach can result in repeated failed attempts to restart in the wrong map. An expedient (though expensive) solution is to cross-correlate with *all* keyframes from *all* maps, before choosing that with the best score. This cost grows linearly with the map size.

An exception to this relocalization strategy is made when two or more trackers are working in the same map, as in our wearable system (see Section 5) where the user has a camera on the hand-held display, and a second active camera mounted on their shoulder. The lost tracker checks whether the other tracker (or trackers) is lost. If not, it attempts to recover on the current map. However, if all trackers are lost the complete search is performed. The first tracker to recover causes the others to abandon their complete searches in favour of attempting to recover on the recovered tracker's map.

4. Using multiple maps in practice

The configuration of the system is sketched in Figure 2. The maps are considered as a central resource, built by multiple cameras performing tracking and mapping. Those "builders" (users who can construct maps) can also use the maps for AR at the same time, as was the emphasis in the single tracker and map version in [9]. However we can also consider a set of users who use the maps solely for tracking or who have limited capacity to build the maps further.

In practical applications for AR outside the laboratory we adopt the following procedure. A builder creates multiple maps of the environment at predominantly non-overlapping sites of interest. During this phase, the builder can explore as freely as necessary to build up a map of the size and detail required. If maps overlap, no attempt is made to merge structural information. This causes indeterminate behaviour, dependent on how the relocalizer recovers. However, with care, overlapping maps can be used effectively. Examples of both these behaviours are shown in the experiments.

Once all of the maps have been made, the system can be

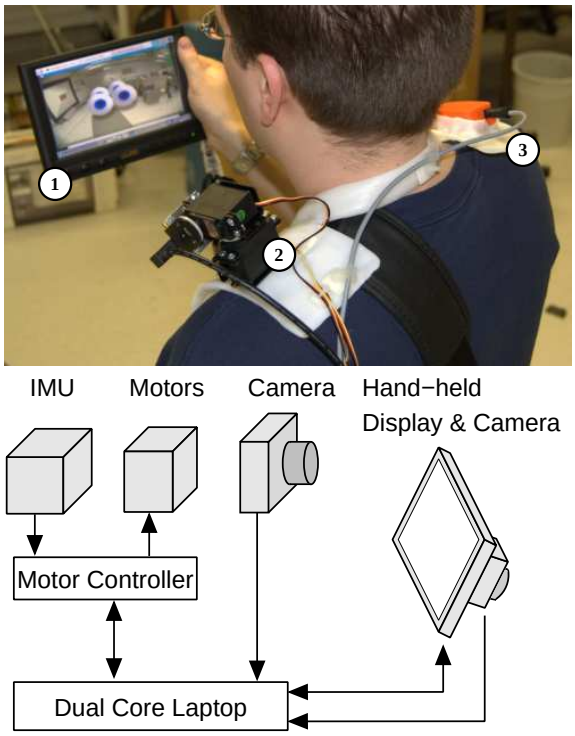


Figure 3. View of the wearable camera system, showing the following elements: (1) Hand-held display with camera mounted on the rear for AR applications. (2) Active camera capable of pan, and tilt. (3) Inertial measurement unit (IMU).

given to the users. Access to the maps could be restricted in several ways. For example, a map might be made "read-only", so no new measurements can be added; or the map might be made writeable within a restricted metric extent, allowing changes to the map since its creation to be incorporated, but stopping the map from growing outside of its bounds where it could potentially interfere with other maps.

Now, when the user moves away from one map the system will become lost and start its relocalization routine. Once another map is found the system will lock to it. We thus allow the user to exploit local maps within large environments, but leave the user to move between maps.

5. Implementation on a wearable vision system

The system with its multiple map and multiple tracker extension has been implemented in C++ under Linux using a dual core 2.20 GHz Intel laptop with 2GB RAM. This supplies sufficient processing power to run simultaneously two trackers and one map maker.

The wearable vision system used for this work is shown in Figure 3. The active camera has servo-driven pan and tilt rotational axes and is mounted along with an inertial measurement unit (IMU) on a shoulder support. The servo controller, executing with associated control functions on an Atmel ATMega128 processor, continuously receives angle data from the IMU, allowing the camera to remain stabilized and, if desired, to fixate on a point in the world. Control messages can be sent to the control electronics either from the user, or from the software, and the camera can enter an active search pattern when the system becomes lost.

The wearer can input information via a hand-held touch screen, which allows mouse-like operations, and includes a virtual on-screen keyboard. The display also has a second camera mounted on its rear side, allowing it to act as a 'magic lens', *i.e*. the user can see the augmented reality associated with a particular map through the display as shown in Figure 3. The display can also be switched to show the shoulder camera's image stream.

6. Experimental evaluation

Three different experiments are presented in the paper: (1) the "desk" sequence shows the robustness, and some limitations of the system, and how the multiple mapping works; (2) the "laboratory" sequence shows how the system can use overlapping maps, and handle mobile objects; (3) the "building" sequence shows how the system can be used to explore large, sprawling environments. The experiments were performed using two cameras, the hand-held camera and the shoulder mounted active camera described above. All the results presented were processed live at 30 Hz and stored directly to video. The video material accompanying the paper is an important supplement to aid the reader's understanding[2].

6.1. Desk sequence

The purpose of this experiment is to demonstrate the multiple mapping capability of the system, its robustness to similar maps, and how it compares to the original single map system. In this experiment 12 maps were made on 15 desks. Each desk is similar (curved shape, with a computer), but has varying amounts of clutter depending on its occupant. The user creates the maps and adds some AR to show to whom the desk belongs, and which research group they are a member of.

The system was able to successfully relocalize onto all of the mapped desks. Frames 6–8 of Figure 4 show the relocalization of three of the maps. The system was even able to relocalize correctly onto two desks (frames 11 and 12) that were sparse in features (frames 9 and 10). When the user is traversing the areas between maps the system becomes lost and attempts to relocalize (frame 5). Once the user arrives at a mapped location the system is able to recover and tracking resumes.

[2]Full versions of all the results videos can be found at `http://www.robots.ox.ac.uk/ActiveVision/Publications/castle_etal_iswc2008`.

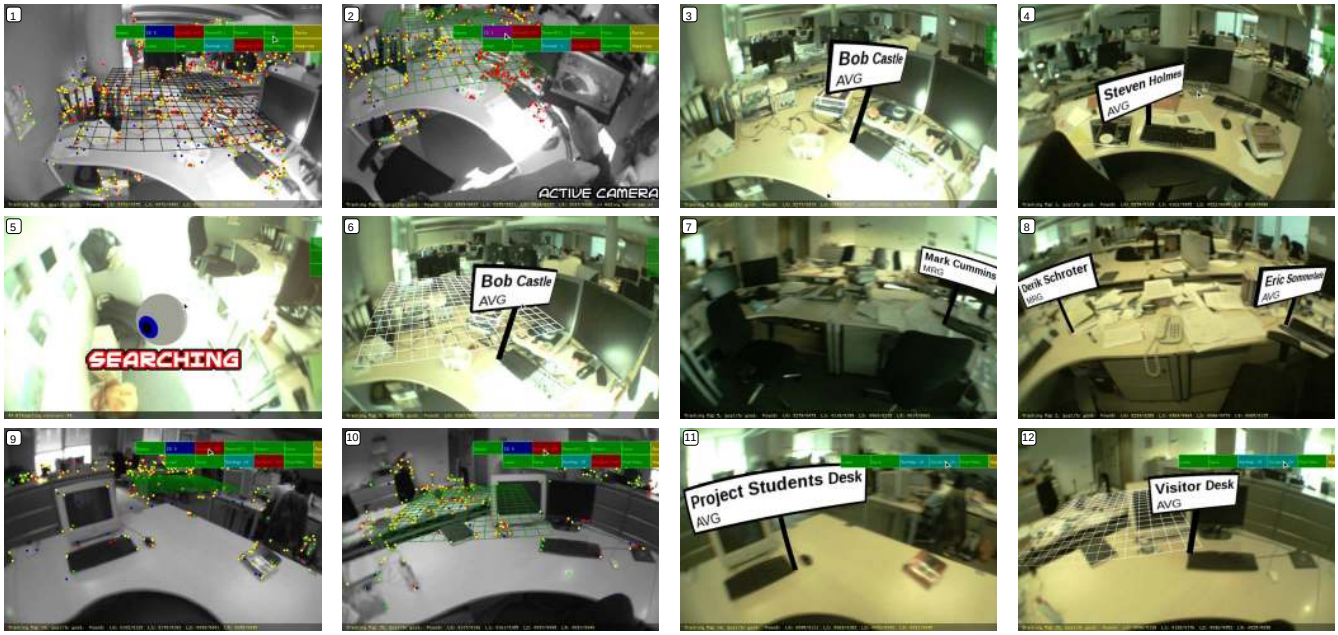


Figure 4. A demonstration both of multiple maps and robustness to self similarity of maps. 12 maps were made of 15 desks, and each desk was augmented with the user's name and research group. (1,2) Hand-held camera and active camera view working in the same map (3) AR added to map, (4) another map created and labelled, (5) attempting relocalization (6–8) successful relocalization on different maps (9–12) Creation of maps on two sparsely featured desks, and subsequent successful relocalization.

At one point during the labelling (frame 7), the map of one user's desk is allowed to grow and cover neighbouring previously mapped desks. This causes the two maps to become overlapped. Because no geometric links exist between the maps, the system has no way of handling the overlap in a defined manner. Instead the system will recover to the map that provides the best pose from the relocalizer, and in this case the system only recovers to the latter map, resulting in the label on the other desk not being seen. This is a limitation of the system, and can usually be avoided during the creation of maps, or used purposely as shown in the next experiment.

Using these multiple maps the system is able to run in real time, adding further keyframes as required, running the bundle adjustment, and allowing accurate maps to be provided to the user. Each map contained between 3 and 37 keyframes and 290 to 1900 map points, resulting in a total of 177 keyframes and 12327 map points. As a comparison, the original single camera and single map system [9] was used to try and create one large map of the same environment. The system was also able to map the same area, creating a map containing 163 keyframes and 10144 map points. However, at this map size the bundle adjustment takes a significant about of time, stopping new keyframe insertion, and hence reducing the freedom to explore. If time is given for the bundle adjustment to run then exploration can continue for a while longer, before having to wait again. Also, a lack of occlusion reasoning in the original system causes a breakdown of the tracking quality heuristics, effectively preventing further keyframe insertion. These issues lead to the system becoming more fragile to use, and the user having to retrace their steps when the system becomes lost more and more frequently, until a suitable relocalization position is found. The tracker, however, is able to continue running at frame rate, no matter the map size, as long as the map can grow in a timely manner. In contrast, using multiple small maps allows a robust tracking and mapping experience, without the fragility and the waiting for the bundle adjustment of the original system.

6.2. Laboratory sequence

The purpose of this experiment was to show how the careful crafting of overlapping maps can be used to encode several levels of information into a scene, and provide robustness to scene change when an object is moved.

Two maps are created: one is a large map of a laboratory, the other is a small map located on the top of a Segway mobile robot. In Figure 5 frames 1 and 2 show the map of the laboratory being created and augmented. Frame 3 shows the robot map with its AR. The smaller robot map is located inside the first, larger, laboratory map, and they are therefore fully overlapping. The system will track the room map until the user is looking at the robot close-up. At this point the system is unable to maintain tracking on the room map and enters relocalization. It relocalizes onto the robot map, and resumes tracking (frame 6). It remains tracking

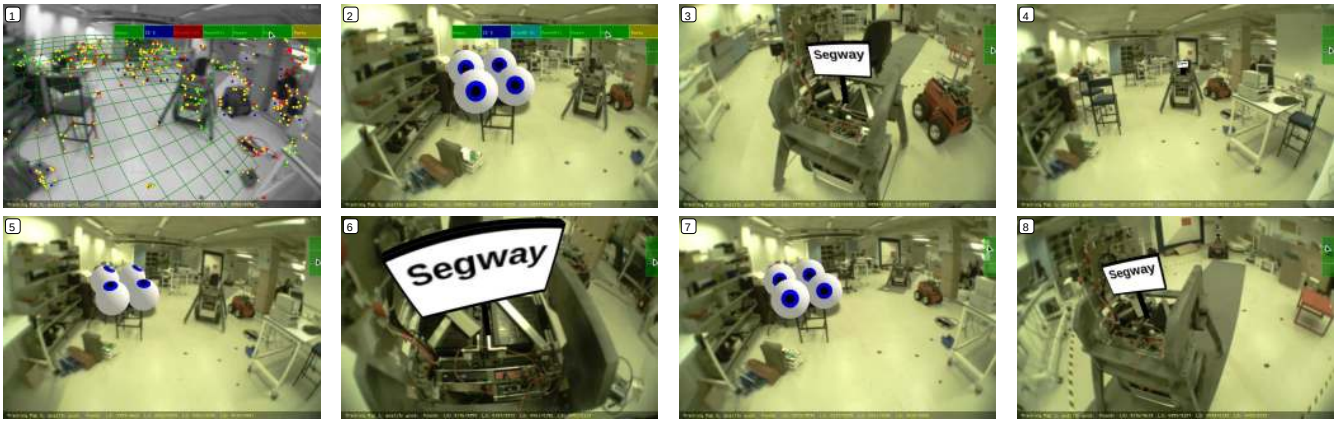


Figure 5. A demonstration of the ability to move between overlapping maps, and relocate maps that have moved. (1,2) Map of room created, and AR overlay added (3) map on mobile robot created, and labelled (4) robot map tracked over a large scale change (5) switching back to the room map (6) switching to the robot map (7,8) after robot has moved relocalization is successful on the room and robot map.

the robot map as the user backs away (frame 4), and only relocalizes to the room map when tracking is lost (frame 5). Overlapping maps used in this way allow several levels of information to be encoded into a scene, with switching obtained by getting close enough to an object, or rapid camera motion, causing blur, or an object disappearing from view.

The main advantage of having a smaller map on the robot, as opposed to having one large map containing all the AR, is that this allows the robot to move to different locations and still maintain its own AR. In frame 7 the robot has moved, but the room map is still able to be tracked as the majority of the scene structure is fixed. When the user approaches the robot in its new location the relocalizer switches the system to the robot map (frame 8).

6.3. Building sequence

This final experiment shows how the system can be used in large sprawling environments, providing AR throughout a large building. Maps were made around the building on multiple floors. Figure 6 shows frames from the sequence. Frames 1–4 and 6 are the five places where maps were made. Frame 2 shows a map created in a particularly feature sparse location. Relocalization into 3 of the maps is shown in frames 5–7. The maps were then reloaded into the system later in the day and four out of the five maps (frames 8–11) successfully relocalized. The system failed to relocalize onto the final map frame 12, which was most likely due to the large increase in brightness in the scene since the map was created.

7. Conclusion

In this paper we have shown that the ability to build multiple small maps around an environment is beneficial to wearable applications, where the wearer can be trusted to move around the world freely, and reach their desired destination. The experiments have also shown that the creation of multiple maps allows the system to scale better than using a single map, and that the use of multiple maps is beneficial for augmenting individual objects, and large sprawling environments. The developed system is scalable, and can grow with the ever increasing performance of computers, and the increase in the number of processing cores. The system allows disconnected maps of the world to be made that can cover a larger total area than a single map could, while maintaining real-time operation.

Integration of the object recognition work done previously [2], will allow a richer AR experience around particular real world objects for the user. It may also allow semantic links between overlapping maps that contain the same objects. Another important development of this work will be to record the spatial location of maps with respect to other maps, and detect overlaps. This would allow maps to be joined if desired, and more intelligent searching during relocalization by only considering spatially near maps. The IMU could assist in this by aiding the estimation of the gross motion of the user. This would help reduce the search time of the relocalizer to a near constant time, regardless of the number of maps.

Acknowledgements

This work was supported by UK Engineering and Physical Science Research Council (grants GR/S97774 and EP/D037077).

References

[1] S. Benhimane and E. Malis. Homography-based 2D visual tracking and servoing. *Special Joint Issue on Robotics and Vision. Journal of Robotics Research*, 26(7):661–676, July 2007.

[2] R. O. Castle, D. J. Gawley, G. Klein, and D. W. Murray. Video-rate recognition and localization for wearable cameras. In *Proc*

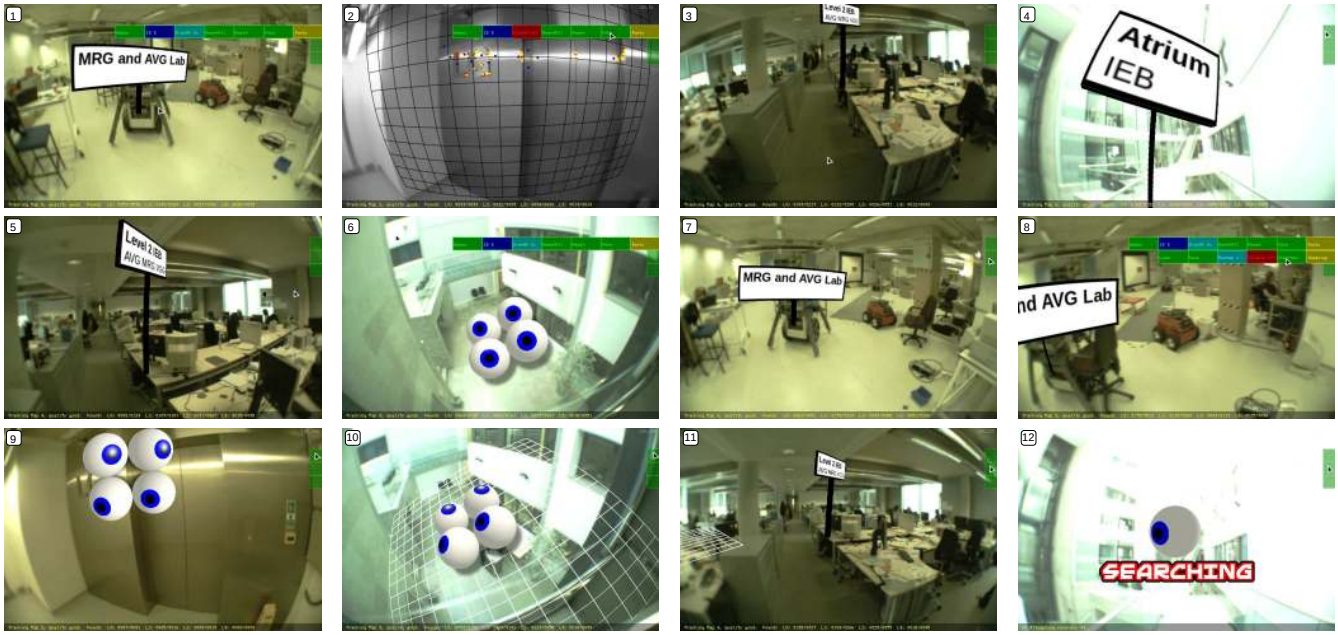


Figure 6. A demonstration of exploring a large scale environment containing five maps, with augmented reality overlays over multiple building floors. (1–4) Creation and labelling of four of the maps (5–7) relocalization onto three of the maps (8–11) successful relocalization at a later time (12) failure to relocalize.

18th British Machine Vision Conference, Warwick, Sept 11-13, 2007, pages 1100–1109, 2007.

[3] A. J. Davison. Real-time simultaneous localisation and mapping with a single camera. In *Proc 9th Int Conf on Computer Vision, Nice, France, Oct 13-16, 2003*, pages 1403–1410, 2003.

[4] A. J. Davison, I. D. Reid, N. Molton, and O. Stasse. MonoSLAM: Real-time single camera SLAM. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(6):1052–1067, 2007.

[5] O. Faugeras and F. Lustman. Motion and structure from motion in a piecewise planar environment. *International Journal of Pattern Recognition and Artificial Intelligence*, 2(3):485–508, 1988.

[6] A. W. Fitzgibbon and A. Zisserman. Automatic camera recovery for closed or open image sequences. In *Proc 5th European Conf on Computer Vision, Freiburg*, volume 1, pages 311–326, 1998.

[7] J. E. Guivant and E. Nebot. Optimization of the simultaneous localization and map-building algorithm for real-time implementation. *IEEE Transactions on Robotics and Automation*, 17(3):242–257, 2001.

[8] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second edition, 2004.

[9] G. Klein and D. W. Murray. Parallel tracking and mapping for small AR workspaces. In *Proc IEEE/ACM 6th Int Symp on Mixed and Augmented Reality, Nara, Japan, Nov 13-16*, 2007.

[10] G. Klein and D. W. Murray. Improving the agility of keyframe-based SLAM. In *Proc 10th European Conf on Computer Vision, Marseille, France, October 12-18*, 2008.

[11] T. Lee and T. Höllerer. Hybrid feature tracking and user interaction for markerless augmented reality. In *Proc 10th Int Conf on Virtual Reality, Reno, NV, March 8–12, 2008*, pages 145–152, 2008.

[12] J. J. Leonard, H. F. Durrant-Whyte, and I. J. Cox. Dynamic map building for an autonomous mobile robot. *International Journal of Robotics Research*, 11(8):286–298, 1992.

[13] J. J. Leonard and H. J. S. Feder. A computationally efficient method for large-scale concurrent mapping and localization. In *Proc 9th Int Symp on Robotics Research, Utah, October 1999*, pages 316–321, 1999.

[14] J. J. Leonard and P. M. Newman. Consistent, convergent and constant-time SLAM. In *Int Joint Conference on Artificial Intelligence, 2003*, pages 1143–1150. Morgan Kaufmann, 2003.

[15] V. Lepetit and P. Fua. Keypoint recognition using randomized trees. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(9):1465–1479, 2006.

[16] W. W. Mayol, A. J. Davison, B. J. Tordoff, and D. W. Murray. Applying active vision and SLAM to wearables. In P. Dario and R. Chatilla, editors, *Robotics Research, The Eleventh International Symposium, Siena 2003*, Springer Tracts in Advanced Robotics, volume 15, pages 325–334. Springer, 2005.

[17] E. Mouragnon, F. Dekeyser, P. Sayd, M. Lhuillier, and M. Dhôme. Real time localization and 3d reconstruction. In *Proc 24th IEEE Conf on Computer Vision and Pattern Recognition, New York NY, 17-22 June, 2006*, pages 363–370, 2006.

[18] D. Nistér. *Automatic dense reconstruction from uncalibrated video sequences*. PhD thesis, Royal Institute of Technology KTH, Stockholm, Sweden, 2001.

[19] D. Nistér, O. Naroditsky, and J. Bergen. Visual odometry for ground vehicle applications. *Journal of Field Robotics*, 23(1), 2006.

[20] M. Pollefeys, R. Koch, and L. Van Gool. Self-calibration and metric reconstruction in spite of varying and unknown internal camera parameters. *International Journal of Computer Vision*, 32(1):7–25, 1999.

[21] E. Rosten and T. Drummond. Machine learning for high-speed corner detection. In *Proc 9th European Conf on Computer Vision, Graz*, volume 1, pages 430–443, 2006.

[22] R. C. Smith and P. Cheeseman. On the representation and estimation of spatial uncertainty. *International Journal of Robotics Research*, 5(4):56–68, 1986.

[23] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. MIT Press, Cambridge MA, 2005.

[24] B. Williams, G. Klein, and I. D. Reid. Real-time SLAM relocalisation. In *Proc 11th Int Conf on Computer Vision, Rio de Janeiro, Brazil, Oct 14-20*, 2007.