Wide-area Augmented Reality using Camera Tracking and Mapping in Multiple Regions

Robert O Castle, Georg Klein, David W Murray*

Active Vision Laboratory, Department of Engineering Science, University of Oxford, Parks Road, Oxford OX1 3PJ, 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. The method is shown working in a progressively larger environments, from desktop to large building.

Key words: augmented reality, structure from motion, localization and mapping, wearable vision.

1. Introduction

Overlaying live video with collocated graphics to generate a sense of augmented reality (AR) for the user of a wearable or hand-held camera requires the continual estimation of the camera's position and orientation relative to a geometric map of its surroundings. This map may be built beforehand — most simply perhaps by specifying the point positions of artificial markers placed in the scene, or by using a CAD model with complexity ranging from a single object or room interior to a streetscape. However, such handcrafting of models has long been regarded as impracticable, and much effort in visual structure from motion (SfM) and simultaneous localization and mapping (SLAM) has been dedicated to acquiring structural models and maps automatically from markerless scenes.

There is much that is common to both approaches, but, if distinction needs to be drawn, SfM is rooted in the off-line optimal reconstruction of both n-point and dense

*Corresponding author. Tel +44 1865 273106.

Email addresses: bob@robots.ox.ac.uk (Robert O Castle), gk@robots.ox.ac.uk (Georg Klein), dwm@robots.ox.ac.uk (David W Murray)

structure (*e.g.* [1] [2] and [3] [4], respectively), while the SLAM's tradition in robotics lies in the on-line recursive recovery of the scene whilst respecting the correlations amongst and between camera and scene entries in the state vector (*e.g.* [5] [6] [7]). There is a middle ground, but work in recursive SfM, as typified by [8] [9], neglected correlations between the camera state and map points, and between map points themselves.

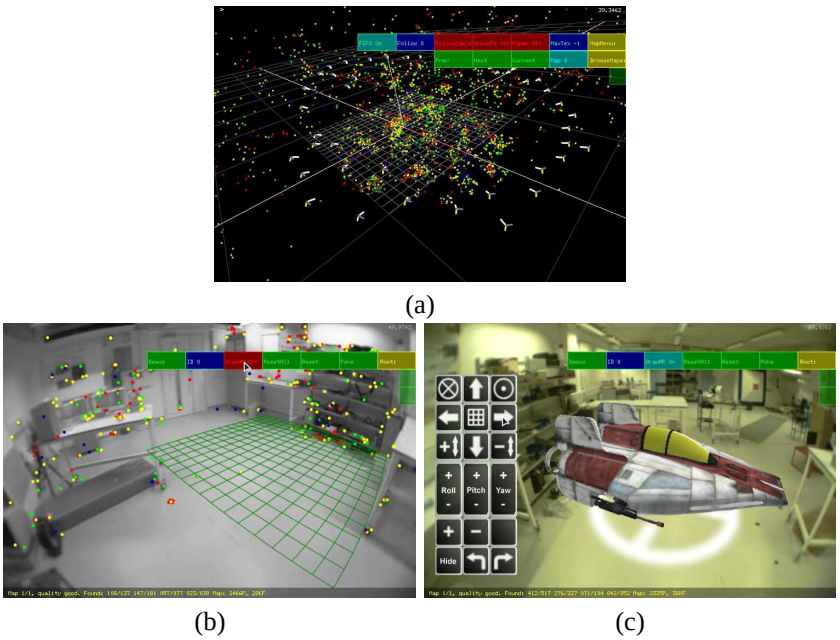
While batch-mode SfM methods dominate in the post-capture insertion of graphics into video, the use of SfM to build maps for AR on the fly has been less common, and in recent years it is recursive SLAM that has appeared better suited to small-scale live AR applications. The monoSLAM algorithm [10] [11] based on the extended Kalman filter has proved formative, and has been applied in demonstrations of augmented reality in basic form [12] [13], in conjunction with hand tracking [14], and with automatic recognition and localization of objects [15] allowing rapid prototyping of a "live guides" to the operation of equipment [16] and so on.

There is obvious appeal in summarizing observations succinctly in a state vector and covariance matrix, but there are costs in doing so. Chief amongst these are, first, that monoSLAM has quadratic complexity in the number of map points, and a practical limit to the map size for 30 Hz operation on current portable processors is around 100. Davison summarizes the approach as generating a sparse map of high quality features [17], but as a consequence the maps are also of limited extent. As with other SLAM methods where complexity requires taming (*e.g.* [18] [19] [20]), it has been demonstrated how to extend the reach of monoSLAM's maps using hierarchical submapping, where maximally sized submaps of features are stitched together with an explicit test for loop closure [21]. Notwithstanding, in many cases it would be desirable just to increase the size of each individual (sub)map.

A second cost arises from the involvement of a strong motion model. Modelling the temporal evolution of the state is fundamental to recursive filtering, as it allows the accumulated wisdom of past measurements to be combined meaningfully with current measurements. Much less fundamentally, but rather importantly in vision, modelling and prediction make data association more reliable. In recursive processing this is all the more important, because errors due to mismatching become locked into the state — there is no later opportunity for review and amendment of error. So important is the need to avoid mismatches in monoSLAM that we find users tend to move the camera over-cautiously to satisfy the motion model, rather than naturally and freely to satisfy themselves.

One approach to managing less predictable motion in tracking is to use a particle filter [22] to represent pose [23], but taking fuller account of correlations between camera pose and scene points in this way is very expensive [24]. A method which takes account of correlations (and indeed avoids the inconsistency inherent in the EKF) and is computationally cheaper is the recent graph-based algorithm of [25] in which graph nodes represent landmark estimates which are conditioned on local observations, and the graph edges represent transformations between the nodes. Because the method optimizes iteratively, it can handle many more landmarks, but its ability to handle agile motion is less clear.

Here though we build a wearable visual AR system upon the authors' earlier method of parallel camera tracking and 3D mapping, PTAM [26] [27], which explicitly ad-



(a)

(b) (c)

Figure 1: Typical output from PTAMM: (a) a 3D map, with the keyframes indicated by their coordinate axes; (b) the user's view when building a map; and (c) the view when adding and manipulating models by touching the screen. Note that the dominant plane, visible in (b), is often useful for positioning augmentations, but there is no restriction on their placement, and the plane plays no special part in the processing.

dresses both problems. In this method (which is more SfM than SLAM), the urgent task of maintaining knowledge of the pose of the camera (camera tracking) is achieved for every frame by resectioning, assuming fixed map points. Meanwhile, in parallel, the map points and selected camera poses are refined using repeated batch-mode bundle adjustments, but with image measurements taken only from the spatially well-separated keyframes. A motion model is used primarily to aid finding matches, allowing the user greater freedom of movement, and video-rate operation is maintained with several *thousands* of point features in each map.

To further extend the range of map coverage, we propose extensions to PTAM to allow multiple cameras to build and utilize multiple maps, with automatic switching between maps. The extension is made quite straightforward by the independence of the tracking and mapping processes. It is demonstrated empirically that the individual 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.

Unlike sub-mapping, where the absolute orientation problem is solved using shared features, we suggest that for many applications of AR in wearable computing there is no imperative to join the separate maps together geometrically. Unlike a robot camera platform, the human user can be relied upon to act on weak spatial and navigational links between individual maps. Here, maps are linked in two ways — for the vision

system by a mechanism which relocalizes a lost camera, and for the user by augmenting them with visual signposts. For a particular application in AR, the maps define the areas of interest, and it follows that the regions in between are uninteresting and need not be mapped at all.

Section 2 reviews the tracking and mapping algorithm upon which our work is based. Section 3 explains the modifications required to allow multiple tracking cameras to work with multiple maps, and Section 4 outlines their practical application on a wearable vision platform built to support this work.

Sections 5 and 6 provide an experimental evaluation of the method. Experiments are undertaken first to demonstrate the viability of the method, dubbed Parallel Tracking and *Multiple* Mapping (PTAMM), and to show that if maps are sufficiently restricted spatially the maps alone can be treated as objects or places. PTAMM is then demonstrated in a prototype application in which visual AR is delivered to a user walking through exhibits in a museum of natural history. Section 7 lists our conclusions and provides more general discussion. An early description of part of this work appeared in [28].

To aid explanation, Fig. 1 highlights relevant items in the output from PTAMM: the 3D map and its keyframe locations, the AR workspace, and the addition of 3D entities to the map. The accompanying video material is also an important supplement, providing evidence of PTAMM's video-rate performance. Video material supporting the experimental results is at
www.robots.ox.ac.uk/ActiveVision/Publications/castle2009A.

2. Parallel tracking and mapping

For a single camera working in a single map, the PTAM algorithm [26] is designed to achieve video-rate recontruction of AR workspaces, and robust camera tracking within them. The spaces are expected to be up to room-sized (with floor area up to, say, $20\,\mathrm{m}^2$, as in Fig. 1) and predominantly rigid. Within these constraints the method is able to handle rapid camera motion, including rotation, and large changes in scale.

2.1. Camera tracking in PTAM

The algorithm in Fig. 2(a) summarizes camera tracking when running normally, using an already initialised 3D map. When a colour frame is delivered by the camera, the tracking thread generates a luminance image from it, sub-samples this into a four-level pyramid, and computes corner features at each scale using the FAST-10 detector [29], chosen for its order of magnitude higher speed than, for example, the Harris and DoG detectors. (See Table 4 of [30].) The colour original is retained for use with the augmented display.

To assist the first round of data association an estimated pose for the camera is obtained from a motion model that assumes constant (or decaying) velocity in the presence (or absence) of measurements. All potentially visible map points are projected from 3D into the image space, up to 50 matches sought at the coarsest scale using proximity and correlation between appropriately warped image patches, and the pose

re-estimated using robust minimization. A second stage of matching and pose optimization is made at the finest scale, adding up to 1 000 matches. For both stages the camera pose $\hat{\boldsymbol{\mu}}$ is estimated as

$$\hat{\boldsymbol{\mu}} = \arg\min_{\boldsymbol{\mu}} \sum_j \rho\left(e_j/\sigma_j\right) , \tag{1}$$

where ρ is a robust measure of the reprojection error

$$e_j = \left|\mathbf{x}_j - \mathbf{x}(\mathbf{K}, \boldsymbol{\mu}, \mathbf{X}_j)\right| . \tag{2}$$

In (2), $\mathbf{x}_j$ is a measurement, $\mathbf{x}(\mathbf{K}, \boldsymbol{\mu}, \mathbf{X}_j)$ is the projection of the corresponding map point $\mathbf{X}_j$, and $\mathbf{K}$ are the camera's calibrated intrinsics and distortion parameters. The measurement variance is assumed to be isotropic in the image, and to scale as 2^{2l} where l is the pyramid level. A Huber M-estimator is used for the cost [31] [32] for which, writing $\beta = (e_j/\sigma_j)$,

$$\rho(\beta) = \begin{cases} \beta^2/2 & \text{if } |\beta| \le \sigma_T \\ \sigma_T(|\beta| - \sigma_T/2) & \text{otherwise.} \end{cases} \tag{3}$$

The distribution threshold σ_T is set equal to the estimate of the error distribution's standard deviation. The minimization is performed using Gauss-Newton, and ten iterations per stage are found adequate.

The fraction of features successfully matched is monitored to provide a measure of tracking quality, and two thresholds are set. If the fraction exceeds the higher, tracking is deemed good; if it falls between them, tracking continues but the frames involved are prevented from becoming keyframes for the mapping process; and if it falls below both thresholds it is assumed that tracking has failed terminally. All the experimental work in this paper uses values of 0.30 and 0.13 for these fractions. Because of the greater number of map points and the greater permitted agility of camera motion, such outright failure occurs less frequently in PTAM than in monoSLAM. When it does, most often because of a very fast twist of the camera, a relocalization method is invoked to recover the lost camera to a pose where tracking can re-start. Discussion of it is deferred until Section 3.2.

2.2. Selecting frames as keyframes

Some of the frames captured are designated as keyframes and play an important role in the mapping process. As the camera moves around the environment, the current frame is selected as a keyframe whenever (i) tracking is deemed good, (ii) the camera has translated by a minimum distance from any previous keyframe, and (iii) the current queue of new keyframes is less than three long. The minimum distance in (ii) is based on the Euclidean distance between the frames and mean depth of points in the scene, causing keyframes near surfaces to be closer together. The new keyframe is added to a queue for the mapping thread to process. The k-th keyframe groups the features $\mathbf{x}_{ik}$ computed in it, the match list between those features and the map points $\mathbf{x}_{ik} \leftrightarrow \mathbf{X}_j$ (although not all features will be matched), and the best estimate of the camera pose $\hat{\boldsymbol{\mu}}_k$ from the tracker.

2.3. Mapping in PTAM

Algorithm (b) in Figure 2 outlines the mapping process in PTAM. It runs separately from tracking, allowing it to take longer than the inter-frame period to compute maps.

A map is initialized at the outset using a five-point stereo algorithm [2] (similar to those in [33] [34]). The first camera viewpoint is selected by the user and its image and feature list becomes the first keyframe, with pose (*i.e.*, rotation and translation) $\boldsymbol{\mu}_1 \equiv \{\mathbf{R}_1, \mathbf{t}_1\} = \{\mathbf{I}, \mathbf{0}\}$. The camera is then moved to a new position with sufficient care to allow features to be tracked using the image alone. The image is captured as the second keyframe, and Nistér's relative pose algorithm [2] is used with RANSAC to compute the new camera pose $\boldsymbol{\mu}_2 = \{\mathbf{R}_2, \mathbf{t}_2\}$. Pairs of matching FAST corners from the two images are then used to triangulate the initial set of 3D scene points $\mathbf{X}_j$ for the map. It is assumed that $|\mathbf{t}_2|$ is some 100 mm so that a reasonable scale can be applied to the map. The scale is, of course, arbitrary and does not affect later processing. The time taken to initialize the map is dominated by the time the user takes to translate the camera carefully and select the initial keyframes — in practice it takes a couple of seconds.

With the map initialized, camera tracking begins as described in Section 2.1, occasionally adding a new keyframe to the queue for processing. The new keyframe and its features are integrated into the map in the following way. New 3D points are added to the map by finding image features which are unmatched, particularly salient, and are distant in the image from any matched feature. To find their 3D positions, a search is made for image matches in the spatially closest keyframe, and the position triangulated using the estimated keyframe poses.

The estimated 3D positions $\{\mathbf{X}_j\}$, $j = 1 \ldots J$ of all map points and keyframe camera poses $\{\boldsymbol{\mu}_k\}$, $k = 2 \ldots K$ (all except the first, which is fixed) are optimized in a bundle adjustment, using Levenberg-Marquardt [35]. The magnitude of the image error is

$$e_{jk} = \left| \mathbf{x}_{jk} - \mathbf{x}(\mathbf{K}_k, \boldsymbol{\mu}_k, \mathbf{X}_j) \right| , \tag{4}$$

where $\mathbf{x}_{jk}$ is the measurement that arose from $\mathbf{X}_j$, and the adjustment is

$$\{\hat{\mathbf{X}}\}, \{\hat{\boldsymbol{\mu}}\} = \arg\min_{\{\mathbf{X}\},\{\boldsymbol{\mu}\}} \sum_j \sum_k v_{jk} \rho\left(e_{jk}/\sigma_{jk}\right) , \tag{5}$$

where the visibility flag

$$v_{jk} = \begin{cases} 1 & \text{if point } j \text{ is visible and matched in } k \\ 0 & \text{otherwise,} \end{cases} \tag{6}$$

and ρ is now a Tukey M-estimator [36],

$$\rho(\beta) = \begin{cases} (\sigma_T^2/6)\left(1 - [1 - (\beta/\sigma_T)^2]^3\right) & \text{if } |\beta| \le \sigma_T \\ (\sigma_T^2/6) & \text{otherwise.} \end{cases} \tag{7}$$

Tukey was found here to give a slight improvement performance over Huber.

In the operating regime, the time taken to execute bundle adjustment is dominated by $O(JK^2)$. However, because this is a batch process and the data retained throughout, it is open to the mapping process to delay performing the complete adjustment

PTAM: Tracking	PTAM: Map Maker

while map initialized and new image available **do**
  Form pyramid, compute corners
  Predict camera pose
  for pyramid levels $l = 2$ then $l = 3$ **do**
  Project map points into image
  Match
  Update camera pose using selected, then all, matches
  end for
  if tracking good and keyframe criteria met **then**
  Offer as keyframe to map maker
  else if tracking lost **then**
  Relocalize
  end if
end while
	Initialize map from two views
	Set running
	while running **do**
	  if new keyframe offered and acceptable **then**
	  Update existing data associations
	  Compute new 3D points from two views
	  else if not locally converged **then**
	  Perform local adjustment
	  else if not globally converged **then**
	  Perform global adjustment
	  else
	  Update data associations
	  end if
	end while

Figure 2: Skeleton code for the tracking and map maker threads in PTAM.

until processor loading permits. As a stopgap when the camera is exploring and new keyframes and measurements need to be added to the map frequently, local bundle adjustments on a fixed and limited number of keyframes are performed. Work on visual odometry [34] has shown that local bundle adjustments in a temporally sliding window were remarkably accurate even without global adjustment. In our case the locale is spatial: there may be some correlation with time, but the locale will include older keyframes if the camera revisits a location after a period of neglect.

3. PTAMM: multiple maps and trackers

3.1. Multiple maps

Measurement using a portable 2.20 GHz dual core processor indicates that the map size at which the ability to explore is curtailed in PTAM is around 150 keyframes and 20 000 3D points. Interpreting the $O(JK^2)$ complexity in terms of such hard limits is awkward however because of the several layers of partial optimization involved: visual odometry from the trackers gives good starting estimates of pose, and these poses and the map points are part-refined using $O(1)$ local adjustments using a fixed number of keyframes. In practice, and for a given processor, a map's extent depends on (i) the degree of exploration, (ii) the degree to which local adjustments suffice until accumulated errors cause the global adjustment to fail, and (iii) occlusion causing the tracker to prohibit addition of further keyframes (see the experiment in Section 5.2).

Here we take the pragmatic view that the user must *always* be able to explore their environment freely, and for this reason propose the building and use of multiple maps, with each sized conservatively and under the control of the user.

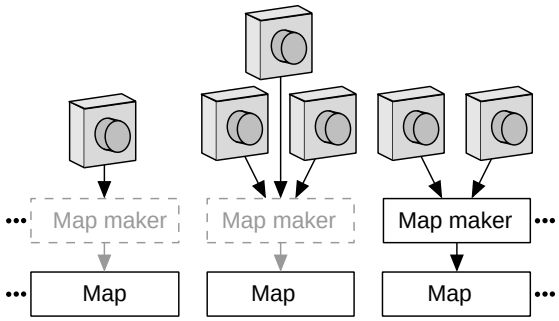


Figure 3: An array of maps, each with their associate map maker fed by single or multiple cameras. The rightmost show the system configuration used.

As suggested by Fig. 3, a notional ideal for managing multiple maps would be to use a separate map maker for each, so ensuring that all maps were optimized, or were being optimized. Aspiring to this on modest hardware requires compromise, not least to fall within the limits on independent threads that can execute together. On the wearable system used here only one map maker is active, and the cameras work in the same map. If a different map is to be used, the map maker and cameras need to switch to it. Were more computing resource available, other approximations to the ideal might be preferred. For example, a roving map maker, with no cameras attached to it, might optimize maps chosen using a paging strategy — *e.g.* giving priority to those that are currently unoptimized but that are likely to be used soon.

3.2. Map switching via relocalization

A number of methods have been reported for the detection of loop-closure or the recovery of the camera location after tracking failure [37] [38] [39] [40] [41]. For the latter task, the original version of PTAM used the randomized tree method of [39] previously demonstrated for camera recovery in monoSLAM. However, randomized trees occupy around 1.3 MB of memory per map point — tolerable for a map with a 10^2 features, but not for maps containing 10^4. Similar difficulties with memory and speed apply to methods based on distinctive features (e.g. [38]), which have been used elsewhere for recognition with monoSLAM [16].

Here a cheaper relocalization method is used which exploits the quite uniform and dense spread of keyframes through a map. A descriptor for each keyframe is created by sub-sampling the image 8-fold (in practice, $640 \times 480 \rightarrow 80 \times 60$), then applying Gaussian blur with scale $\sigma = 5$. The current camera image (similarly subsampled) is compared to those from keyframes using zero mean sum of square differences.

The keyframe that has the least difference is accepted, the camera's pose is set to that of the keyframe, and a rotation correction estimated using a direct second order minimization [42] of the sum of squared differences. If tracking fails to restart, the relocalizer is invoked again — though always using the camera's current image.

In multiple map PTAMM, relocalization is also used to switch between maps: but when tracking fails it is uncertain whether the camera is looking at an area belonging

to the same map, or of another map, or indeed at an unmapped area. To resolve this, the relocalization mechanism cross-correlates with all of the keyframes from all maps. In the current implementation, the cost of search grows linearly with the map size, but each comparison calculation takes less than 2×10^{-2} ms allowing the 30 or so large maps that could be held in the active memory of 2 GB to be searched in under 67 ms, or 2 frame periods.

If the recovered camera lies in the current map, tracking resumes as normal; but if in a new map, the tracker and map maker switch maps and resume tracking from the estimated pose, calling up any AR attached to the new map.

When using rather than building maps, extended periods of tracking failure indicate that the wearer and camera have moved out of all mapped regions. PTAMM's tracker continues to search its maps using the most recently captured image, and provides indication of this to the wearer.

3.3. Multiple camera trackers

Because the tracking process is largely independent of map making, it proves straightforward to allow multiple cameras to add information into any one map. A map is initialized using just one camera so that the coordinate system is defined uniquely. Thereafter, any number of cameras can use those 3D points to determine their poses from frame to frame. Additional cameras determine their pose relative to the map using the relocalization method outlined above.

The more interesting question is how to allow multiple cameras to build maps, *i.e.* when to insert keyframes into the map. From a geometrical standpoint this is also answered straightforwardly, because in single camera PTAM new keyframes are determined by spatial separation of poses, not temporal separation. With multiple cameras, each camera tracker offers frames that individually satisfy the earlier criteria to the map maker, and the map maker decides whether or not to insert the keyframes into the map and run local or global adjustments. Within certain constraints on photometric similarity, additional views can be utilized at will, and multiple trackers can work on a single map simultaneously or at different times.

When multiple but "associated" cameras trackers are employed, as on the wearable system described below, the relocalization strategy is modified. If one of the trackers becomes lost but the other is not, search will only occur in the current map. If all trackers are lost the complete search is performed by all trackers, and the first tracker to recover causes the other to abandon their complete search in favour of attempting to find their positions in the recovered tracker's map.

4. Implementation and usage

PTAMM is implemented in C++ and runs under the Linux OS on a portable Intel dual core 2.20 GHz processor, part of the wearable vision system shown in Fig. 4. This has two cameras, one on a servo-driven pan-tilt mechanism worn on the shoulder (CamW), and the other (CamH) mounted on the back of a hand-held touch screen which acts as a "magic lens", allowing the user to see the scene beyond with augmentations overlaid. Both cameras deliver 640×480 pixel imagery at 30 Hz, and have horizontal fields of view of 94°.

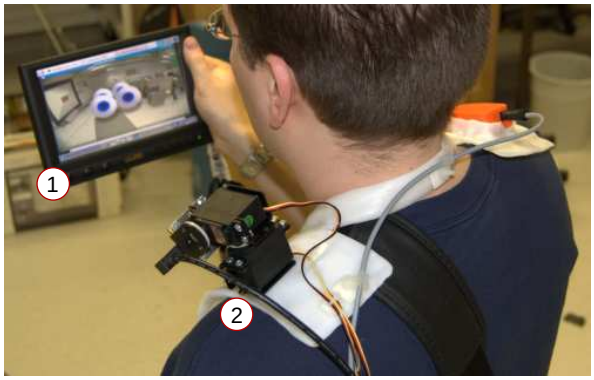


Figure 4: View of the wearable system, showing (1) Hand-held display with camera (CamH) mounted on the rear for AR applications and (2) the Wearable pan-tilt camera (CamW) on the left shoulder. The inertial measurement unit on the right shoulder is not used in this work.

The maps are considered as the central resource. They are built by one or more cameras performing tracking and mapping. During this phase, the wearer can explore as freely as necessary to build up a map of the size and detail required. Although the maps are usually non-overlapping sites of special interest, they can overlap and no attempt is made to merge structural information. This can cause uncertain behaviour when relocalizing, but with care overlapping maps can be used effectively. Examples of both these behaviours are shown in the experiments.

Although 3D graphics can be added to the maps in a post-processing phase, PTAMM also allows them to be inserted into the map while it is being built, and then viewed while the map continues to be enlarged. Once built, it is likely that maps would be handed to users with access restricted in some way. For example, a map might be used read-only (as in some experiments here), or it might allow addition of further augmentations only, or the map might be made fully writeable only within its original bounds to avoid interference with other maps.

5. Results: basic operation

The following sections illustrate first the basic functionality of PTAMM and then a prototype application in a larger environment. All experiments were performed with the wearable system using either both its cameras (CamH and CamW) or, if a single camera, the hand-held CamH alone. All the results presented were processed live at 30 Hz and stored directly to video. (As noted earlier, video material is at www.robots.ox.ac.uk/ActiveVision/Publications/castle2009A.)

The experiments in this section show (i) how multiple cameras can add to the same map, (ii) how map switching works in practice, (iii) how overlapping maps can be exploited, and (iv) explore the potential for building maps in a wide variety of locations.

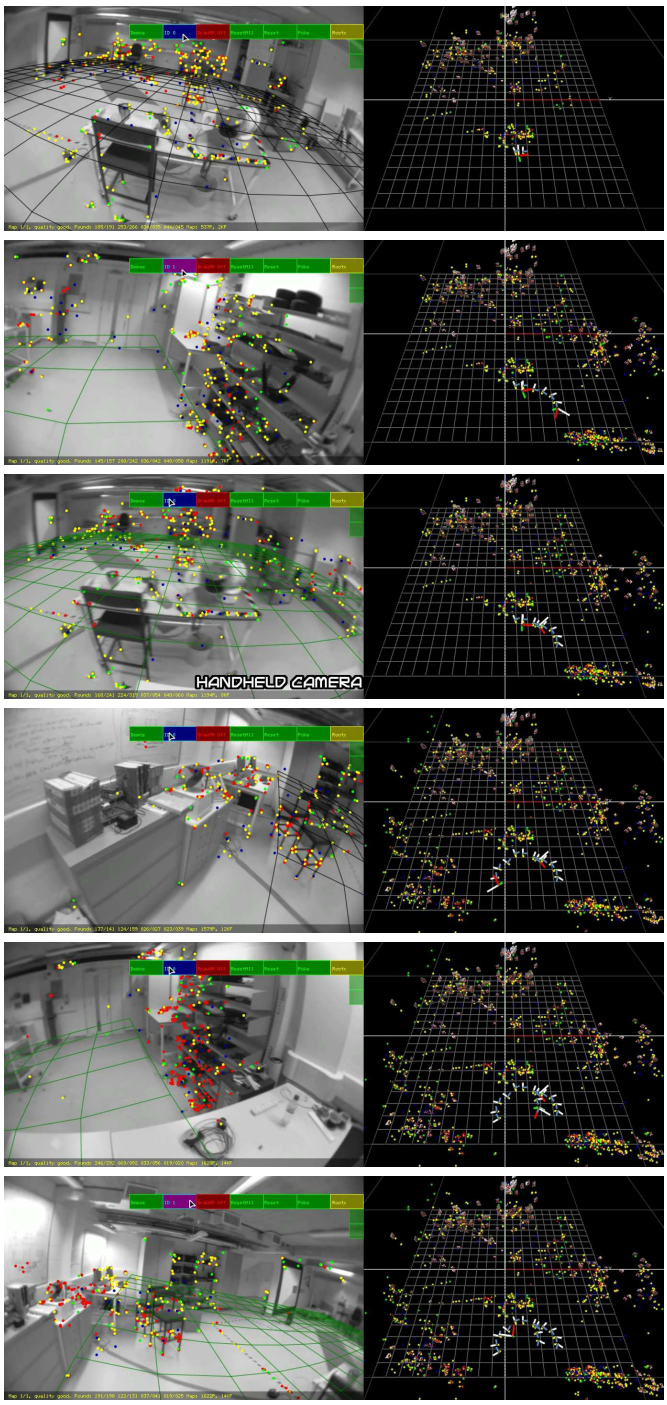


Figure 5: The wearable shoulder mounted camera (CamW) and hand-held camera (CamH) building and using the same map. (a) The map initialized by CamW. CamH then localizes in the map and (b) adds keyframes to the map. CamW is still (c) tracking, and (d) adding keyframes to map. (e) CamW tracking in the area mapped by CamH, and (f) *vice versa*.

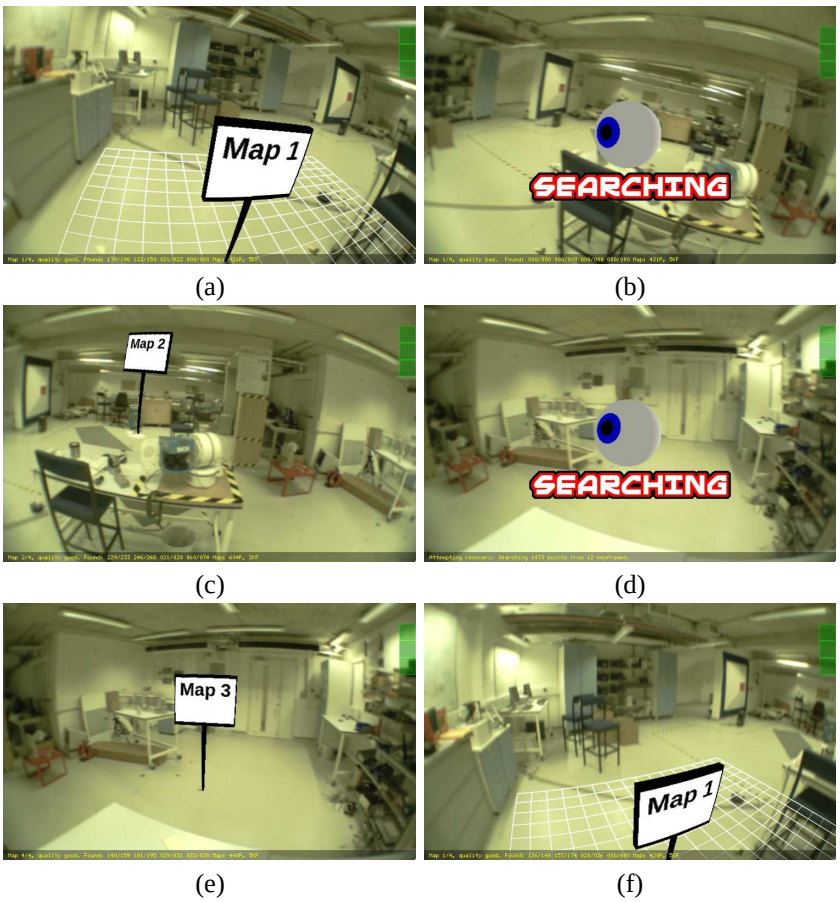


Figure 6: A single camera switching between several maps with (a, c, e, f) periods of tracking in maps 1, 2, 3, and 1, respectively, interspersed with periods (b, d) of being lost and attempting to relocalize between maps.

5.1. Multiple cameras

Fig. 5 shows frames from a sequence where the two cameras together build and use a single map. The map is initialized by the wearable camera CamW, which begins to track and supply keyframes to the map maker. Once the map has been established, the hand-held camera CamH is allowed to (re)localize itself within it, and it too tracks and adds keyframes to the map simultaneously with CamW. Either camera can track through areas mapped by the other, and add keyframes as and when they are required.

5.2. Multiple maps

Fig. 6 illustrates map switching behaviour with the hand-held camera. Three maps were created, augmented with signposts, and set in read-only mode so no further map points or keyframes could be added. When the camera leaves the boundary of map 1

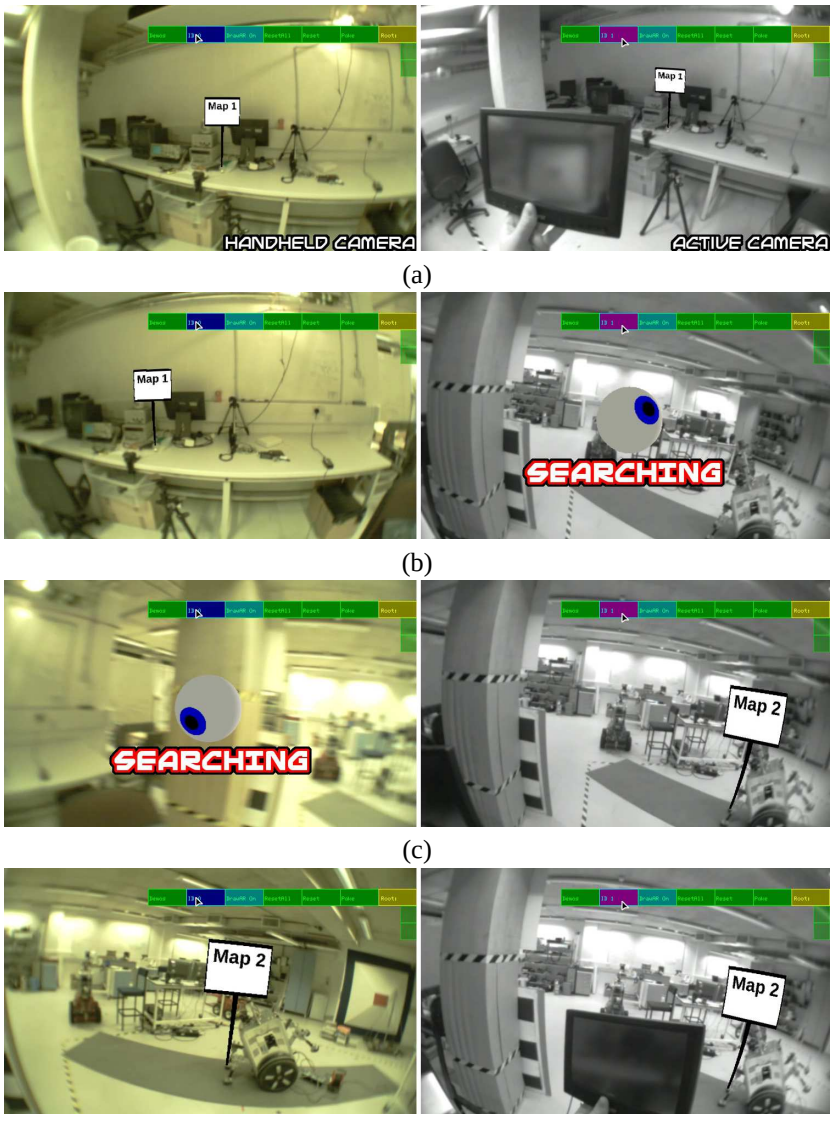


Figure 7: The two associated cameras on the wearable switching between two maps. (a) Both cameras in map 1. (b) CamH remains in map 1, while CamW looks at map 2 and is lost. (c) CamH now points at map 2 and becomes lost, and CamW relocalizes in map 2. (d) As CamH is looking at map 2, it is now able to relocalize.

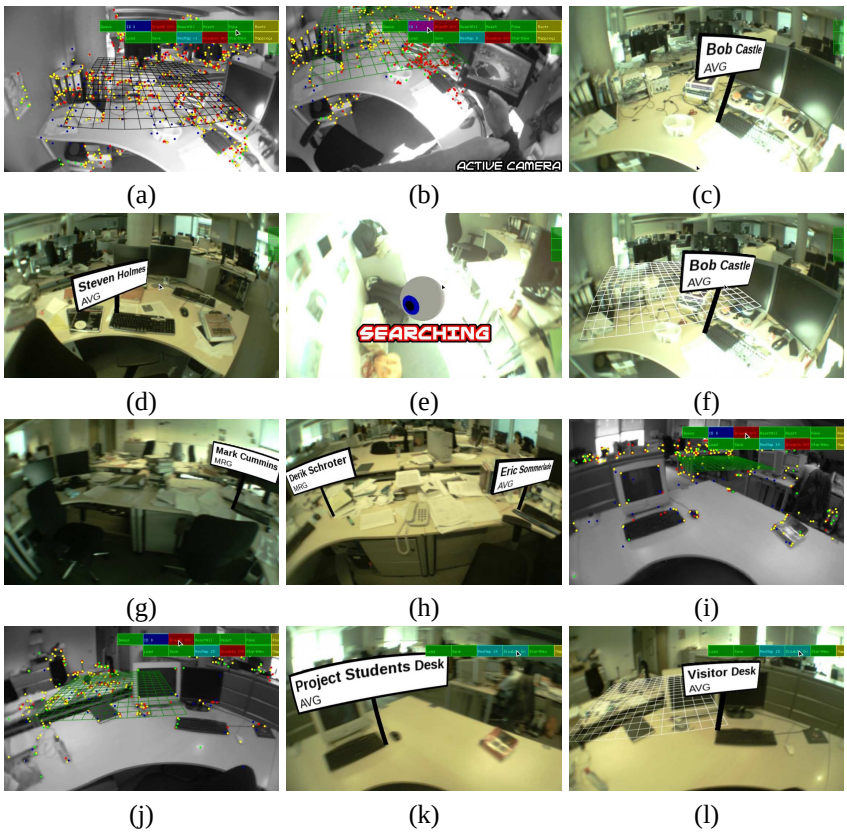


Figure 8: A demonstration both of multiple maps and robustness to self similarity of maps. 12 maps were made of 15 desks, and each desk was augmented with the user's name and research group. (a,b) Hand-held camera and active camera view working in the same map, and (c) AR added to map. (d) Another map created and labelled. (e) Attempting relocalization, and (f–h) successful relocalization on different maps. (i–l) Creation of maps on two sparsely featured desks, and subsequent successful relocalization.

it becomes lost and attempts to relocalize (6b). Experience indicates that displaying an active icon when lost (here were use a rolling eyeball) encourages the user to engage in the search process. At (c) it enters map 2 and relocalizes and tracks again; and so on to map 3. The order in which the camera enters maps is unimportant, as no prior assumption is made as to the location of the lost camera — all maps are searched.

Fig. 7 shows an example of switching using two associated cameras which are, recall, constrained to work in the same map. When the second camera (here the wearable CamW) becomes lost, it can recover only into the the current map. This is the case even if it is observing a different map (as in Fig. 7b), and so in this contrived example it remains lost. When both cameras become lost, the first to relocalize causes the other to abandon its global search and attempt to recover to the same map.

The experiment of Fig. 8 shows how the structural detail in the maps distinguishes areas of broadly similar appearance. Twelve maps were made from 15 desks in the

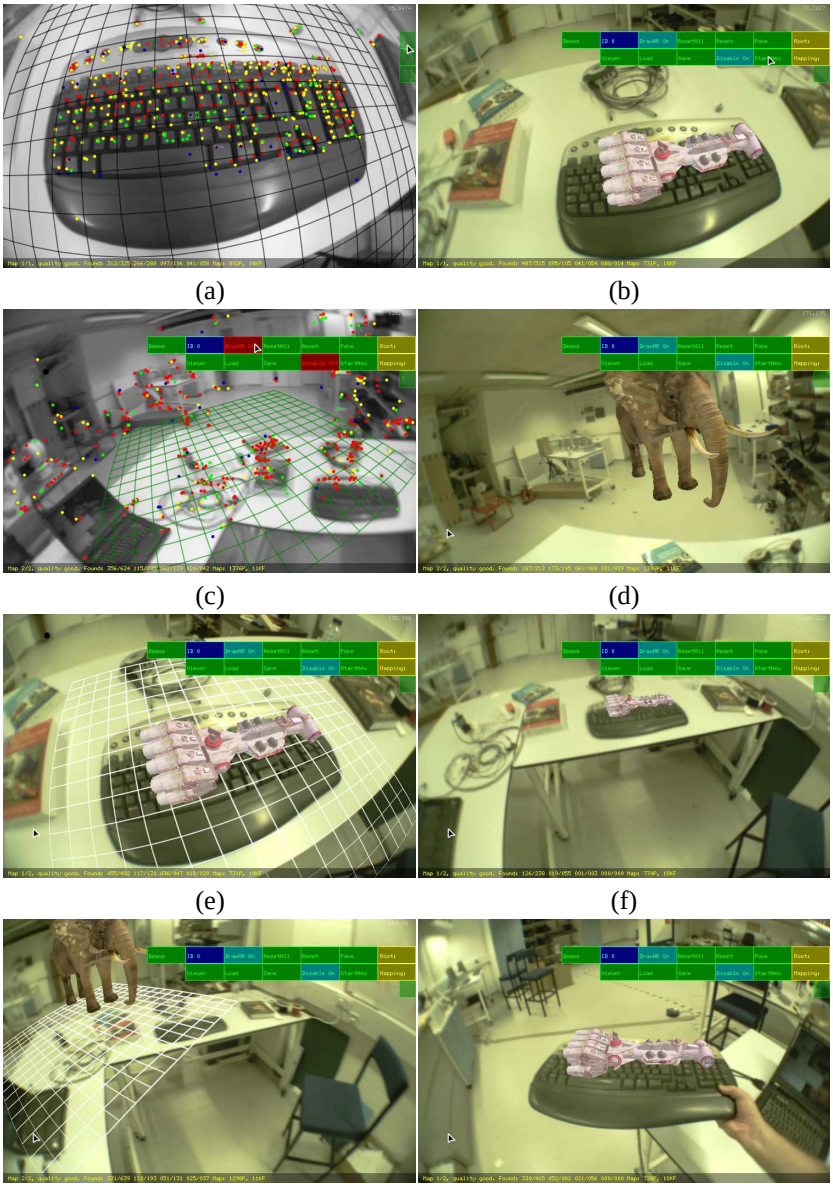


Figure 9: (a,b) A fine scale map is created on a keyboard and augmented with an arbitrary graphic. (c,d) Coarse scale map of room created and augmented. (e,f) The camera relocalizes onto the keyboard map and tracks it over a substantial scale change. (g) When tracking breaks, the camera recovers to the room map. (h) Maps built on objects move with the object.

laboratory. Each desk is similar (curved shape, with a computer), but is differentiated by the various clutter left by its occupant. In Fig. 8(a-c) the user created the first map and added a 3D graphic to show to whom the desk belongs, and similarly for the remainder. The system was able successfully to relocalize onto each of the mapped desks. The desks in (i) and (j) are most similar, but PTAMM is able to distinguish the two correctly as shown in (k) and (l).

The desk maps contained between 3 and 37 keyframes and 290 to 1 900 map points — a total of 177 keyframes and 12 327 map points. As a comparison, single-camera and single-map PTAM was used to create one large map of broadly the same region, with 163 keyframes and 10 144 map points. However, PTAM has no occlusion reasoning, and 3D map points that lay behind various objects in the room (pillars, monitors, *etc.*) were projected into the image as potentially visible, but were consequently unmatched. Although tracking continued, it was deemed poor, preventing new views being tagged as keyframes, and bringing further building of the large map to a halt.

5.3. Overlapping maps and mobile maps

Because there are no explicit links between maps there is no way to detect overlaps. If the two maps are of similar scale they are likely to have similar keyframes and when a camera relocalizes it may or may not be into the correct map. Although a remedy is required for this, overlapping maps at different scales can be used to good effect, allowing detailed maps to move around inside other maps. In the example shown in Fig. 9, a fine scale map is built upon a computer keyboard, and a second coarse scale map is built for the surrounding room. The keyboard exists in both maps, but is not observed in any detail in the room map. When close to the keyboard, there is insufficient detail to track the room and the camera relocalizes onto the keyboard (Fig. 9e) which it tracks until the distance is such that few features are observed on it, at which point the camera relocalizes into the room's map (Fig. 9g).

Because there is no geometrical linkage between maps, the map on the keyboard can move around with the object and could coexist with another map (Fig. 9h).

5.4. More varied locations

Fig. 10 shows frames cut from a sequence in which the user revisits different locations inside and outside a building. The purpose of the experiment is to demonstrate that the method can cope with a variety of everyday locations and lighting conditions, and that scenes do not have to be specially prepared. As before, maps were constructed using PTAMM and augmented with arbitrary graphics additions (spacecraft, Statue of Liberty, fighting machine, sofa, *etc*), then set to be read-only. Once created the user was able to explore the building in an unstructured manner, with maps and artefacts recovered as the user entered a map.

6. Results: larger scale operation

6.1. Human computer interaction

This extended experiment involved 45 minutes of continuous use of the system to build maps and add augmentations for an AR tour of displays in the Oxford University

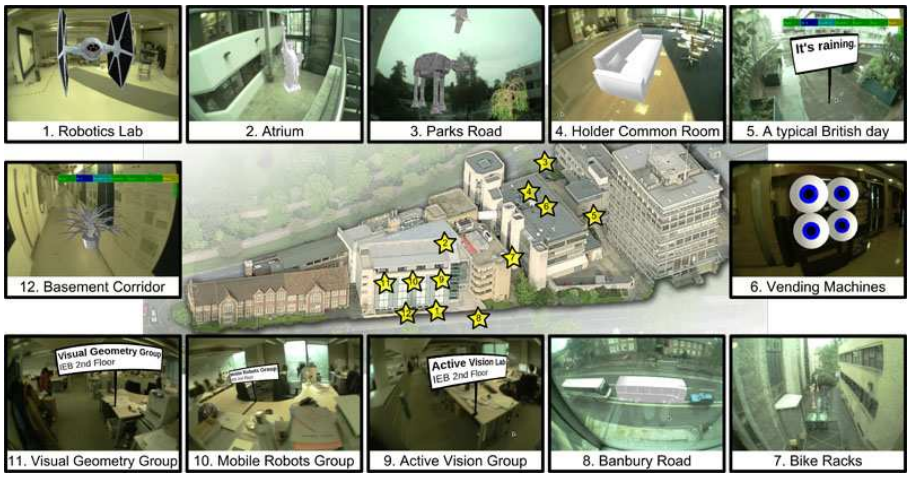


Figure 10: Multiple maps built at several locations within and outside a building.

Museum of Natural History, followed by 20 minutes walking around using the maps. The experiment's aim was to highlight those parts that are robust to use outside the laboratory. Results were obtained using the single camera on the back of the hand-held display. Both map building and AR usage (Fig. 11) took place during regular opening hours, and we note that neither phase of the operation caused undue concern to, or interest from, other museum visitors.

Perhaps the separate exhibits within the museum fit too well the pattern of "islands of interest" to be a stern test of the desirability such of separate maps. However, one advantage of the approach not apparent in earlier work is that the task of making each separate map is a conveniently bounded exercise. First, at no time was the builder concerned whether all parts of the museum had been mapped. Secondly, small maps make it easier to repeat if the mapping is interrupted and the real-time performance allow immediate review of the quality of the resulting map.

Fig. 12(a) shows the map and keyframes created while walking away from the museum's entrance. They demonstrate (as did Fig. 9(a) and (f)), PTAM's ability to maintain track over large changes in scale. The the strong horizontals in the building's facade are evident in the 3D map points, as is the archway at the entrance. Two dinosaur models were placed either side of the arch (Fig. 12b), and are still stably located when some 50 m distant at (c).

Now inside the museum, Fig. 13(a) shows the 3D points and keyframes in the map built around the skeletons of two elephants. Panels (b) and (c) show the raw video and with the 3D augmentations of the complete animals overlaid. Well-placed augmentations can look very convincing, and this example in particular shows how AR could enhance an exhibit, filling in spatial relationships that are not obvious from the skeletons alone. The process can be envisages in reverse, with the skeleton augmenting the complete animal.

We note two items relevant to the interface design for placement of models. First

(a) (b)

Figure 11: The AR system used to create maps and augment the museum exhibits: (a) elephants and (b) stuffed wallaby *etc*.

(a) (b) (c)

Figure 12: (a) The map and keyframes constructed in front of the museum's entrance. (b) Two dinosaur models are set down, and remain stably located as the user moves away by some 50 m.

it is difficult to place models into the scene by eye and hand, and instead we use correspondence between 3D points in the map and on the model. While this is sufficient for simple signage and 3D models that are not closely related to the scene structure, for the placement of objects like the elephants it would be beneficial to have models that could be warped by other than a mere 3D similarity transformations and which might, with suitable user guidance, mould themselves to selected map points.

Fig. 14(a) shows the map created around a table exhibit of stuffed animals. The keyframes are close to the table, but the structure of the cabinets to either side is captured too. A graphic of a tree has been added in (b). While this appears satisfactory in (b), subfigure (c) reminds us that while pixels in the augmented graphics are depth ordered, those in live video feed are not and will always be occluded by the AR. The underlying pointilliste map of corner features is unlikely to provide sufficient depth coverage to remedy this, and the use of dense stereo would seem the obvious method of solving this.

6.2. PTAM

The extended experiment also provided examples of shortcomings in feature matching in PTAM. For speed, this uses zero-mean sum of squared-differences between fea-

(a) (b) (c)

Figure 13: (a) The map and keyframes constructed around a pair of elephant skeletons. (b) A view of the original, and (c) with a model of the full animal overlaid.

(a) (b) (c)

Figure 14: (a) The 3D map build around a table display. The added tree graphic is properly positioned and looks convincing in (b), but the lack of depth ordering on the live video causes occlusion errors.

ture patches centred on FAST corners.

The first example appears in the failure to close a loop around one of the exhibits. Fig. 15 shows that rather similar keyframes are not overlapping. This may have arisen because the loop was traversed too quickly for a global adjustment to have run, and the successive local adjustments have allowed motion drift to accumulate. Over and above the standard reprojection and matching, PTAM does not make explicit higher-level checks for loop closure. Computing and matching stronger descriptors from the keyframes using the methods proposed and refined in [37] [38] respectively would be an approriate approach, although one that would challenge the computational budget. Using higher order features for tracking such as edges could also help with tracking robustness as demonstrated in [27].

A second difficulty occurred when tracking a highly textured region. A map was created successfully from the region of grass around that shown in Fig. 16(a), but only by moving the camera steadily. The AR dinosaur was added in Fig. 16(b), but note that the footprints are laid into the grass. However, as the camera is moved less carefully back towards the map origin, the features on the grass become mismatched and the map drifts along the grass Fig. 16(c), and eventually tracking fails. This is difficult imagery, and failure is not surprising: the point of interest here is that maps for AR should not be built with a care that will not be replicated by the user.

6.3. Relocalizer

While the use of the relocalization for map switching works well overall, the particular relocalizer used was designed to be the cheapest that would work well within

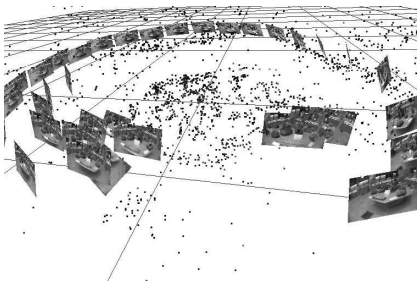


Figure 15: The accumulation of motion drift caused a failure to rematch at the end of this loop.

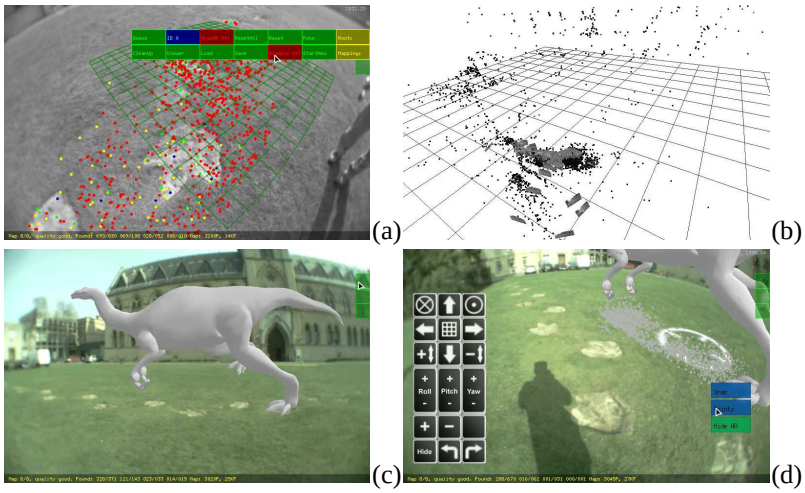


Figure 16: Map building in a challenging environment. (a) Point features on textured region. (b) The 3D map built, and (c) an AR dinosaur added. In (d) the map (and hence overlay) has become misplaced.

a single map where (i) the likelihood of perceptual aliasing in the imagery is low, and (ii) the likelihood of aliasing followed by successful tracking all but negligible.

Fig. 17 shows an example where the wrong map was chosen but where tracking succeeded, at least for a second or so, before a further tracking failure after which the correct map was chosen. In this case the maps were being used in read-only mode, but had the user been building maps the incorrectly chosen map would have been corrupted.

The museum has a repetitive internal structure of bays between columns, with each bay housing similar display cabinetry. The obvious cause of aliasing would seem to between two of these. Remarkably here, however, as Fig. 17(a) shows, the mix-up is with the map of the front of the building.

The relocalizer needs a stronger method of appearance testing, and perhaps to incorporate further geometrical tests before trying to track. Several methods were mentioned earlier, but it would be convenient to use the same method to aid relocalization as is adopted for detecting loop closure.

(a) (b) (c)

Figure 17: Relocalization and recovery failure. (a) Incorrect localization, followed by (b) incorrect tracking, followed by (c) tracking failure.

7. Conclusion

This paper has shown that the ability to build multiple modestly-sized maps within an environment is useful in wearable applications in which the wearer needs to, and can be trusted to, move around the world and between mapped regions without detailed guidance.

It has been shown that the method of parallel tracking and mapping provides the required geometric underpinning, but also conveniently splits the processing in a way that can be exploited to allow multiple cameras to build and work in multiple maps. In turn this allows much larger areas to be managed.

The resulting system, PTAMM, has been demonstrated in a range of experiments of increasing scale, showing that multiple maps allow for augmentation of individual objects and large sprawling environments. No particular impediment has been found to future upward scaling of the system as further computing resources become available.

Using PTAMM an AR tour of the natural history museum was able to be created, mapping and placing AR only at locations of interest. The system relocalized correctly for the majority of the time allowing the user to freely explore the museum as they desired. The ability to build maps and augment them in real time allowed different layouts to be experimented with providing instant feedback. Improving the user interface to have more features like those found in a CAD modelling program would improve the user's ability to place models where they desired. Using multiple maps allows unique AR to be placed on each exhibit, and if a display is moved the map would still be found on the exhibit. Also, if an exhibit is modified, only the map for that exhibit needs to be updated or redone. If a single large map had been used, any changes to the locations or content of the exhibits would mean either recapturing the whole map or carefully editing the map to remove the invalid exhibit features, and then updating the map with the new changes and reimplementing the AR. Loop closure also becomes a larger issue as map sizes increase, allowing more opportunity for measurement drift.

Acknowledgements

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

References

[1] H. C. Longuet-Higgins., A computer algorithm for reconstructing a scene from two projections., Nature 293 (1981) 133–135.

[2] D. Nistér, An efficient solution to the five-point relative pose problem, IEEE Transactions on Pattern Analysis and Machine Intelligence 26 (6) (2004) 756–777.

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

[4] A. W. Fitzgibbon, A. Zisserman, Automatic camera recovery for closed or open image sequences, in: Proc 5th European Conf on Computer Vision, Vol. 1, 1998, pp. 311–326.

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

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

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

[8] C. G. Harris, J. M. Pike, 3D positional integration from image sequences, in: Proc 3rd Alvey Vision Conference, 1987, pp. 233–236.

[9] Z. Zhang, O. Faugeras, 3D Dynamic Scene Analysis, Springer-Verlag, 1992.

[10] A. J. Davison, Real-time simultaneous localisation and mapping with a single camera, in: Proc 9th IEEE Int Conf on Computer Vision, 2003, pp. II: 1403–1410.

[11] A. J. Davison, I. D. Reid, N. D. Molton, O. Stasse, MonoSLAM: Real-time single camera SLAM, IEEE Transactions on Pattern Analysis and Machine Intelligence 26 (6) (2007) 1052–1067.

[12] A. J. Davison, W. W. Mayol, D. W. Murray, Real-time localisation and mapping with wearable active vision, in: Proc 2nd IEEE/ACM Int Symp on Mixed and Augmented Reality, 2003, pp. 18–27.

[13] D. Chekhlov, A. Gee, A. Calway, W. Mayol-Cuevas, Ninja on a plane: Automatic discovery of physical planes for augmented reality using visual SLAM, in: Proc 6th IEEE/ACM Int Symp on Mixed and Augmented Reality, 2007.

[14] W. W. Mayol, A. J. Davison, B. J. Tordoff, N. D. Molton, D. W. Murray, Interaction between hand and wearable camera in 2D and 3D environments, in: Proc 15th British Machine Vision Conference, 2004.

[15] R. O. Castle, D. J. Gawley, G. Klein, D. W. Murray, Video-rate recognition and location, in: Proc 17th British Machine Vision Conference, 2007.

[16] R. O. Castle, G. Klein, D. W. Murray, Combining monoSLAM with object recognition for scene augmentation using a wearable camera, Image and Vision Computing In Press. doi:doi:10.1016/j.imavis.2010.03.009.

[17] A. J. Davison, Vision-based SLAM in real-time, in: J. Martí (Ed.), Proc IbPRIA 2007, no. 4477 in LNCS, Springer-Verlag Berlin Heidelberg, 2007, pp. 9–12.

[18] J. J. Leonard, H. J. S. Feder, A computationally efficient method for large-scale concurrent mapping and localization, in: Proc 9th Int Symp on Robotics Research, 1999, pp. 316–321.

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

[20] P. M. Newman, J. J. Leonard, Consistent, convergent and constant-time SLAM, in: Int Joint Conference on Artificial Intelligence, Morgan Kaufmann, 2003.

[21] L. A. Clemente, A. J. Davison, I. D. Reid, J. Neira, J. D. Tardós, Increasing the size of maps mapping large loops with a single hand-held camera, in: Proc Robotics: Science and Systems Conference, 2007.

[22] M. Isard, A. Blake, Contour tracking by stochastic propagation of conditional density, in: Proc 4th European Conf on Computer Vision, 1996, pp. 343–356.

[23] M. Pupilli, A. Calway, Real-time camera tracking will resilience to erratic motion, in: Proc 24th IEEE Conf on Computer Vision and Pattern Recognition, 2006.

[24] M. Montemerlo, S. Thrun, D. Koller, B. Wegbreit, Fastslam 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges, in: Int Joint Conference on Artificial Intelligence, 2003.

[25] E. Eade, T. Drummond, Monocular SLAM as a graph of coalesced observations, in: Proc 11th IEEE Int Conf on Computer Vision, 2007.

[26] G. Klein, D. W. Murray, Parallel tracking and mapping for small AR workspaces, in: Proc 6th IEEE/ACM Int Symp on Mixed and Augmented Reality, 2007.

[27] G. Klein, D. W. Murray, Improving the agility of keyframe-based SLAM, in: Proc 10th European Conf on Computer Vision, 2008.

[28] R. O. Castle, G. Klein, D. W. Murray, Video-rate localization in multiple maps for wearable augmented reality, in: Proc 12th IEEE Int Symp on Wearable Computers, 2008.

[29] E. Rosten, T. Drummond, Machine learning for high-speed corner detection, in: Proc 9th European Conf on Computer Vision, 2006.

[30] E. Rosten, R. Porter, T. Drummond, Faster and better: a machine learning approach to corner detection, IEEE Transactions on Pattern Analysis and Machine Intelligence 32 (1) (2010) 105–119.

[31] P. J. Huber, Robust Statistics, John Willey and Sons, 1981.

[32] Z. Y. Zhang, Parameter estimation techniques – a tutorial with application to conic fitting, Image and Vision Computing 15 (1) (1997) 59–76.

[33] C. Engels, H. Stewénius, D. Nistér, Bundle adjustment rules, in: Proc Symposium on Photogrammetric Computer Vision, Bonn, Germany, Sep 20-22, 2006.

[34] E. Mouragnon, M. Lhuillier, M. Dhome, F. Dekeyser, P. Sayd, Real time localisation and 3d reconstruction, in: Proc 24th IEEE Conf on Computer Vision and Pattern Recognition, 2006, pp. I: 363–370.

[35] D. W. Marquardt, An algorithm for the least-squares estimation of non-linear parameters, J Soc Indust Appl Math 11 (2) (1963) 431–441.

[36] J. W. Tukey, Exploratory Data Analysis, Addison-Wesley, Reading MA, 1977.

[37] P. M. Newman, K. Ho, SLAM-loop closing with visually salient features, in: Proc 2005 IEEE Int Conf on Robotics and Automation, 2005, pp. 644–651.

[38] M. Cummins, P. Newman, Highly scalable appearance-only SLAM - FAB-MAP 2.0, in: Proc Robotics: Science and Systems Conference, 2009.

[39] B. Williams, G. Klein, I. D. Reid, Real-time SLAM relocalisation, in: Proc 11th IEEE Int Conf on Computer Vision, 2007.

[40] E. Eade, T. Drummond, Unified loop closing and recovery for real time monocular SLAM, in: Proc 18th British Machine Vision Conference, 2008.

[41] A. Irschara, C. Zach, J. Frahm, H. Bischof, From structure-from-motion point clouds to fast location recognition, in: Proc 27th IEEE Conf on Computer Vision and Pattern Recognition, 2009.

[42] S. Benhimane, E. Malis, Homography-based 2D visual tracking and servoing, International Journal of Robotics Research 26 (7) (2007) 661–676.