Automatic Relocalisation for a Single-Camera Simultaneous Localisation and Mapping System

Brian Williams, Paul Smith and Ian Reid
Department of Engineering Science, University of Oxford, UK
{bpw,pas,ian}@robots.ox.ac.uk

***Abstract*— We describe a fast method to relocalise a monocular visual SLAM (Simultaneous Localisation and Mapping) system after tracking failure. The monocular SLAM system stores the 3D locations of visual landmarks, together with a local image patch. When the system becomes lost, candidate matches are obtained using correlation, then the pose of the camera is solved via an efficient implementation of RANSAC using a three-point-pose algorithm. We demonstrate the usefulness of this method within visual SLAM: (i) we show tracking can reliably resume after tracking failure due to occlusions, motion blur or unmodelled rapid motions; (ii) we show how the method can be used as an adjunct for a proposal distribution in a particle filter framework; (iii) during successful tracking we use idle cycles to test if the current map overlaps with a previously-built map, and we provide a solution to aligning the two maps by splicing the camera trajectories in a consistent and optimal way.**

I. INTRODUCTION

This paper is concerned with the problem of real-time localisation and mapping using only a single camera. We are particularly concerned with the issue of robustness to such real-world tracking conditions as occlusions or fast motion, and to this end we provide a solution to re-localisation of the camera when lost, and evaluate its use in different scenarios.

Though others [1], [2], [3] have investigated recursive structure from motion using Extended Kalman Filtering (or variants thereof), Davison [4] demonstrated the first real-time version of this. Like the prior art, Davison used the Extended Kalman Filter (EKF) to estimate the pose of the camera while creating a sparse map of point features. Davison's system offers good performance when tracking, but becomes lost during fast camera motions or occlusions, when a lack of measurements of visual landmarks causes the camera pose uncertainty to grow rapidly.

Pupilli and Calway [5] address the problem of robustness to sudden motion and occlusions in the context of camera localisation (though they do not have a rigorous approach to mapping, so it not truly a 'SLAM' system). They achieve a degree of robustness using the now standard technique of filtering the state – in this case the camera pose – using a set of particles to represent the possibly multimodal distribution of the camera location. Their system is able to cope with significant amounts of occlusion and camera shake by the (random) scattering of particles that occurs naturally as camera uncertainty increases. As soon as one of the particles lands at a location sufficiently close to the true pose, it receives high weight and the resampling process concentrates the particles nearby.

A more rigorous approach to single-camera SLAM using particles was presented by Eade and Drummond [6], who implement the FastSLAM 2.0 algorithm [7] for a bearing-only sensor. The benefit of the FastSLAM framework arises via the Rao-Blackwellization [8] of the filter, which reduces the computational complexity of filter updates to $O(N)$ (from $O(N^2)$ in the case of the EKF). However, the system remains prone to tracking failure during rapid motions.

In this paper we make use of the fact that a SLAM system, by its very nature, builds a map of the environment. Given the three-dimensional points, and their corresponding image projections, a well-known result [9] states that the pose of the camera can be recovered (there may be as many as four solutions for the pose, but importantly this is a finite number, not a parametric family). We thus hypothesize matches between map features and features found in an image in order to localise the camera in a bottom-up fashion. Note that in contrast to image-to-image matching such as used in Nistèr's [10] system, where five point correspondences are sought, a map-to-image match requires only three.

Though this is unremarkable in itself, we demonstrate in this paper how by using an efficient version of RANSAC [11] we can achieve reliable, near-real-time localisation. As well as the straightforward addition of automatic recovery to Davison's EKF method, we also show the utility of this bottom-up approach in the context of particle filtering. We further demonstrate how the pose recovery module can greatly assist with the problem of map alignment in single-camera SLAM. When sudden motion causes the camera view to move to an unmapped location, our single camera SLAM system automatically begins a new map. However it is important to determine the position of this new map relative to previously-mapped regions as soon as possible. By detecting when the system has returned to a previously mapped region using the pose recovery algorithm (running in idle cycles during normal tracking), we show how tracking the trajectory relative to both the current and the previous map leads to a map alignment method well-suited to single-camera SLAM.

The remainder of the paper is organised as follows: First, we briefly describe Davison's EKF-based single-camera SLAM system. Then, the recovery module is outlined, and results are given on its usefulness for both EKF- and particle-filter-based systems. Finally, our method of map alignment for a vision-based SLAM system which makes use of the recovery module is shown.

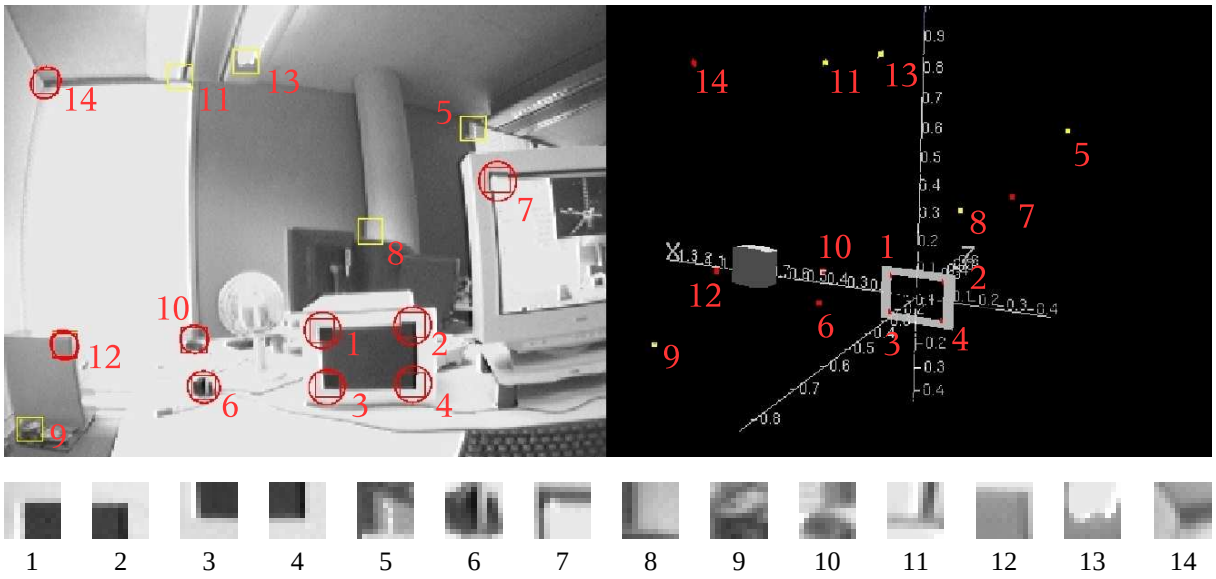


Fig. 1. The EKF-based SLAM system of Davison [4] creates a map of current features when tracking. The camera image is shown on the left, and the three-dimensional map and camera location on the right. The image patches identifying each feature are shown and numbered below, corresponding with the numbers in the image and map view.

II. EKF-BASED MONOSLAM

The automatic recovery module described in this paper was implemented as an extension to the EKF-based SLAM system developed by Davison [4], but the approach is general and can be used for any vision-based system that uses point features. We will outline Davison's system to give a better understanding of single-camera SLAM before the recovery module is explained.

The pose of the camera, $\hat{\boldsymbol{x}}_p$, is represented by a position vector, $\boldsymbol{r}$, and a quaternion, $\boldsymbol{q}$:

$$\hat{\boldsymbol{x}}_p = \begin{pmatrix} \boldsymbol{r} \\ \boldsymbol{q} \end{pmatrix} = \begin{pmatrix} x & y & z & q_0 & q_x & q_y & q_z \end{pmatrix}^T,$$

and point features are represented by a 3D position vector:

$$\hat{\boldsymbol{y}}_i = \begin{pmatrix} x & y & z \end{pmatrix}^T.$$

The total state estimate is represented as a multivariate Gaussian with mean

$$\hat{\boldsymbol{x}} = \begin{pmatrix} \hat{\boldsymbol{x}}_v & \hat{\boldsymbol{y}}_1 & \hat{\boldsymbol{y}}_2 & \dots \end{pmatrix}^T, \tag{1}$$

and covariance

$$\mathsf{P} = \begin{pmatrix} \mathsf{P}_{xx} & \mathsf{P}_{xy_1} & \mathsf{P}_{xy_2} & \dots \\ \mathsf{P}_{y_1x} & \mathsf{P}_{y_1y_1} & \mathsf{P}_{y_1y_2} & \dots \\ \mathsf{P}_{y_2x} & \mathsf{P}_{y_2y_1} & \mathsf{P}_{y_2y_2} & \dots \\ \vdots & \vdots & \vdots & \ddots \end{pmatrix}. \tag{2}$$

When each new frame arrives, predictions are made using the pose estimate to determine a 3σ search ellipse in the image for each feature; Davison calls this process 'Active Search'. Each feature stores an 11×11 image patch as a descriptor, and correlation is performed between this patch and the pixels in the search region to determine the actual feature measurement in the image. The difference between these observed feature locations and their predicted positions is used by the EKF to update both the camera pose estimate and the map. Fig. 1 shows Davison's system tracking. The three-dimensional camera pose estimate and the location of the features in the map are shown on the right, the patch for each feature is shown below, and the search regions for the observed features in the image can be seen on the left.

Using Active Search is important to allow the system to run in real time, because the cost of correlation across the whole image is too high. Of course, the observations will fail if the camera becomes occluded or the motion is too fast (either causing motion blur or failing to conform to the 'constant velocity' motion model). With no measurements, the pose of the camera becomes more uncertain, leading to larger search regions. In theory, the search region should still contain the feature, and those should later be matched again and the system then recovers. In practice, this does not happen: frequently, the system will find a patch in the growing search region which looks similar enough to the true feature but is not in fact correct. If this incorrect observation is used, the map will be irretrievably corrupted. Therefore, the EKF system cannot be trusted to recover on its own.

This motivates the use of our separate recovery module. When the system has no successful observations, the recovery algorithm is run, which will keep trying to determine the camera pose using the map created so far. When the occlusion or fast motions have ceased, the pose is determined, and the EKF is reset without the danger of making incorrect observations. In robotics, this problem (global localisation) has been investigated before, but usually only in 2D using range sensors e.g. [12].

III. FINDING THE CAMERA POSE

The recovery module generates candidate poses from hypothesised matches between locations in the current frame and the features in the map. A set of three potential matches

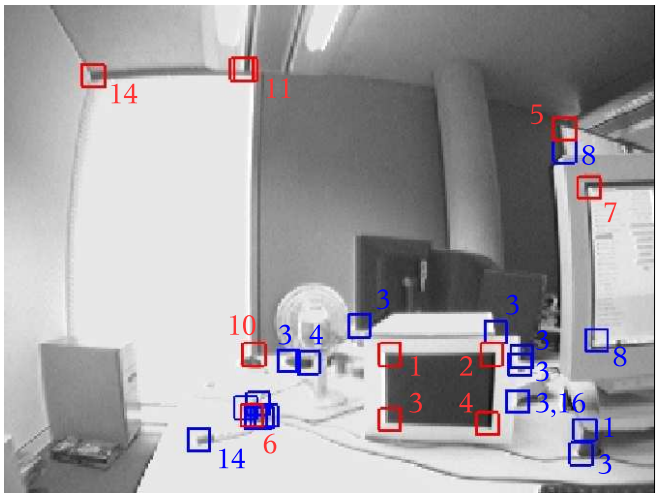


Fig. 2. Potential matches in the recovery image for the features from the map of Fig. 1. Correlation is performed between the patch for each map feature and the 'corners' detected in this image. Correct matches (red) were found for many of the features, but there were also many false detections (blue) and failed detections, as well as multiple matches for some features.

is selected, and all the camera poses consistent with this set are calculated using the three-point-pose algorithm. These poses are evaluated by looking for consensus amongst the other matches found in the image. If a pose with a large consensus is found, that pose is assumed to be correct.

A. Finding Potential Matches

The map features in Davison's system are initialised at regions in the image where the response to the Shi-Tomasi [13] corner detector is strong. Therefore, to find matches to these features in the image, our system proceeds as follows. First, the Shi-Tomasi detector is run across the whole image. Then, an exhaustive correlation is performed between each of these corner points and the image patches for the entire map. This correlation is performed not only on the corner point, but also within ± 2 pixels since the precise location of the detected corner point is to some extent view-dependent.

A number of potential matches to map features are found in the image, as shown in Fig. 2. Note that not all the matches are correct (e.g. feature 3 is matched to several image locations). Also, matches were not found for some of the map features present in the image (e.g. features 8, 9, 12 and 13) due to the Shi-Tomasi corner not being detected.

B. Selecting Sets of Matches

Sets of three matches are chosen randomly using only two constraints:

1) No two matches can come from the same map feature.
2) No two matches can come from the same corner point in the image.

This random selection was found to be sufficient for the size of maps we considered, but as map size increases, the number of combinations of three matches rapidly increases. More intelligent ways to select sets among a large number of possibilities (for example co-visibility and correlation score) were explored, as will be discussed in section V-C.

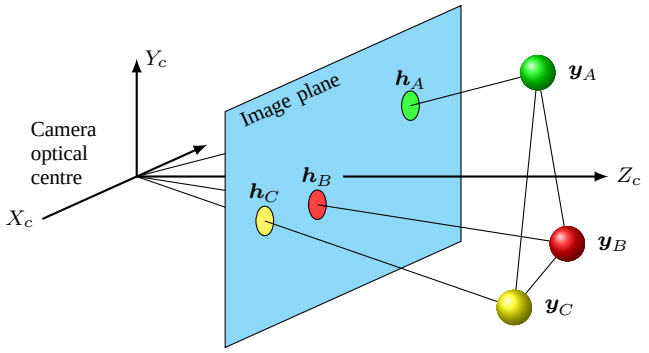


Fig. 3. The three-point-pose algorithm [9] finds up to four valid poses for the camera given the position in 3D ($\boldsymbol{y_A}$, $\boldsymbol{y_B}$ and $\boldsymbol{y_C}$) of three points and their corresponding projections in the image ($\boldsymbol{h_A}$, $\boldsymbol{h_B}$ and $\boldsymbol{h_C}$).

C. The Three-Point-Pose Algorithm

The three-point-pose algorithm is used to determine the camera pose indicated by each set of matches. Because the problem is underconstrained, up to four poses for the camera are valid given the known arrangement of three points in 3D ($\boldsymbol{y}_A$, $\boldsymbol{y}_B$ and $\boldsymbol{y}_C$) and their image projections ($\boldsymbol{h}_A$, $\boldsymbol{h}_B$ and $\boldsymbol{h}_C$). Several methods exist [9], of which we have selected the method of Fischler and Bolles [14] for its numerical stability. The algorithm involves solving a fourth-degree polynomial which is found by rearranging the simultaneous equations specified by the geometry shown in Fig. 3. The triangle linking the points is known from the map geometry, and the angles to the points are known from the image projections. The four-fold ambiguity in pose is resolved by evaluating all the potential poses.

D. Evaluating Potential Poses

For each candidate pose, the image projections of the remaining map features are calculated. If this pose is correct, the projections will be at the true locations in the image for those features, and it is likely that a match will have been found for that feature at that point (Fig. 4). By checking how many matches are found at the locations predicted by the pose, a consensus score is given to each pose. In our experience to date, only rarely is there more than one successful prediction and so we take a pose to be correct if it successfully predicts two or more features. Of course, this fails in the case where the map contains regions with similar appearence and geometry. Though the photometric information inherent in our features renders them more distinctive than, say, geometric data such as laser scans – reducing the problem of multiple solutions to the point where we have not regularly encountered this – we recognise that in larger environments this is inevitable. One possible solution – not implemented – would be to use multiple separate EKFs with each hypothesis until the ambiguity is resolved. Alternatively, particle filters provide a natural mechanism for the maintenance of many hypotheses. These remain for future work.

To speed up the evaluation, the $T_{d,d}$ test developed by Matas [11] is used. When the sets of three matches are

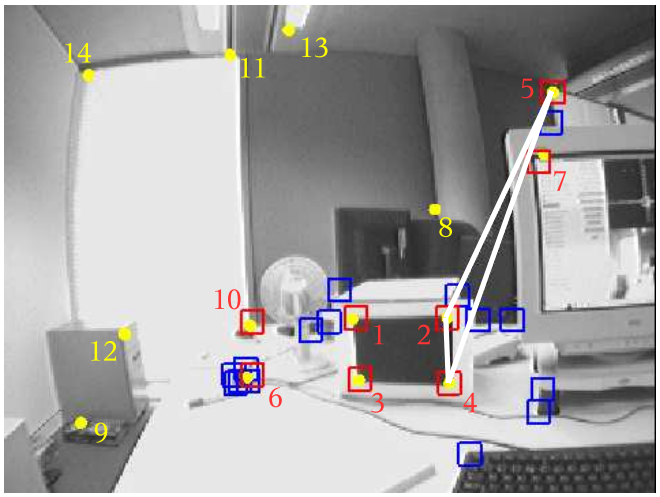


Fig. 4. The evaluation of one of the poses given by the three-point-pose algorithm for the three matches forming the white triangle (features 2, 4, and 5). The projections for the other map features according to this pose are shown as yellow circles. The pose finds matches nearby for five of the projections (features 1, 3, 6, 7, 10), and so is considered to be the correct pose.

chosen, a fourth feature is also randomly selected as an evaluation feature. This feature is the first to be projected to evaluate the pose. If this prediction does not find a match then that pose is abandoned. It is possible that good poses are thrown away, but because of the great speed-up this test gives, many more poses are checked.

A time limit is also set for the algorithm. If the correct pose is not found within this time limit then the algorithm gives up: a new frame is taken from the camera and the algorithm is run again. This ensures that, in cases when the camera is occluded for a short period of time, new frames are periodically tried. A time limit of 200ms was chosen which was felt to be enough time to try a reasonable number of poses. If the time limit were much higher, the pose could be too far out of date when found.

IV. UPDATING THE SLAM SYSTEM

Having recovered the camera pose, for continued correct operation of the EKF, the uncertainty in this pose and the coupling to the map must be correctly modelled. The features that were not used in the three-point-pose algorithm are uncorrelated to the new camera pose so the corresponding covariance terms, P_{xy}, are set to zero. The uncertainty in the camera pose and the covariance terms which couple this to the three features could be determined by first order perturbation. Likewise this estimate could be improved via a batch estimation using all the matches. Though straightforward, such a procedure essentially replicates the machinery of the EKF itself up to the linearisation errors in the EKF. Hence, a correct and expedient solution is obtained by feeding the three image measurements into the Kalman filter, having first set the camera pose uncertainty sufficiently large that it can be treated essentially as uninformative. Note that because the innovation will be identically zero for the new pose, this will not result in any alteration to the pose estimate.

Fig. 5. To test the recovery module, sudden motion was simulated by removing the intermediate frames as the camera panned left from the image on the left to that on the right.

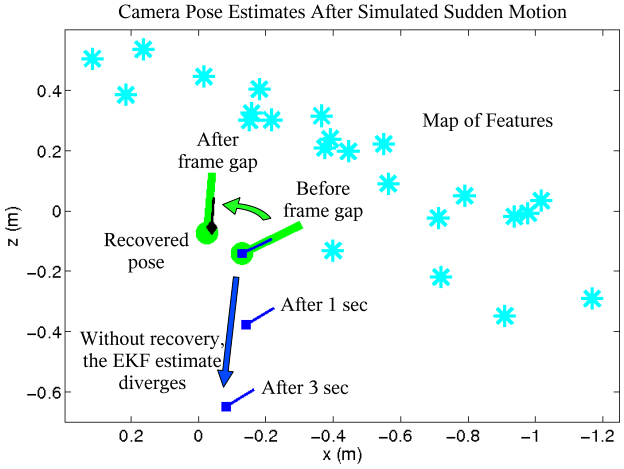


Fig. 6. The pose estimates for the ordinary EKF (square) and the EKF with the recovery module (diamond) are shown after the simulated sudden motion in Fig. 5. The 'ground truth' (circle) is found by running the EKF without removing the intermediate frames. The recovery module immediately detects the failure of observations and suggest a pose close enough to the true pose that the EKF can correct it and continue. Without the recovery module, the EKF assumes it still at the old pose, and makes incorrect observations leading to divergence.

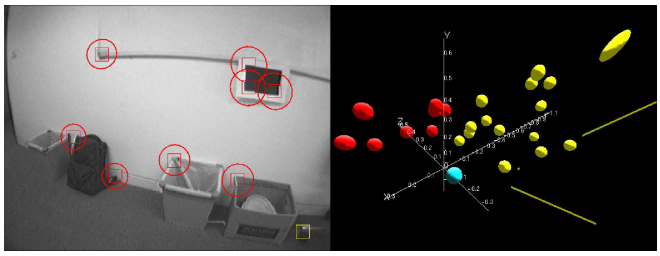
V. POSE RECOVERY RESULTS

A. Pose Recovery for the Extended Kalman Filter

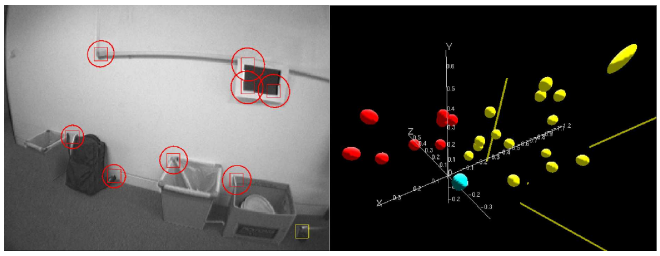
To test the recovery, module Davison's system [4] was run on an image sequence to build a map. At one point in the sequence, the camera pans left so that it observes a previously-mapped area. Sudden movement was simulated by removing the intermediate frames (Fig. 5). Fig. 6 shows the robustness to this sudden movement for the EKF system with (diamond) and without (square) the recovery module. An estimated ground truth (circle) is given by running the EKF system without removing the intermediate frames.

Unsurprisingly, because it has no measurements, the EKF fails catastrophically. In more detail, what actually happens is that, in the absence of observations, the EKF continues to predict the camera location with growing uncertainty. However, this prediction is sufficiently far from the true value that either no further measurements are ever possible or erroneous matches are acquired. The result in either case is rapid divergence.

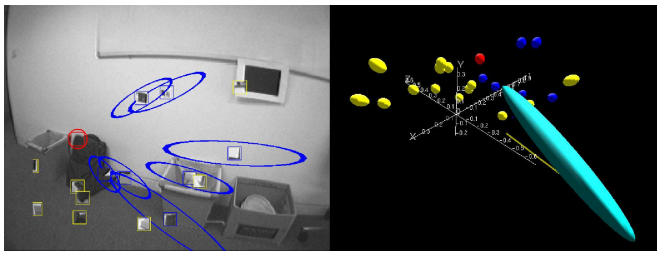
In contrast, when using the recovery module, as soon as no measurements are available, recovery is initiated. When restarted with the recovered pose, the EKF converges quickly



(a) 'Ground truth': Pose and feature observations for the EKF system when it has not undergone the simulated sudden motion.



(b) With recovery: The system has relocalised after the sudden motion, with a pose very close to the 'ground truth'.



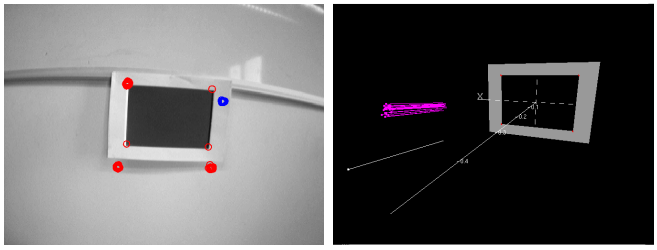
(c) Without recovery: The EKF fails after sudden motion. Many of the attempted observations have failed (blue), and the one 'successful' observation is where it has mistaken the corner of the office chair for the black bag (poor data association). The system believes it is still facing the old direction even though almost no features match.

Fig. 7. Recovery for EKF SLAM. The estimates for the map and camera are shown with 3σ uncertainty ellipsoids. The colours indicate the camera (cyan), a successfully-observed feature (red), an unsuccessfully-observed feature (blue), and a feature which was not chosen for observation (yellow).

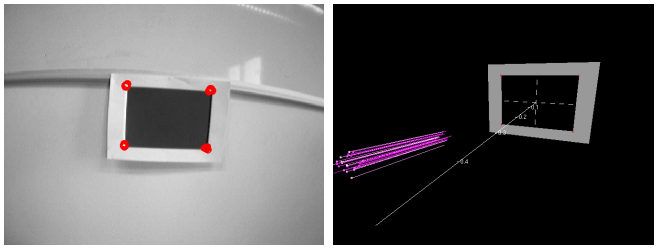
to its correct value again. The camera pose estimates after the simulated sudden movement can be seen in Fig. 7. The pose estimate for the system with the recovery module is close to the 'ground truth' estimate where the frames were not removed.

B. Pose Recovery for a Particle Filter

Data-driven localisation methods such as the one we have described also fit naturally into a particle-filter framework. At each iteration in a particle filter algorithm, particles are drawn from a proposal distribution and then reweighted using image measurements so that the particle set is a valid approximation of the true posterior. A typical proposal distribution takes into account the system's dynamical model and associated uncertainty. When the system is "lost", recovery occurs if a particle sampled from this proposal distribution coincides with the true pose. The likelihood of this is increased dramatically if the proposal distribution incorporates data driven samples in addition to those generated from the dynamical prior. This was suggested by (amongst others)



(a) The particle filter has locked onto the shadow instead of the black rectangle. A single particle is created using the pose suggested by the recovery module. The search region for each particle is drawn red if the correlation was successful and blue if it failed.



(b) Resampling causes the whole particle set to converge on the correct pose which was suggested by the recovery module.

Fig. 8. Recovery for Particle Filter

[15] and here we demonstrate our recovery module as a sample generator.

A particle filter was implemented to track the camera pose using observations of the four corners of a known black rectangle. At times, the filter lost track and would lock on to the shadow of the sheet which looked similar. A larger number of particles (we used only 25) could be used with a more noisy motion model to pull the system out of this local minimum, but instead, we used the recovery module. Periodically, the recovery module is run and a single particle representing the suggested pose is added into the particle set. In Fig. 8(a) this single particle can be seen away from the particle cluster. The predicted observations in the image for the whole particle set can be seen, indicating that the rest of the particle set has locked onto the shadow while the single particle from the recovery module has locked onto the true rectangle. After resampling, the whole particle set has a better pose estimate since the single particle, with four successful observations, received a much higher weight.

The recovery module could thus be useful for FastSLAM-based systems as well as the EKF-based one we have used. However, there is a difficulty in deciding exactly how to incorporate the pose suggestion in FastSLAM since the particles represent trajectories with a map estimate rather than just pose.

C. Accuracy, Reliability and Timing

To test the performance of the recovery module we used Davison's system to perform SLAM on a 45-second sequence while the recovery module was run at each frame without updating the SLAM state. Fig. 9 shows the total time taken to recover the pose (red bars) as the size of the map (blue line) increases. The gaps in the sequence are where

the pose was not recovered. In this sequence, the recovery algorithm succeeded within the time limit of 200ms in 84% of the frames. The cases where recovery was not possible were due to an insufficient number of correct matches being found, while due to the random nature of the system there will always be cases where a correct choice of matches are not tried within the time limit.

In the later half of the sequence, the time taken to find the pose becomes more erratic. With a larger map, more matches are found, leading to a much larger number of combinations of three matches with which to determine potential poses. Therefore, for larger maps, a method should be used which gives a higher probability of selecting a correct set of three matches. We explored weighting the selection according to co-visibility and correlation score, but did not find this to be computationally effective for small maps.

The time taken for correlation to find potential matches (black bars) scales linearly with the size of the map and the number of Shi-Tomasi features detected in the recovery image. For larger maps, a feature matching algorithm which does not scale with map size would be required.

A breakdown of the timing for each aspect of the recovery on a typical run (frame 1200) is shown in the table below. Note that the time for generating and evaluating poses can vary, due to the random pose hypothesis creation.

Corner detection	4 ms
Correlation	88 ms
Three-point-pose and evaluation	5* ms
Overhead	1 ms
Total	**98 ms**

For the pose to be useful, a low error in the image projection is more important than in the 3D position or orientation because, after recovery, the SLAM system can correct the pose if the features are found near where they are expected in the image. The search regions are typically of radius of ten pixels, and for this 45-second sequence, the projection error was within ten pixels in 93% of the successful recoveries.

Davison's system, when combined with the recovery module, was found to be very robust to both sudden motion and occlusion. The module is fast enough to allow the combined system to be used in real-time although there can be a small lag, after becoming lost, while the pose is recovered.

VI. MAP ALIGNMENT USING THE RECOVERY MODULE

In the normal running of the SLAM system, a quick rotation can cause the observations to suddenly be of an unmapped region of the world. Pose recovery is of course not possible at this stage, and instead the SLAM system should begin a fresh map. Crucially, if the camera returns to an area where the old map is visible, this should be detected and the relative pose of the two maps should be found, so that the maps can be joined.

While the system is making its fresh map, the recovery module can be run periodically to try to detect the camera

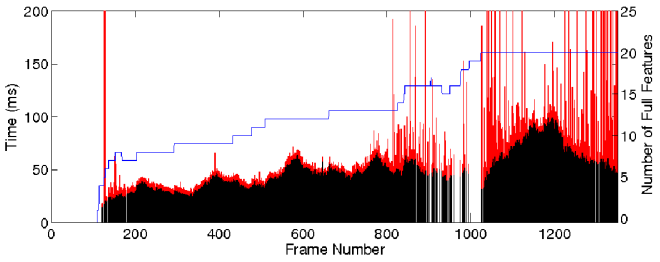


Fig. 9. The recovery module was run on every frame of a 45-second sequence. The number of features in the map (blue line) grows as the SLAM system runs. The total time taken (red bars) by the recovery module is shown when it was successful (in 84% of frames). A large fraction of this time is taken up by the correlation to find feature matches (black bars).

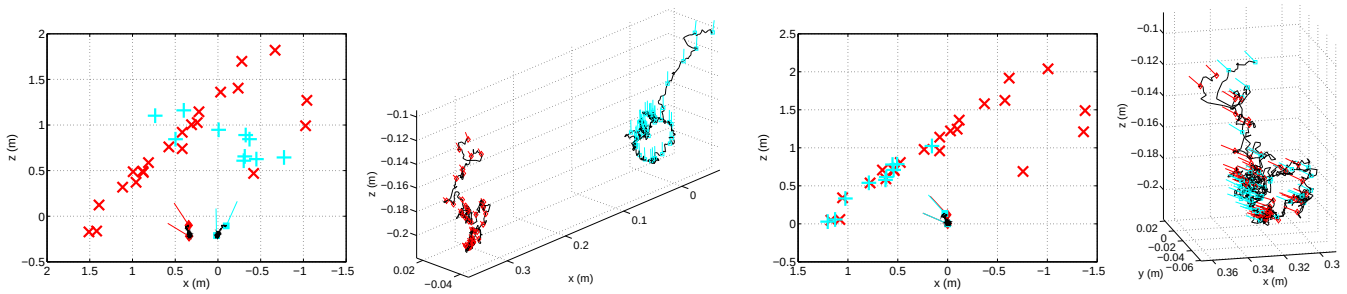
pose relative to the old map. When this overlap is detected, the difference between the current pose estimate and the recovered pose indicates the relative orientation and translation of the two maps. However, this is not enough to align the maps in a single-camera system due to the ambiguity in scale.

Alternatively, the map features themselves could be used for the alignment. This method would be robust to scale difference, but unfortunately is not well suited to real-time single-camera SLAM systems, where the maps created are purposely very sparse so that the tracking can be done in real time. With such sparse maps, it is unlikely that there will be any features common to both maps, and a minimum of three shared features would be needed to specify the alignment.

Instead, we have chosen to use a camera trajectory common to the two coordinate frames. When the recovery module detects that the camera has returned to a region in the old map, the recovered pose is used to initialise a second EKF to track the camera in the old map. Meanwhile, the camera motion is still tracked in the current map using the first EKF. The camera trajectory in the two maps will be at different orientations and scales to suit the map, but crucially, the actual camera motion is identical. By finding the transformation between these trajectories, we find the transformation needed to align the two maps.

Note that a trajectory in this context is a sequence of 6D poses (position and orientation), not simply a sequence of positions. To obtain an initial estimate of the alignment transformation, the orientation of the cameras along the trajectory is used to estimate the rotation between the two. A value for the scale is then obtained by measuring the overall length of the trajectories. Finally, an estimate for the translation offset is determined given the other four parameters. This initial estimate is refined via non-linear optimisation of the Mahalanobis distance between pairs of poses in the respective trajectories.

We used Davison's system to make a map using the sequence shown in Fig. 5. Initially, the camera makes a map of the region to the left of the office chair. When the camera suddenly moves to the region to the right of the chair, a new map is begun. Later in the sequence, the camera pans back to the first region as shown in Fig. 5. At this point, the recovery algorithm finds a pose for the camera in the



(a) Maps before alignment. (b) Trajectories before alignment. (c) Aligned maps. (d) Aligned trajectories.

Fig. 10. Map alignment for two maps generated from different parts of the sequence in Fig. 5. When the recovery module detects that a previously mapped region is visible, the system runs SLAM simultaneously in the current (red $\times$) and the previous (cyan $+$) map. The trajectory estimated relative to the two maps (b) can be aligned (d) to find the relative pose and scale of the two maps. After alignment (c), the line of features from both mapslying on and near the wall of the room can clearly be made out.

original map made of this region indicating that the maps have begun to overlap.

After the overlap is detected, the system tracks the camera in each map using a different EKF. Fig. 10 shows a top-down view of the estimated trajectory in the current map (red $\times$) and in the old map (cyan $+$). These two trajectories are then aligned to find the joined map shown on the right of Fig. 10. The line of features lying on and near the wall of the room can clearly be made out.

The current SLAM system takes 10ms per frame, leaving 23ms to run the recovery module. At this speed, a map overlap check could be made at 3Hz. Once an overlap is found, there is also enough time to run the second EKF needed for the trajectory alignment.

VII. CONCLUSIONS

The recovery module presented here finds the pose of a camera using a map of point features created by a single-camera SLAM system. Correlation is used to find potential matches in the current image for features in the map. These matches are then used to determine candidate poses via the three-point-pose algorithm. Pose hypotheses are verified by projecting the other map features in a RANSAC framework.

The module was shown to reliably relocalise an EKF-based SLAM system after sudden movement or occlusion, thus making the system much more robust. We also showed how the module could be used to aid a particle filter tracking system by creating a small number of extra particles according to the recovered pose. Finally, the module was used while tracking to detect when the camera returned to a region previously seen in another map, and a method was shown which aligns the two maps using the camera trajectory. This alignment is not dependent on common map features and is able to determine the relative scale between the maps, both of which are a particular challenge for single-camera SLAM. This system is useful not only for independent map alignment, but could also be used to recognise loop closures.

VIII. ACKNOWLEDGEMENTS

This work has benefited greatly for many insightful conversations with members – past and present – of the Active Vision Lab and the GRPTR team at University of Zaragoza. We gratefully acknowledge the financial support of the EPSRC (grant GR/T24685 and a studentship to BPW) and the Royal Society (International Joint Project).

REFERENCES

[1] C. G. Harris. Tracking with rigid models. In *Active Vision*. MIT Press, 1992.

[2] A. Chiuso, P. Favaro, H. Jin, and S. Soatto. Structure from motion causally integrated over time. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(4):523–535, 2002.

[3] P. F. McLauchlan. A batch/recursive algorithm for 3D scene reconstruction. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 738–743, 2000.

[4] A. J. Davison. Real-time simultaneous localisation and mapping with a single camera. In *Proc. IEEE International Conference on Computer Vision*, pages 1403–1410, 2003.

[5] M. Pupilli and A. Calway. Real-time camera tracking using a particle filter. In *Proc. British Machine Vision Conference*, pages 519–528, 2005.

[6] E. Eade and T. Drummond. Scalable monocular SLAM. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 469–476, 2006.

[7] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit. FastSLAM 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges. In *Proc. International Joint Conference on Artificial Intelligence*, pages 1151–1156, 2003.

[8] A. Doucet, N. Freitas, K. Murphy, and S. Russell. Rao-Blackwellised particle filtering for dynamic bayesian networks. In *Proc. Conference on Uncertainty in Artificial Intelligence*, pages 176–183, 2000.

[9] R. M. Haralick, C. Lee, K. Ottenberg, and M. Nolle. Review and analysis of solutions of the three point perspective problem. *International Journal of Computer Vision*, 13(3):91–110, 1994.

[10] D. Nistér. An efficient solution to the five-point relative pose problem. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 195–202, 2003.

[11] J. Matas and O. Chum. Randomized RANSAC with $t_{d,d}$ test. *Image and Vision Computing*, 22(10):837–842, 2004.

[12] J. Neira, Tardós J. D., and J. A. Castellanos. Linear time vehicle relocation in SLAM. In *Proc. International Conference on Robotics and Automation*, pages 427–433, 2003.

[13] J. Shi and C. Tomasi. Good features to track. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 593–600, 1994.

[14] M. A. Fischler and R. C. Bolles. RANdom SAmple Consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[15] S. Thrun, D. Fox, and W. Burgard. Monte carlo localisation with mixture proposal distribution. In *AAAI National Conference on Artificial Intelligence*, pages 859–865, 2000.