Reducing mismatching under time-pressure by reasoning about visibility and occlusion

S Wangsiripitak

D W Murray

Department of Engineering Science University of Oxford Parks Road, Oxford, OX1 3PJ www.robots.ox.ac.uk/ActiveVision

Abstract

Three methods are explored which help indicate whether feature points are potentially visible or occluded in the matching phase of the keyframe-based real-time visual SLAM system. The first derives a measure of potential visibility from the angular proximity to keyframes in which they were observed and globally adjusted, and preferentially selects those with high visibility when tracking the camera position between keyframes. It is found that sorting and selecting features within image bins spread over the image improves tracking stability. The second method automatically recognizes and locates 3D polyhedral objects alongside the point map, and uses them to determine occlusion. The third method uses the map points themselves to grow surfaces. The performance of each is tested on live and recorded sequences.

1 Introduction

Establishing image correspondence is central to the recovery of camera pose and scene structure from multiple views. Where processing time is unlimited, the use of point features with rich descriptors, aided by densification using iconic pixel matching and verification using robust fitting does much to eliminate mismatching in non-pathological imagery. In real-time structure from motion and SLAM, however, there is little time to linger at the image level, features are often described by only a few surrounding pixels, and by the time more has been learnt about the scene there is no recourse to past imagery.

The difficulties generated by this time pressure differ according to the size of the maps generated. In visual SLAM based on the EKF [3] [4] computational complexity is dominated by the square of the number of scene points, and the 3D maps which can be updated in real-time are quite small in extent — around 10^2 points on current machines. Avoiding mismatches is then all the more important. But now small is beautiful, as more time can be devoted to each point. The maintenance of a complete covariance matrix places useful bounds on the image region where matches may be found, the compact map restricts the opportunity for occlusion, and the sparse structure makes it relatively cheap to search for coplanarity between map points using hypothesize and test. This has been used both to detect occlusion and to reduce the size of the scene representation [6] [1]. A disadvantage of sparsity, however, is that the structure is little more than a set of pegs on which to hang the computation of camera pose.

By contrast, at a typical operating point in keyframe-based SLAM using visual odometry and bundle adjustment [10] complexity is linear in the number of map points and quadratic¹ in the number of keyframes, and the map size and extent can be larger — 10^4 points, say. While the structure recovered is more representative of the scene, the opportunity for mismatching is greater, particularly as the scene can easily extend to involve occluding surfaces.

In both approaches, failure to match at all can be as damaging as matching incorrectly. The quality of camera tracking is often assessed using the ratio of the number of map points that are successfully matched to the number of points that are considered *potentially visible*. If this ratio falls below some acceptable threshold, tracking may be deemed to have failed.

This paper describes the exploration of methods which help determine whether points are potentially visible in the matching phase of the keyframe-based system of [10]. The first sorts image features according to the angular proximity to keyframes in which they were observed and globally adjusted, and chooses high scoring points when tracking the camera position between keyframes. It is found that sorting and selecting points within image bins spread over the image improves tracking stability, while also reducing the cost of sorting. The second method involves higher level processing, and uses automatic recognition of the faces of a 3D polyhedral object to locate the object within the point map and hence to determine if map points are certainly occluded. The third method lies somewhere in between, and uses the reconstructed map points themselves to grow surfaces.

Section 2 gives a brief overview the underlying structure from motion system. Section 3 describes the derivation of potential visibility and Section 4 describes the use of objects and surfaces to detect occlusion. Each section describes experiments carried out on live and stored imagery. The paper concludes in Section 5.

2 Background

2

Our investigations are based on the parallel tracking and mapping method of [10] where concurrent threads balance (i) the frame-rate task of camera pose estimation assuming a known map with (ii) the lower-rate optimization using bundle-adjustment of a subset of poses (keyframes) and the entire map structure.

Fig. 1(a) summarizes the camera tracking thread using an already initialized 3D map. On capture, each frame is sub-sampled into a four-level pyramid, and corner features found at each scale by the FAST-10 detector [16]. A prior for the camera pose is estimated using a constant velocity $\alpha\beta$ filter which switches to an exponentially decaying model in the absence of measurements. Potentially visible map points are projected into the image, and up to 60 matches sought at the coarsest scale to estimate the pose μ using robust minimization. Then up to 1000 matches are sought at coarse and fine scales and the pose re-optimized. For both stages, the optimum is found using some ten iterations of Gauss-Newton with a Tukey M-estimator cost function (see [19]) based on the reprojection errors $e_i = |\mathbf{x}_i - \mathbf{x}(\boldsymbol{\mu}, \mathbf{X}_i, \mathbf{C})|$, where \mathbf{x}_i is the measurement and $\mathbf{x}(\boldsymbol{\mu}, \mathbf{X}_i, \mathbf{C})$ is the generative model involving scene point \mathbf{X}_i and known camera intrinsics and lens distortion parameters \mathbf{C} .

The fraction of potentially visible features that is successfully matched is monitored to provide a measure of tracking quality. Two thresholds are set. If the fraction exceeds the higher, tracking is deemed good, and if it falls between them, tracking continues but the frames involved are prevented from becoming keyframes for the mapping process. If it falls below both, tracking is assumed to have failed terminally, and a relocalization method is invoked to recover the lost camera to a pose where tracking is able to resume [11].

¹The quadratic dependency eventually becomes cubic.

(a) Tracking	(b) Mapping
while map initialized & new image captured do	Initialize map from two views
Form pyramid, compute corners	Set running
Predict camera pose	while running do
for coarse then fine scale in pyramid do	if new keyframe offered and acceptable then
Project map points into image	Update existing data associations
Match	Compute new 3D points from two views
Update pose from matches	else if not locally converged then
end for	Perform local adjustment
if tracking good & keyframe criteria met then	else if not globally converged then
Offer as keyframe to map maker	Perform global adjustment
else if tracking lost then	else
Relocalize	Update data associations
end if	end if
end while	end while

Figure 1: Code summary of the threads for (a) camera tracking and (b) 3D scene mapping in PTAM.

Fig. 1(b) summarizes the mapping process. A starting camera pose is selected by the user and its image and list of features form the initial keyframe, with pose $\boldsymbol{\mu}_1 \equiv \{\mathbf{R}_1, \mathbf{T}_1\} = \{\mathbf{I}, \mathbf{0}\}$. The camera is moved carefully to a new position to allow features to be tracked in the image. This image is captured as the second keyframe, and Nistér's relative pose algorithm [15] used to derive the 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 $\{\mathbf{X}_i\}$. In practice, care is taken to make $|\mathbf{T}_2 - \mathbf{T}_1| \approx 100 \text{ mm}$ so that a reasonable, but fundamentally arbitrary, scale can be applied to depth and speed.

With the map initialized, camera tracking begins as described earlier, and new keyframes added and stored when required for spatial coverage. The estimated 3D positions $\{\mathbf{X}_i\}$, $i = 1 \dots I$ of all map points and keyframe camera poses $\{\boldsymbol{\mu}_k\}$, $k = 2 \dots K$ (*i.e.*, all except the first, which is fixed) are optimized in a bundle adjustment, using Levenberg-Marquardt [13] to minimize the reprojection error wrapped in a Tukey M-estimator cost function [17].

3 Potential visibility

A feature F_i is entered into the map at \mathbf{X}_i using measurements from the first two keyframes in which is observed. Let their indices be k_{i1} and k_{i2} . In [10] a notional normal direction to the surface underlying the patch is hypothesized to lie along the direction of the camera's optic axis in the first keyframe, $\mathbf{n}_i = -\mathbf{z}(k_{i1})$. Two uses are made of that surface normal in later calculations. First it is used to warp the image patch around a feature for correlation matching. Second, it plays a part in the derivation of potential visibility. A feature is deemed potentially visible if (i) its projected position is in image, (ii) the angle between the feature's normal and the current optic axis is less than a threshold (60° is used), (iii) the warped image patch does not become too small, and (iv) the point is neither too near nor too far from the camera. (In practice, the last two require the warped image patch to be matched in one of levels 0-2 of the current image's pyramid.)

The criteria are broadly successful. However, determining potential visibility by measuring the angle between the current optical axis of the camera and that in only the *first* frame in which the features is observed appears unnecessarily crude. The normal from a single view should lie along the viewing direction not the optic axis and account should be taken of all keyframes in which the feature is matched.

Here we wish to improve on the hypothesis of the surface normal from a sequence and frame and to propose a more probabilistic measure of potential visibility. The motivation is



Figure 2: Feature F_i viewed by a camera whose optic axis is along **z**. Keyframe k_{i1} is the *first* keyframe, and keyframe k_i is *any* keyframe in which F_i is observed. The current frame is that used by the tracking thread and is therefore not a keyframe.

that, as mentioned earlier, shortage of time allows only a restricted number of map features to be used in the camera tracking thread, and using those that are more likely to be visible should not only help tracking numerically, but also allow more realistic monitoring of tracking failure through the fraction of potentially visible points that are matched. Later in the paper we consider more absolute determination of visibility through occlusion by static objects and surfaces in the map.

3.1 A measure of visibility

As the camera moves in the environment, the *i*-th feature F_i may be observed in different keyframes. A list L_i of keyframes in which the feature is observed is maintained $L_i = \{k_{i1}, k_{i2}, \ldots\}$, which allows the projected ray from the feature to the camera centre to be recalled in the world coordinate system as $\mathbf{V}_{ik_i} = \mathbf{T}_{k_i} - \mathbf{X}_i$, $k_i \in L_i$, as shown in Fig. 2.

During camera tracking (where the current camera frame is *de facto* not a keyframe) if feature F_i is predicted to project into the image along ray $\mathbf{V}_i = \mathbf{T} - \mathbf{X}_i$ at the predicted camera pose, a measure of alignment with the closest viewing direction in the feature's keyframe list is sought

$$\boldsymbol{\alpha}_{ik_i} = \max_{k_i \in L_i} \left[\frac{\mathbf{V}_{ik_i} \cdot \mathbf{V}_i}{|\mathbf{V}_{ik_i}| |\mathbf{V}_i|} \right]$$
(1)

and the feature's current visibility score $v_i = [0, 1]$ is determined from

$$v_i = \begin{cases} (\alpha_{ij} - \alpha_0)(1 - \alpha_0)^{-1} & \text{if } \alpha_{ij} > \alpha_0 \\ 0 & \text{otherwise.} \end{cases}$$
(2)

In later experiments, $\alpha_0 = 1/\sqrt{2}$ is used, corresponding to a cut-off angle of 45°.

The linear roll-off in α provides a cheap computation for a flatter distribution in angular deviation. However the test for this approach is not in the merit of individual values, but whether it provides an increased chance of choosing a better subset of features during tracking. Recall that during tracking the pose of the camera is re-estimated using first some 60 features at coarse scale, then at most 1000 points from coarse to fine scales in least squares using a Tukey M-Estimator. There is insufficient time either to use all points, or to perform stronger robust estimation based on sampling, such as employed in [5]. With a binary estimation of visibility, the selection of points within those regarded as visible is random. Here instead we can sort the features by visibility score and use the features with the highest scores first.



Figure 3: (a) Images from the desk sequence where the camera's view direction spans a wide angle. (b) Comparison of performance over time using (i) the original, (ii) sorted visibility across the entire image, (iii) sorted visibility in regions. The performance measure is the ratio of successfully matched features. (c,d) The maps recovered using (i) and (iii) respectively.

3.2 Spatial coverage

An observed side effect of selecting on visibility scores v computed across the entire image is that the features selected tend to be clustered at a small number of locations in the scene and image, rather than spread throughout. In some scenes this causes the estimated pose of even a *stationary* camera to cycle between two or more poses: the change in estimated pose causes a new set of clustered features to be selected, again changing the pose, all in a deterministic fashion. This interesting but unwelcome phenomenon is avoided by requiring points to be selected from throughout the image. The feature visibilities are sorted within nimage bins. If p points are to be used in all, an attempt is made to use the most visible p/nfrom each. If a bin is exhausted, the demand from the remainder becomes p/(n-1), and so on.

3.3 Warping

In common with many schemes, [10] used a warped image patch for feature matching, but one based on the feature in the first keyframe. Another small modification made here is to warp the patch created on the fly from the image of the keyframe closest in angle, according to the rotation and distance of the current camera from it.

3.4 Experiments and Results

The modified camera tracking and map-making methods are implemented as threaded processes in C++, and are run here on a dual core 2.66 GHz cpu. Grey level images of size 640×480 pixels are provided at 30 Hz from a CCD camera over a Firewire link.



Figure 4: Stability in (a) position and (b) orientation of the camera using points selected (i) randomly using the original binary visibility score, and (ii, iii) in decreasing order of the modified score sorted across the entire image and within image regions, respectively.

Fig. 3(a) shows stills from the sequence of a cluttered desk as the camera is translated (of necessity) and rotated through some 90° over four seconds. The comparative performance of (i) the original binary visibility measure, (ii) the proposed visibility measure sorted over the image, and (iii) the proposed measure sorted in image regions is shown in Fig. 3(b), where the ratio of points matched to points considered potentially visible and tracked is shown as a function of frame number. The falling ratio value of the original eventually causes tracking to be stopped early (at about 60°) because, although there is no occlusion caused by other objects, visibility is derived from the first keyframe alone. The modified methods sometimes result in a smaller ratio but one that is stable.

Figs. 3(c,d) compare the maps recovered with the original and with the graded visibility in regions. Apart from the premature end to tracking in the original, there is no significant difference between them. The more robust tracking performance with the lower (but steady) ratio of matches is not being bought at the cost of overall map density or quality.

To assess the effect of the modifications on pose recovery in the tracking thread, the estimation was iterated ten times on each image of a sequence, and errors from the mean evaluated for position and orientation. Figs. 4(a) and (b) shows these error magnitudes for the desk sequence using selection of points using (i) the original visibility score; (ii) the modified visibility score across the entire image; and (iii) the modified score within distributed image regions. The original shows the greatest error, and tracking ceases when the apparent fraction of potentially visible features being tracked and matched fell below threshold. The modified visibility measure sorted within regions performs best overall.



Figure 5: Extra time of using sorted visibility in image regions, shown with the number of potentially visible points.

Fig. 5 shows the extra time (black) of using sorted visibility in image bins, and the number of points deemed visible (red) which affects the sorting time. The number of all points in the map (blue) is also shown for reference. The camera movement that induces another set of points closest in angle, and points clustered in some image bins, causes the ordering time to jump intermittently. The extra time required for each frame is 5.33 ms on average.

4 Detecting occlusion

The method of increasing the likelihood of picking visible points cannot indicate that points are occluded — the point maps remain transparent. The paper now explores two methods of detecting occlusion.

4.1 Using modelled objects

The first of these augments the map with known 3D objects which are static and form part of the surroundings' structure. It determines whether parts of the map are occluded by them. Object detection, recognition and pose initialization are performed using an extension of [2] to fully 3D objects.

An additional thread, running independently of those in Fig. 1, computes SIFT features [12] in the keyframes alone, and compares their descriptors with those stored in a database. If sufficient support is found, the object is flagged as potentially visible. As the objects here are polyhedral, mismatching is reduced by grouping coplanar features in the database and seeking only matches which are consistent with a single homography transforming image to database feature positions.

Once an object has been found visible in two or more keyframes, the scene positions of keypoints with matches are optimized, taking the keyframe camera poses from the mapping thread as fixed known values. With just two views an algebraic residual is minimized [8], but as more keyframes are added Levenberg-Marquardt is used to minimize the overall projection error. The scaling, rotation and translation in the similarity transformation between the points in the object frame X_B and the recovered scene points X are found by minimizing $\sum ||X_i - sRX_{Bi} - T||^2$ using Horn's closed-form solution to the absolute orientation problem [9]. Note that the object scale is adjusted to the scale of the map, not *vice versa*. This keeps the adjustment local rather than global, which is preferable as there are drifts in map scale over time and space. (For moving objects, the identity and initial pose of the object are further verified by attempting to track crease edges and albedo markings held in the model using a robust RAPiD tracker [7]. Here however we consider only stationary objects.)

Fig. 6(a) shows stills from an experimental run where some 30% of the potentially visible points built early in the sequence (40% of image area) become occluded by a pillar in the bottom left image. As shown in Fig. 6(b), without occlusion detection tracking fails when the percentage of supposedly visible features that are successfully matched falls off.

The pillar was then modelled as an object and some texture applied so that it could be recognized and located automatically. Fig. 6(c) shows the results with occlusion detection. As in Fig. 3(b) results are shown without the visibility measure, with the visibility measure sorted across the entire image, and with visibility sorted in image regions. Again, the last measure performs best.

4.2 Growing higher order structure

While explicit modelling of object in visual SLAM has proved worthwhile when the objects move or are likely to move [18], we find taking a similar approach to static structural



Figure 6: (a) Stills from the basement sequence where a pillar occludes a large part of the map. (b) Without occlusion detection, tracking fails quickly. (c) With occlusion detection tracking continues. The performance is further improved by including the visibility measure of § 3.



Figure 7: (a) Planar fragments are fitted to points which are visible within image bins, which are being tracked and matched, and which are not assigned to another fragment. (b) Example (enlarged from Fig. 8(c)).

elements of the scene unduly onerous. A lower-level approach to occlusion detection is through the growth of scene surfaces from point structure. This has been demonstrated by Carranza and Calway [1] using planar surfaces in EKF-based visual SLAM maps. Because EKF-SLAM's maps are sparse, and each point typically more accurately located in 3D, they found it feasible to generate each planar hypothesis using three scene points alone, and to test that the scene viewed within each triangle was indeed planar using image warping. Their principal goal was not occlusion detection *per se*, but the reduction of the state dimension by replacing many points with a single plane.

However, with denser point maps this approach becomes prohibitively expensive. Here instead we compute planar surfaces using the 3D map points rather than the image as direct support. Our aim is to reconstruct many planar surfaces to allow reasoning about occlusion and timely prevention of tracking failure rather than to achieve the fully dense reconstruction of scene structure proposed by Newcombe and Davison [14].

At every frame (*i.e.* at the tracking frame rate of 30 fps) an attempt is made to fit plane fragments using robust methods. The seed points selected are those 3D map points which (i) are visible within an image bin (Fig. 7), (ii) are being tracked and matched, and (iii) are as yet unassigned to an existing planar fragment. To support a hypothesis, all points which satisfy (i) are used.

As soon as a plane fragment — each described in the map by location, surface normal and convex hull points — is created, it can be used in further determination of visibility (i) above. However, to save time testing for occlusion (rather than to generate more convincing scene reconstructions) a process merges the fragments into larger planes and further grows



Figure 8: (a) The approximate camera path between a set of library shelves and (b) stills from the sequence at the camera positions shown in (a). (c) shows the fragments early in the sequence, (d) and (e) show the merged planes detected (cf surfaces (1) and (2) in (a)). (f) shows the detection of surface (3) allowing occluded map points to be ignored. (See supplementary video.)

the planes by accreting points.

Fig. 8 illustrates the processing. Fig. 8(a) shows a stylized plan of the camera motion between a set of library shelves, with (b) showing stills cut from the sequence at (approximately) the camera positions sketched in (a). Without occlusion detection, the parallel tracking and mapping method prevents further keyframes being added at the third position, when the fraction of successfully tracked points falls sharply, and tracking fails shortly afterwards. Fig. 8(c) shows the 3D map and planar fragments built at the start of the sequence. Fig. 8(d) shows the larger plane corresponding to surface (1) in (a) created by merging and point accretion. Fig. 8(e) shows the addition of end of the shelving (2 in (a)), and (f) shows the map with the surface (3) detected and added. This allows the tracking process to consider only points on the visible surface allowing the addition of keyframes to continue. A video of this sequence is provided.



Figure 9: The set of camera positions computed using occlusion detection in front of surface (3), shown enlarged from 8(f). Circled is an erroneous camera position obtained after turning off occlusion detection.

Fig. 9 shows the succession of camera positions in front of the surface (3) of Fig. 8 recovered using occlusion detection. (An enlargement of bottom of Fig. 8(f).) To illustrate the effect of seeing occluded points, at the end of the sequence occlusion detection was turned off and the sequence reversed. The computed camera rapidly spun out of the scene.

5 Conclusions

This paper has shown that the performance of a real-time structure from motion algorithm can be improved by preferentially selecting points for matching using a simple visibility measure. It was found that sorting and selecting features within image regions, and requiring points to be drawn from all regions, improved tracking stability.

Two methods of detecting occlusion have been explored: the automatic recognition and pose initialization of 3D polyhedral objects within the 3D point map, and the growth of planar surfaces from the points themselves. The latter holds greater promise for general scenes.

References

- J. M. Carranza and A. Calway. Efficiently increasing map density in visual SLAM using planar features with adaptive measurement. In Proc 20th British Machine Vision Conference, 2009.
- [2] R. O. Castle and D. W. Murray. Object recognition and localization while tracking and mapping. In Proc 8th IEEE/ACM Int Symp on Mixed and Augmented Reality, pages 179–180, 2009.
- [3] A. J. Davison. Real-time simultaneous localisation and mapping with a single camera. In Proc 9th IEEE Int Conf on Computer Vision, pages II: 1403–1410, 2003.
- [4] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse. MonoSLAM: Real-time single camera SLAM. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(6):1052–1067, 2007.
- [5] E. Eade and T. Drummond. Scalable monocular SLAM. In Proc 24th IEEE Conf on Computer Vision and Pattern Recognition, pages 469–476, 2006.
- [6] A. P. Gee, D. Cheklov, W. Mayol, and A. Calway. Discovering planes and collapsing the state space in visual SLAM. In *Proc 18th British Machine Vision Conference*, 2007.
- [7] C G Harris and C Stennett. RAPiD a video rate object tracker. In Proc 1st British Machine Vision Conference, pages 73–77. BMVA Press, 1990.
- [8] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2000.
- [9] B. K. P. Horn, H. M. Hilden, and S. Negahdaripour. Closed form solution of absolute orientation using orthonormal matrices. *Journal of the Optical Society of America*, A, 5(7):1127–1135, 1988.
- [10] G. Klein and D. W. Murray. Parallel tracking and mapping for small AR workspaces. In Proc 6th IEEE/ACM Int Symp on Mixed and Augmented Reality, 2007.
- [11] G. Klein and D. W. Murray. Improving the Agility of Keyframe-Based SLAM. In Proc 10th European Conf on Computer Vision, 2008.
- [12] D.G. Lowe. Distinctive image features from scale-invariant keypoints. International Journal of Computer Vision, 60(2):91–110, 2004.
- [13] D. W. Marquardt. An algorithm for the least-squares estimation of non-linear parameters. J Soc Indust Appl Math, 11(2):431–441, 1963.
- [14] R. A. Newcombe and A. J. Davison. Live dense reconstruction with a single moving camera. In Proc 28th IEEE Conf on Computer Vision and Pattern Recognition, 2010.
- [15] D. Nistér. An efficient solution to the five-point relative pose problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(6):756–777, 2004.
- [16] E. Rosten and T. Drummond. Machine learning for high-speed corner detection. In Proc 9th European Conf on Computer Vision, 2006.
- [17] J. W. Tukey. Exploratory Data Analysis. Addison-Wesley, Reading MA, 1977.
- [18] S. Wangsiripitak and D. W. Murray. Avoiding moving outliers in visual slam by tracking moving objects. In Proc 2009 IEEE Int Conf on Robotics and Automation, 2009.
- [19] Z. Y. Zhang. Parameter estimation techniques a tutorial with application to conic fitting. *Image and Vision Computing*, 15(1):59–76, 1997.