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

S Wangsiripitak

D W Murray

Establishing image correspondence is central to the recovery of camera pose and 3D structure from multiple views. When working "off-line", much can be done to eliminate mismatching in non-pathological imagery. However, in real-time structure from motion and SLAM there is little time to linger at the image level, features are often described by only a few surrounding pixels, and there is often no recourse to re-examine earlier imagery. Pressure of time generates difficulties which differ according to the size of the 3D maps computable at frame-rate. In EKF-based SLAM [3] the small map size (say 10^2 points) makes mis-matching intolerable — but the maintenance of covariance bounds the image regions where matches may be found, the compact map restricts the opportunity for occlusion, and the sparse structure makes it cheap to search for occluding surfaces using hypothesize and test [1] [5]. In contrast, keyframe-based SLAM using bundle adjustment [6] handles a larger map (say 10⁴ points), which is more representative of the scene and more tolerant of mismatching --- but 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, because the quality of camera tracking is often assessed using the ratio of the number of successfully matched points to the number in their superset which were deemed potentially visible.

This paper discusses three methods which help indicate whether and to what degree feature points are visible or occluded in the matching phase of the keyframe-based real-time visual SLAM system of [6]. The first is concerned with feature visibility and is used in conjunction with either of the other two which are concerned with detecting occlusion of features.

Visibility. We derive a measure of the potential visibility of features using the angular proximity to keyframes in which they were observed, and preferentially select those with high visibility when tracking the camera position between keyframes. The *i*-th feature point F_i in a map is flagged as potentially visible if its predicted projection lies in the image, and the angle between the camera's current viewing direction \mathbf{V}_i and the estimated normal of the surface underlying the feature is below a threshold. The normal direction \mathbf{n}_i is estimated from the observed viewing direction \mathbf{V}_{ik_i} , using all the keyframes k_{ij} in the set $L_i = \{k_{i1}, k_{i2}, \ldots\}$ in which feature F_i is observed (Fig. 1(a)). The one closest in angle to \mathbf{V}_i is used as \mathbf{n}_i to derive an analogue visibility score $0 \le v_i \le 1$.



Figure 1: (a) Keyframe geometry. (b) Using [6] the tracked ratio (red) falls over time and tracking eventually fails. Using the new measure the tracked ratio (yellow) is maintained. Distributing points through the image (green) gives a similar though reduced fraction, but greater overall stability in performance.

In [6], a feature's normal was estimated using only the optic axis $-\mathbf{z}(k_{i1})$ of the first keyframe in which the features was observed, and the visibility designation was entirely binary, $v_i = \{0, 1\}$. During camera tracking there is often insufficient time to use all matches to recover pose, and using a binary estimation of visibility the selection of points within those deemed visible is effectively random. Here instead we sort the features by visibility score, and use the features with the highest scores first, providing an increased chance of choosing a better subset of features during tracking. A further improvement in tracking stability, and one which reduces the cost of visibility sorting, is obtained by dividing the image

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

into bins, and sorting and selecting points by visibility within them.

Occlusion detection (1). The second method developed here is one which involves higher level processing to determine feature occlusion. 3D objects, which are static and form part of the surroundings' structure, are recognized by computing SIFT features [7] in the keyframes, and by comparison with those stored in a database. Using the keyframe poses computed from [6] the 3D positions of the SIFT features are determined by triangulation, using a linear algebraic method for two views and Levenberg-Marquardt method when more keyframes are added [2]. The similarity transformation between the reconstructed and database 3D positions is recovered to locate the object in the 3D map. Then, during camera tracking, map points are checked for occlusion by the object. Results are shown in Fig. 2.



Figure 2: (a) A pillar occludes part of the map. (b,c) Ratios of tracked points to potentially visible points (without (red) and with (yellow,green) visibility measures) (b) without and (c) with occlusion detection.

Occlusion detection (2). A lower-level approach to occlusion detection is through the growth of scene surfaces from the point structure itself. 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, (ii) are being tracked and matched, and (iii) are as yet unassigned to an existing planar fragment (Fig. 3(a)). To support a hypothesis, all points which satisfy (i) are considered. 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 the planes by accreting points. Fig. 3(b) shows fitted fragments, and Fig. 3(c) shows their growth into larger surfaces.



Figure 3: Planar fragments seeded from image bins (a,b) and grown into larger surfaces (c).

- J. M. Carranza and A. Calway. Efficiently increasing map density in visual SLAM using planar features with adaptive measurement. In *Proc 19th 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, 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.
- [4] E. Eade and T. Drummond. Scalable monocular SLAM. In Proc 24th IEEE Conf on Computer Vision and Pattern Recognition, pages 469–476, 2006.
- [5] A. P. Gee, D. Cheklov, W. Mayol, and A. Calway. Discovering planes and collapsing the state space in visual SLAM. In Proc 17th British Machine Vision Conference, 2007.
- [6] 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.
- [7] D.G. Lowe. Distinctive image features from scale-invariant keypoints. International Journal of Computer Vision, 60(2):91–110, 2004.