Estimation and Tracking of Partial Planar Templates to Improve VSLAM

Abdelsalam Masoud, William Hoff

Colorado School of Mines, Golden, Colorado, USA
{amasoud,whoff}@mines.edu

Abstract. We present an algorithm that can segment and track partial planar templates, from a sequence of images taken from a moving camera. By "partial planar template", we mean that the template is the projection of a surface patch that is only partially planar; some of the points may correspond to other surfaces. The algorithm segments each image template to identify the pixels that belong to the dominant plane, and determines the three dimensional structure of that plane. We show that our algorithm improves the accuracy of visual simultaneous localization and mapping (VSLAM), especially in scenes where surface discontinuities are common.

Keywords: Tracking, visual SLAM, structure from motion.

1 Introduction

For mobile robot applications, it is important to perform Simultaneous Localization and Mapping (SLAM). Visual sensors (*i.e.*, cameras) are attractive for SLAM due to their low cost and low power. Much research has been performed on VSLAM (visual SLAM), including approaches using a single camera, which we focus on here. Most approaches detect feature points in the environment, using an interest point operator (*e.g.*, [1]) that looks for small textured image templates. These templates are then tracked through subsequent images, and their 3D locations, along with the camera motion, are estimated.

In order to track image templates, most existing algorithms assume (either implicitly or explicitly) that each image template is the projection of a single planar surface patch (*e.g.*, [2], [3]). The Lucas-Kanade algorithm and its variants [4] is a good example of methods to track planar templates. These algorithms compute the image deformation of reference template $T(\mathbf{x})$, so as to minimize the sum of squared differences between $T(\mathbf{x})$ and a region of the current image $I(\mathbf{x})$. If the patch is planar, then its appearance can be accurately predicted in subsequent images.

For example, a homography (projective) transformation can accurately model the deformation of the image template from the reference image to the current image. Even if a surface is curved, it can appear to be locally planar if the patch size is small enough. However, as the distance between the reference camera and the current cam-

era increases, the prediction error of a curved patch also increases. Tracking will eventually fail when the camera has moved far enough.

A more difficult problem occurs when the template encompasses two disjoint surfaces, which may be widely separated in depth. Unfortunately, such templates often are detected by interest point operators, because the boundary between the surfaces often yields good image texture. However, even small camera motion will cause tracking to fail in such cases.

Some environments have many non-planar surfaces. Figure 1 shows the top 64 points that were automatically detected by an interest point operator [1], using a template window size of 15×15 pixels. By visual inspection, 36 of these templates encompass more than one surface. Tracking of these templates will fail after a short time, using tracking algorithms that make the single-plane assumption.

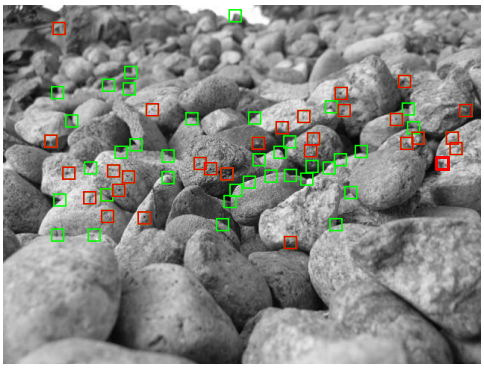

Fig. 1. Interest points detected in a scene. Templates that appear to encompass more than one surface are shown as green; the others are shown as red.

If we can model the true 3D structure of a patch, then we can more accurately predict its appearance, and potentially track it over a longer distance. When a feature is visible over a large visual angle, the error in its estimated 3D location is reduced. This also improves the accuracy of the derived camera poses.

In this work, we present such an algorithm that estimates the 3D structure of a patch, as it is tracked through a sequence of images. We assume that the image template is the projection of a planar surface, but some of the points in the template may not belong to that surface (*i.e.*, they may belong to other surfaces). We automatically identify the points belonging to the "dominant" plane of the patch, and the parameters of that plane. This allows us to accurately predict the appearance of the pixels belonging to the dominant plane, and ignore the others. As a result, the algorithm is better able to track templates that encompass surface discontinuities. A preliminary version of this work was published in [5].

The contribution of this paper is that we have incorporated the new partial plane tracking algorithm into a standard VSLAM algorithm. We show that the new approach achieves significantly better accuracy, as compared to a VSLAM algorithm that uses the standard whole plane tracking method.

Note that we make no assumptions about the color distributions of the regions (*e.g.*, that one region is lighter than another). As a result, methods such as mean-shift [6] that segment and track based on color differences are not applicable.

The remainder of this paper is organized as follows: Section 2 describes the new template tracking method, and Section 3 describes the overall VSLAM algorithm. Section 4 presents experimental results, and Section 5 provides conclusions.

2 Modeling and Tracking of Partial Planar Templates

Each template to be tracked is represented by a small square subimage (we use size 15×15 pixels) from the image where the template was first detected. We use a standard interest point operator [1] to detect points to track. Let $\mathbf{X}_0$ be the 3D point location at the center of the template, and $\mathbf{x}_{ref}$ be the corresponding image point.

We assume that a template is the image projection of a planar surface patch in the scene, which we call the "dominant plane". However, only some of the points in the template are projections of the dominant plane; hence we use the name "partial planar template". The other points may be projected from other surfaces (we do not try to model these other surfaces; only the dominant plane). Let $p_{ref}(\mathbf{x})$ represent the probability that image point **x** in the reference image belongs to the dominant plane for the template. Initially, $p_{ref}(\mathbf{x}) = 0.5$ for all points in the template. Note that for conventional algorithms, $p_{ref}(\mathbf{x})$ is always equal to 1.0, because they assume that the template is the projection of a single planar surface.

Initially, we have no 3D information on new templates, and so we use a 2D method (normalized cross correlation) to track new templates to subsequent images. After the camera has translated sufficiently far, we can perform triangulation to estimate the 3D position of the center of the template (we use a threshold of 2.0 degrees of visual angle between the first and last observations). We also initialize the surface normal vector of the template to point away from the camera, as done by [2] and others.

As each new image is acquired, three steps are performed on each template: (1) matching, (2) updating the surface normal, and (3) updating the probability mask. These are described in the subsections below.

2.1 Matching

To match a template to a new image, we warp the template from the reference image to the current image in order to predict its appearance. A plane is represented by the equation $\mathbf{n}^T\mathbf{X} = d$, where **n** is the normal vector, *d* is the perpendicular distance to the origin, and **X** is any point on the plane. We can transform a planar surface from one image to the other using the homography matrix given by

$$\mathbf{H} = \mathbf{K}(\mathbf{R} + \mathbf{t}\,\mathbf{n}^T/d)\mathbf{K}^{-1} \tag{1}$$

where **K** is the camera intrinsic parameter matrix, **n** is the surface normal, and **R** and **t** are estimated rotation and translation between the cameras, respectively [2].

Let $\mathbf{w}(\mathbf{x};\boldsymbol{\beta})$ denote the warping function that implements the homography transformation as described above, where $\boldsymbol{\beta}$ is a vector of parameters, consisting of the set $\{\mathbf{R}, \mathbf{t}, \mathbf{n}, d\}$. Note that d can be found from the location of the template center using $d = \mathbf{n}^T\mathbf{X}_0$. The warp $\mathbf{w}(\mathbf{x};\boldsymbol{\beta})$ takes the pixel $\mathbf{x}_1$ in the first image and maps it to location $\mathbf{x}_2 = \mathbf{w}(\mathbf{x}_1;\boldsymbol{\beta})$ in the second image. We warp the template to the current image using

$$T_{curr}^{(\boldsymbol{\beta})}(\mathbf{x}) = T_{ref}(\mathbf{w}(\mathbf{x};\boldsymbol{\beta})). \tag{2}$$

The superscript $\boldsymbol{\beta}$ is used to emphasize that the result depends on the parameters $\boldsymbol{\beta}$. Similarly, let $p_{curr}^{(\boldsymbol{\beta})}(\mathbf{x})$ be the probability mask for the template, warped from the reference image to the current image.

To match the template to the current image, we search a 2D region around the predicted location to minimize the sum of squared differences, weighted by the probability at each pixel:

$$\Delta\mathbf{x} = \arg\min \sum_{\mathbf{x}\in N_{H\times W}(\mathbf{x}_c)} \left[T_{curr}^{(\boldsymbol{\beta})}(\mathbf{x}) - I_{curr}(\mathbf{x}+\Delta\mathbf{x})\right]^2 p_{curr}^{(\boldsymbol{\beta})}(\mathbf{x}) \tag{3}$$

where $N_{HxW}(\mathbf{x}_c)$ is the $H \times W$ neighborhood surrounding the predicted location of the template center. The size of the search neighborhood is determined by the uncertainty of the camera pose and the uncertainty of the 3D location of the point. Since the size is usually relatively small, an exhaustive search is performed.

If the sum of squared differences, weighted by the probability is below an empirically derived threshold (we used a value of 40.0 in this work), we consider the template to be successfully matched to the location $\mathbf{x}_0 = \mathbf{x}_c + \Delta\mathbf{x}$ in the current image.

2.2 Updating the Surface Normal

After a template has been matched, we update its surface normal, which is parameterized by the two direction angles (θ,ϕ). We search for the angles that minimize the sum of square differences weighted by the probability:

$$(\theta,\phi) = \arg\min \sum_{\mathbf{x}\in N_{H\times W}(\mathbf{x}_c)} \left[T_{curr}^{(\boldsymbol{\beta})}(\mathbf{x}) - I_{curr}(\mathbf{x}+\Delta\mathbf{x})\right]^2 p_{curr}^{(\boldsymbol{\beta})}(\mathbf{x}) \tag{4}$$

A non-linear optimization algorithm [7] is used for the search. Since we usually have a fairly good estimate of the surface normal from the previous images, there is usually a fairly small correction to the angles. However, in the case of highly slanted patches, the correction from the initial orientation can be larger.

2.3 Updating the Probability Mask

We next update the probability mask $p_{ref}(\mathbf{x})$. We first take the region of the current image where the template matched, and map it back to the reference image.

$$T_{match}^{(\boldsymbol{\beta})}(\mathbf{x}) = I_{curr}(\mathbf{w}^{-1}(\mathbf{x};\boldsymbol{\beta})) \tag{5}$$

The residual error between the template in the reference image and the corresponding region in the current image is:

$$r(\mathbf{x}) = T_{ref}(\mathbf{x}) - T_{match}^{(\boldsymbol{\beta})}(\mathbf{x}) \quad (6)$$

If the point **x** belongs to the dominant plane, then the magnitude of the residual ***r(x)*** should be small. We assume that the residuals are normally distributed, so that the probability of measuring residual *r*, given that the point is on the dominant plane, is $p(r|\mathbf{x} \in D) = N(0, \sigma_D^2)$, where σ_D is the standard deviation of residuals for points on the dominant plane.

Similarly, if the point **x** does not belong to the dominant plane (*i.e.*, it belongs to the "background"), then the probability of the residuals is $p(r|\mathbf{x} \in B) = N(0, \sigma_B^2)$, where σ_B is the standard deviation of residuals for points in the background.

The probability mask is estimated recursively. Given the probability mask estimated from the residuals at times 1 through *t*-1, we can estimate the probability at time *t* using the discrete Bayes filter [8]:

$$p(\mathbf{x} \in D \mid r_{1:t}) = \eta\, p(r_t \mid \mathbf{x} \in D)\, p(\mathbf{x} \in D \mid r_{1:t-1}) \quad (7)$$

$$p(\mathbf{x} \in B \mid r_{1:t}) = \eta\, p(r_t \mid \mathbf{x} \in B)\, p(\mathbf{x} \in B \mid r_{1:t-1}) \quad (8)$$

We choose η so that $p(\mathbf{x} \in D \mid r_{1:t}) + p(\mathbf{x} \in B \mid r_{1:t}) = 1$. Then the probability mask is $p_{ref}(\mathbf{x}) = p(\mathbf{x} \in D \mid r_{1:t})$.

2.4 Example of Synthetic Image

To illustrate our method, we show results on a synthetic image. Two planar surfaces were created in a checkerboard pattern (Figure 2, left). Both planes were perpendicular to the camera's line of sight. The two surfaces were textured with random noise using the same parameters. Next, interest points were detected in the synthetic image (Figure 2, right), using 15×15 templates.

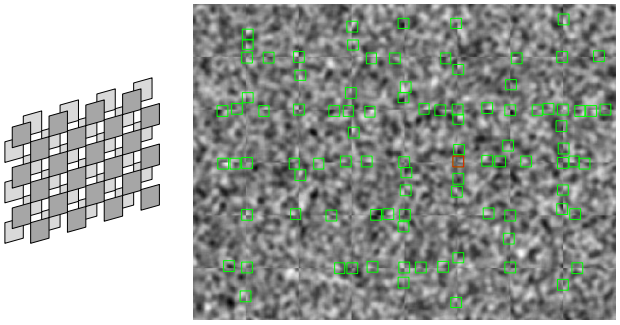

Fig. 2. Synthetic image. (Left) 3D structure. (Right) Interest points.

The camera was then translated to the right. The result of tracking one of the templates (the template outlined in red in Figure 2) is shown in Figure 3. This template encompasses two planar regions, because it is located at one of the corners of the checker-board pattern. The evolution of the probability mask over time is shown in the bottom row. By the end of the sequence, the algorithm appeared to correctly seg-

ment the template and identify the pixels belonging to the dominant (foreground) plane, with only a few errors.

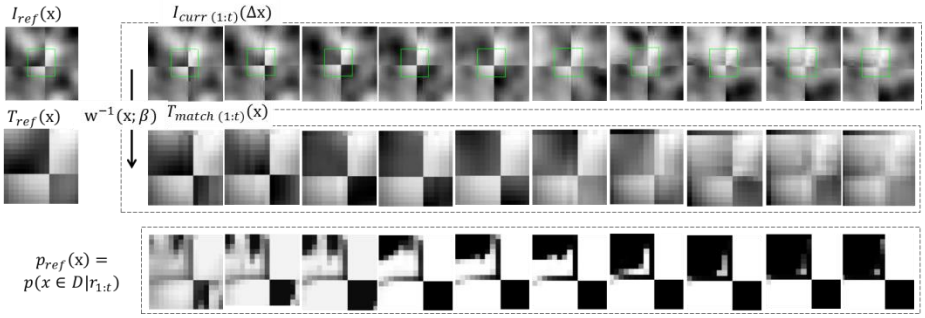

Fig. 3. The reference template (left column) is tracked for 50 frames. The results for every 5th frame are shown. (Top row) Image region surrounding the tracked point. (Middle row) The matched template, warped back to the reference image. (Bottom row) Probability mask, where white indicates a high probability of the point to belong to the dominant plane.

3 Integration of Tracking Method with VSLAM

We integrated the tracking method with a conventional VSLAM algorithm similar to [3]. The algorithm determines the poses of the cameras and the 3D positions of tracked points from images (Figure 4). Some images are designated as "keyframes", and these poses are refined using bundle adjustment over a sliding window.

```
While input data is available do
  Get next image
  Track existing points to the new image
  Estimate pose from 2D to 3D point correspondences
  Initialize new 3D points using triangulation
  Update surface normal and probability masks
  If this pose is a keyframe
    Do bundle adjustment over last N keyframes
    Acquire new points to track
  End If
End While
```

Fig. 4. VSLAM algorithm pseudocode.

The "pose estimation" step is performed using the points which have 3D information. A standard PnP ("Perspective-n-Point") algorithm is used, with a RANSAC [9] approach to eliminate outliers. We next initialize new 3D points from 2D points by performing triangulation if there is sufficient visual angle between observations.

An image is designated as a "keyframe" if the camera has moved sufficiently far from the previous keyframe. Bundle adjustment is performed over the last N=3 keyframes. We collect all points that were observed in the N keyframes, and optimize

their 3D locations as well as the poses of the *N* keyframes, to minimize the reprojection error of the points. We acquire new points if the number of points being tracked in the current image falls below a threshold (we used a threshold of 200 points).

As with all VSLAM algorithms that use a single moving camera, there is a scale ambiguity in the results if only image measurements are used. Some additional information must be used to resolve the ambiguity. In this work we resolve the ambiguity by physically measuring the pose of the second keyframe (the first keyframe is considered to be at the origin). This defines a true metric scale for all results.

4 Experimental Results

We compared the performance of the VSLAM algorithm using the "partial plane" tracking method to the same algorithm using the "whole plane" tracking method. The whole plane method is identical to the partial plane method, except that the probability mask $p_{ref}(\mathbf{x})$ is always equal to 1.0, and we skip the step of updating the probability mask. Therefore, any improvement in performance can be attributed to the use of the new partial plane tracking method.

Three datasets were used to test the algorithm. The first two are image sequences that we collected (called "CSM Indoor" and "CSM Outdoor"), shown in Figure 5.

Fig. 5. Sample images from "CSM Indoor" and "CSM Outdoor" datasets.

The "CSM Indoor" dataset consists of 40 images, in which the camera translates in the XZ plane and simultaneously rotates. The estimated camera poses and 3D point positions are shown in Figure 6.

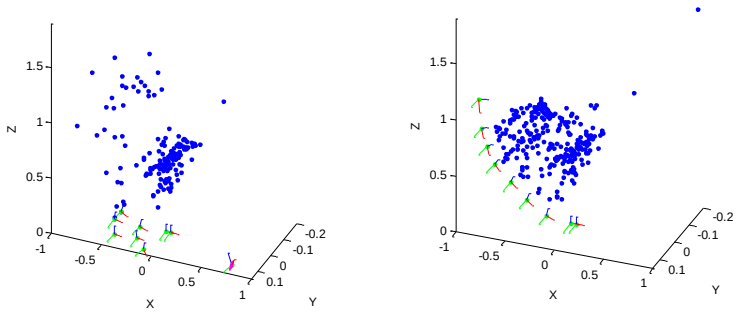

Fig. 6. Estimated camera poses and structure, for "CSM Indoor" dataset. (Left) Results using whole plane method. (Right) Results using partial plane method.

Figure 7 (left) shows the estimated camera trajectories for the two methods, as well as the ground truth. As can be seen, the whole plane method tracks well in the beginning, but then the positional error starts to increase significantly. This is due to the fact that the number of tracked points starts to decline, because the whole plane tracking method cannot track partial planar templates for very long. Figure 7 (right) shows the number of inlier 3D points being tracked at each frame.

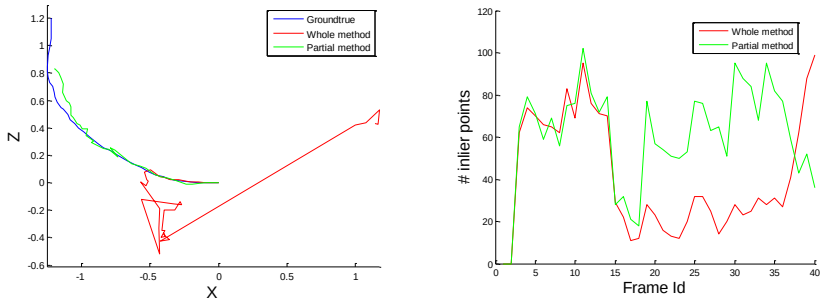

Fig. 7. "CSM Indoor" scene. (Left) Camera trajectories. (Right) Number of tracked points.

The remaining dataset is from the Computer Vision Group at Zhejiang University, and has been used by the research community to evaluate structure-from-motion algorithms [10]. A sample image and the estimated camera poses and 3D point positions are shown in Figure 8.

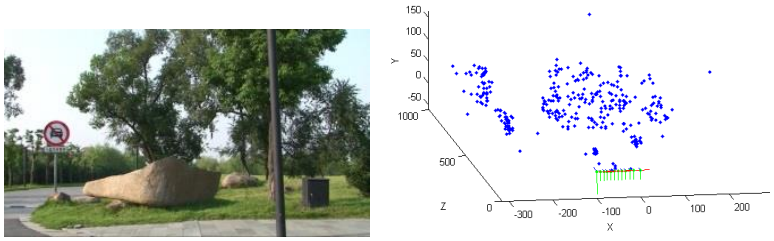

Fig. 8. "ACTS Road" scene. (Left) Sample image from dataset. (Right) Reconstructed scene, showing 3D point locations and camera poses at keyframes for the partial planar method.

Table 1 summarizes the results from the three datasets. As can be seen, features can be tracked for a greater number of frames using the partial plane method than with the whole plane method. The average number of inlier 3D points being tracked in each frame is greater using the partial plane method. The partial plane method also yields significantly more accurate camera poses.

5 Conclusions

We have presented a novel algorithm to segment and track partial planar templates, using a sequence of images from a moving camera. Unlike existing algorithms that

assume a feature arises from a single planar patch, our algorithm can handle the case where the patch encompasses more than one surface. We showed that our algorithm can estimate and track such features over a longer time, compared to algorithms that assume a patch contains only a single plane. We also showed that the method significantly improves the performance of a VSLAM algorithm, in terms of the accuracy of motion and structure estimates.

Table 1. Experimental results comparing whole plane and partial plane methods. RMSE (root mean squared error) of camera translation (in meters) and orientation (in radians) is shown with respect to ground truth. We did not have ground truth camera orientations for the CSM sequences, so RMSE of orientation is unknown for those.

Sequence	*Tracking plane method*	*Mean # frames tracked*	*Mean # inlier 3D points*	*RMSE translation*	*RMSE orientation*
CSM Indoor	Whole plane	4.6	41.4	1.18	-
	Partial plane	5.7	60.9	0.20	-
CSM Outdoor	Whole plane	7.5	34.1	2.51	-
	Partial plane	8.3	54.8	1.23	-
ACTS Road	Whole plane	51.3	18.0	24.97	0.394
	Partial plane	85.8	30.9	2.15	0.035

References

1. J. Shi and C. Tomasi (1994). "Good features to track." *Proc. of CVPR*, pp. 593-600, 1994.
2. A. Davison, I. Reid, N. Molton, and O. Stasse (2007). "MonoSLAM: Real-time single camera SLAM." *IEEE Trans. on PAMI*, 29(6):1052-1067.
3. G. Klein and D. Murray (2007). "Parallel tracking and mapping for small AR workspaces." *Proc. of ISMAR '07*, pp. 1-10.
4. S. Baker and I. Matthews (2004). "Lucas-Kanade 20 years on: a unifying framework". *Int'l J. of Computer Vision*, Vol. 56, pp. 221-255.
5. A. Masoud and W. Hoff (2014). "Segmentation and tracking of partial planar templates," *Proc. of Winter Conf. on Applications of Computer Vision*, pp. 1128-1133.
6. D. Comaniciu and P Meer (2002). "Mean shift: A robust approach toward feature space analysis." *IEEE Trans. on PAMI*, 24(5):603–619.
7. J. Lagarias, *et al*. (1998). "Convergence properties of the Nelder-Mead simplex method in low dimensions." *SIAM Journal of Optimization*, 9(1):112-147.
8. S. Thrun, W. Burgard, and D. Fox (2005). "Probabilistic Robotics." *MIT Press*.
9. M. Fischler and R. Bolles (1981). "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography." *Comm. of ACM*, 24(6):381-39.
10. G. Zhang, J. Jia, T. Wong, and H. Bao (2009). "Consistent depth maps recovery from a video sequence." *IEEE Trans on PAMI*, 31(6):974-988.