Segmentation and Tracking of Nonplanar Templates to Improve VSLAM

Abdelsalam Masoud, William Hoff

Department of Electrical Engineering and Computer Science, Colorado School of Mines, Golden,Co, USA, 80401

Abstract

In this paper we address the problem of visual simultaneous localization and mapping (VSLAM) using a single camera as the sole sensor. A VSLAM system estimates its position and orientation (pose) by tracking distinct landmarks in the environment using its camera. Most approaches detect feature points in the environment, using an interest point operator that looks for small textured image templates. Existing algorithms typically assume that an image template is the projection of a single planar surface patch. However, if the template is actually the projection of a nonplanar surface, tracking will eventually fail. We present an algorithm that estimates the 3D structure of a nonplanar template as it is tracked through a sequence of images. Using the 3D structure, the algorithm can more accurately predict the appearance of the template, and as a result, is better able to track the template. We then evaluate the benefit of using the new feature tracking method in VSLAM, and demonstrate that the new algorithm can track points longer, and achieve better accuracy, than if the standard single-plane feature tracking method is used. The approach is especially effective in scenes where surface discontinuities are common.

Keywords: `Tracking, visual SLAM, structure from motion.`

1. Introduction

For many mobile robot applications, it is important to estimate the robot's location and map its environment. A mobile robot must answer two primary questions: "Where am I?" and "What is the structure of my environment?" Navigating an unknown environment is a key task for mobile robot applications. The robot needs to keep track of where it is in the world, "localization", and simultaneously build a map of the environment from a sequence of landmark measurements, "mapping", to navigate through it. Localization and mapping must be performed simultaneously, a problem commonly known as Simultaneous Localization And Mapping (SLAM). The SLAM problem has been studied extensively in the robotics community [1].

Visual sensors (*i.e.*, cameras) are attractive for SLAM due to their low cost and low power. Much research has been performed on visual SLAM (VSLAM). Some methods use a single camera [2], [3] and others use stereo cameras [4], [5]. We focus on the single camera (monocular) case in this paper. Single cameras are more ubiquitous than stereovision cameras. They are present on electronic consumer products such as laptops, PDAs, and cellular phones. In addition to robot navigation, VSLAM has many other applications, including augmented reality and generating dense three-dimensional models from video.

A VSLAM system estimates its position and orientation (pose) by tracking distinct landmarks in the environment using its camera. Most approaches detect feature points in the environment, using an interest point operator (*e.g.*, [6]) that looks for small textured image templates. These templates are then tracked through subsequent images, and their 3D locations, along with the camera poses, 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.*, [7, 8]). The Lucas-Kanade algorithm and its variants [9] are a good example of methods to track planar templates. These algorithms

Email addresses: `amasoud@mines.edu` (Abdelsalam Masoud), `whoff@mines.edu` (William Hoff)

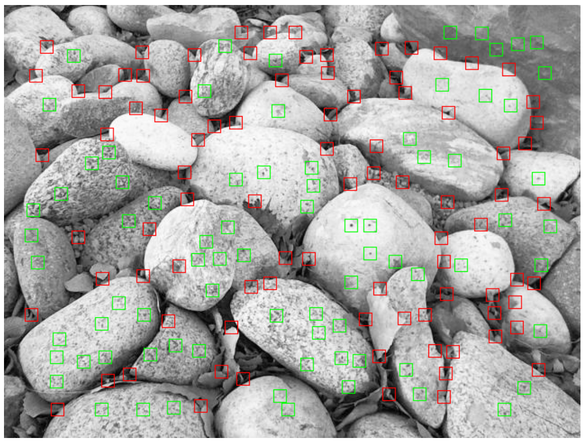


Figure 1: Interest points detected in outdoor rocks scene. Templates that appear to encompass more than one surface are shown as red; the others are shown as green.

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 camera 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 a distinctive image texture. A single homography transformation cannot handle the change in appearance of the tracked image template. Even small camera motion will cause tracking to fail in such cases.

Some environments have many nonplanar surfaces. Figure 1 shows the top 167 points that were automatically detected by an interest point operator [6], using a template window size of 15×15 pixels. By visual inspection, 89 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. Although this analysis is qualitative, it does indicate that outdoor natural scenes can contain many discontinuities.

Discontinuities can also occur in manmade indoor environments. Figure 2 shows a scene containing a set of objects. Most of the objects are planar, but discontinuities occur at the boundaries of the planes. 154 interest points were automatically detected, and by visual inspection, 52 of these encompass more than one surface.

Consider the patch that has a blue circle around it, as shown in Figure 2. This patch encompasses two surfaces: a portion of a leaf is in front of a background. We automatically tracked this patch through a sequence of images as the camera translated to the right, using the new method developed in this paper. Figure 3 shows five images from this sequence. The image region corresponding to the leaf remains relatively constant, but the other portion of the template changes appearance drastically. Nevertheless, we were still able to track this patch through the sequence. A method that assumed that the patch consisted of a single surface would not be able to track this template.

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. Davison [7] found that a wide field of view (FOV) lens improved the accuracy of VSLAM versus a narrow FOV because the features are visible for a longer tracking distance as the camera moves.This also improves the accuracy of the derived camera poses. However, when a feature

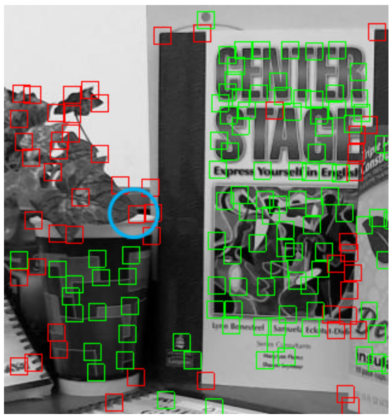


Figure 2: Example of indoor scene containing planar and nonplanar image patches. Templates that appear to encompass more than one surface are shown as red; the others are shown as green.

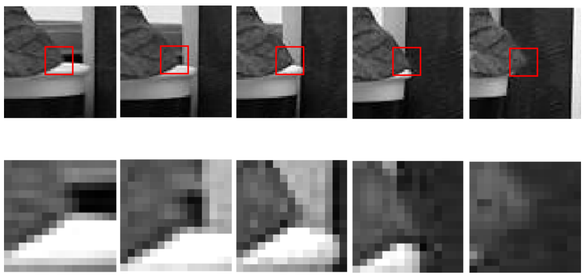


Figure 3: One patch from Figure 3 was tracked through a sequence of five images. Although the overall appearance of the template changes, the portion corresponding to the leaf remains constant.

is observed over a larger visual angle, it will have a larger change in appearance.

Our hypothesis is that by estimating the actual shape of a nonplanar image patch, its appearance can be more accurately predicted from different viewpoints; therefore, it is possible to track the nonplanar image patch over a larger visual angle. This is expected to improve the accuracy of VSLAM.

The contribution of this work is the development of an algorithm that estimates the 3D structure of a nonplanar patch as it is tracked through a sequence of images. Our approach is to model an image patch as "partially planar". Namely, we assume that the image patch 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). As described in Section 2, we automatically identify the points belonging to the "dominant" plane of the patch, and estimate 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. We then incorporate the new feature tracking algorithm into a standard VSLAM approach, as described in Section 3. We then compare the accuracy of the derived poses to that of a similar VSLAM algorithm that uses the standard single-plane feature tracking algorithm (Section 4). A preliminary version of this work was published in [10].

Below is a list of assumptions:

- As in most VSLAM algorithms, we assume that the scene is static and only the camera is moving. However, the algorithm should be able to tolerate small regions of the image corresponding to moving objects, since it is able to identify and reject outliers.
- As in most VSLAM algorithms, we use gray scale images, rather than color.

- We make no assumptions about the relative brightness of the regions (*e.g.*, that the foreground region is uniformly brighter or darker than the background region).

- We require no other sensors (such as a compass or inertial sensors), although the information provided by additional sensors could potentially be used to improve accuracy.

A related field of study is called "Structure from Motion" (SfM), which also attempts to recover camera poses and scene structure from a set of images. Unlike VSLAM, SfM algorithms typically operate in batch mode (*i.e.*, they process the images after all have been collected), and are not real time. SfM algorithms have been very successful in large scale reconstruction (*e.g.*, [11]) in a variety of scenes.

The difference in our work is as follows: Existing SfM algorithms typically track many points (*e.g.*, many thousands). With so many points, some of them are likely to be on planar surfaces. Therefore, even though tracking fails for points on nonplanar surfaces, there are enough remaining points on planar surfaces for these algorithms to estimate camera motion adequately. However, in VSLAM applications, we need to limit the number of points being tracked to maintain real-time operation. If a significant percentage of these points are on nonplanar surfaces, then failure to track them can result in a very low number of remaining points. This can adversely affect the accuracy.

Our work therefore addresses the case in which we are tracking a relatively low number of points (to allow real-time operation on limited computing hardware), and a high percentage of points are on nonplanar surfaces. In these cases we show that our new method dramatically improves the accuracy of VSLAM (as detailed in Section 4).

2. Modeling and tracking of partial planar templates

This section describes the new method to model and track 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 [6] 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. Let $\mathbf{n}$ be the unit vector that is normal to this plane. However, instead of requiring that *all* points in the template be projections of the same plane, we relax this requirement to say that only *some* of the points in the template are projections of the plane; hence we use the name "partial planar template". We call this plane the "dominant plane" for the template. The other points may be projected from other surfaces (we do not try to model these other surfaces; only the dominant plane).

In particular, we use a "probability mask" to represent the likelihood that each pixel is on the dominant plane. Let $p_{ref}(\mathbf{x})$ represent the probability that image point $\mathbf{x}$ in the reference image belongs to the dominant plane (D) for the template. Initially, $p_{ref}(\mathbf{x}) = 0.5$ for all points in the template, meaning that we have no knowledge of whether a point is on the dominant plane or not. Note that for conventional algorithms, $p_{ref}(\mathbf{x})$ is always equal to 1.0 for all pixels, because they assume that the template is the projection of a single planar surface. A related approach is that of [12], who also use a mask to indicate which pixels lie on a plane. However, their mask is created by thresholding the pixel variance using an empirically defined threshold; whereas our mask is rigorously defined using probability.

Note that the dominant plane could be either the foreground surface (*i.e.*, closer to the camera) or the background surface (*i.e.*, further from the camera). The name "dominant plane" simply means that it corresponds to the largest area of the template. It is possible that the algorithm will start tracking a background surface as the dominant plane. When this happens, tracking can fail if a foreground surface starts to occlude the background surface. We show examples of this in Section 4.

Initially, we have no 3D information on new templates. After the camera has translated sufficiently far, we can perform triangulation to estimate the 3D position of the center of the template. We also initialize the surface normal vector of the template to point directly away from the camera.

As each new image is acquired, our method incrementally updates the structure and probability mask of each template. Thus, the template can be tracked more reliably in subsequent images, since its appearance can be predicted more accurately. This approach is similar to that used by [13, 14, 15], who also track image regions and segment them using probability masks. However, their methods were not applied to VSLAM. In VSLAM, the characteristics of the tracked templates are very different. For example, they are much smaller and can be considered locally as the projection of a planar surface.

The subsections below describe the steps in detail: (1) matching, (2) updating the surface normal, and (3) updating the probability mask. The entire process is illustrated on a synthetic image.

2.1. Matching

This section describes the processing steps needed to match a template from the reference image to a new image.

If the template has 3D information (*i.e.*, we have already performed triangulation or bundle adjustment on this point), we use the predicted pose of the camera to determine where to search for the template in the new image. In this work, the pose of the camera in the previous image is simply used as the predicted pose (however, a more accurate method would be to use the estimated velocity of the camera to predict the pose). The location of the template center $\mathbf{x}_c$ is predicted in the current image by projecting the 3D point of the template center onto the current image, using the pose of the current camera.

If the template has only 2D information (*i.e.*, it is a new template and we have not yet performed triangulation or bundle adjustment on this point), then we simply center the search for the template in the new image at the same location that it was found in the previous image.

It is then necessary to warp the template from the reference image to the current image in order to predict its appearance, so that it can be matched in the current image. A plane is represented by the equation $\mathbf{n}^T\mathbf{X} = d$, where $\mathbf{n}$ is the normal vector, d is the perpendicular distance to the origin, and $\mathbf{X}$ 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 $\mathbf{K}$ is the camera intrinsic parameter matrix, $\mathbf{n}$ is the surface normal, and $\mathbf{R}$ and $\mathbf{t}$ are estimated rotation and translation between the cameras, respectively [7]. Note that d can be found from the location of the template center using $d = \mathbf{n}^T\mathbf{X}_0$.

Let $\mathbf{w}(\mathbf{x};\mathbf{P})$ denote the warping function that implements the homography transformation as described above, where $\mathbf{P}$ is a vector of parameters, consisting of the set $\mathbf{R},\ \mathbf{t},\ \mathbf{n}$, d. The warp $\mathbf{w}(\mathbf{x};\mathbf{P})$ takes the pixel $\mathbf{x}_1$ in the first image and maps it to location $\mathbf{x}_2 = \mathbf{w}(\mathbf{x}_1;\mathbf{P})$ in the second image. If the template has only 2D information, then the warping is just the identity transformation (*i.e.*, the predicted template is the same as the reference template). The template is warped to the current image using

$$T^{(\mathbf{P})}_{curr}(\mathbf{x}) = T_{ref}(\mathbf{w}(\mathbf{x};\mathbf{P})) \tag{2}$$

The superscript $\mathbf{P}$ is used to emphasize that the result depends on the parameters $\mathbf{P}$. Similarly, let $p^{(\mathbf{P})}_{curr}(\mathbf{x})$ be the probability mask for the template, warped from the reference to the current image, using the same warp. Figure 4 shows an example of a 3D template extracted from an image. The left column shows the template from the reference image warped to the current image. The right column shows the probability mask from the reference image warped to the current image.

To match the template to the current image, a 2D region around the predicted location is exhaustively searched to minimize the sum of squared differences (SSD), weighted by the probability at each pixel:

$$\triangle\mathbf{x} = \underset{\triangle\mathbf{x}}{\operatorname{argmin}} \sum_{\mathbf{x}\in N(\mathbf{x}_c)} |T^{(\mathbf{P})}_{curr}(\mathbf{x}) - I_{curr}(\mathbf{x}+\triangle\mathbf{x})|^2 p^{(\mathbf{P})}_{curr}(\mathbf{x}) \tag{3}$$

where $N(\mathbf{x}_c)$ is the $H \times W$ neighborhood surrounding the predicted location of the template center. In this work, the size of the search neighborhood was simply fixed as 80 pixels in height by 160 pixels in width. A more efficient method would be to determine the size by considering the uncertainty in the camera pose and the uncertainty in the 3D location of the point. This could result in a smaller search area, especially when

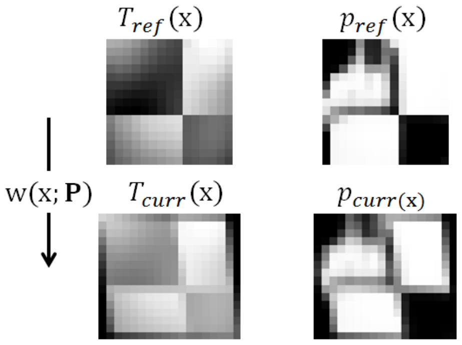


Figure 4: Template and probability mask, warped from reference to current image.

the uncertainties are small. However, although less efficient, the fixed size neighborhood that we used was sufficient to successfully track points in all the datasets used in this paper. The search neighborhood was larger in width than height because the motion of the camera was primarily horizontal translation in the datasets that were used.

In the case of a 2D template, we use a very conservative estimate for the size of the search neighborhood. We assume the point could have any depth between d_{min} and d_{max}. The width of the search region is determined by these limits. We set d_{max} to a very large value and d_{min} to a very small value to ensure that the search region is large enough to contain the point, regardless of its 3D location.

The probability mask values are high for pixels on the dominant plane, and low for pixels that are not on the dominant plane. Thus, by weighting the sum of squared differences by the probability, the method ignores the pixels that are not on the dominant plane and their residual errors do not affect the result.

If the minimum sum of squared differences, weighted by the probability is below an empirically derived threshold (a value of 40.0 is used in this work, where the image pixel values range from 0..255), the template is considered to be successfully matched to the location $\mathbf{x}_0 = \mathbf{x}_c + \triangle\mathbf{x}$ in the current image. The value of 40.0 was chosen high enough so that the weighted SSD of all correctly matched points is below this value, regardless of image noise. Although some incorrectly matched points also pass this criteria, they are subsequently identified as outliers and are filtered out by the RANSAC-based pose estimation step (described in Section 3).

2.2. Updating the surface normal

After a template has been matched and the pose of the camera was estimated, its surface normal is updated. This step is performed only for 3D templates, since 2D templates do not yet have surface normals. Our method is similar to that used by [7] and [16]. The normal vector is parameterized by the two direction angles (θ, ϕ). The normal vector in spherical coordinates is

$$\vec{\mathbf{n}} = \begin{bmatrix} cos\phi \\ sin\phi cos\theta \\ sin\phi sin\theta \end{bmatrix} \tag{4}$$

where ϕ is the angle from the x axis and θ is the angle from the y-axis in the $y - z$ plane.

The angles are estimated by minimizing the sum of squared differences weighted by the probability:

$$(\theta, \phi) = \underset{\theta,\phi}{\text{argmin}} \sum_{\mathbf{x} \in N(\mathbf{x}_c)} |T^{(\mathbf{P})}_{curr}(\mathbf{x}) - I_{curr}(\mathbf{x} + \triangle\mathbf{x})|^2 p^{(\mathbf{P})}_{curr}(\mathbf{x}) \tag{5}$$

Note that this search varies the surface normal $\mathbf{n}(\theta, \phi)$, but centers the placement of the template at the offset $\triangle\mathbf{x}$ that was found from the matching step. Each change in the surface normal changes the appearance of the warped template and probability mask.

A non-linear optimization algorithm [17] is used for the search. Since there is usually a fairly good estimate of the surface normal from the previous images, there is usually a fairly small correction to the angles.

2.3. Updating the probability mask

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

$$T^{(\mathbf{P})}_{match}(\mathbf{x}) = I_{curr}(\mathbf{w}^{-1}(\mathbf{x};\mathbf{P})) \tag{6}$$

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

$$r(\mathbf{x}) = T_{ref}(\mathbf{x}) - T^{(\mathbf{P})}_{match}(\mathbf{x}) \tag{7}$$

If the point $\mathbf{x}$ belongs to the dominant plane, then the magnitude of the residual $r(\mathbf{x})$ should be small. This is because the warping transformation should accurately map the reference image to the current image, and the values at the corresponding points in the two images should be close. It is assumed that the residuals are normally distributed so that the probability of measuring residual r, given that the point is on the dominant plane, is $p(r|\mathbf{x} \in D) = N(\mu_D, \sigma^2{}_D)$, where μ_D and σ_D are the mean and standard deviation of residuals for points on the dominant plane.

Similarly, if the point $\mathbf{x}$ does not belong to the dominant plane, then the warping transformation does not accurately map this portion of the reference image to the current image. In this case, the warp can map the point to anywhere in the current image, and the magnitude of the residual in this case is expected to be large. We assume that the probability of the residuals is $p(r|\mathbf{x} \in \overline{D}) = N(\mu_{\overline{D}}, \sigma^2{}_{\overline{D}})$, where $\mu_{\overline{D}}$ and $\sigma_{\overline{D}}$ are the mean and standard deviation of residuals of residuals for points not on the dominant plane. In Appendix A, it is described how to automatically estimate the parameters μ_D, σ_D, $\mu_{\overline{D}}$, and $\sigma_{\overline{D}}$ from the image template.

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

$$p(\mathbf{x} \in D|r_{1:t}) = \eta p(r_t|\mathbf{x} \in D)p(\mathbf{x} \in D|r_{1:t-1}) \tag{8}$$

$$p(\mathbf{x} \in \overline{D}|r_{1:t}) = \eta p(r_t|\mathbf{x} \in \overline{D})p(\mathbf{x} \in \overline{D}|r_{1:t-1}) \tag{9}$$

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

2.4. Example of tracking

To illustrate our method, we show results on a synthetic image. Two planar surfaces were created such that the frontal surface was in a checkerboard pattern (Figure 5, left). Both planes were perpendicular to the first camera's line of sight. The two surfaces were textured with random noise using the same parameters. Next, interest points were detected in the first (reference) image (Figure 5, right), using 15×15 templates.

The camera was then translated to the right. The result of tracking one of the templates (the template outlined in red in Figure 5) is shown in Figure 6. This template encompasses two planar regions, because it is located at one of the corners of the checkerboard pattern. The left column of this figure shows the portion of the reference image (top row) from which the template (middle row) was extracted.

The evolution of the probability mask over time is shown in the bottom row. White values indicate a high probability of the point to belong to the dominant plane. By the end of the sequence, the algorithm appeared to correctly segment the template and identify the pixels belonging to the dominant plane, with only a few errors.

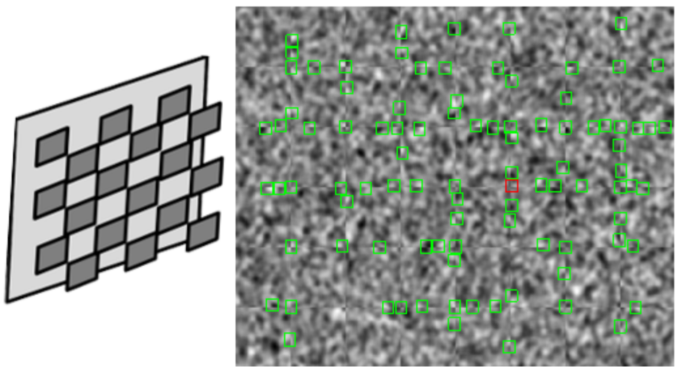


Figure 5: Synthetic image. (Left) 3D structure. (Right) Interest points.

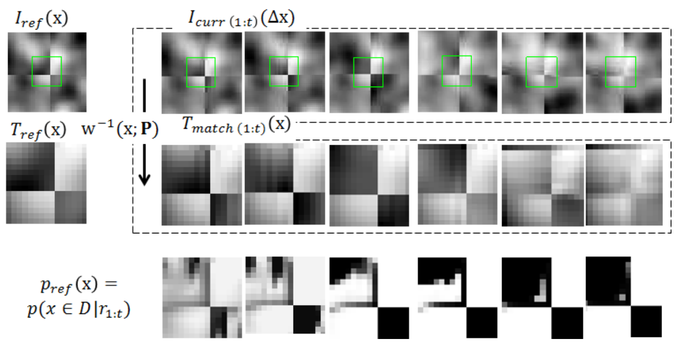


Figure 6: The reference template (left column) is tracked for 50 frames. The results for every 5th frame are shown. (Top row) Image region surrounding the tracked point. (Middle row) The matched template, warped back to the reference image. (Bottom row) Probability mask, where white indicates a high probability of the point to belong to the dominant plane.

3. Integration of tracking method with VSLAM

This section describes the integration of the new partial plane feature tracking method into a complete VSLAM algorithm similar to the algorithm developed by Klein and Murray, called Parallel Tracking and Mapping (PTAM) [8]. Like PTAM, we assume that the camera intrinsic parameters are known and the camera is moving smoothly through the environment. Similar to their approach, we perform a local bundle adjustment (BA) over a sliding window of keyframes.

The main differences between our VSLAM algorithm and theirs are as follows:

- We use the more accurate homography warping transformation to predict the appearance of templates, rather than the faster (but less accurate) affine transformation.
- We do not implement multi-threading and do not perform a global BA in a background thread. Global BA should improve the results, especially for longer image sequences, although the sequences that we used in this paper were all rather short. However, as shown in Section 4, our new feature tracking method significantly improves performance even for short sequences. If we did incorporate global BA and apply the system to longer sequences, we expect the improvement to be even larger.
- We initialize the scale factor by physically measuring the relative pose between two keyframes, rather than assuming an arbitrary displacement of 10 cm between the first two keyframes.

- We use the RANSAC algorithm to compute pose and eliminate outliers, rather than a least squares approach with a robust M-estimator. While the latter approach could possibly work on our datasets, we chose the RANSAC method due to its tolerance to outliers caused by dynamic scene changes.

We chose the PTAM algorithm as a model because the algorithm is well known and excellent results have been shown in the literature. However, the feature tracking method developed in this paper could be used in any VSLAM algorithm that detects and tracks feature points (for example, an algorithm based on Kalman filtering, such as [2] EKF-based algorithm).

As will be described in the next section, we compare the performance of the VSLAM algorithm using the new "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 for all points, 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.

In PTAM [8], an image is designated as a "keyframe" if the camera has moved sufficiently far from any previous keyframes. The minimum distance used depends on the mean depth of observed features, so that keyframes are spaced closer together when the camera is near a surface, and further apart when observing distant objects.

In our work, we use the same criteria as PTAM to designate images as keyframes. However, we manually select keyframes instead of letting the algorithm automatically choose the keyframes. The reason is that we want to do an accurate comparison of the partial plane feature tracking method with the whole plane tracking method. We want both versions of the algorithm to use the same keyframes, so that the pose accuracy at those same keyframes can be directly compared. If such a comparison was not needed, then automatic keyframe selection could be used.

The processing begins by acquiring the first image of the sequence. Feature interest points are detected, and this image is used as the first keyframe. Subsequent images are processed according to the algorithm whose pseudocode is given in Algorithm 1.

```
1  while input data is available do
2      Get next image ;
3      Track existing points to the new image ;
4      Estimate pose from 2D to 3D point correspondences;
5      Initialize new 3D points using triangulation;
6      Update the matched points that are inliers ;
7      if this pose is a keyframe then
8          Do bundle adjustment over last N keyframes;
9          Acquire new points to track ;
10     end
11 end
```

Algorithm 1: VSLAM algorithm pseudocode.

The following example illustrates the processing steps on the synthetic image sequence described in Section 2.4. The processing begins by detecting interest points in the first image (Figure 7), using 15 $\times$ 15 templates. We use the Shi-Tomasi method [6] which finds points at which both eigenvalues of the autocorrelation matrix are high. A threshold of 10 was used for the minimum eigenvalue. From this image, 172 interest points were detected, and a significant fraction of them encompass more than one surface. To illustrate the processing, one of the templates (outlined in red) was chosen and is shown in the expanded view. This template encompasses two planar regions because it is located at one of the corners of the checkerboard pattern.

The algorithm now enters the main processing loop in which it acquires and processes each new image in the sequence.

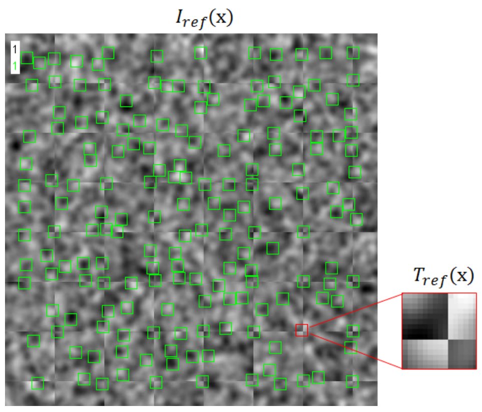


Figure 7: Interest points that were detected in the first image, along with an expanded view of one of the image templates.

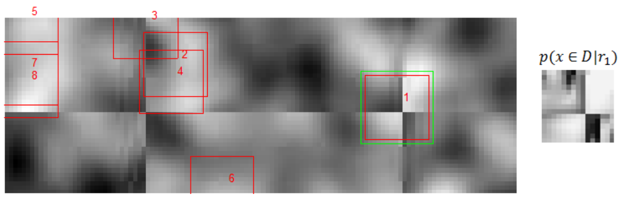


Figure 8: Example of the matched template in the current image (left image), and the estimated mask probability of the dominant image plane (right image).

3.1. Initial processing with 2D templates

Initially, there are no 3D templates, and the algorithm tracks only 2D templates. Templates are tracked to the current image (line 3 in the pseudocode). In the case of a 2D template, the search region is defined by the upper and lower limits of the depth of the point, d_{max} and d_{min}. In this example, d_{max} was set to a very large value (1000) and d_{min} was set to a very small value (1). Figure 8, left shows the search region in the second image for the template from Figure 7.

The location in the search region with the minimum SSD is found, by exhaustively searching the region. Figure 8 shows the top 8 locations with the smallest SSD values. The location with the minimum SSD is labeled "1" and is the correct match.

At this point, the algorithm will try to estimate the pose of the camera from 2D to 3D point correspondences (line 4), if any are available. However, since no 3D points have been initialized yet, this step is skipped. Also since no pose is available, the triangulation step is skipped (line 5).

The next step is to update the matched points (line 6). In the case of 2D points, this step updates the probability mask $p_{ref}(\mathbf{x})$, based on the residual errors, using the method described in Section 2.3. Figure 8, right shows the updated probability mask, conditioned on the first measured residuals. As can be seen, the upper right and lower left regions have relatively higher probability values. Thus, the algorithm is beginning to segment the dominant plane in this template.

The current image is not a keyframe (line 7), so steps 8-9 are skipped. Processing continues as above with additional images; that is, points are tracked and their probability masks are updated, until a keyframe is acquired. In this example, the 5^{th} image is a keyframe. We provide the relative pose between the first and second keyframes to the algorithm. This establishes the scale factor [8].

Once the pose between the first two keyframes is known, the system can perform triangulation to estimate 3D point locations (line 5). In order to triangulate a point, the visual angle between the two observations must be greater than a threshold (we use a threshold of 2.0 degrees); otherwise the uncertainty is too large.

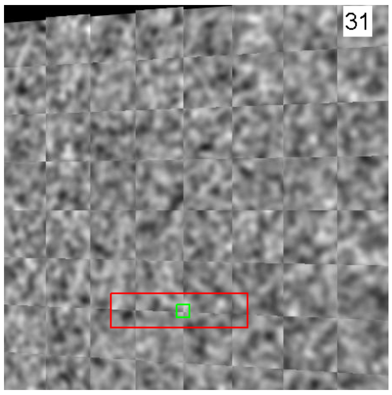


Figure 9: Search region in image 31 (red) with the location of the matched template (green).

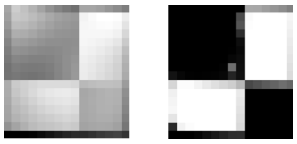


Figure 10: Result of warping the template and probability mask to image $I_{31}(x)$. (Left) Warped template $T_{31}(\mathbf{x})$. (Right) Warped probability mask $p_{31}(\mathbf{x})$.

For newly initialized 3D points, we also initialize the surface normal vector of the template to point away from the camera.

3.2. Processing with 3D templates

We now describe the processing steps after some 3D points have been initialized. 3D templates are tracked to the current image (line 3 in the pseudocode). The predicted location of the template in the current image is determined by projecting the 3D point of the template center onto the current image, using the predicted pose of the current camera. In this work, we simply use the last camera pose as the predicted pose. Figure 9 shows the search region in the 31^{st} image of the sequence, for the example template from Figure 7.

To match the template to the current image, the reference template must be warped to its expected appearance in the current image, using the method described in Section 2.1. Figure 10 shows the result of warping the reference template $T_{ref}(\mathbf{x})$ to the current image $T_{31}(\mathbf{x})$. Also shown is the probability mask warped to the current image, $p_{31}(\mathbf{x})$.

Next, the matching location in the current image is estimated by minimizing the SSD errors weighted by the probability; *i.e.*,

$$\triangle\mathbf{x} = \underset{\triangle\mathbf{x}}{\operatorname{argmin}} \sum_{\mathbf{x}\in N(\mathbf{x}_c)} |T_{31}^{(\mathbf{P})}(\mathbf{x}) - I_{31}(\mathbf{x}+\triangle\mathbf{x})|^2 p_{31}^{(\mathbf{P})}(\mathbf{x}) \tag{10}$$

The matching location is shown by the green box in Figure 9.

The pose estimation step (line 4) uses the Perspective-n-Point (PnP) [18] method. This method estimates the pose of the camera from 2D to 3D point correspondences. To identify outliers due to mismatched points, we use a method called MLESAC (Maximum Likelihood Estimator Sample Consensus) [19]. This algorithm is very similar to RANSAC [20] but it evaluates the likelihood of the hypothesis by representing the error distribution as a mixture model, thus avoiding the use of arbitrary thresholds. In this example, the pose estimation step found 182 inlier 2D to 3D point correspondences, as shown in Figure 11.

The updating step (line 6) for 3D points consists of three steps:

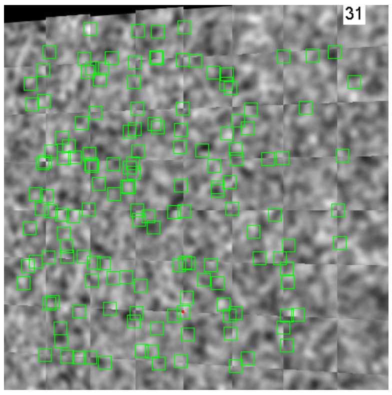


Figure 11: Inlier points after pose estimation. The matched location of the example template is labeled by a red spot.

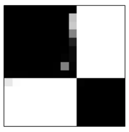


Figure 12: Updated probability mask for the example template, $p(x = D|r_{1:31})$.

1. The point is triangulated again using the current image observation and all previous image observations. However, if the point location has already been computed by bundle adjustment, we skip this step. The rationale is that BA provides the most accurate 3D location information for the point, but if BA has not yet been performed for this point, triangulation using all observations provides the most accurate 3D location information.
2. The surface normal vector for the template is updated using the method described in in Section 2.2.
3. The probability mask for the template is updated using the method described in Section 2.3.

Figure 12 shows the updated probability mask for the template after image 31 has been processed. As can be seen, by this point the algorithm has cleanly segmented the dominant plane.

Image 32 happens to be a keyframe (line 7), so BA is performed over the last $N = 3$ keyframes (line 8). In our application, we do not have loop closures. We follow the approach of [21, 8] and others that do real time estimation, and use a "sliding window" of keyframes. This allows us to limit the number of points and poses that need to be solved for.

Let A be the set of the most recent N keyframes (including the current keyframe) which we want to solve for. Let P be the set of points that are visible in any of these keyframes. Let the set B contain any additional keyframe for which a measurement of any point in P has been made. Local bundle adjustment optimizes the poses of the most recent N keyframes, and all of the map points seen by these, using all of the measurements ever made of these points. Therefore, minimization becomes:

$$\{\{\beta_i\}, \{\mathbf{X}_j\}\} = \underset{\{\beta_i\}\in A, \{\mathbf{X}_j\}\in P}{\operatorname{argmin}} \sum_{i\in A\cup B} \sum_{j\in P} |\mathbf{x}_{ij} - \mathbf{f}(\mathbf{X}_j, \beta_i)|^2. \quad (11)$$

where $\mathbf{x}_{\mathrm{ij}}$ are the measured image points. The function $\mathbf{f}()$ estimates the 2D predicted image points from a projected 3D point $\mathbf{X}_j$ using the camera pose β_i.

We collect all points that were observed in the N keyframes, and optimize their 3D locations as well as the poses of the $N - 1$ keyframes, to minimize the reprojection error of the points. The result is shown in Figure 13.

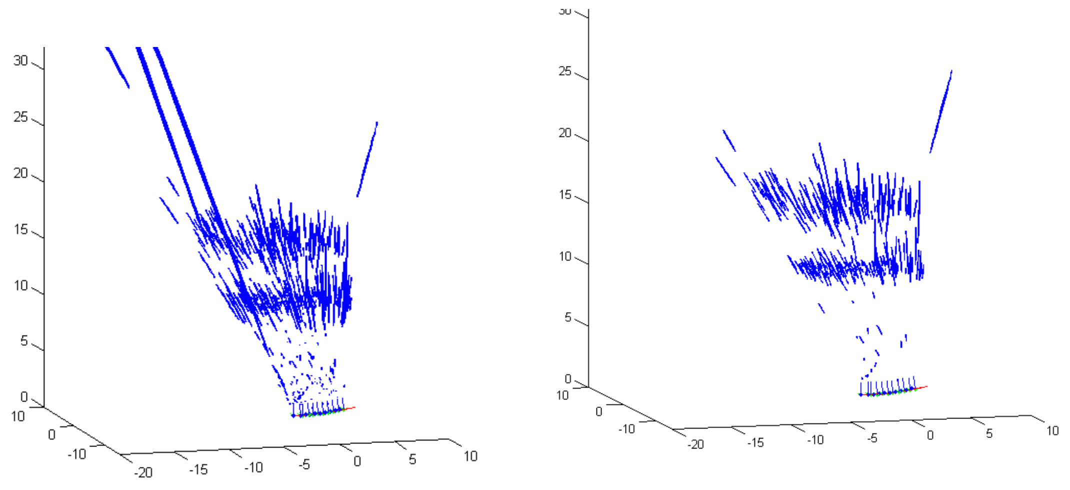


Figure 13: Estimated 3D map and estimated camera locations before BA (left image), and after BA (right image). The uncertainties of point positions are shown as ellipsoids.

Finally, we acquire new points (line 9) if the number of points being tracked in the current image falls below a threshold. The threshold is an adjustable parameter; in this example we used a threshold of 200 points. As we show in the next section, increasing the number of points provides more accurate results, but is slower. Note that the actual number of detected points can be below the threshold, if the image does not have enough textured regions from which good interest points can be extracted. New points are required to be more than a minimum distance from existing points, in order to avoid clusters of points that are very close together. In this work we use a minimum local distance of 1.5 * *(size of the tracking template)* = 1.5*(15)= 23 pixels. In other words, the centers of the templates cannot be closer than 23 pixels from each other.

Figure 14 shows new feature points added at this step, to reach a total of 200 points. Newly added 2D points are labeled by red squares with size 15×15, and existing inlier 3D points are labeled by green squares with size 15×15. The small blue points indicate that the point was used for BA.

The VSLAM algorithm continues to read new images from the sequence and repeats the previous steps until reaching the last image in the data set.

4. Experimental results

This section provides an evaluation of the performance of the VSLAM algorithm using a number of different datasets; including synthetic, indoor and outdoor sequences. We compare the performance of the VSLAM algorithm using the new "partial plane" tracking method (PPM) to the same algorithm using the "whole plane" tracking method (WPM). The whole plane method is identical to the partial plane method, except that image templates are not segmented into a planar region and a nonplanar region; instead, the entire template is modeled as a single planar region. Therefore, any improvement in performance can be attributed to the use of the new partial plane tracking method.

4.1. Datasets

The hypothesis of this paper is that by estimating the actual shape of a nonplanar image patch, the corresponding image template can be tracked over a larger visual angle, as compared to traditional algorithms that assume a template is the projection of a single planar patch. This is expected to improve the accuracy of VSLAM. To test this hypothesis, we chose datasets of scenes in which a substantial fraction of the images contained nonplanar patches. The datasets were also chosen such that points were visible over a large visual

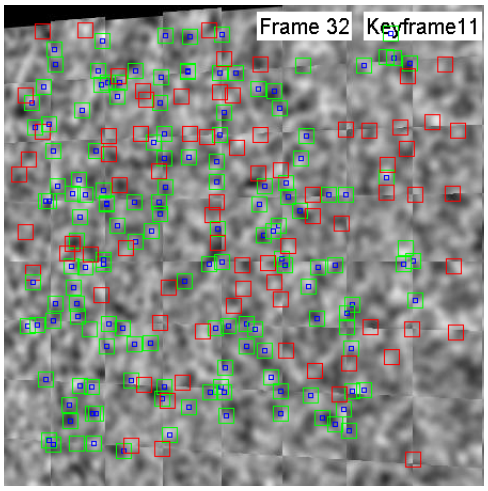


Figure 14: Example of newly acquired new 2D points (red) and existing inlier 3D points (green).

angle.

The second of the two criteria was satisfied when the camera motion was primarily a sideways motion; *i.e.*, the translation was perpendicular to the line of sight, with possibly a small amount of rotation about the vertical axis. Another type of motion that satisfies this condition is an “orbital” motion, in which the camera moves in a circular motion about a fixed point, and simultaneously rotates in order to keep that point in the field of view.

We used six datasets in the evaluation: One was a synthetic image dataset; two were datasets that we took ourselves, and three were datasets that have been used in other research on SfM and VSLAM. These are described in the subsections below.

4.1.1. Synthetic images

A synthetic image dataset was created to evaluate the algorithm. The synthetic dataset has the advantage that the pose of the virtual cameras and the structure of the scene are known perfectly. The scene was designed to contain a large fraction of nonplanar regions. It consists of two planar surfaces that were colored with a random image texture. The image texture consisted of uniform random noise, followed by processing by a Gaussian low pass filter with $\sigma = 3$. The two planes were oriented perpendicular to the camera in its first position, and were positioned at distances of 10 and 15 units from the camera, respectively. The plane closest to the camera was perforated in a checkerboard pattern, such that the plane further away was visible through the holes.

The camera was then translated in the $x - z$ plane. The motion was primarily to the right, with a small amount of forward motion (see Figure 15). Specifically, the trajectory of the camera followed a cosine function, of the form $Z = Acos(\frac{2\pi x}{\lambda})$, where the amplitude was $A = 1.2$ and the wavelength was $\lambda = 20$. The camera also rotated so that its line of sight was always perpendicular to the tangent of the trajectory.

The size of the images was set to 450×450 pixels. The field of view of the camera was 46 degrees, horizontally and vertically. Images were taken at increments of 0.3 units of translation in the x direction, for a total of 34 images. Figure 16 shows three images from the sequence, at times 1, 11, and 21.

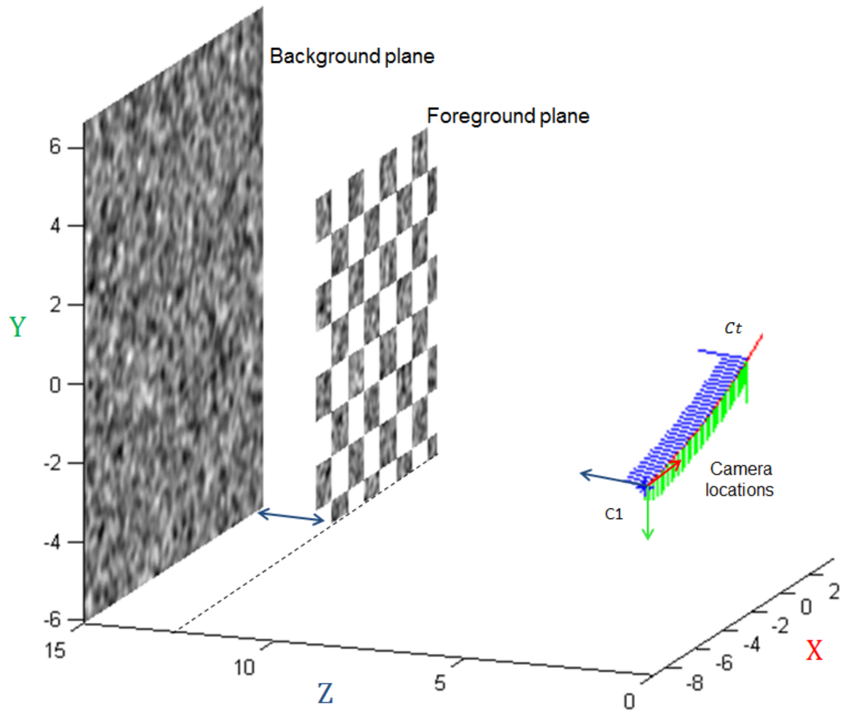


Figure 15: 3D structure of the scene for the synthetic images.

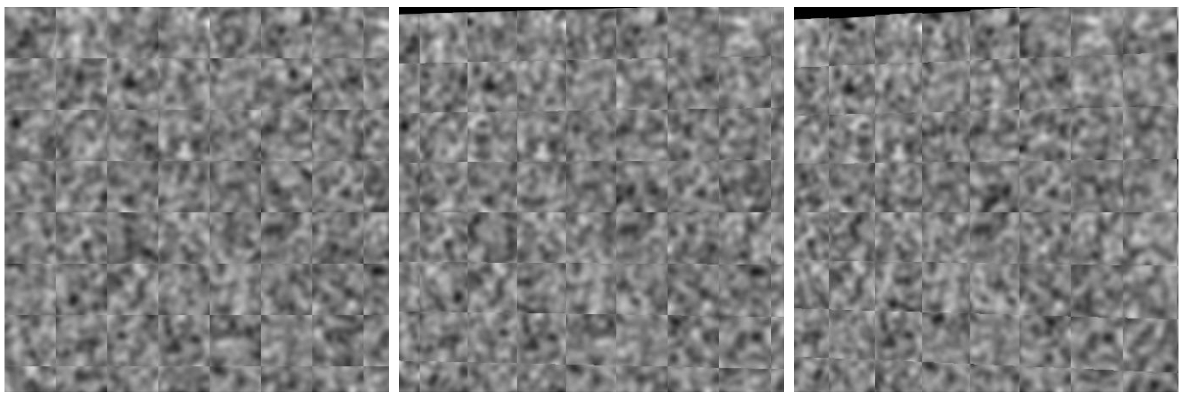


Figure 16: Images 1, 11 and 21 from the synthetic image sequence.

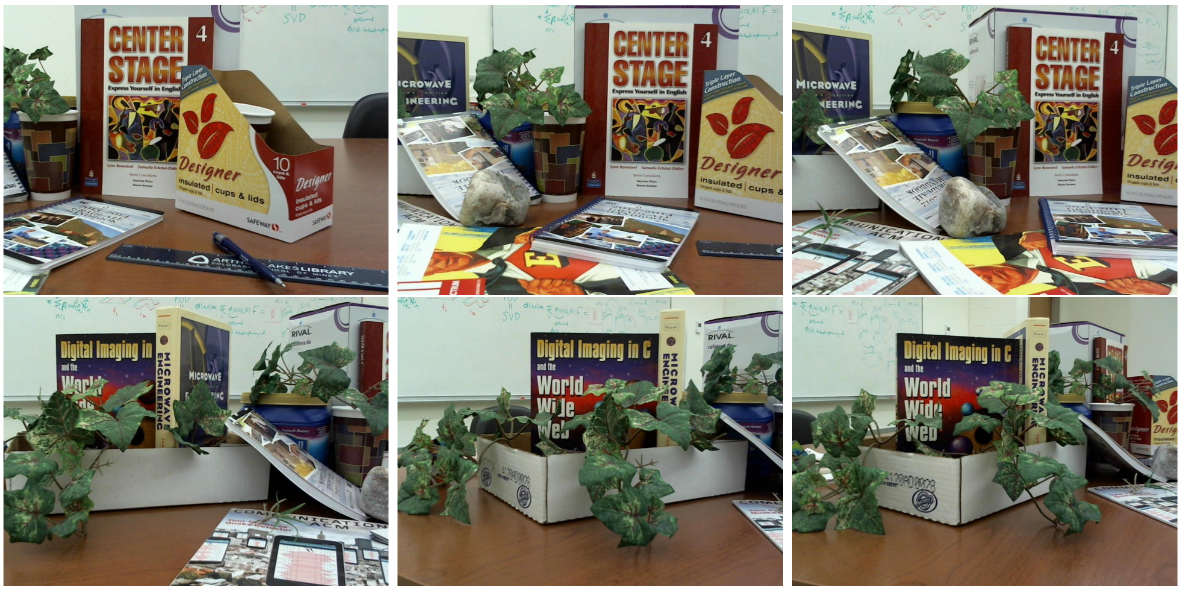


Figure 17: Images 1, 6, 11, 16, 21, and 26 from the "CSM Indoor" image sequence.

4.1.2. CSM datasets

We collected two image sequences using a Logictech USB camera, of an indoor scene and an outdoor scene. The images were 640 × 480 pixels and the camera intrinsic parameter matrix was

$$\mathbf{K} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 531.77 & 0 & 319.97 \\ 0 & 530.78 & 235.21 \\ 0 & 0 & 1 \end{bmatrix}$$

In each scene, the camera was mounted on a tripod, and was manually moved in a circular arc trajectory, and rotated such that the camera always pointed toward the center of the arc. We measured the position of the camera as each image was captured; however, the orientation of the camera was not measured. Thus, the ground truth consists only of the camera translational positions.

The "CSM Indoor" scene consisted of a set of objects on a table. The scene contained large planar regions (*e.g.*, the sides of the boxes, the table and the wall) as well as some portions that have a lot of nonplanar regions (*e.g.*, the leaves on the potted plant). The length of the sequence was 40 images. Although the lighting in the scene was constant during the recording of the images, the brightness of some objects changes somewhat as images are taken from different viewpoints. Figure 17 shows 6 images from the sequence, at times 1, 6, 11, 16, 21, and 26.

The "CSM Outdoor" scene consisted of large rocks in the foreground and bushes in the background. Large portions of the scene contain nonplanar regions. The length of the sequence was 50 images. Figure 18 shows 6 images from the sequence, at times 1, 6, 11, 16, 21, and 26.

4.1.3. ACTS datasets

We used three image sequences from the Automatic Camera Tracking System (ACTS) [22, 23] dataset. This dataset has been used to evaluate several SfM algorithms in the literature. The images are taken by a handheld camera and are of size 960 × 540 pixels. We used the SfM code provided by the authors to estimate the camera poses for each image, and used this as the ground truth.

The "ACTS Road" scene consists of rocks and trees that are relatively distant from the camera. The camera translates to the left without rotating during the sequence. The length of the sequence was 100

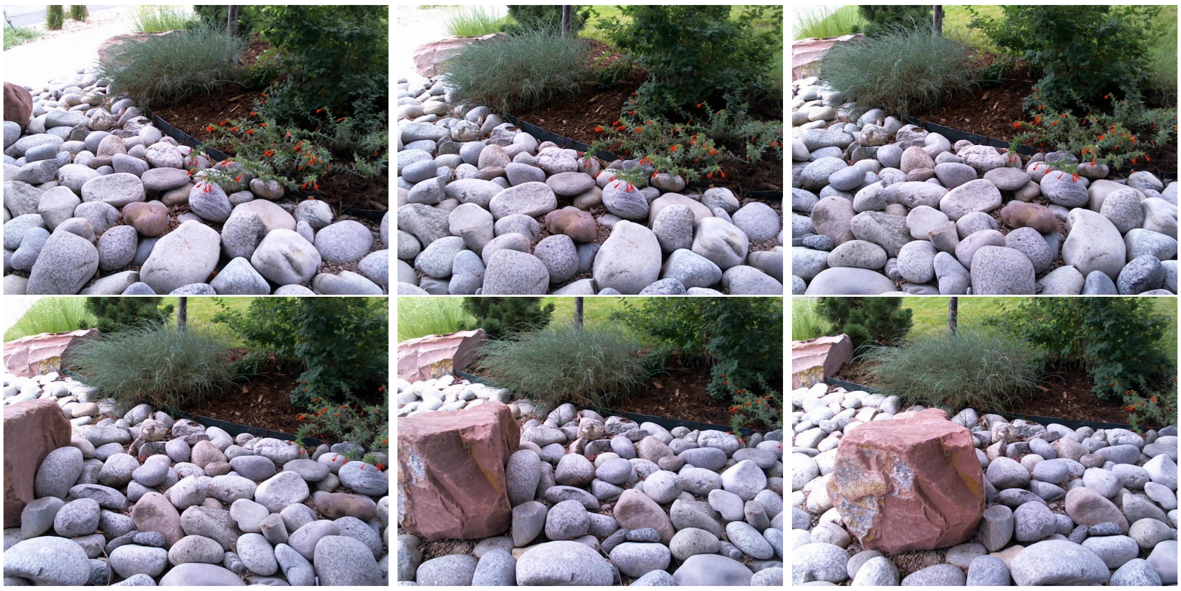


Figure 18: Images 1, 6, 11, 16, 21, and 26 from the "CSM Outdoor" image sequence.

images. The camera intrinsic parameter matrix as computed by the ACTS SfM code was

$$\mathbf{K} = \begin{bmatrix} 1221.23 & 0 & 479.5 \\ 0 & 1221.23 & 269.5 \\ 0 & 0 & 1 \end{bmatrix}$$

Figure 19 shows 6 images from the sequence, at times 1, 11, 21, 31, 41 and 51.

The "ACTS Flower" scene consists of a flower pot that is relatively close to the camera. The camera moves to the right in an "orbiting" motion around the flower pot, keeping it more or less centered in the field of view. The length of the sequence was 100 images. The camera intrinsic parameter matrix as computed by the ACTS SfM code was

$$\mathbf{K} = \begin{bmatrix} 1139.34 & 0 & 479.5 \\ 0 & 1139.34 & 269.5 \\ 0 & 0 & 1 \end{bmatrix}$$

Figure 20 shows 6 images from the sequence, at times 30, 40, 50, 60, 70, and 80.

The "ACTS Lawn" scene consists of a statue of a man on a bench that is relatively close to the camera. In the distant background are trees and buildings. The camera translates to the left and also rotates to keep the man in the field of view. The length of the sequence was 100 images. The camera intrinsic parameter matrix as computed by the ACTS SfM code was

$$\mathbf{K} = \begin{bmatrix} 1139.79 & 0 & 479.5 \\ 0 & 1139.79 & 269.5 \\ 0 & 0 & 1 \end{bmatrix}$$

Figure 21 shows 6 images from the sequence, at times 0, 20, 40, 60, 80 and 100.

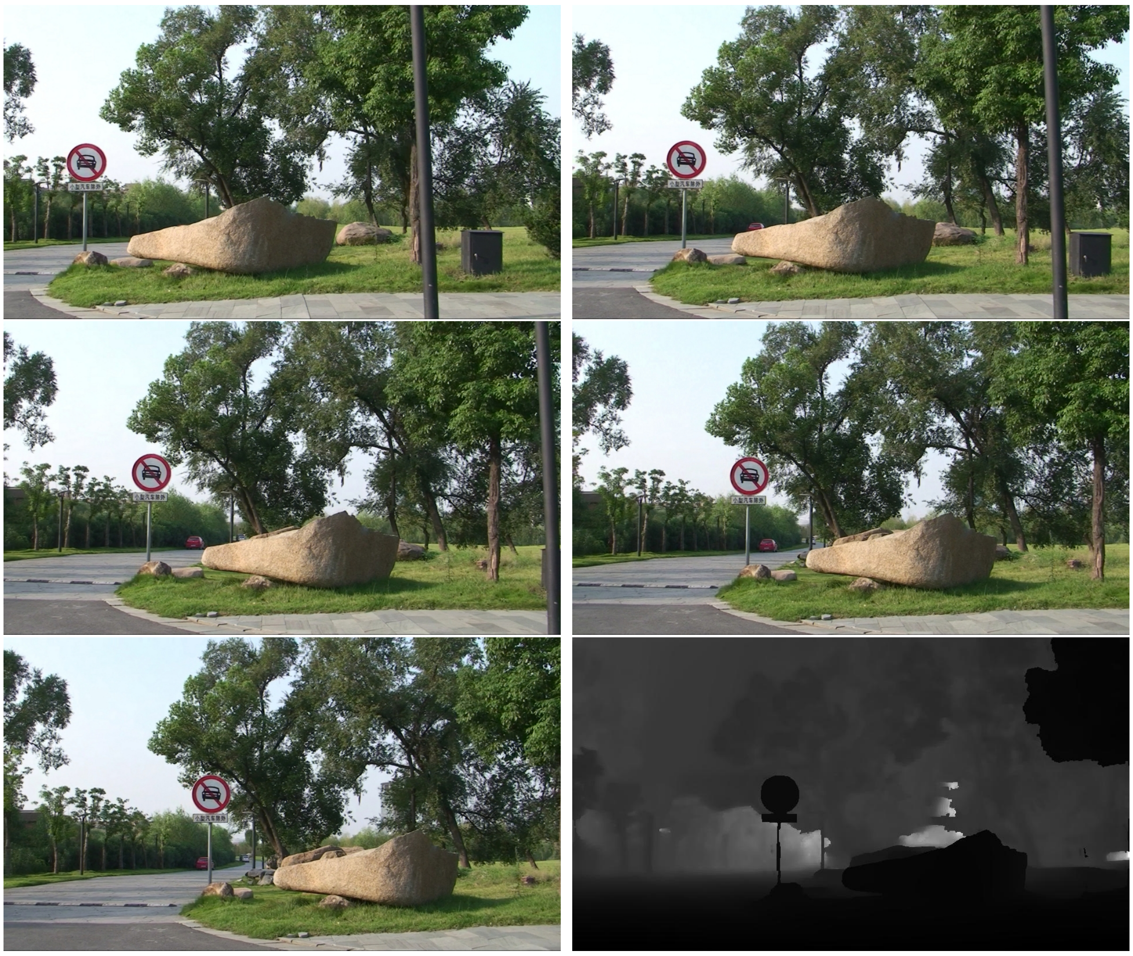


Figure 19: Images 1, 11, 21, 31 and 41 from the “ACTS Road” image sequence.

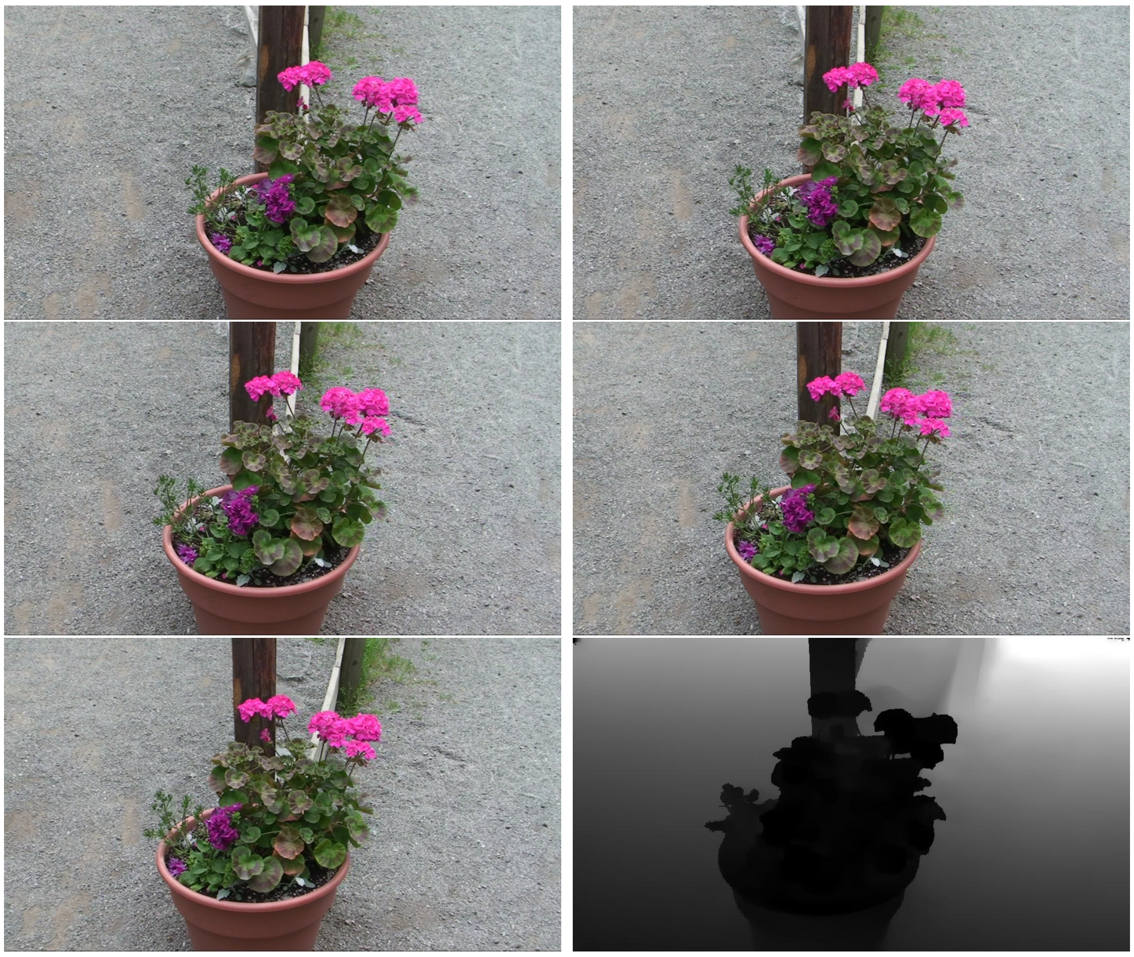


Figure 20: Images 30, 40, 50, 60, 70, and 80 from the "ACTS Flower" image sequence.

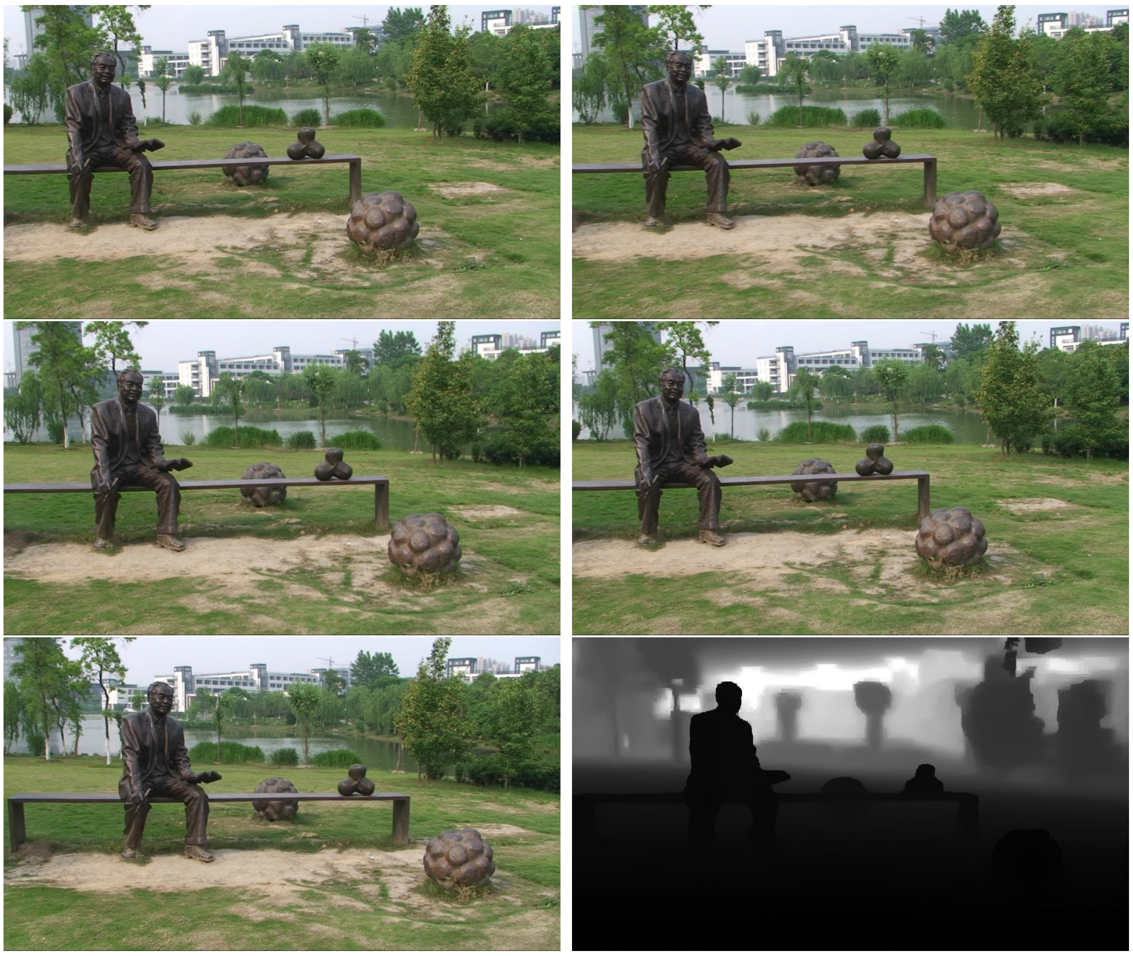


Figure 21: Images 0, 20, 40, 60, 80 and 100 from the “ACTS Lawn” image sequence.

4.1.4. KITTI datasets

The method is further evaluated on the KITTI odometry benchmark[24]. This dataset is taken from a forward-facing vehicle-mounted camera. The camera moves at high speed and is subject to significant lighting changes. This dataset is more challenging than our other datasets because features are not visible in the images for a long time. The camera images are cropped to a size of 1226 × 370 pixels.

The “KITTI 04 & 06” scenes consist of urban environments with moving objects such as other cars. The camera moves forward without rotating during the sequences. The length of each sequence was 135 images. The camera intrinsic parameter matrix of the KITTI 04 & 06 was

$$\mathbf{K} = \begin{bmatrix} 707.0912 & 0 & 601.8873 \\ 0 & 707.0912 & 183.1104 \\ 0 & 0 & 1 \end{bmatrix}$$

Figure 22 shows the first images of the KITTI 04 & 06 sequences.

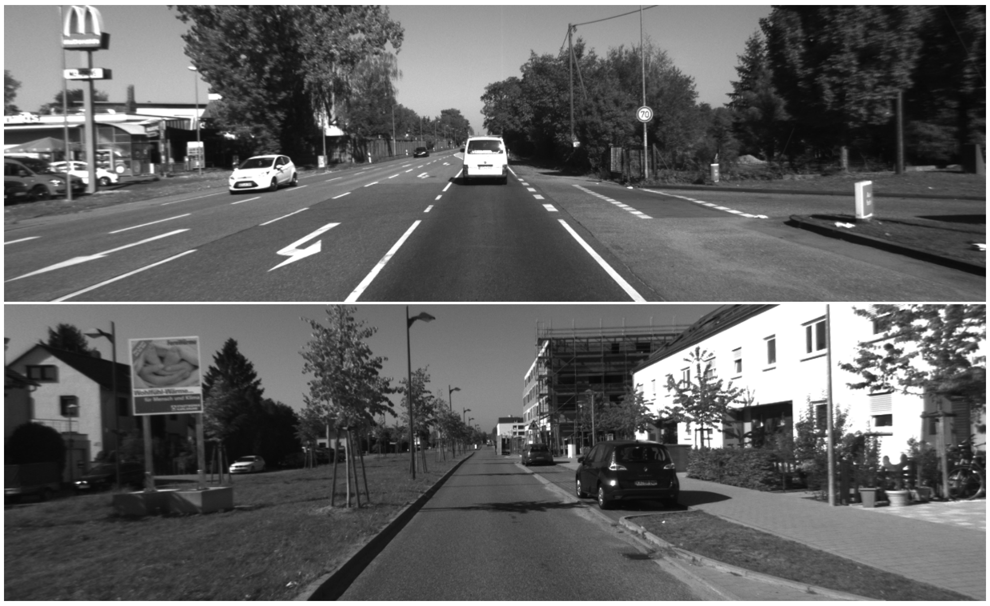


Figure 22: The image at time 1 from the "KITTI" image sequence. (Top) the KITTI04 image. (Bottom) the KITTI06 image.

4.2. Evaluation results

This section shows the results of the VSLAM algorithm on each of the datasets above. In each, we plot the estimated camera trajectory in the $x - z$ plane (*i.e.*, the ground plane). We also show a 3D plot of the estimated camera poses and the reconstructed scene points. The algorithm is run twice; once using the new "partial plane" tracking method and once using the "whole plane" tracking method.

To evaluate the results, the following metrics were used:

1. The absolute camera position error was computed as the Euclidean distance between each of the ground truth positions and estimated camera positions. The root-mean-square error (RMS error) was calculated using

$$\text{RMS error} = \sqrt{\frac{\sum_{i=1}^{n}(\mathbf{t}_{Est_i} - \mathbf{t}_{GT_i})^2}{n}} \tag{12}$$

2. The absolute camera orientation error was evaluated by computing the rotation $\mathbf{R}$ that transforms the estimated camera orientation into the ground truth camera orientation (for those sequences where ground truth orientation was available). This rotation can be expressed as a rotation of θ about some axis $\mathbf{k}$ [25]. The angle θ is given by

$$\theta = cos^{-1}(\frac{tr(\mathbf{R}) - 1}{2}) \tag{13}$$

   The root-mean-square error (RMS error) of this angle was calculated.

3. We expect that the partial plane method will be able to track points longer, on average, than the whole plane method. Therefore, in each keyframe, we counted the number of frames each point had been tracked, and recorded the average of this number.
4. We expect that the partial plane method will be able to track more points, on average, than the whole plane method. Therefore, in each keyframe, we counted the number of tracked 3D points. Only *inlier* points were counted; meaning those points that were labeled as inliers by the RANSAC-based pose estimation algorithm.
5. Finally, we recorded the error of the reconstructed points. This error was calculated as the difference between the depth of the estimated point and the depth of the ground truth point. (Ground truth point positions were only available for the synthetic image sequence.)

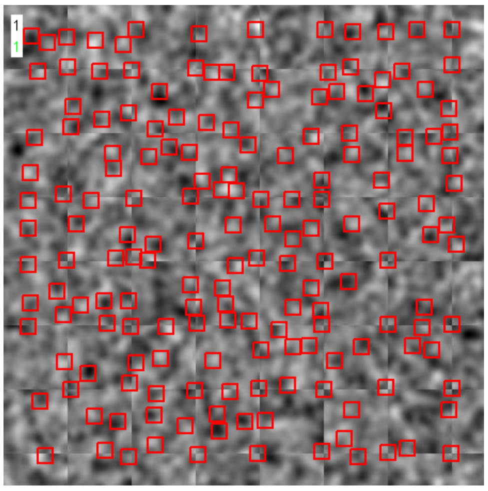


Figure 23: Interest points detected in the first image of the synthetic sequence.

As previously described, in each sequence the first frame is designated as a keyframe, and its pose is considered to be the origin of the world coordinate system. Next, we designate another frame as the second keyframe, and provide the relative pose of the second keyframe relative to the first keyframe, to the system. This establishes the absolute scale for the reconstruction.

The choice of which frame is the second keyframe varies from sequence to sequence. The criterion for choosing the second keyframe was as follows. We choose the first frame in which triangulation can be performed on a majority of the points in the scene. In other words, we pick the first frame where the camera has translated far enough such that the visual angle between the observations of points exceeds the minimum threshold of 2 degrees.

Additional keyframes are added at regular intervals. The interval between keyframes is manually chosen, and is different for each sequence. As previously described, instead of manually selecting the keyframes, we could have the algorithm choose the keyframes. An image could be designated as a keyframe if the camera has moved sufficiently far from any previous keyframes (where the minimum distance depends on the mean depth of observed features). However, we manually select keyframes so that we can do an accurate comparison of the partial plane feature tracking method with the whole plane tracking method. We want both versions of the algorithm to use the same keyframes, so that the pose accuracy at those same keyframes can be directly compared. If such a comparison was not needed, then automatic keyframe selection could be used.

In the results shown in Sections 4.3 through 4.5, a maximum of 200 points were tracked. In Section 4.6, we show the results of an experiment in which this number was varied.

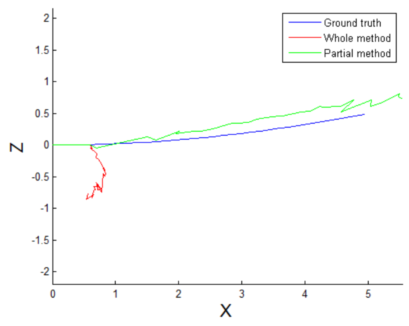
4.3. Synthetic images

In this sequence, the 5^{th} image was designated as the second keyframe, and its pose was provided to the algorithm to establish the scale. After that, every subsequent 5^{th} frame was designated as a keyframe.

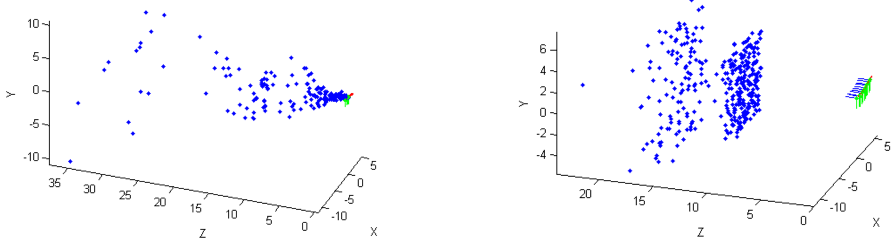
Figure 23 shows the interest points detected in the first image of the synthetic sequence. By visual inspection, 108 of the initial 172 points encompass more than one surface; that is, they are nonplanar.

Figure 24(a) shows the trajectory of the camera as plotted on the ground $(x-z)$ plane of the world. Both PPM and WPM initially follow the ground truth exactly, up until about $x = 0.8$. This pose corresponds to the 2^{nd} keyframe. After that, PPM continues to follow the ground truth fairly accurately, but WPM diverges sharply from the ground truth trajectory.

Figure 24(b) shows a 3D plot of the reconstructed scene. As can be seen, the new PPM algorithm appears to reconstruct the structure of the scene fairly accurately. The points are clustered at the depth of the two surfaces, at 10 and 15. However, the WPM results have a lot of error.



(a)



(b)

Figure 24: Results for synthetic sequence. (a) Trajectory of camera position in the ground $(x - z)$ plane. The camera starts at the left at position (0,0), and travels to the right. (b) Reconstructed scene, showing 3D point locations and camera poses at keyframes for WPM (left) and PPM (right).

Figure 25 shows some template examples that were tracked by PPM through the image sequence, along with their final probability images. The algorithm appeared to correctly segment these templates and identify the pixels belonging to the dominant plane, with only a few errors. 495

Figure 26(a) shows the camera translation as a function of image frame number. Figure 26(b) shows the average number of frames that each point has been tracked as a function of frame number. As can be seen, PPM consistently tracks points longer than WPM.

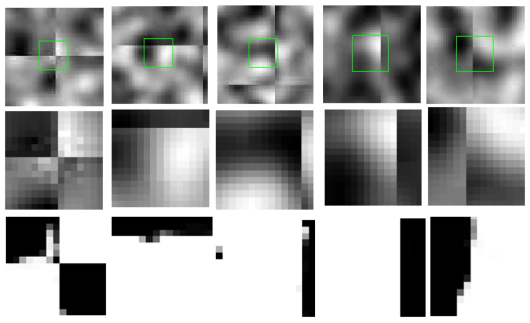


Figure 25: Example templates tracked by PPM. Top row is the location of the templates in the reference image. Middle row is the reference image templates $T_{ref}(\mathbf{x})$, and bottom row is the final probability image, *i.e.* $p(\mathbf{x} \in D|r_{1:t})$.

Figure 26(c) shows the track length for each point, sorted by length. As can be seen, PPM has more points with longer track lengths. Note that the total number of points processed by the system is actually larger for WPM. This is because WPM must acquire more new points to track because it is not able to track points as long as PPM. 500

Figure 26(d) shows the number of 3D points being tracked, as a function of frame number. Only inlier points are counted. As can be seen, PPM consistently is able to track more points than WPM.

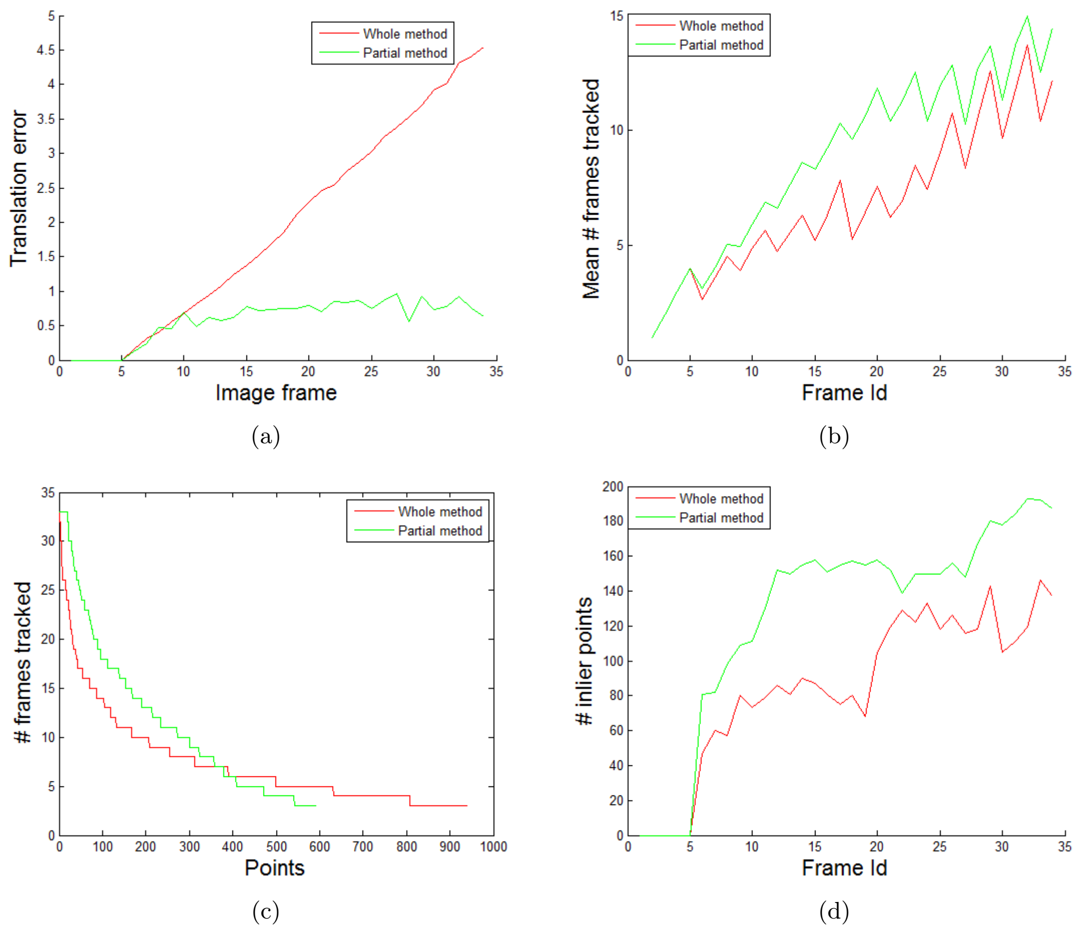


Figure 26: Results for synthetic sequence. (a) Camera pose error. (b) Average number of frames that points are tracked, as a function of frame number. (c) Track lengths for each point, sorted by length. (d) Number of inlier 3D points being tracked, for each frame.

We examined the results more closely in order to understand why WPM begins to fail after the 2nd keyframe. Figure 27 shows the tracked points in frame 6 for WPM (left) and PPM (right). The inlier 3D points are shown as green boxes. There are 55 inlier 3D points for WPM and 88 inlier 3D points for PPM. We manually examined the matches to determine which inlier 3D points were incorrect. By visual inspection, 11 of the 55 points are incorrect for WPM and only 3 of 88 are incorrect for PPM.

In other words, 20% of the matches are incorrect for WPM. The mismatches cause the estimated pose to be in error, and the error grows with subsequent frames. Looking at the two points that were not able to be tracked by PPM, it appears that the algorithm started to track the background (further) surface as the dominant plane, and tracking failed because the foreground surface occluded the background surface.

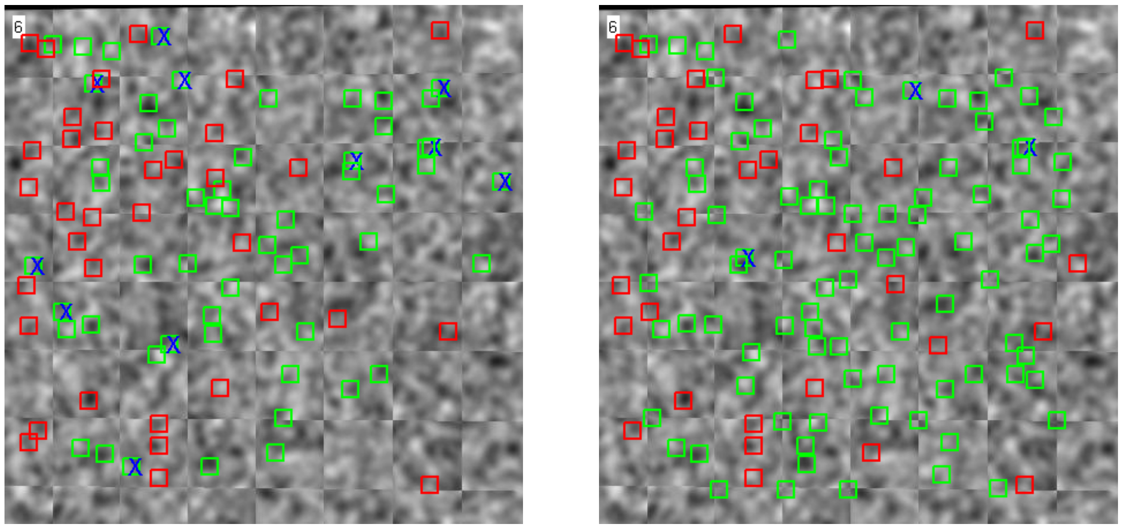


Figure 27: Tracked points in frame 6, for WPM (left) and PPM (right). 2D points are labeled by red, 3D points are labeled by green. Those 3D points that were matched incorrectly are marked by "×".

Finally, Figure 28 shows a histogram of the depth error for the reconstructed points. As can be seen, the PPM results are much more accurate than the WPM results.

A summary of the quantitative results for the synthetic sequence is shown in Table 1, and includes the average number of frames that points are tracked, the average number of inlier 3D points in each frame, RMS translation error of camera poses, RMS angle error of camera poses, and the RMS 3D error of estimated point locations.

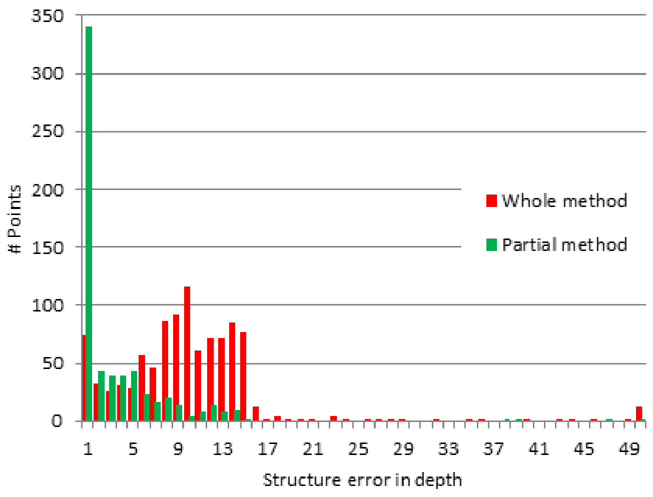


Figure 28: Histogram of the depth error for reconstructed 3D points.

In summary, PPM performed much better than WPM on the synthetic image sequence. This was expected due to the large fraction of nonplanar surfaces in the scene.

Table 1: Comparison of WPM and PPM for the synthetic sequence.

Test	WPM	PPM
Mean # frames tracked	9.33	15.02
# inlier 3D points	70.94	116
RMS error translation	2.53	0.16
RMS error $(\hat{\theta})$ angle	1.37	0.09
RMS error of 3D points	9.85	3.38

4.4. CSM Sequences

This section shows and discusses the experimental results of the VSLAM algorithm using both methods, WPM and PPM, on the CSM Indoor and the CSM Outdoor image sequences.

4.4.1. CSM Indoor

In the CSM Indoor sequence, the 2^{nd} image was designated as the second keyframe, and its pose was provided to the algorithm to establish the scale. After that, every subsequent 5^{th} frame was designated as a keyframe.

The results on this sequence are shown in Figure 29, Figure 30, and Figure 31. Both PPM and WPM do fairly well on this sequence. The error is similar for both methods, although PPM is slightly more accurate than WPM. In both methods the translation error slowly grows over time. Both methods track about the same number of points up until about frame 20, when the number of points being tracked by PPM becomes consistently larger than WPM.

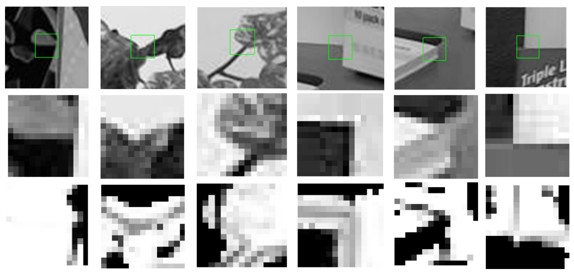
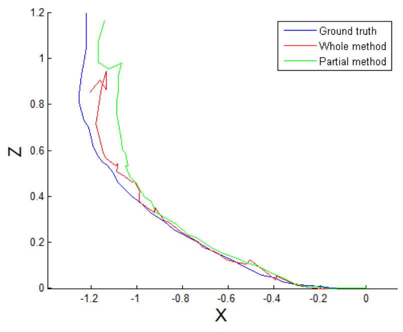


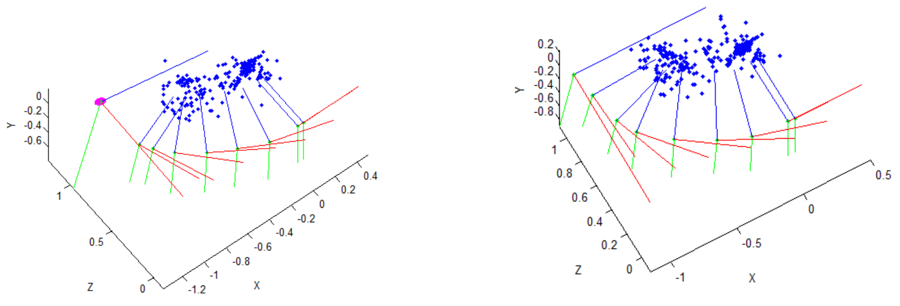
Figure 29: Example templates tracked by PPM. Top row is the location of the templates in the reference image. Middle row is the reference image templates $T_{ref}(\mathbf{x})$, and bottom row is the final probability image, *i.e.*, $p(\mathbf{x} \in D|r_{1:t})$. From left to right, t = 15, 10, 7, 5, 12, 11 respectively.

Looking at the input sequence, the camera undergoes a rapid movement around frame 20. This could cause the WPM to lose track of templates that encompass more than one surface. Figure 32 shows the points that are being tracked at image number 23, for the two methods. As can be seen, PPM at this time is tracking more 3D points, and many of these are nonplanar.

In summary, the performance of PPM was about the same as WPM on the "CSM Indoor" image sequence. This may be due to the fact that there are large planar surfaces in the scene, so that the WPM algorithm was able to track an adequate number of points.

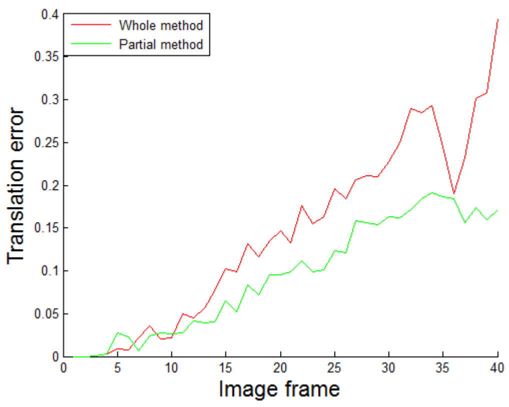


(a)

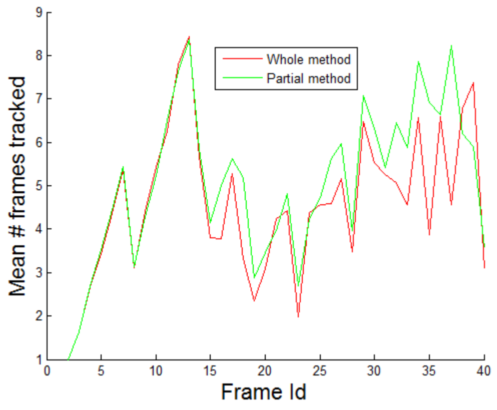


(b)

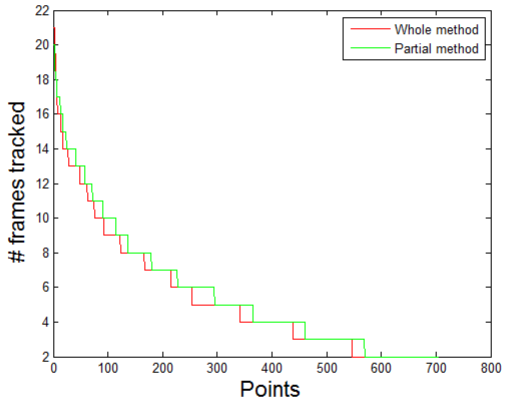
Figure 30: Results for "CSM Indoor". (a) Trajectory of camera position in the ground $(x - z)$ plane . (b) Reconstructed scene, showing 3D point locations and camera poses at keyframes for WPM (left) and PPM (right).



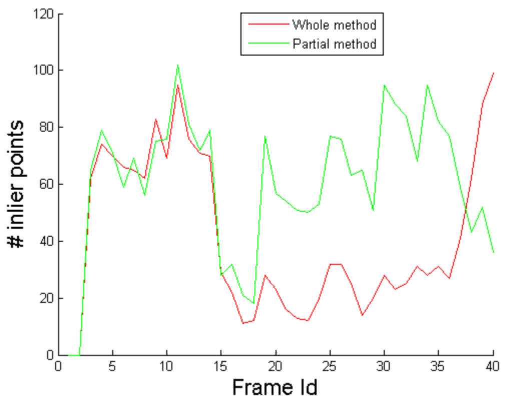
(a)



(b)



(c)



(d)

Figure 31: Results for "CSM Indoor". (a) Camera pose error. (b) Average number of frames that points are tracked, as a function of frame number. (c) Track lengths for each point, sorted by length. (d) Number of inlier 3D points being tracked, for each frame.

4.4.2. CSM outdoor

In the CSM Outdoor sequence, the 2^{nd} image was designated as the second keyframe, and its pose was provided to the algorithm to establish the scale. After that, every subsequent 4^{th} frame was designated as a keyframe.

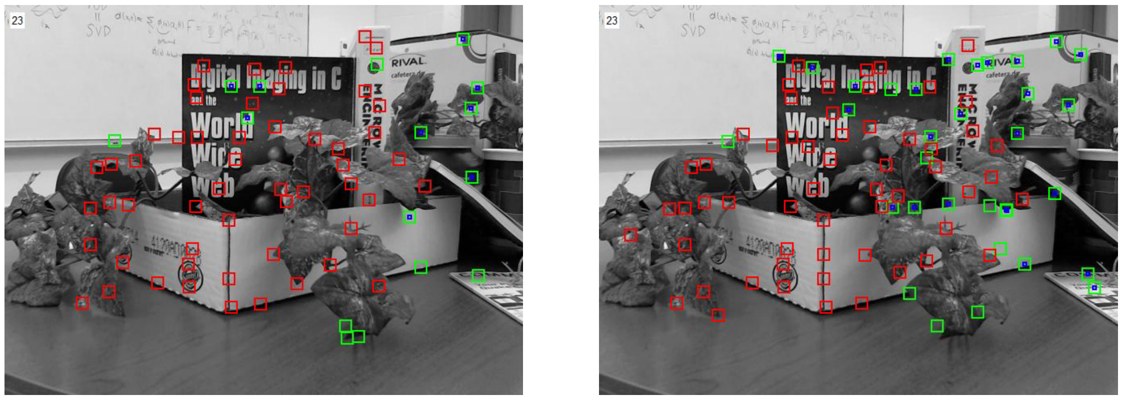
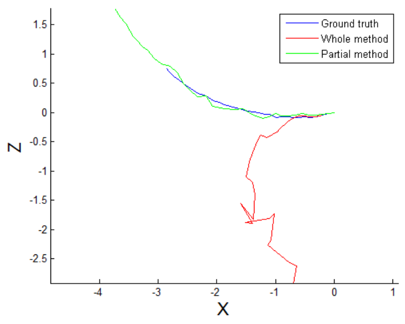
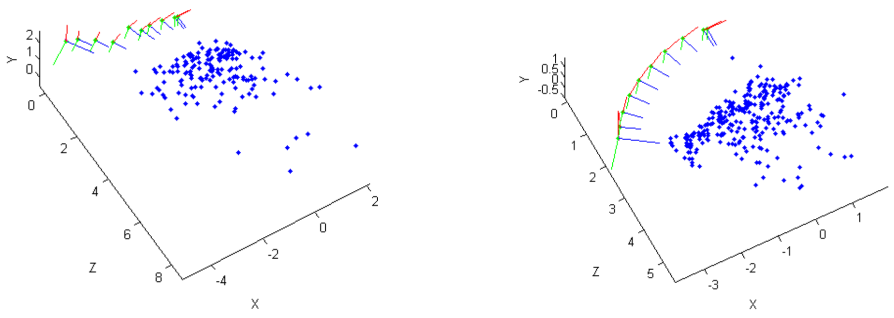


Figure 32: Image number 23 of the "CSM Indoor" sequence, showing tracked 2D points (red) and 3D points (green). Points that have been used in bundle adjustment are marked with a blue dot. (Left) Results from WPM. (Right) Results from PPM.

545 The results for this sequence are shown in Figure 33, Figure 34, and Figure 35. WPM diverges sharply from the ground truth trajectory after about frame 15. PPM is significantly more accurate than WPM, although its translation error does grow gradually over time. It is evident that PPM can track points for much longer than WPM, and the number of points being tracked by PPM is consistently larger than WPM.



(a)



(b)

Figure 33: Results for "CSM Outdoor". (a) Trajectory of camera position in the ground $(x-z)$ plane . (b) Reconstructed scene, showing 3D point locations and camera poses at keyframes for WPM (left) and PPM (right).

To understand why the WPM error grew suddenly, Figure 36 shows the points that are being tracked at image number 14, for the two methods. As can be seen, PPM is tracking a larger number of 3D points at this time. The relatively few number of points being tracked by WPM could cause the estimated pose to have a large error.

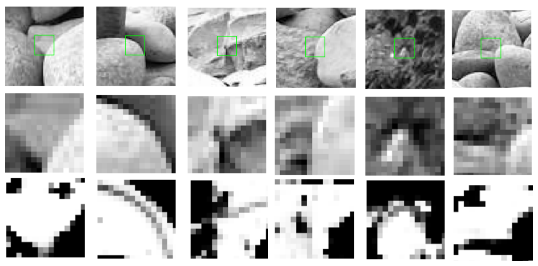


Figure 34: Example templates tracked by PPM. Top row is the location of the templates in the reference image. Middle row is the reference image templates $T_{ref}(\mathbf{x})$, and bottom row is the final probability image, *i.e.*, $p(\mathbf{x} \in D|r_{1:t})$. From left to right, t = 10, 5, 14, 6, 12, 22 respectively.

In summary, the performance of PPM was much better than WPM on the "CSM Outdoor" sequence. This may be due to the fact that there are many nonplanar surfaces in the scene, so that the WPM algorithm was not able to track an adequate number of points.

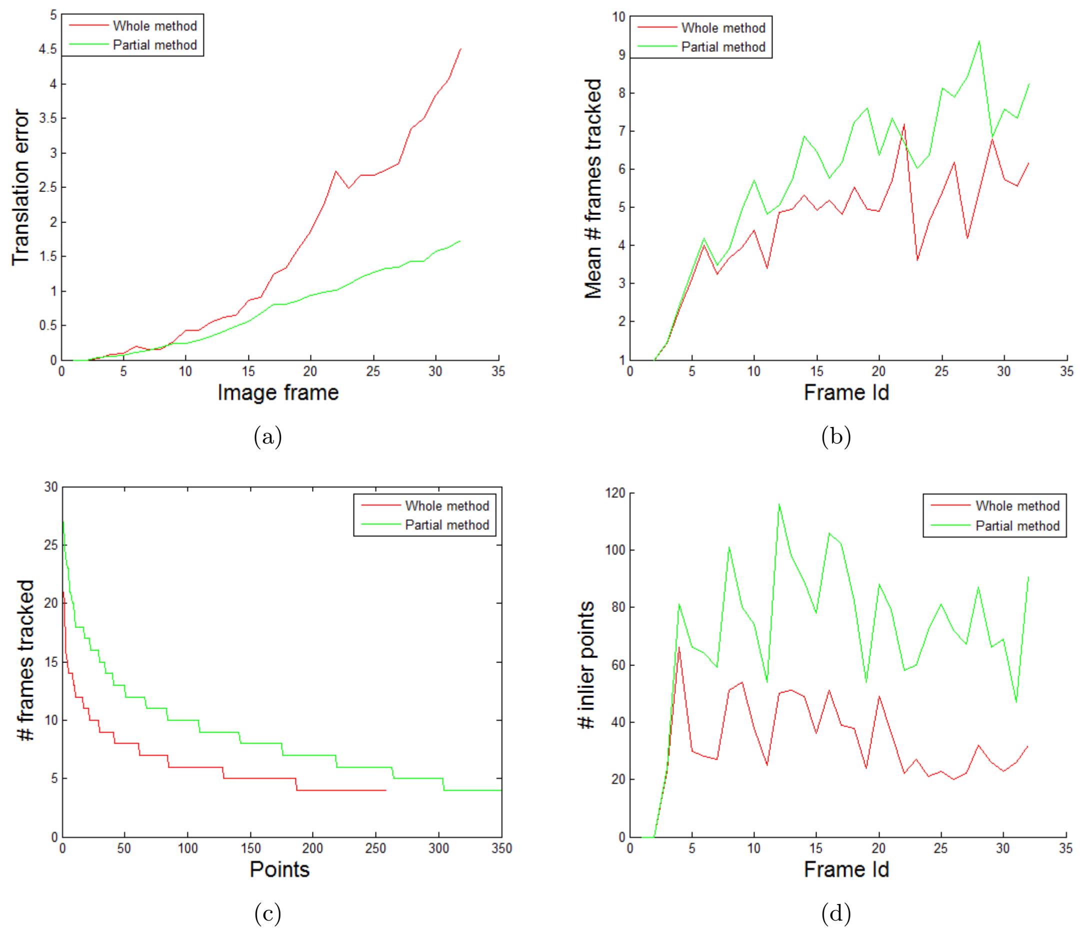


Figure 35: Results for "CSM Outdoor". (a) Camera pose error. (b) Average number of frames that points are tracked, as a function of frame number. (c) Track lengths for each point, sorted by length. (d) Number of inlier 3D points being tracked, for each frame.

4.5. ACTS sequences

This section shows the experimental results of the VSLAM algorithm using both methods, WPM and PPM, on the ACTS image sequences.

4.5.1. ACTS Road

In the ACTS Road sequence, the 15^{th} image was designated as the second keyframe, and its pose was provided to the algorithm to establish the scale. After that, every subsequent 8^{th} frame was designated as a keyframe. The results for this sequence are shown in Figure 37 and Figure 38. The WPM trajectory diverges sharply from the ground truth after about frame 45. Also, after frame 45, PPM is tracking points for much longer periods than WPM.

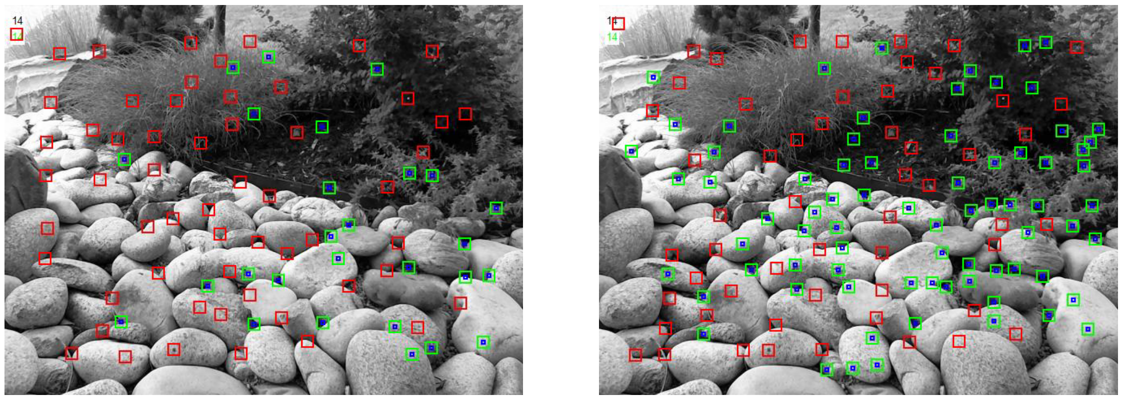
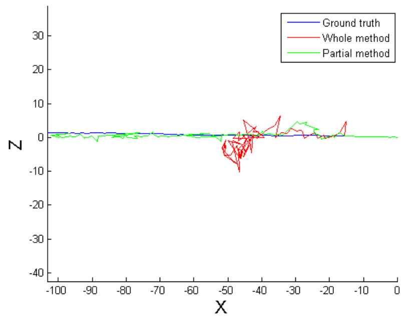
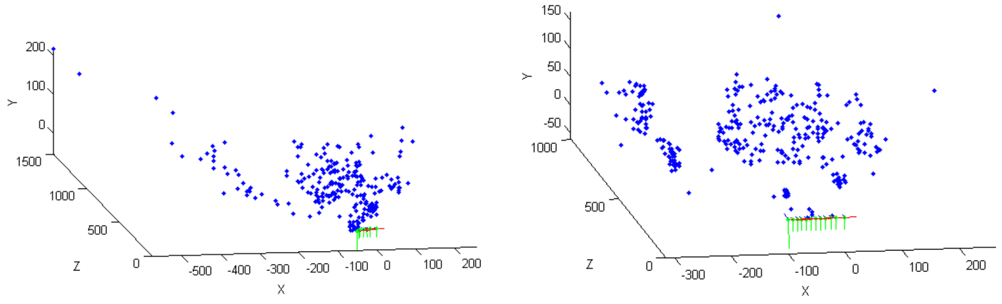


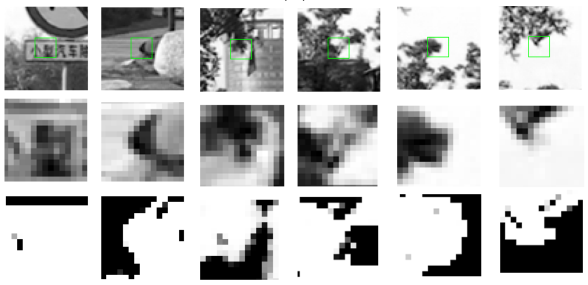
Figure 36: Image number 14 of the “CSM Outdoor” sequence, showing tracked 2D points (red) and 3D points (green). Points that have been used in bundle adjustment are marked with a blue dot. (Left) Results from WPM. (Right) Results from PPM.



(a)



(b)



(c)

Figure 37: Results for “ACTS Road”. (a) Trajectory of camera position in the ground $(x-z)$ plane . (b) Reconstructed scene, showing 3D point locations and camera poses at keyframes for WPM (left) and PPM (right). (c) Example templates tracked by PPM. Top row is the location of the templates in the reference image. Middle row is the reference image templates $T_{ref}(\mathbf{x})$, and bottom row is the final probability image, *i.e.*, $p(\mathbf{x} \in D|r_{1:t})$. From left to right, t = 55, 61, 19, 49, 65, 70 respectively.

To help understand the behavior of the algorithm when WPM failed, Figure 39 shows the points that are being tracked at image number 47, for the two methods. As can be seen, PPM is tracking a larger number of points at this time.

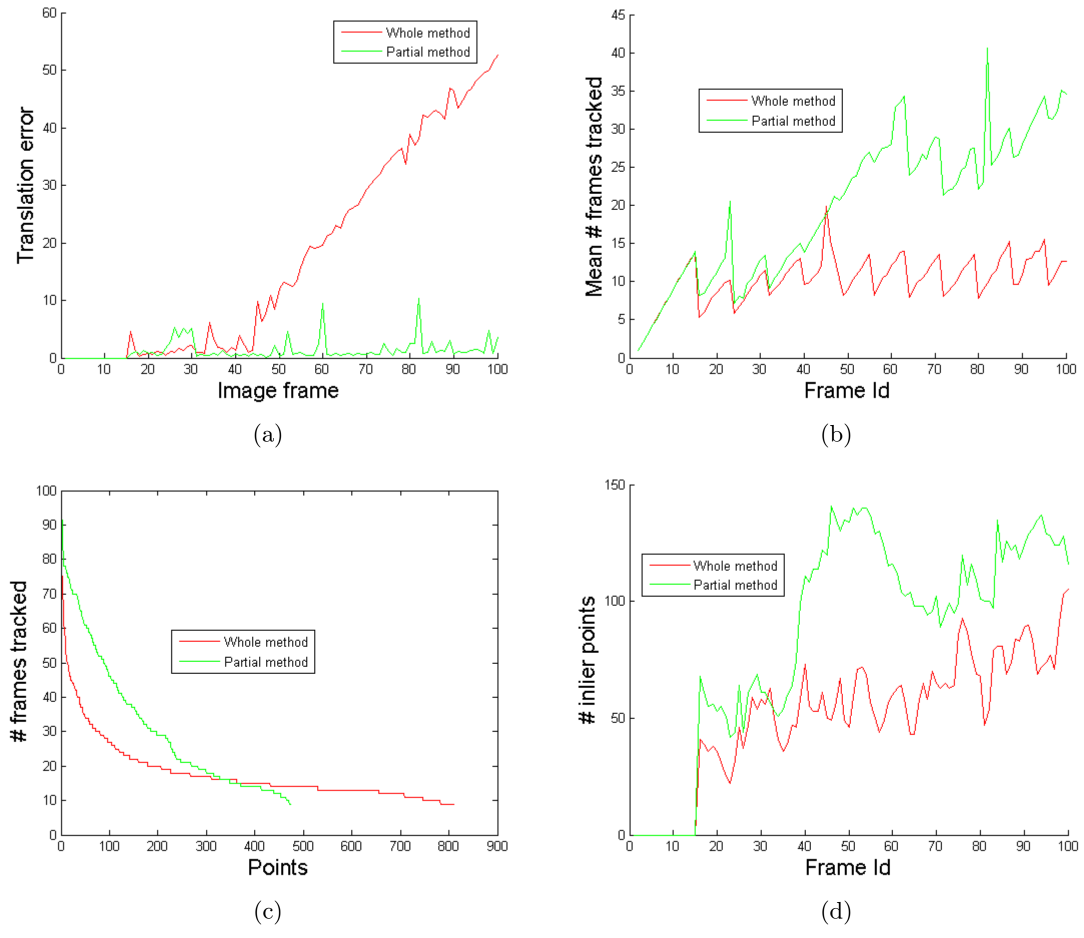


Figure 38: Results for “ACTS Road”. (a) Camera pose error. (b) Average number of frames that points are tracked, as a function of frame number. (c) Track lengths for each point, sorted by length. (d) Number of inlier 3D points being tracked, for each frame.

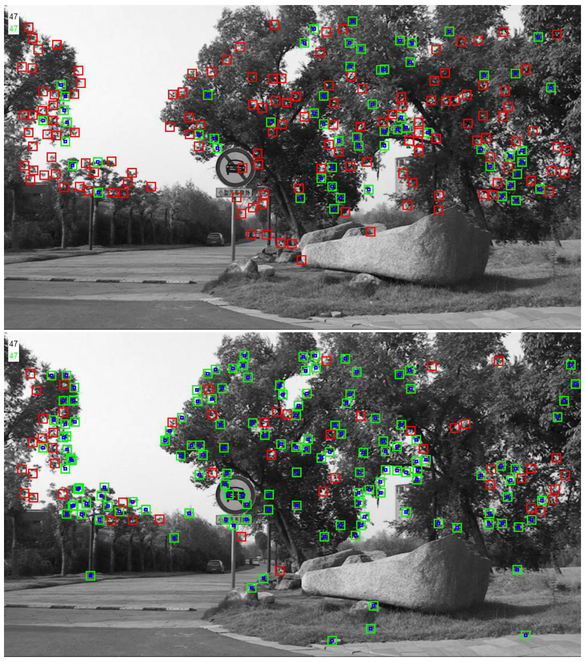


Figure 39: Image number 47 of the "ACTS Road" sequence, showing tracked 2D points (red) and 3D points (green). Points that have been used in bundle adjustment are marked with a blue dot. (Top) Results from WPM. (Bottom) Results from PPM.

Figure 40 shows a histogram of the depth error for the reconstructed points. As can be seen, the PPM results are much more accurate than the WPM results.

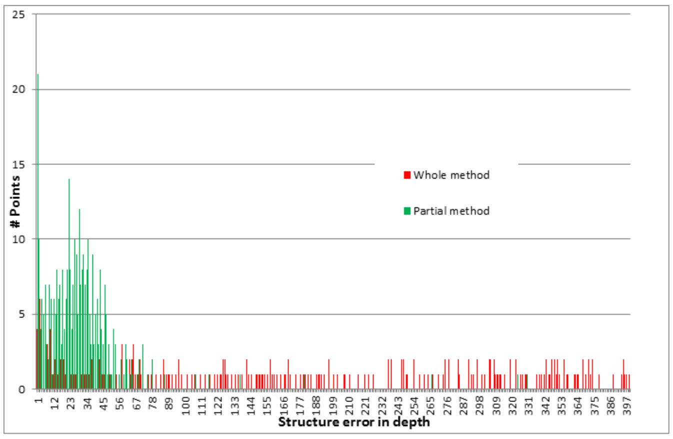
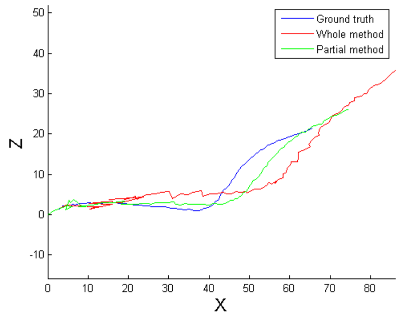
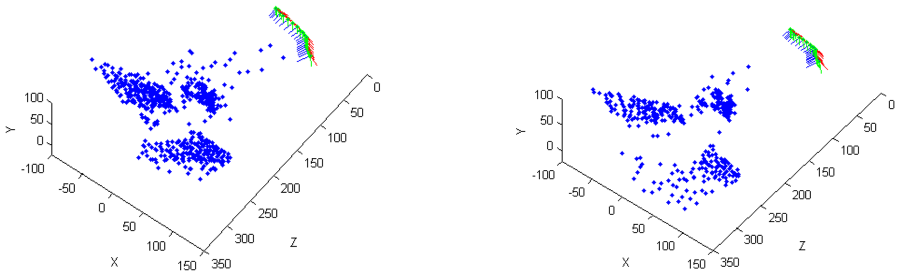


Figure 40: Histogram of the depth error for reconstructed 3D points of "ACTS Road".

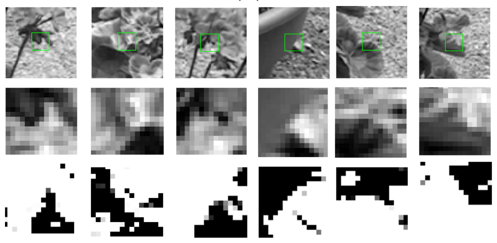
In summary, PPM performed much better than WPM in this sequence. The WPM results had large errors about midway through the "ACTS Road" image sequence, due to the fact that it could not track nonplanar templates. PPM was able to maintain good results all the way through the sequence.



(a)



(b)



(c)

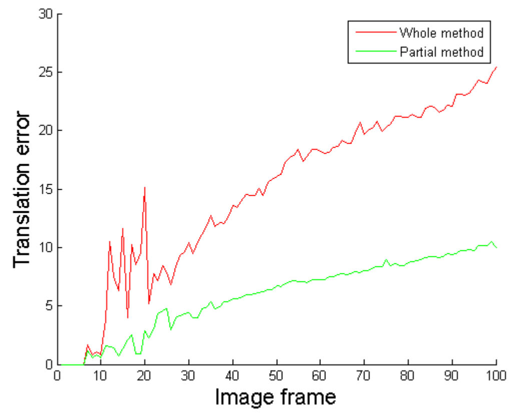
Figure 41: Results for “ACTS Flower”. (a) Trajectory of camera position in the ground $(x-z)$ plane . (b) Reconstructed scene, showing 3D point locations and camera poses at keyframes for WPM (left) and PPM (right). (c) Example templates tracked by PPM. Top row is the location of the templates in the reference image. Middle row is the reference image templates $T_{ref}(\mathbf{x})$, and bottom row is the final probability image, *i.e.*, $p(\mathbf{x} \in D|r_{1:t})$. From left to right, t = 95, 54, 36, 83, 90, 76 respectively.

4.5.2. ACTS Flower

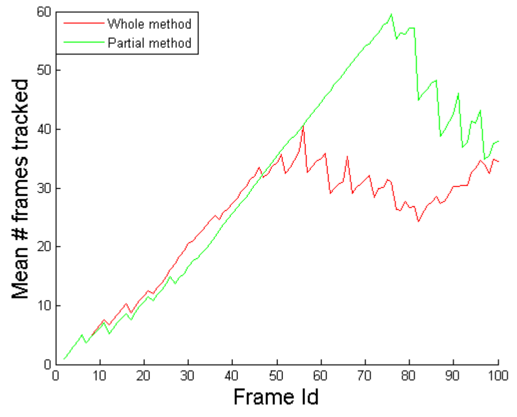
In the ACTS Flower sequence, the 6^{th} image was designated as the second keyframe, and its pose was provided to the algorithm to establish the scale. After that, every subsequent 15^{th} frame was designated as a keyframe.

The results for this sequence are shown in Figure 41 and Figure 42. Both algorithms were able to follow the ground truth trajectory fairly well, although PPM is consistently more accurate than WPM throughout the sequence. Between images 30 to 70, PPM is tracking a larger number of 3D points than WPM. Figure 43 shows the points that are being tracked at image number 61, for the two methods.

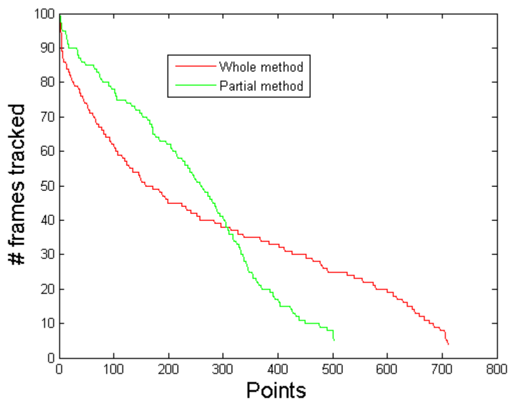
In summary, the performance of the two methods is similar, although the accuracy of PPM is somewhat better than WPM. This scene had large planar areas (*i.e.*, the ground surrounding the flower pot), so WPM may have been able to track enough points in these areas to obtain good pose accuracy.



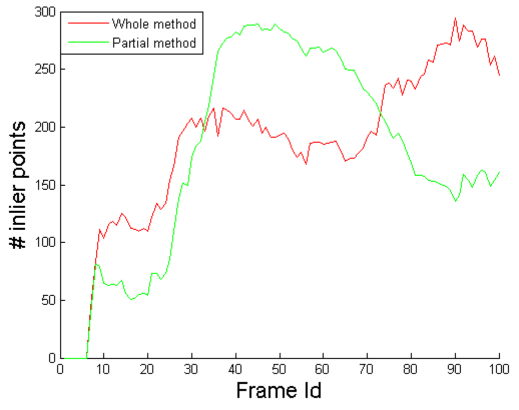
(a)



(b)



(c)



(d)

Figure 42: Results for "ACTS Flower". (a) Camera pose error. (b) Average number of frames that points are tracked, as a function of frame number. (c) Track lengths for each point, sorted by length. (d) Number of inlier 3D points being tracked, for each frame.

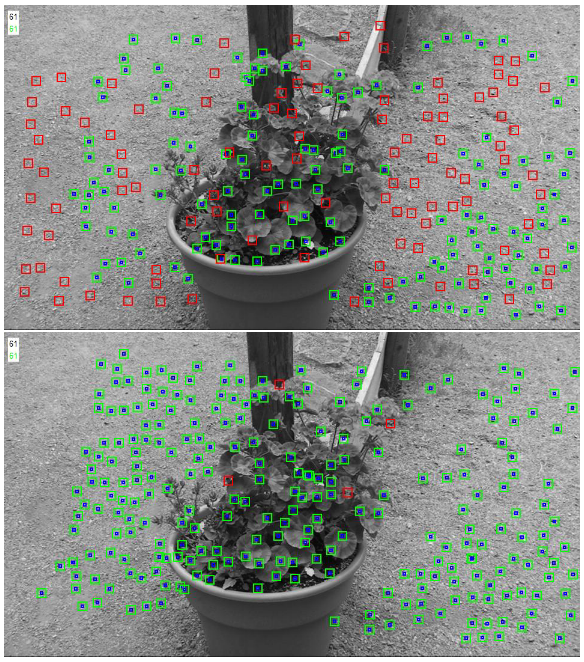


Figure 43: Image number 61 of the "ACTS Flower" sequence, showing tracked 2D points (red) and 3D points (green). Points that have been used in bundle adjustment are marked with a blue dot. (Top) Results from WPM. (Bottom) Results from PPM.

Figure 44 shows a histogram of the depth error for the reconstructed points. As can be seen, the PPM
585 results are much more accurate than the WPM results.

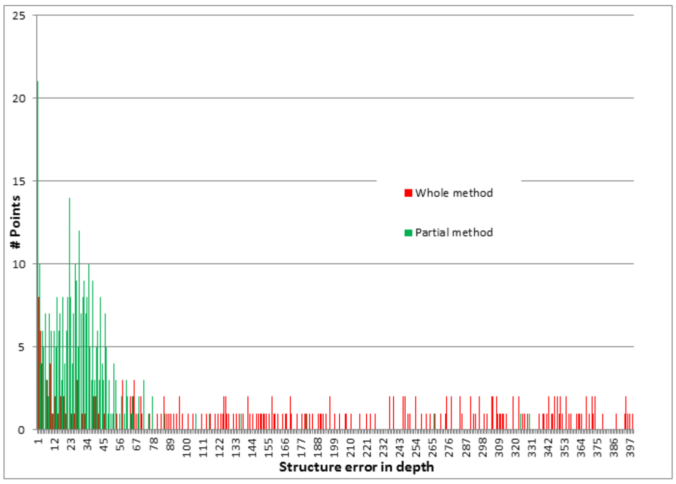
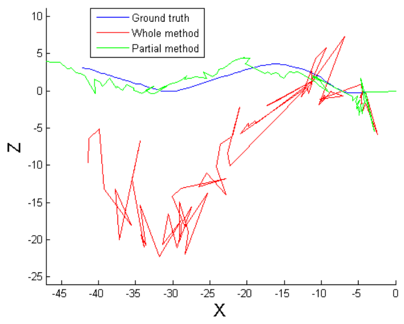


Figure 44: Histogram of the depth error for reconstructed 3D points of "ACTS Road".

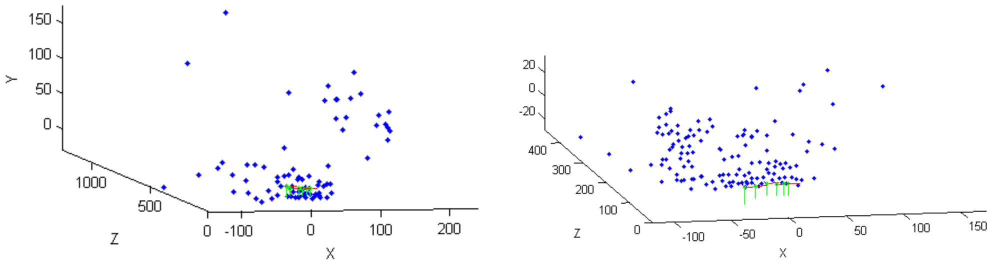
4.5.3. ACTS Lawn

In the ACTS Lawn sequence, the 12th image was designated as the second keyframe, and its pose was provided to the algorithm to establish the scale. After that, every subsequent 20th frame was designated as a keyframe.

590 The results for this sequence are shown in Figure 45, Figure 46, and Figure 47. The WPM results diverge sharply from the ground truth shortly into the sequence, while PPM follows the ground truth closely throughout the sequence.

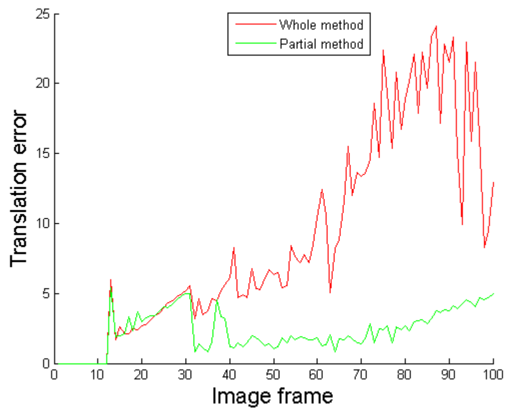


(a)

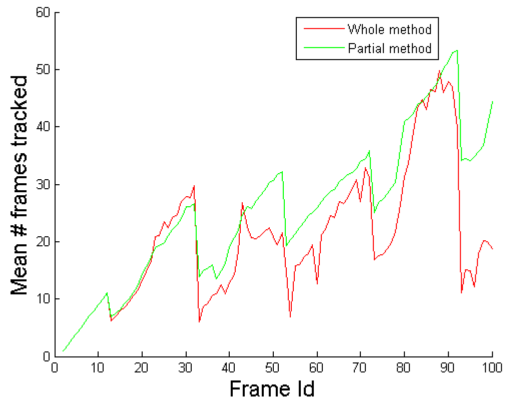


(b)

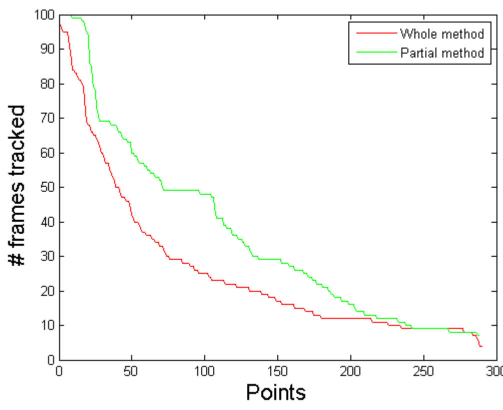
Figure 45: Results for "ACTS Lawn". (a) Trajectory of camera position in the ground $(x - z)$ plane . (b) Reconstructed scene, showing 3D point locations and camera poses at keyframes for WPM (left) and PPM (right).



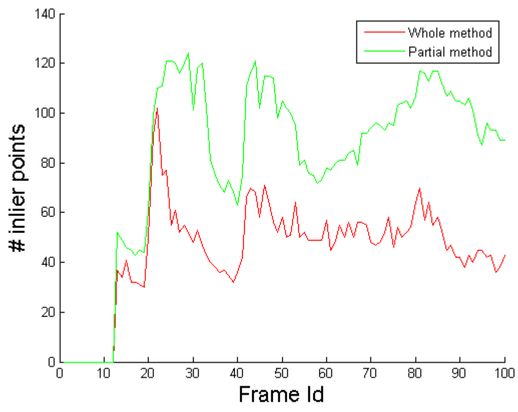
(a)



(b)



(c)



(d)

Figure 46: Results for "ACTS Lawn". (a) Camera pose error. (b) Average number of frames that points are tracked, as a function of frame number. (c) Track lengths for each point, sorted by length. (d) Number of inlier 3D points being tracked, for each frame.

Figure 48 shows the points that are being tracked at image number 32, for the two methods. As can be seen, PPM is tracking a larger number of 3D points.

595 Another way to visualize the number of frames that points are tracked is to display the points using colors that correspond to the number of frames. As Figure 50 shows, there are many more points being tracked for 30 frames or more in the PPM results than in the WPM results.

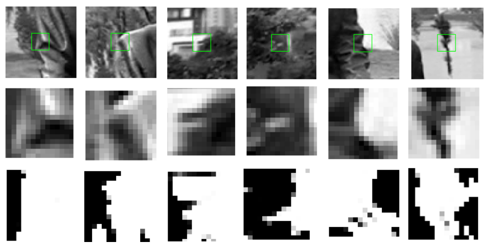


Figure 47: Example templates tracked by PPM. Top row is the location of the templates in the reference image. Middle row is the reference image templates $T_{ref}(\mathbf{x})$, and bottom row is the final probability image, *i.e.*, $p(\mathbf{x} \in D|r_{1:t})$. From left to right, t = 13, 98, 15, 18, 15, 36 respectively.

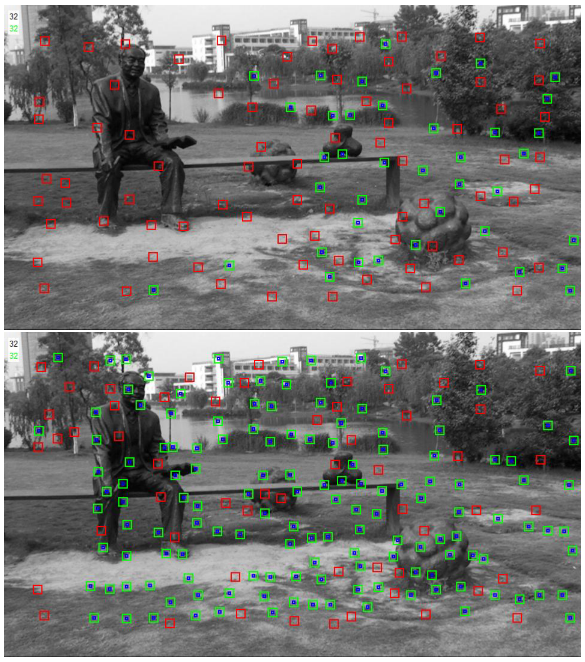


Figure 48: Image number 32 of the "ACTS Lawn" sequence, showing tracked 2D points (red) and 3D points (green). Points that have been used in bundle adjustment are marked with a blue dot. (Top) Results from WPM. (Bottom) Results from PPM.

Figure 49 shows a histogram of the depth error for the reconstructed points. As can be seen, the PPM results are much more accurate than the WPM results.

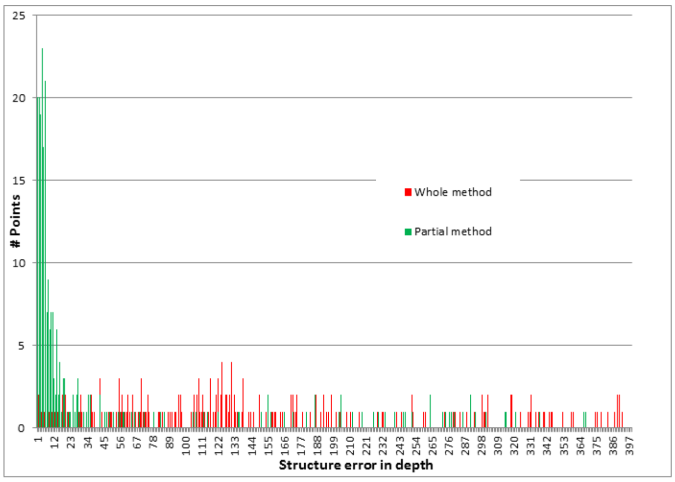


Figure 49: Histogram of the depth error for reconstructed 3D points of "ACTS Road".

600 In summary, the performance of PPM was much better than WPM in this sequence, possibly due to the large change in depth between the foreground object (the man on the bench) and the background. WPM had difficulty tracking points on the occluding boundary of the foreground object, as can be seen in Figure 30.

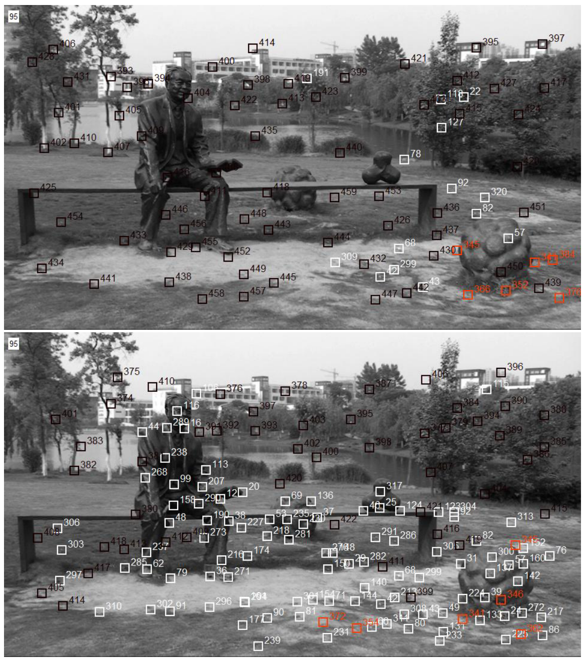


Figure 50: Image number 95 in the "ACTS Lawn" sequence, where the colors indicate how many frames each point is tracked. The colors are shown according to a heat map scheme, where the lowest value is shown as dark red (meaning the point was tracked 10 times or less), and the highest values are shown as white (meaning the point was tracked 30 times or more). (Top) Results from WPM. (Bottom) Results from PPM.

4.6. KITTI sequences

605 This section shows the experimental results of the VSLAM algorithm using both methods, WPM and PPM, on the KITTI odometry benchmark image sequences.

4.6.1. KITTI 04

In the KITTI 04 sequence, the 10th image was designated as the second keyframe, and its pose was provided to the algorithm to establish the scale. After that, every subsequent 3rd frame was designated as a keyframe. The results for this sequence are shown in Figure 51 and Figure 52. The WPM trajectory diverges sharply from the ground truth after about frame 21. Also, after frame 21, PPM is tracking points for much longer periods than WPM.

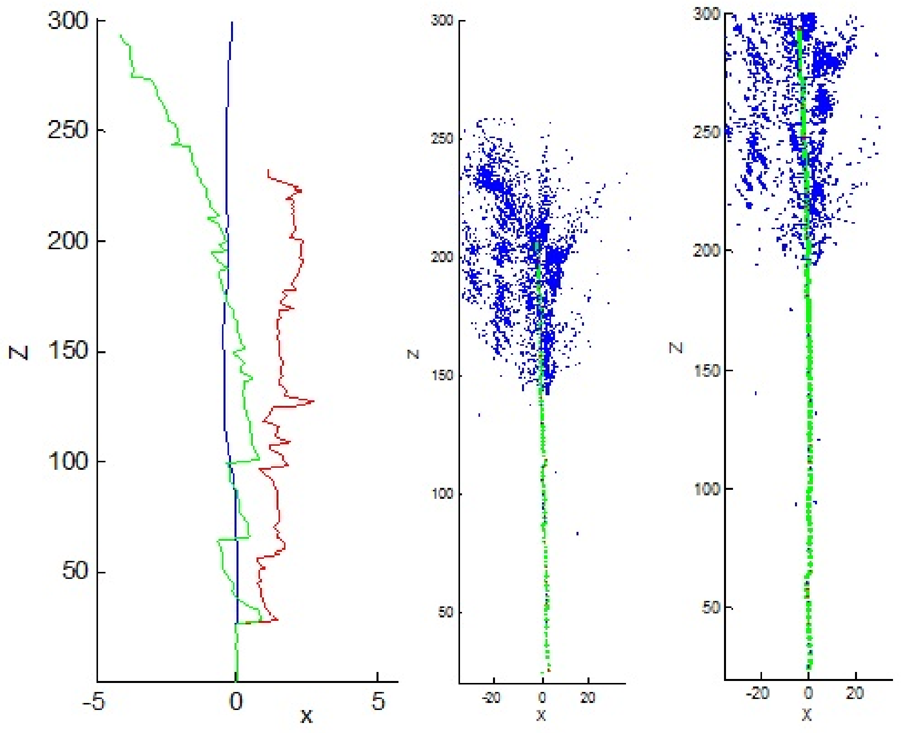
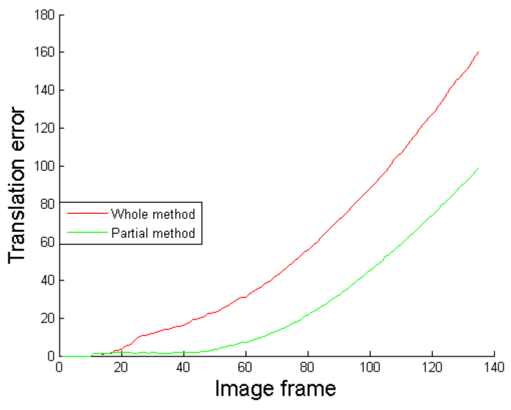
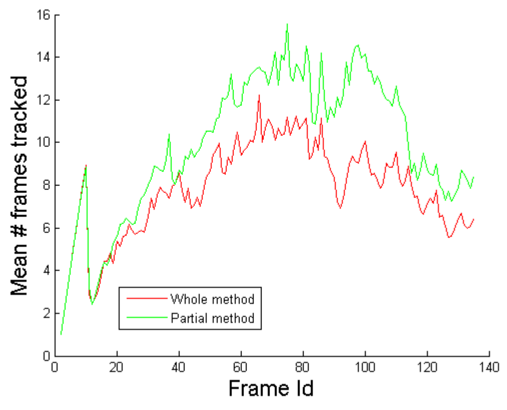


Figure 51: Results for "KITTI 04". (Left) Trajectory of camera position in the ground ($x - z$) plane . Reconstructed scene, showing 3D point locations and camera poses at keyframes for WPM (middle) and PPM (right).

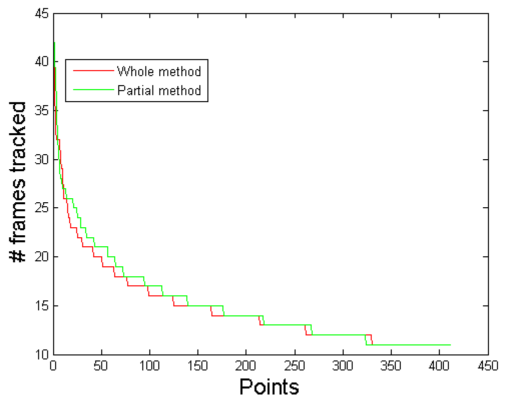
To help understand the behavior of the algorithm when WPM failed, Figure 53 shows the points that are being tracked at image number 21, for the two methods. As can be seen, PPM is tracking a larger number of points at this time.



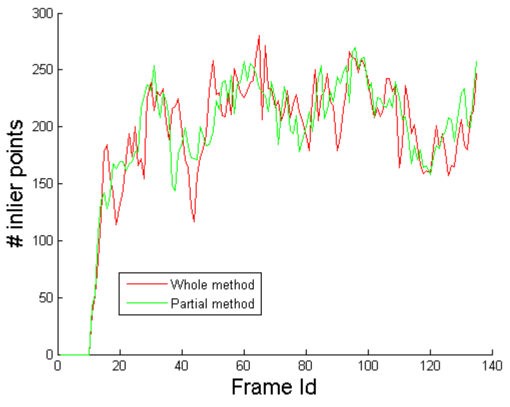
(a)



(b)



(c)



(d)

Figure 52: Results for "KITTI 04". (a) Camera pose error. (b) Average number of frames that points are tracked, as a function of frame number. (c) Track lengths for each point, sorted by length. (d) Number of inlier 3D points being tracked, for each frame.

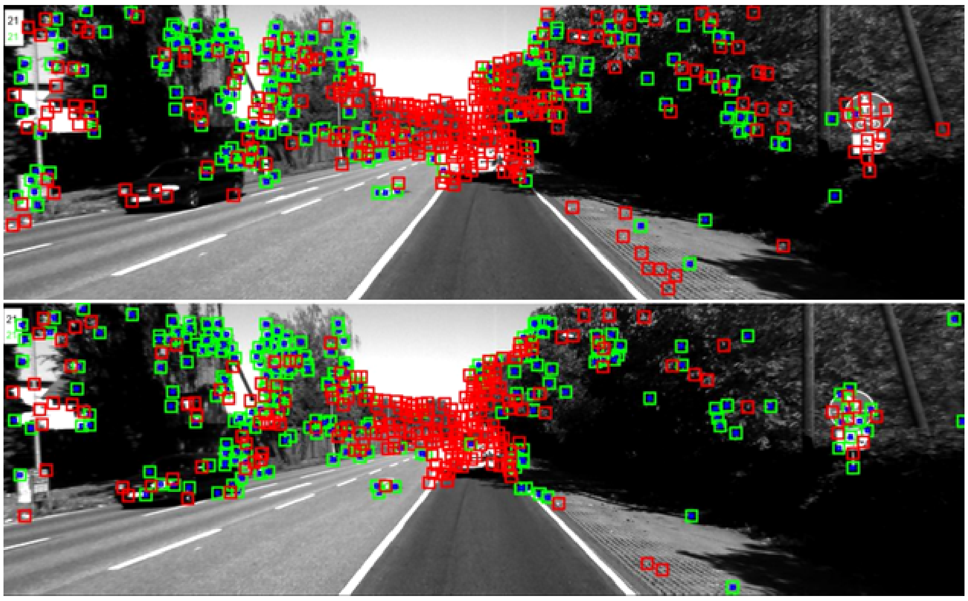


Figure 53: Image number 21 of the "KITTI 04" sequence, showing tracked 2D points (red) and 3D points (green). Points that have been used in bundle adjustment are marked with a blue dot. (Top) Results from WPM. (Bottom) Results from PPM.

4.6.2. KITTI 06

In the KITTI 06 sequence, the 10th image was designated as the second keyframe, and its pose was provided to the algorithm to establish the scale. After that, every subsequent 3rd frame was designated as a keyframe. The results for this sequence are shown in Figure 54 and Figure 55. The WPM trajectory diverges sharply from the ground truth after about frame 26. Also, after frame 26, PPM is tracking points for much longer periods than WPM.

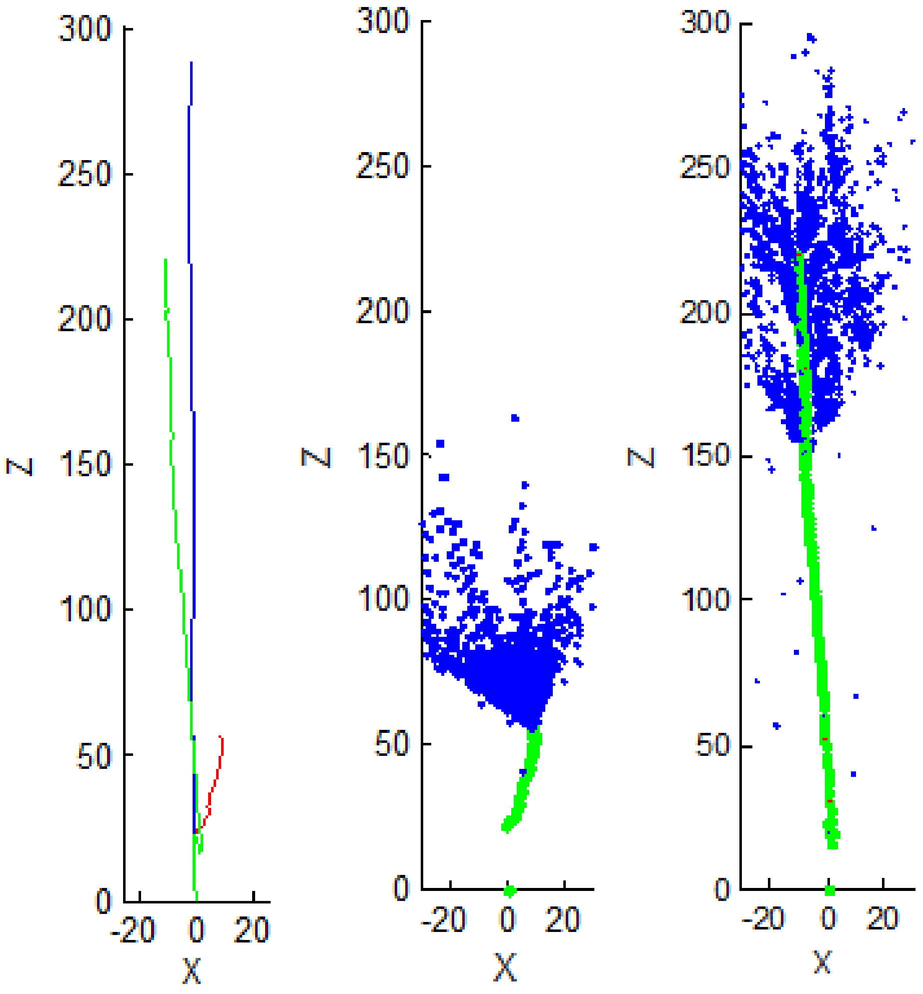
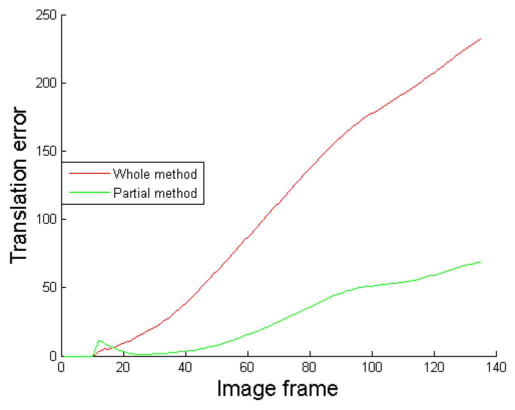
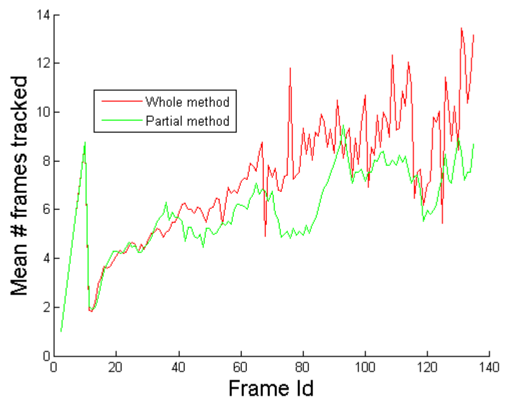


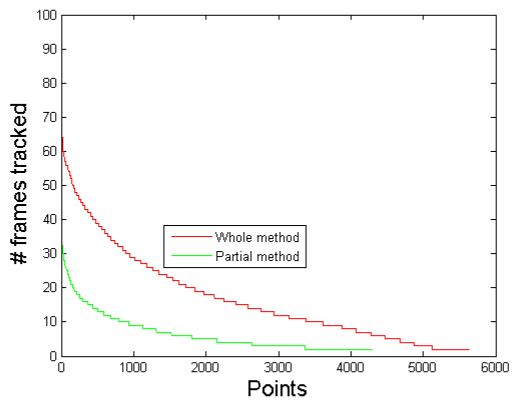
Figure 54: Results for “KITTI 06”. (left) Trajectory of camera position in the ground ($x - z$) plane . Reconstructed scene, showing 3D point locations and camera poses at keyframes for WPM (middle) and PPM (right).



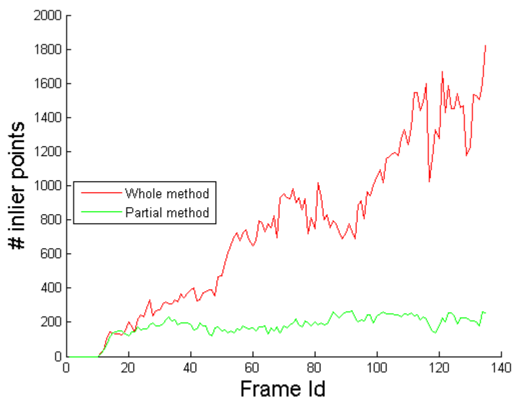
(a)



(b)



(c)



(d)

Figure 55: Results for "KITTI 06". (a) Camera pose error. (b) Average number of frames that points are tracked, as a function of frame number. (c) Track lengths for each point, sorted by length. (d) Number of inlier 3D points being tracked, for each frame.

4.7. Summary

A quantitative summary of the results is shown in Table 2. For each of the real image sequences:

- PPM tracked a higher average number of points per frame than WPM.
- PPM was able to track points longer on average than WPM.
- The RMS error for translation and orientation was lower for PPM than WPM (ground truth orientation was not available for the CSM sequences).

The preceding results were obtained using a threshold of 150 points in each keyframe. Namely, if the number of tracked points fell below 150 points, new points were acquired to try to raise the number up to 150.

We performed an additional experiment to determine the effect of this number on the results. Table 3 shows the RMS translation error for each method, using a threshold of 150, 200, and 300 points. As can be seen, the accuracy improves when more points are used, for both methods. However, in each case, PPM is better than WPM.

By adding points, WPM can achieve almost the same accuracy as PPM (in most cases). For example, WPM using 300 points has an RMS error of 0.2 m on the CSM Indoor dataset. This is comparable to the

RMS error of 0.21 m achieved by PPM with only 150 points. However, it is desirable to limit the number of points to achieve real time performance.

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. Such cases often occur in outdoor natural scenes, which can contain many discontinuities. Even in man-made indoor environments, which typically contain planar surfaces, discontinuities can occur at the boundaries of the planes.

Table 2: Comparison of WPM and PPM for all real image sequences.

Sequence	Ground truth path length	Tracking method	Mean # inlier 3D points	Mean # frames tracked	RMS error translation	RMS error (radians)
CSM Indoor	2.01	WPM	54	5.4	0.2	-
		PPM	60.9	5.8	0.1	-
CSM Outdoor	3.12	WPM	32.4	6.31	2.51	-
		PPM	70.7	8.55	0.91	-
ACTS Road	103.41	WPM	51.3	18.0	24.97	0.394
		PPM	85.8	30.9	2.15	0.035
ACTS Flower	77.83	WPM	41.4	37.9	16.32	0.88
		PPM	60.9	48.9	6.66	0.512
ACTS Lawn	44.96	WPM	44.7	25.9	11.43	0.267
		PPM	81.8	36.3	2.75	0.071
KITTI 04	25606	WPM	187.5	7.78	635.61	-
		PPM	190	9.89	307.1	-
KITTI 06	12014	WPM	733.8	7.2	1250	-
		PPM	176.7	6	325	-

Table 3: Effect of the number of tracked points on RMS error.

	VSLAM based tracking method	RMS error translation of acquired points		
		150	200	300
CSM Indoor	WPM	1.182	0.25	0.2
	PPM	0.21	0.1	0.09
CSM Outdoor	WPM	3.51	2.5	1.47
	PPM	1.23	0.9	0.8
ACTS Road	WPM	26.33	24.97	21.1
	PPM	17.45	2.15	1.07
ACTS Flower	WPM	32.28	16.32	17.77
	PPM	7.17	6.66	3.13
ACTS Lawn	WPM	13. 3	11.43	3.21
	PPM	3.39	2.75	1.05

The partial planar tracking method was incorporated into a VSLAM algorithm and evaluations were done using a number of different datasets; including synthetic, indoor and outdoor sequences. The performance of the VSLAM algorithm using the new partial plane tracking method (PPM) was compared to the same algorithm using the whole plane tracking method (WPM).

The results showed that the algorithm with PPM can estimate and track features over a larger distance, compared to the algorithm with WPM. The average number of tracked points was also greater for PPM than for WPM. Finally, the accuracy of the reconstructed camera poses and scene structure was better for PPM than WPM.

For systems that must operate in real time, the number of tracked points must be kept small. In this case, the new method is particularly advantageous. If a significant number of the points are on nonplanar surfaces, then failure to track them can result in a low number of remaining points, which can adversely affect accuracy.

Our main contribution is for the case in which we are tracking a relatively low number of points (to allow real-time operation on limited computing hardware), and a high percentage of points are on nonplanar surfaces. In these cases we show that our new method dramatically improves the accuracy of VSLAM.

Acknowledgments

We acknowledge the Ministry of Higher Education and Scientific Research of Libya who sponsored the Libyan North American Scholarship Program.

References

[1] S. Thrun, W. Burgard, D. Fox, Probabilistic robotics, Cambridge, MA: MIT press, 2005.

[2] A. Davison, Real-time simultaneous localisation and mapping with a single camera 2 (2003) 1403–1410.

[3] N. Karlsson, E. D. Bernardo, The vslam algorithm for robust localization and mapping, International Conference of Robotics and Automation (ICRA) (2005) 24–29.

[4] P. Elinas, R. Sim, J. Little, Sigmaslam: stereo vision slam using the rao-blackwellised particle filter and a novel mixture proposal distribution (2006) 1564–1570.

[5] S. Se, D. Lowe, J. Little, Mobile robot localization and mapping with uncertainty using scale-invariant visual landmarks, The international Journal of Robotics Research (2002) 735–758.

[6] J. Shi, C. Tomasi, Good features to track, IEEE Conference on Computer Vision and Pattern Recognition (1994) 593–600.

[7] A. Davison, I. Reid, N. Molton, O. Stasse, Monoslam: Real-time single camera slam, IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI) 29 (6) (2007) 1052–1067.

[8] G. Klein, D. Murray, Parallel tracking and mapping for small ar workspaces, International Symposium on Robotics Research on Mixed and Augmented Reality, 2007.

[9] S. Baker, I. Matthews, Lucas-kanade 20 years on: A unifying framework, International Journal of Computer Vision 56 (3) (2004) 221–255.

[10] A. Masoud, W. Hoff, Segmentation and tracking of partial planar templates, IEEE Winter Conference on the Applications of Computer Vision (WACV) (2014) 1128–1133.

[11] S. Seitz, N. Snavely, R. Szeliski, Photo tourism: Exploring photo collections in 3d, In ACM Transactions on Graphics (SIGGRAPH) 25 (3) (2006) 835–846.

[12] H. Wuest, F. Wientapper, D. Stricker, Acquisition of high quality planar patch features, Advances in Visual Computing. Springer Berlin Heidelberg (2008) 530–539.

[13] N. Grammalidis, L. Bleris, M. G. Strintzis, Using the expectation-maximization algorithm for depth estimation and segmentation of multi-view images, Proceedings. First International Symposium on 3D Data Processing Visualization and Transmission (2002) 686–689.

[14] G. Hager, P. Belhumeur, Efficient region tracking with parametric models of geometry and illumination, IEEE Trans on PAMI 20 (10) (1998) 1025–1039.

[15] S. Oron, A. Bar-Hillel, S. Avidan, Extended lucas-kanade tracking, Computer Vision ECCV (2014) 142–156.

[16] N. Molton, A. Davison, I. Reid, Locally planar patch features for real-time structure from motion, BMVC (2004) 1–10.

[17] J. Lagarias, et al., Convergence properties of the nelder-mead simplex method in low dimensions, SIAM Journal of Optimization 9 (1) (1998) 112–147.

[18] F. Moreno-Noguer, P. Fua, V. Lepetit, Accurate non-iterative o(n) solution to the pnp problem, IEEE 11th International Conference on Computer Vision, 2007.

[19] P. Torr, A. Zisserman, Mlesac: a new robust estimator with application to estimating image geometry, Computer Vision and Image Understanding 78 (1) (2000) 138–156.

[20] M. Fischler, R. Bolles, Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography, In Communications of the ACM 24 (6) (1981) 381–395.

[21] E. Mouragnon, M. Lhuillier, M. D. F. Dekeyser, P. Sayd, Real time localization and 3d reconstruction, IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR) 1 (2006) 363–370.

[22] G. Zhang, J. Jia, T. Wong, H. Bao, Recovering consistent video depth maps via bundle optimization, IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2008) 1–8.

[23] G. Zhang, J. Jia, T. Wong, H. Bao, Consistent depth maps recovery from a video sequence, IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI) 31 (6) (2009) 974–988.

[24] C. S. A. Geiger, P. Lenz, R. Urtasun., Vision meets robotics: The kitti dataset, International Journal of Robotics Research 20 (32) (2013) 1229–1235.

[25] J. Craig, Introduction to Robotics: Mechanics and Control, 2nd ed., Reading, Mass: Addison-Wesley, 1990.

Appendix A. Normal distribution of reference image template

In this appendix, we describe how to estimate σ_D and $\sigma_{\overline{D}}$ for a given template as shown in Figure A.1(a). This is done when the template is initialized. As described in Section 2.3, σ_D is the standard deviation of residuals for points on the dominant plane.

Image residual error can arise from image noise and also from mis-registration. The total residual variance is the sum of the two variances; i.e., $\sigma_D^2 = \sigma_n^2 + \sigma_\epsilon^2$, where σ_n^2 is the variance due to noise and σ_ϵ^2 is the variance due to mis-registration. We use a constant value of $\sigma_n = 1$ for the image noise.

To estimate σ_ϵ^2 we assume that the positional errors are normally distributed with a small standard deviation; i.e., $p(\triangle\mathbf{x}) = N(0, \sigma^2{}_{\mathbf{x}})$, where $\triangle\mathbf{x}$ is the positional error after aligning the reference template with the current image. In our work, we use $\sigma_x^2 = 1$. Now, the image residual error between the reference image template and the matched region in the current image, for some positional error $\triangle\mathbf{x}$, is

$$r(\mathbf{x}) = T_{ref}(\mathbf{x} + \triangle\mathbf{x}) - T_{match}(\mathbf{x}) \tag{A.1}$$

Prior to actually matching the template, we do not know where the template will end up being matched to, in the current image. However, we can estimate the residual if we assume that the matched region of the current image is identical to the reference template, after alignment. Therefore, we can estimate the residual error for a positional error $\triangle\mathbf{x}$ by comparing the reference template with a shifted version of itself:

$$r(\mathbf{x}) = T_{ref}(\mathbf{x} + \triangle\mathbf{x}) - T_{ref}(\mathbf{x}) \tag{A.2}$$

The variance of the image residual error is then estimated using:

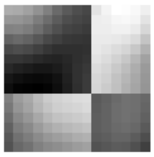
$$\sigma_\epsilon^2 = \frac{1}{N}\sum_{\triangle\mathbf{x}}[r(\mathbf{x}) - \mu]^2 P_D(\triangle\mathbf{x}) \tag{A.3}$$

Note that the variance σ_ϵ^2 is different at each pixel in the template. For example, in the vicinity of a large intensity step edge, σ_ϵ^2 is large, as shown in Figure A.1(b).

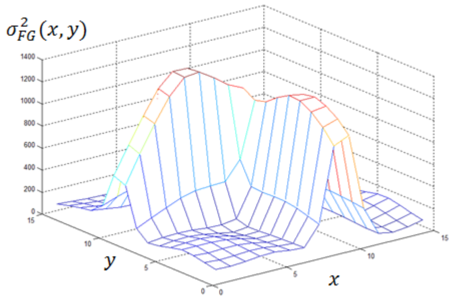
We next estimate $\sigma_{\overline{D}}$, which is the standard deviation of residuals for points not on the dominant plane (e.g., in the background). In this case, the reference template is not accurately aligned to the current image, and can be mapped to anywhere in the search region. We assume that the probability of the positional error $P_{\overline{D}}(\triangle\mathbf{x})$ is a uniform probability distribution over the size of the search region. The image residual variance is estimated as $\sigma_{\overline{D}}^2 = \sigma_n^2 + \sigma_\epsilon^2$, where the variance of the image residual error is estimated using:

$$\sigma_\epsilon^2 = \frac{1}{N}\sum_{\triangle\mathbf{x}}[r(\mathbf{x}) - \mu]^2 P_{\overline{D}}(\triangle\mathbf{x}) \tag{A.4}$$

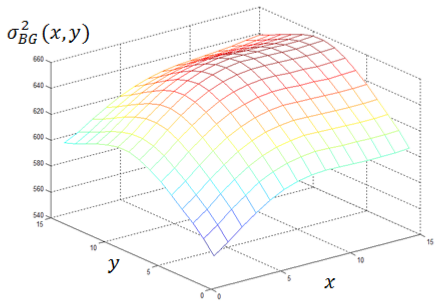
An example of $\sigma_{\overline{D}}^2$ is shown in Figure A.1(c).



(a)



(b)



(c)

Figure A.1: (a) Example of a reference image template $T_{ref}(\mathbf{x})_{15\times15}$. (b) Standard deviation of residuals for points on the dominant plane and (c) for points not on the dominant plane in the reference image template