2014 IEEE International Conference on Robotics & Automation (ICRA)
Hong Kong Convention and Exhibition Center
May 31 - June 7, 2014. Hong Kong, China

Using Superpixels in Monocular SLAM

Alejo Concha and Javier Civera[1]

***Abstract*— Monocular SLAM and Structure from Motion have been traditionally based on finding point correspondences in highly-textured image areas. Large textureless regions, usually found in indoor and urban environments, are difficult to reconstruct by these systems.**

In this paper we augment for the first time the traditional point-based monocular SLAM maps with superpixels. Superpixels are middle-level features consisting of image regions of homogeneous texture. We propose a novel scheme for superpixel matching, 3D initialization and optimization that overcomes the difficulties of salient point-based approaches in these areas of homogeneous texture.

Our experimental results show the validity of our approach. First, we compare our proposal with a state-of-the-art multiview stereo system; being able to reconstruct the textureless regions that the latest cannot. Secondly, we present experimental results of our algorithm integrated with the point-based PTAM [1]; estimating, now in real-time, the superpixel textureless areas. Finally, we show the accuracy of the presented algorithm with a quantitative analysis of the estimation error.

I. Introduction

The extense research performed in monocular SLAM [1] and Structure from Motion (SfM) [2] has produced many excellent results. Traditionally these methods have used salient point features extracted in areas with high image gradients –e.g. SIFT [3] or SURF [4]–; and local descriptors containing grey-level or gradient information. These methods that use *low-level features* are able to reconstruct highly textured areas; but show a poor performance when the image texture is low. Recent research has been able to *densify* traditional visual 3D reconstructions [5], [6]; but large untextured image regions are still a challenge.

On the other hand, very recent work has incorporated the use of *high-level features* in visual SLAM and SfM. In these works object categories –car, screen– [7]; object instances –a particular chair or table model– [8]; or the room layout [9] are the semantic features that are explicitly modelled in the estimation. Maps at object or layout level contributes to densify the model of the scene in SfM and SLAM. Nevertheless, the use of these features suffer from the usual low precision-recall rates in the classification of some objects and layout parts, and an accurate and dense semantic reconstruction is still far from reach.

In this paper we propose for the first time the use of superpixels as a *middle-level* feature –between *low-level* points and lines and *high-level* objects and layout– to represent low-texture areas in monocular SLAM. Such low-texture regions are difficult to reconstruct with other types of features. Superpixels are defined as a set of pixels that are close in the image and in the color spaces; and are local, coherent and preserve most of the structure of the image [10].

While the main advantage of the superpixels lies in their capability of describing poorly-textured areas, they have several drawbacks that have prevented their use in SfM and monocular SLAM: First, their repeatability is low and highly dependent on specularities, camera bright/gain and the content of the image. Second, due to this low repeatability and the lack of robust descriptors, the matching is difficult.

The contribution of this paper is the use of the superpixel contours as its only descriptor, and the proposal of a matching and an optimization algorithm to estimate their position and orientation in 3D. We assume that superpixels correspond to planar surfaces, use a homography model and minimize the distance between the contours. Our optimization of the superpixel parameters runs in two steps: First, we use a Monte Carlo approach to initialize new superpixels in the map. After they are initialized, we actively search for correspondences in other frames and minimize a robust cost function of the reprojection error to optimize their geometric parameters.

We have integrated our superpixel reconstruction algorithm into the SLAM system known as *Parallel Tracking and Mapping* (PTAM) [1]. The original PTAM produces a point-based 3D map and estimates the camera pose in real-time at 30 frames per second. We experimentally demonstrate that our proposed scheme is able to augment PTAM map with superpixels also in real-time. Finally, we have also performed an error analysis in order to show the accuracy of our algorithm.

We believe that the interest of using superpixels as mid-level features in monocular SLAM resides in the following two points:

- Superpixels allow us to describe textureless regions for which other representations have some limitations: Low-level features are usually designed to describe highly-textured areas. High-level features do not present yet the robustness of low-level ones.
- The use of superpixels might help to reduce the cost of scene mapping from visual sensors. Approaches like DTAM [6] present dense pixel-wise reconstructions that might be essential in certain applications –e.g. augmented reality– but come at the price of a large parallel computation in the GPU. For some other applications, a higher-level though non-pixel-accurate reconstruction might be enough. Superpixels group pixels that are close in the image and color spaces; but still keeps the essentials of the image.

The structure of the paper is as follows: Section II out-

[1]Alejo and Javier are with the I3A, Universidad de Zaragoza {alejocb,jcivera}@unizar.es

lines the related work. Section III describes very briefly the algorithm we use to extract the superpixels. Section IV details the main contributions of the paper, that is, the superpixel matching and optimization algorithms. We show our experimental results in section V; and present our conclusions and lines for future work in section VI.

II. Related Work

A. Visual SLAM and SfM using low-level features

Low-level local features have been predominantly used in SLAM and SfM in the latest decades; and hence the literature corpus is huge. SIFT descriptors have been typically used in SfM [2], [11]; where wide-baseline views are likely to appear and invariance is needed. More simple descriptors, like image templates, have been used in SLAM [12], [1] where the baseline between video frames is small and the template warping can be predicted. Line features have been less used [13], [14] due to the difficulties in the matching.

More recent research has extended 3D reconstruction beyond salient features. [5] presented a method implemented as a match, expand, and filter procedure, starting from a set of salient points, and iteratively expanding them. DTAM [6] estimates in real-time a dense scene reconstruction using a smooth regulariser. But still, in both cases, large textureless areas are a problem. It is this latest problem the one we address in this paper.

B. Visual SLAM and SfM using high-level features

High-level features have just only been started to be used in SfM and visual SLAM. [7] jointly optimizes the object category recognition and registration with the standard SfM point-based reconstruction and camera pose estimation. The performance of both processes benefit from the correlation between them.

In the visual SLAM domain, [15] is able to recognize and register a set of known objects in a monocular SLAM map through the local features in them. [8] goes a step further, and uses known objects directly as the map features.

The layout of a room is used in [16], [9] as a high-level feature –in the latest work jointly with multiview geometry–; for which its 3D parameters are estimated from the image information.

C. Superpixels and SfM

Piecewise-planar 3D reconstructions have been proposed from different angles in monocular SLAM and SfM [17], [18]; but they have not used superpixels due to the above mentioned low repeatability. The use of superpixels in SfM has only been explored in [19]. This work performs a multi-view reconstruction of superpixels in a structured Manhattan-like scene with high texture. In this paper, we are able to reconstruct superpixels in *any* direction in non-Manhattan scene with low texture. Also, we demonstrate the real-time performance of the algorithm when integrated within a SLAM system. Differently from [19], we use the contour of the superpixels as their only descriptor.

III. Superpixels

We use the algorithm by Felzenszwalb et al. [20] to segment the PTAM keyframes into superpixels. In this paper, the image $\mathcal{I}$ is modelled as an undirected graph $G(V, E)$ where the vertices V are the pixels in the image $\mathbf{u} \in V$. For every edge between neighbouring pixels $(\mathbf{u}_i, \mathbf{u}_j) \in E$, a dissimilarity measure is defined based on the color difference of the two pixels $w((\mathbf{u}_i, \mathbf{u}_j)) = |\mathcal{I}(\mathbf{u}_i) - \mathcal{I}(\mathbf{u}_j)|$. The goal is to obtain a partition $\mathcal{S}$ such that each component $\mathcal{C} \in \mathcal{S}$ contains *similar* elements. This basically means that edges between vertices in the same component should have low weights and edges between vertices in different components should have higher weights. The internal difference of a component $Int(\mathcal{C})$ is defined as the maximum dissimilarity w between the edges of the minimum spanning tree of the component $\mathcal{C}$.

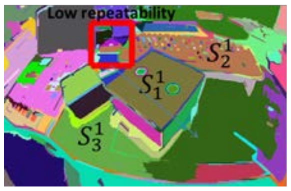
The algorithm starts with n vertices and m edges, being each vertex its own component. E is sorted into $(o_1, ..., o_m)$ by non-decreasing edge weight and the following step is repeated m times: If the edge q, where $q = 1, ..., m$, has its vertices in different components and $w(o_q)$ is small compared to the internal difference of both components, then the two components are merged.

We use this specific superpixel method because it adapts to the image contours accurately and the superpixels size is irregular. They are as large as they need to in order to occupy the homogeneous colour region and does not have a prefixed regular size like, for example, [21]. There are 3 configuration parameters: the threshold function, the minimum component size enforced by post-processing, and the standard deviation used to smooth the input image before segmentation. These are respectively set to 200, 20 and 1 in our paper.

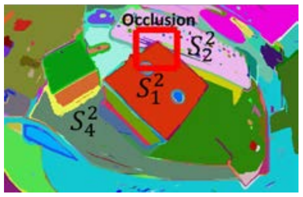
Figure 1 shows an illustrative example of the superpixel segmentation of two SLAM keyframes using [20], the results of our 3D reconstruction algorithm and the associated challenges. Figures 1(a) and 1(b) show the two frames of a sequence that have been selected as keyframes by PTAM. Figures 1(c) and 1(d) show the resulting superpixels for these two keyframes. Notice how the pixels have been clustered in larger entities; but the structure of the scene is mostly captured. Observe superpixels s_1^1 and s_1^2 in Figures 1(c) and 1(d): Notice how they accurately capture the same untextured area of the scene, that is the top of the box. Now look at superpixels s_2^1 and s_2^2. While they also both capture the keyboard, their repeatability is lower: In Figure 1(c) the superpixel is extended to the left into the mouse (red square in the figure). Notice also that this superpixel is partially occluded, and the occlusion is different in both keyframes due to the different viewpoint. In Figure 1(e) we show that our algorithm is able to reconstruct both superpixels in spite of the challenges mentioned for superpixel s_2. Finally, look at superpixels s_3^1 and s_4^2 as an example of an even lower repeatability of the superpixels: Due to the reflection on the table, both superpixels correspond to the same area but the overlap is small. Our algorithm correctly considers that these two superpixels do not match.

(a) Keyframe #1

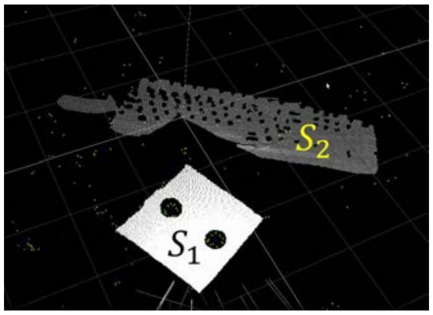
(b) Keyframe #2



(c) Keyframe #1 superpixels.



(d) Keyframe #2 superpixels.



(e) 3D reconstruction of superpixels s_1 and s_2 from keyframes #1 and #2.

Fig. 1. Superpixels example. Figures 1(a) and 1(b) show two PTAM keyframes of a sequence. Figures 1(c) and 1(d) show the superpixel segmentation. Notice that the superpixel s_1 is imaged in both keyframes (s_1^1 and s_1^2) quite accurately. The superpixel s_2, though seen in both keyframes (s_2^1 and s_2^2), presents a low repeatability: observe in a red square in Figure 1(c) that s_2^1 is extended outside the keyboard, while s_2^2 is not. Note also that the occlusion is different due to the different viewpoint of both superpixels. The algorithm we propose is able to reconstruct correctly the superpixels s_1 and s_2, see Figure 1(e). Superpixels s_3 and s_4 are in the same region, but they are so different that our algorithm does not consider them to match.

IV. Mapping points and superpixels

A. Map State Vector

We define the state of our map as $\mathbf{x} = \left(\mathbf{x}_C^\top \;\; \mathbf{x}_M^\top\right)^\top$; where $\mathbf{x}_C$ corresponds to camera states and $\mathbf{x}_M$ is the map model. The camera state vector $\mathbf{x}_C$ stores the camera state of m selected keyframes that summarize the geometry of the scene $\mathbf{x}_C = \left(\mathbf{c}_1^\top \;\ldots\; \mathbf{c}_j^\top \;\ldots\; \mathbf{c}_l^\top \;\ldots\; \mathbf{c}_m^\top\right)$. The state vector $\mathbf{c}_j$ for each camera j is composed by its position $\mathbf{t}_j$ and orientation $\mathbf{r}_j$ in a common reference frame $\mathbf{c}_j = \left(\mathbf{r}_j^\top \;\; \mathbf{t}_j^\top\right)^\top$.

The map model contains n point features and q superpixel features (see Fig 2)

$$\mathbf{x}_M \;=\; \left(\mathbf{p}_1^\top \ldots \mathbf{p}_i^\top \ldots \mathbf{p}_n^\top \; \mathbf{s}_1^\top \ldots \mathbf{s}_k^\top \ldots \mathbf{s}_q^\top\right)^\top \quad (1)$$

Every point $\mathbf{p}_i$ is modelled with its Euclidean coordinates in a common reference frame $\mathbf{p}_i = \left(X_i^W \; Y_i^W \; Z_i^W\right)^\top$.

We make the assumption that every superpixel corresponds to an approximately planar surface; and we store in the state vector the azimuth-elevation angles of its normal and its distance to the origin $\mathbf{s}_k = \left(\theta_k \; \phi_k \; d_k\right)^\top$. The plane normal $\mathbf{n}_k$ can be computed as $\mathbf{n}_k = \left(cos(\theta_k)sin(\phi_k) \; sin(\theta_k)sin(\phi_k) \; cos(\phi_k)\right)^\top$.

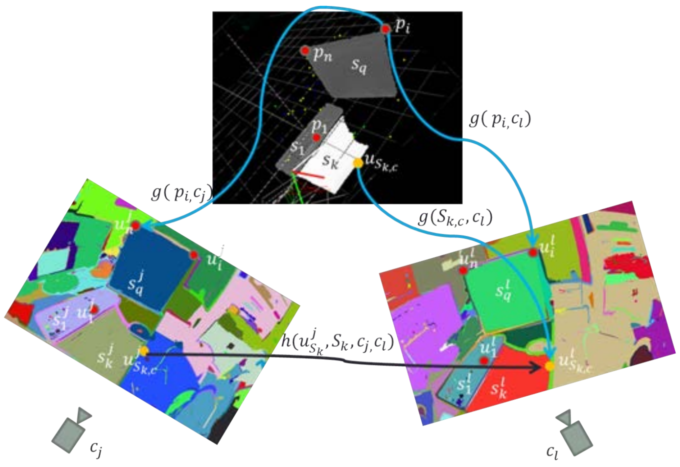


Fig. 2. Notation convention for the main concepts discussed in the paper.

B. Projection Model

We use the FOV projection model of [1]. Every point $\mathbf{p}_i = \left(X_i^W \; Y_i^W \; Z_i^W\right)^\top$ is imaged in the camera $\mathbf{c}_j$ at coordinates $\mathbf{u}_i^j = \begin{pmatrix} x_i^j \\ y_i^j \end{pmatrix} = \mathbf{g}\left(\mathbf{p}_i, \mathbf{c}_j\right)$ following the equations

$$\begin{pmatrix} x_i^j \\ y_i^j \end{pmatrix} = \begin{pmatrix} x_0 \\ y_0 \end{pmatrix} + \begin{pmatrix} f_x & 0 \\ 0 & f_y \end{pmatrix} \frac{r}{r'} \begin{pmatrix} \frac{X_i^{c_j}}{Z_i^{c_j}} \\ \frac{Y_i^{c_j}}{Z_i^{c_j}} \end{pmatrix} \quad (2)$$

$$r = \sqrt{\frac{X_i^{c_j\,2} + Y_i^{c_j\,2}}{Z_i^{c_j\,2}}}, \quad r' = \arctan\left(2r\tan\left(\frac{w}{2}\right)\right) \quad (3)$$

Where $\left(f_x \;\; f_y\right)^\top$ is the focal length at horizontal and vertical direction, $\left(x_0 \;\; y_0\right)^\top$ the principal point and w the distortion coefficient. $\left(X_i^{c_j} \; Y_i^{c_j} \; Z_i^{c_j}\right)^\top$ are the Euclidean coordinates of the point $\mathbf{p}_i$ in the reference frame of the camera $\mathbf{c}_j$. See Figure 2 for an illustrative scheme of the model.

C. Homography Model

We will assume that image superpixels correspond to planar areas in the scene; and hence will use the homography model for their optimization. We believe that this is a reasonable assumption in man-made environments. In any case, notice that every superpixel that does not hold this assumption will be rejected in our initialization step by the robust cost function and will not be added to the map.

Every pixel $\mathbf{u}_{s_k}^j$ belonging to the superpixel $\mathbf{s}_k$ in image j is mapped into the image coordinates in $\mathbf{u}_{s_k}^l$ in image l via the homography model $\mathbf{H}_{s_k}$

$$\mathbf{u}^l_{s_k} = \mathbf{h}\left(\mathbf{u}^j_{s_k}, \mathbf{s}_k, \mathbf{c}_j, \mathbf{c}_l\right) = \mathbf{H}_{s_k}\mathbf{u}^j_{s_k} \quad (4)$$

where the homography matrix $\mathbf{H}_{s_k}$ is computed as

$$\mathbf{H}_{s_k} = \mathbf{K}_l\left(\mathbf{R} + \mathbf{t}\mathbf{n}_k^T/d_k\right)\mathbf{K}_j^{-1} \quad (5)$$

Where $\mathbf{R}$ and $\mathbf{t}$ are the relative rotation and translation between cameras j and l. $\mathbf{n}_k$ and d_k are the estimated normal and distance of the superpixel s_k in the reference frame of the camera j. See Figure 2 for an illustrative scheme of the model.

D. Superpixel Initialization

Superpixels are initialized from two keyframes $\mathbf{c}_j$ and $\mathbf{c}_l$ for which we have their poses from the PTAM estimation. As the first step, we extract all the superpixels $\mathcal{S}_j$ and $\mathcal{S}_l$ from the two keyframes using the code from [20]. We use a Monte Carlo approach for the correspondence search: For every superpixel $\mathbf{s}_k$ in $\mathcal{S}_j$ we create several hypotheses $\mathbf{s}^h_k$ spread over the possible configurations (see section IV-G for practical details). The superpixel hypotheses are ranked according to the reprojection error of the superpixel contours

$$\epsilon_{s^h_{k,c}} = ||\mathbf{u}^l_{s^h_{k,c}} - \mathbf{h}\left(\mathbf{u}^j_{s^h_{k,c}}, \mathbf{s}_k, \mathbf{t}_j, \mathbf{t}_l\right)|| \quad (6)$$

The superpixel hypothesis $\mathbf{s}^h_k$ with the smallest error $\epsilon_{s^h_{k,c}}$ is taken as the initial seed for the map optimization if it is below a certain threshold.

E. Superpixel Optimization

As in [19], we make the assumption that the camera poses computed from a point-based Bundle Adjustment are accurate enough and optimize separately the components of our state vector. First, we estimate the camera poses and point positions by minimising the standard Bundle Adjustment cost function

$$\{\hat{\mathbf{x}}_C, \hat{\mathbf{p}}_1, \ldots, \hat{\mathbf{p}}_n\} = \underset{\mathbf{x}_C, \mathbf{p}_1, \ldots, \mathbf{p}_n}{\arg\min} \sum_{j=1}^{m}\sum_{i=1}^{n} \rho(\epsilon^j_i) \ , \quad (7)$$

where ϵ^j_i the reprojection error of every map point $\mathbf{p}_i$ in a camera $\mathbf{c}_j$

$$\epsilon^j_i = \mathbf{u}^j_i - \mathbf{g}\left(\mathbf{p}_i, \mathbf{c}_j\right) \ . \quad (8)$$

After that, we estimate the superpixel parameters assuming that the camera poses are known

$$\{\hat{\mathbf{s}}_1, \ldots, \hat{\mathbf{s}}_q\} = \underset{\mathbf{s}_1, \ldots, \mathbf{s}_q}{\arg\min} \sum_{k=1}^{q}\sum_{c} \rho(\epsilon_{s_{k,c}}) \ , \quad (9)$$

where $\epsilon_{s_{k,c}}$ is the reprojection error of the superpixel contours

$$\epsilon_{s_{k,c}} = \mathbf{u}^l_{s_{k,c}} - \mathbf{h}\left(\mathbf{u}^j_{s_{k,c}}, \mathbf{s}_k, \mathbf{c}_j, \mathbf{c}_l\right) \ . \quad (10)$$

In the above equation, $\mathbf{u}^l_{s_{k,c}}$ refers to the pixels $\mathbf{u}$ that belong to the contour c of the superpixel s_k in the image l.

Instead of minimizing directly both errors, we minimize a robust function of them $\rho(\epsilon)$. Specifically, we use the t-distribution [22] and hence our robust error function is

$$\rho(\epsilon) = \epsilon^2 \frac{\Gamma(\frac{v+1}{2})}{\Gamma(\frac{v}{2})\sqrt{\pi v \sigma^2}} \left(1 + \frac{1}{v}\frac{(\epsilon-\mu)^2}{\sigma^2}\right)^{\left(-\frac{v+1}{2}\right)} \quad (11)$$

Where v corresponds to the degrees of freedom and is set to 5, Γ is the gamma function and μ and σ are the mean and the standard deviation of the error ϵ.

Using a robust cost function is of key importance for the reconstruction of 3D superpixels; as their repeatability is low and the contours might slightly vary from one keyframe to the next one.

F. Active Matching

Once a superpixel is initialized from two keyframes, as described in section IV-D, we search for it in the next keyframes actively. We project the superpixel contours in the new keyframe and we search for the closest superpixel within a neighboring area. If the reprojection error is under a threshold, we consider it as the match of the initialized one.

The reprojection error for a 3D contour point $\mathbf{p}_{s_{k,c}}$ of a superpixel $\mathbf{s}_k$ in a camera $\mathbf{c}_j$ is computed using the FOV model $\mathbf{g}$ from section IV-B.

$$\epsilon_j = \mathbf{u}^j_{s_{k,c}} - \mathbf{g}(\mathbf{p}_{s_{k,c}}, \mathbf{c}_j) \ ; \quad (12)$$

Where $\mathbf{u}^j_{s_{k,c}}$ stands for the closest point to $\mathbf{g}(\mathbf{p}_{s_{k,c}}, \mathbf{c}_j)$ in contour of superpixel $\mathbf{s}_k$ in camera j.

G. Practical Considerations

In the superpixel initialization step (section IV-D) we use a hierarchichal Monte Carlo approach to estimate the normal and distance to the origin. In a first step, θ is in the range $[0, \pi]$, ϕ is in the range $[0, \pi]$ and d is a distance in the range $[0.66 d_{min}, 1.5 d_{max}]$. The 3D points which reprojection lie in the cameras we want to reconstruct are selected to calculate d_{min} and d_{max} which are the minimum and maximum distance respectively from the 3D points to the camera origin. We use 500 superpixel hypotheses. After that, in a second step we take as seed the best hypothesis until the current iteration (lower reprojection error in equation 6) and test a new set of hypotheses with the angle θ in the range $[\theta_{best} - 0.25,\ \theta_{best} + 0.25]$, the angle ϕ in the range $[\phi_{best} - 0.25,\ \phi_{best} + 0.25]$ and d in the range $[d_{best} - (d_{max} - d_{min})/50),\ d_{best} + (d_{max} - d_{min})/50]$. In this second step we test 350 hypotheses. In a third step we test 150 hypotheses within the intervals $[\theta_{best} - 0.125,\ \theta_{best} + 0.125]$, $[\phi_{best} - 0.125,\ \phi_{best} + 0.125]$ and $[d_{best} - (d_{max} - d_{min})/25),\ d_{best} + (d_{max} - d_{min})/25]$ for θ, ϕ and d respectively. The threshold for the superpixel correspondences used in the initialization and active matching (sections IV-D and IV-F) is the following: With the best matching hypothesis, we compute the left-right and right-left reprojection error (from image j to image l and from image l to image j) and we count the percentage of contour points which error is under a threshold of 4.5 pixels. If this percentage is higher than 66% we consider that the superpixels match.

After initialization, every time a new keyframe is added we project all the superpixels in the map and apply the active matching algorithm in section IV-F. We use the same matching threshold defined in the previous paragraph; if a correspondence is found we add it as a constraint to the optimization described in secion IV-E. If there is a superpixel without matches, we try to initizalize it with the previous keyframes and the algorithm in section IV-D.

PTAM adds a new keyframe when the camera is a minimum distance away from the nearest keyframe already in the map. This minimum distance depends on the depths of observed features, so it will be bigger when we are far from the keypoints. For our purposes the keyframes added by PTAM are sometimes too close to each other; and we need more parallax for our superpixels estimation. For this reason, we did a relatively large displacement in initialization and we only add a PTAM keyframe if the camera translation is close to the initialization motion.

V. Experimental Results

A. Comparison against Multiview Stereo

The aim of this first experiment is to compare our reconstruction algorithm with a state-of-the-art point-based one oriented to create dense reconstructions. We have chosen the multiview stereo called PMVS [5], due to the availability of the code. PMVS usually receives as input a set of wide-baseline images, their poses being estimated by the Bundle Adjustment optimization of the software package Bundler [11]. In order to make a fair comparison, we run both PMVS and our superpixel SLAM algorithm on a set of 752×480 images with pose information from Bundler. Notice that this working mode should be similar to operating in the mapping thread of PTAM; as this latest one also operates on a set of frames with a seed pose information. The only difference is that PTAM selects automatically the keyframe set; but we think this does not influence our algorithm. In any case, in section V-B we also present experimental results inside a PTAM framework.

Figure 3 summarizes the result of this experiment. Figures 3(a), 3(b) and 3(c) show 3 of the images of the scene we chose for our experiment. Notice that the scene is mostly composed of two untextured walls in a lab. Figure 3(d) shows a 3D reconstruction of salient points, estimated using Bundler [11]. Only the few textured areas of the scene, corresponding to the posters and the backpack, are reconstructed. Multiview stereo approaches [5] are able to produce a more dense reconstruction from an initial Bundle Adjustment one. But, as shown in Figure 3(e), the large and untextured wall regions are still a challenge for these techniques. Figure 3(f) shows the 3D estimation of image superpixels, using the algorithm proposed in this paper. Large and untextured areas can be correctly modelled as superpixels; and hence reconstructed. Finally, Figures 3(g) and 3(h) show a joint reconstruction of points and superpixels. Observe that the combination of the two features is able to produce an accurate and almost dense model of the imaged scene.

TABLE I
Computational Cost

		Computational Cost [seconds]		
Image pair	Superpixels	Extraction	Initialization	Optimization
1	2	0.3758	0.7003	0.0055
2	2	0.3553	1.5447	0.0036
3	2	0.3603	3.0382	0.0056

Table I details the cost of our superpixel reconstruction algorithm for 3 different image pairs in the above experiment. For each experiment, 2 superpixels per image pair were matched, initialized and reconstructed. This computational cost was measured in a computer with a 3.5 GHz Intel Core i7-3770K processor and 8.0 GB of RAM memory. Superpixel extraction using the algorithm in [20] takes around 0.18 seconds per image.

The superpixel initialization is the more expensive step and its cost is highly variable, depending on the specific superpixels to match. This large cost is due to the high number of superpixel hypotheses. In our experiments it took from 0.7 seconds in the first experiment and up to 3 seconds in the third one. Though this latest cost might seem high, it is worth noticing several factors. Firstly, the initialization is only made once per superpixel. Once a superpixel is initialized, the following optimizations described in secion IV-E only take some milliseconds. And secondly, this cost might be easily optimized either by introducing stronger priors on the superpixel parameters –and hence reducing the number of hypotheses–, or by a parallel evaluation of such hypotheses in the GPU.

Looking at this costs, we consider reasonable the use of this algorithm in the mapping thread of the real-time Parallel Tracking and Mapping system of [1]. This is what we evaluate in the next section.

B. Using Superpixels in Monocular Parallel Tracking and Mapping

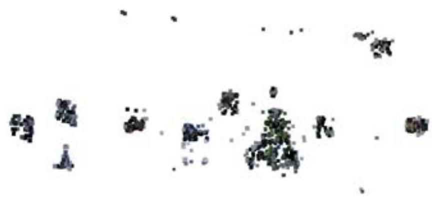
The aim of this experiment is to demonstrate real-time performance of the proposed superpixel reconstruction algorithm within a point-based SLAM system (PTAM [1]). We run our system in 3 live image sequences, taken by a VGA camera in a desktop scene. In the desktop, we deployed several textureless boxes in an unorganized setting, in order to demonstrate that our system can reconstruct superpixels in any orientation.

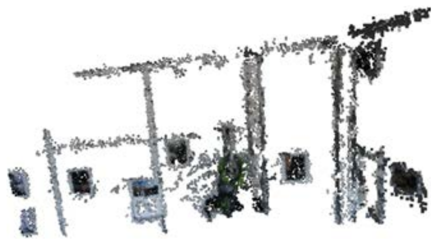
In Figures 4 and 5 we show the results of two of the experiments. In both figures the top image shows the reprojection of the reconstructed 3D points, the middle figure shows the reprojection of the reconstructed superpixels, and the third image shows a 3D view of the estimated superpixels. Notice the accuracy of the reconstruction in both cases, and how the use of superpixels allow to reconstruct the

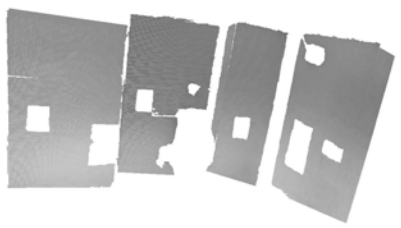
(a) Image 1 of a low-textured lab scene.

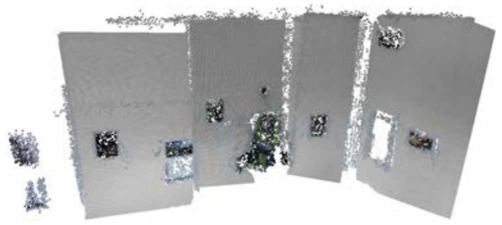
(b) Image 2 of a low-textured lab scene.

(c) Image 3 of a low-textured lab scene.


(d) Point-based reconstruction using Bundle Adjustment [11]. Notice the sparsity of the reconstruction.


(e) Point-based reconstruction using Multiview Stereo [5]. The reconstruction has been extended, but still the large untextured walls cannot be reconstructed.


(f) 3D reconstruction of the largest superpixels, corresponding to the large untextured areas of the walls.


(g) Side view of the 3D reconstruction of points and superpixels

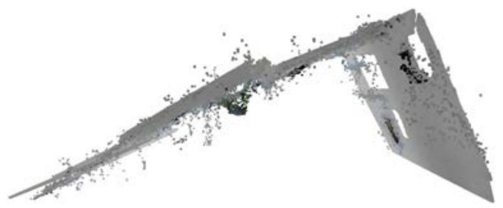

(h) Top view of the 3D reconstruction of points and superpixels

Fig. 3. Comparison of the proposed algorithm with a Bundle Adjustment and a Multiview Stereo reconstructions.

textureless boxes and the table. As in the previous section, the reconstruction is much more dense than using point features alone. Both experiments were done in real-time. The video accompanying the paper[1] shows a real-time experiment in the same scene.

Figure 6 shows an image from our third real-time SLAM experiment. In this case, we focus on showing the reprojection error in the Figure 6(e). Notice how the contour points of the superpixel in the top of the box overlap with the reprojected contour points of the corresponding superpixel in the second keyframe.

C. Quantitative Analysis with Ground Truth Depth

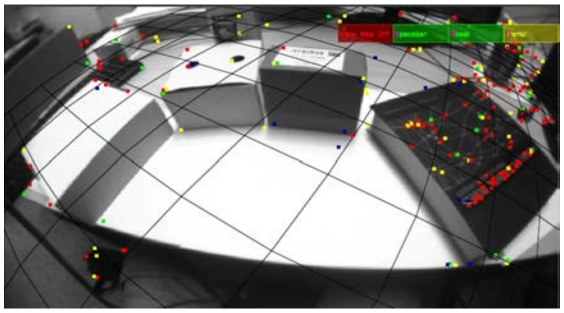
The goal of this section is to provide a quantitative analysis of the achievable accuracy of our algorithm. In order to do that, we recorded several image sequences with a *RGB-D* camera, used the depth channel *D* as the ground truth depth, and estimated 3D superpixel reconstructions using the *RGB* data. A first thing to notice is that the superpixel estimation error will be highly dependant on the image parallax [2]. We evaluated the errors in two different experimental settings: A low-parallax one in the lab environment of section V-A, and several high-parallax ones in the desktop environment with sequences similar to section V-B. In the lab –low-parallax– sequence, our algorithm estimated 12 superpixels. In the desktop –high-parallax– sequences, we estimated 5 superpixels.

Before computing the error, we have to normalise the scale s of our monocular 3D superpixel reconstruction. We do that by minimizing, for every keyframe j and every pixel $\mathbf{u}_j$ that was reconstructed, a robust cost function ρ the depth error between the *D* channel $D(\mathbf{u}_j)$ and the estimated depth $d(\mathbf{u}_j)$ multiplied by the scale factor s.

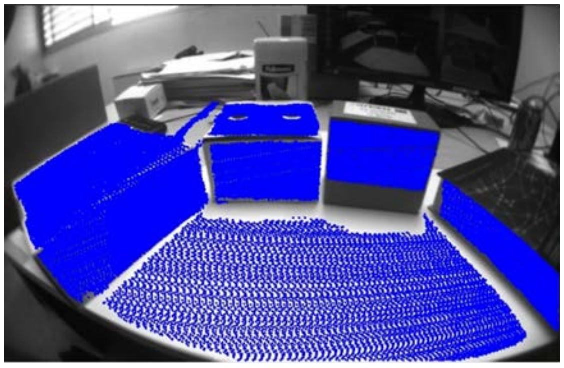
$$\hat{s} = \arg\min_{s} \sum_{j} \rho(D(\mathbf{u}_j) - s \times d(\mathbf{u}_j)) \ . \qquad (13)$$

We can extract the ground truth parameters for the superpixel $\mathbf{s}_k^{GT} = (\theta_k^{GT} \ \phi_k^{GT} \ d_k^{GT})^\top$ from the *RGB-D* data, by a least-squares optimization. We define the superpixel error as $(\Delta\theta_k \ \Delta\phi_k \ \Delta d_k)^\top = (\theta_k^{GT} - \theta_k \ \phi_k^{GT} - \phi_k \ d_k^{GT} - sd_k)^\top$. We can also compute the error of each point $\Delta P = ||\mathbf{P}_k^{GT} - s\mathbf{P}_k||$ belonging to a superpixel k. The ground truth value

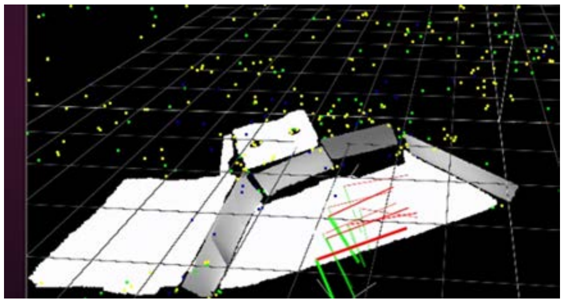
[1]The video is also available at `http://youtu.be/Cv7PeXGfU_E`



(a) Keyframe

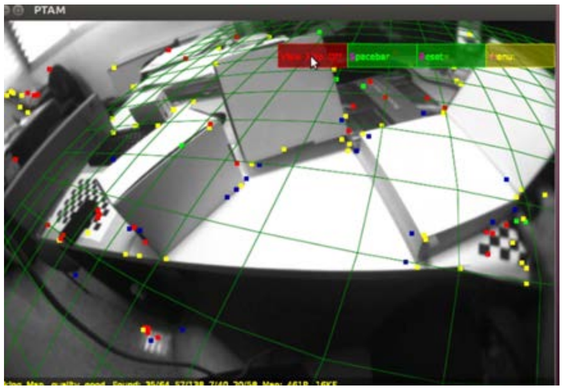


(b) Reprojected superpixels

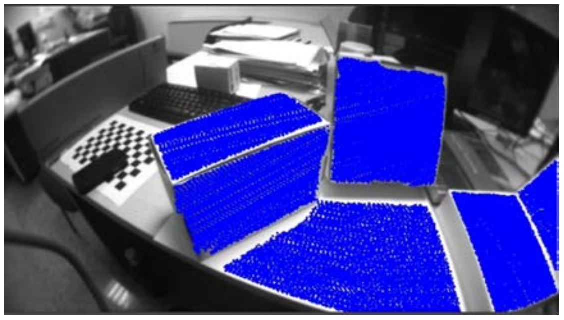


(c) 3D reconstruction

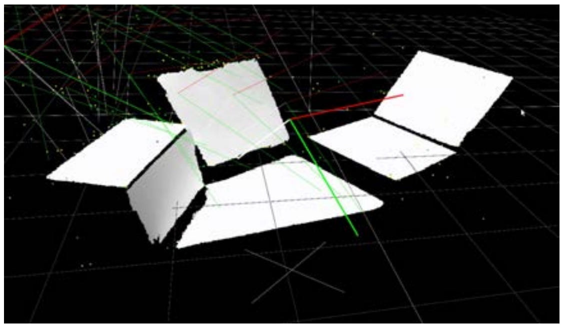
Fig. 4. Fig 4(a) is a PTAM keyframe of a sequence. Fig 4(b) shows the projection of the of the reconstructed superpixels and Fig 4(c) is the 3D view of the estimated superpixels.



(a) Keyframe



(b) Reprojected superpixels



(c) 3D reconstruction

Fig. 5. Fig 5(a) is a PTAM keyframe of a sequence. Fig 5(b) shows the reprojection of the of the reconstructed superpixels and Fig 5(c) is the 3D view of the estimated superpixels.

$\mathbf{P}_k^{GT}$ is extracted from the *RGB-D* keyframe. The estimated value $\mathbf{P}_k$ is extracted from the intersection of the superpixel plane $\mathbf{s}_k$ and the backprojected ray from the keyframe. Table II shows the median errors for the plane parameters $(\Delta\theta_k \ \Delta\phi_k \ \Delta d_k)^\top$ and the points ΔP belonging to the superpixels.

Notice first the large difference in the metric parameters (Δd and ΔP) between the low-parallax and high-parallax sequences. Such values will be highly affected by the parallax. The error for the d parameter will depend on the normal, so we find more relevant the error ΔP of the points belonging to the superpixel. In any case, notice the high accuracy of the reconstruction. Angular errors are around $2°$, and the median point error can be lower than $2cm$ for high-parallax sequences of desktop scenarios –with typical distances of $1m$– and around $8cm$ for low-parallax sequences of a room environment –with typical distances of $10m$.

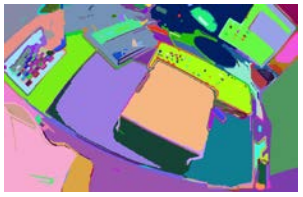
VI. CONCLUSIONS

In this paper, we have explored for the first time the use of superpixels in monocular SLAM. We have proposed algorithms for superpixel contour-based matching, 3D initialization, active matching and optimization. Our experimental results show that the use of superpixels allow to reconstruct scene areas where the texture is low. We have compared our algorithm against the state-of-the-art multiview stereo code PMVS [5], and demonstrated that using superpixels allow to reconstruct large and untextured areas that PMVS cannot. We have integrated our contributions in a point-based Parallel Tracking and Mapping system [1] and we have shown that our algorithm can work as a parallel thread in a real-time SLAM system, allowing to fill in areas with low texture.

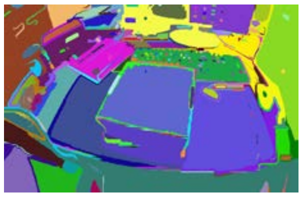
In the experimental results of this paper, we have ob-

(a) Keyframe #1

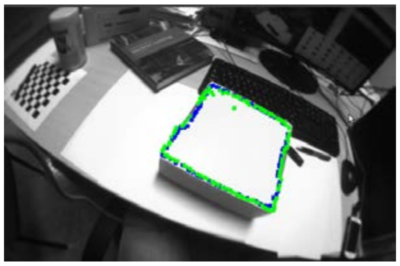
(b) Keyframe #2



(c) Keyframe #1 superpixels.



(d) Keyframe #2 superpixels.



(e) Keyframe #1 reprojection error. In blue the predicted superpixel contour, in green the extracted superpixel.

Fig. 6. Superpixel reprojection error.

TABLE II

MEDIAN ERRORS FOR THE SUPERPIXEL 3D PARAMETERS

	Low Parallax (Lab)	High Parallax (Desktop)
$\Delta\theta[^\circ]$	2.3	1.1
$\Delta\phi[^\circ]$	2.2	3.4
$\Delta d[cm]$	9.7	2.6
$\Delta P[cm]$	7.8	1.7

served some topics that we would like to address as future work. Firstly, as already commented, we believe that the key advantage of the superpixels lies in their capacity for representing low-textured areas. On the other hand, state-of-the-art superpixels tend to be unstable and difficult to match. We would like to explore if superpixels could be made more stable; either learning over past data or using the 3D prior information extracted from the SLAM estimation. We are also interested on making an exhaustive comparison of the superpixel features against local point features and object features regarding repeatability, accuracy and descriptive power.

ACKNOWLEDGMENT

The research here was funded by the Spanish government (Ministerio de Economía y Competitividad) through the projects with references IPT-2012-1309-430000 and DPI2012-32168. We would like to thank the anonymous reviewers of this paper for their insightful comments; that helped to improve the paper.

REFERENCES

[1] G. Klein and D. Murray, "Parallel tracking and mapping for small AR workspaces," in *Sixth IEEE and ACM International Symposium on Mixed and Augmented Reality*, 2007.

[2] R. I. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, 2004.

[3] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[4] H. Bay, A. Ess, T. Tuytelaars, and L. V. Gool, "SURF: Speeded up robust features," *Computer Vision and Image Understanding*, vol. 110, no. 3, pp. 346–359, 2008.

[5] Y. Furukawa and J. Ponce, "Accurate, dense, and robust multiview stereopsis," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 8, pp. 1362–1376, 2010.

[6] R. A. Newcombe, S. J. Lovegrove, and A. J. Davison, "DTAM: Dense tracking and mapping in real-time," in *2011 IEEE International Conference on Computer Vision (ICCV)*, 2011, pp. 2320–2327.

[7] S. Y. Bao and S. Savarese, "Semantic structure from motion," in *2011 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011, pp. 2025–2032.

[8] R. F. Salas-Moreno, R. A. Newcombe, H. Strasdat, P. H. Kelly, and A. J. Davison, "SLAM++: Simultaneous localisation and mapping at the level of objects," in *2013 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013.

[9] A. Flint, D. Murray, and I. Reid, "Manhattan scene understanding using monocular, stereo, and 3d features," in *2011 IEEE International Conference on Computer Vision (ICCV)*, 2011, pp. 2228–2235.

[10] X. Ren and J. Malik, "Learning a classification model for segmentation," in *Ninth IEEE International Conference on Computer Vision*, 2003, pp. 10–17.

[11] N. Snavely, S. Seitz, and R. Szeliski, "Modeling the world from internet photo collections," *International Journal of Computer Vision*, vol. 80, no. 2, pp. 189–210, 2008.

[12] A. J. Davison, N. D. Molton, I. D. Reid, and O. Stasse, "MonoSLAM: Real-time single camera SLAM," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. June, pp. 1052–1067, 2007.

[13] G. Klein and D. Murray, "Improving the Agility of Keyframe-Based SLAM," in *Proceedings of the 10th European Conference on Computer Vision: Part II*. Springer, 2008, pp. 802–815.

[14] J. Solà, T. Vidal-Calleja, J. Civera, and J. M. M. Montiel, "Impact of landmark parametrization on monocular ekf-slam with points and lines," *International journal of computer vision*, vol. 97, no. 3, pp. 339–368, 2012.

[15] J. Civera, D. Gálvez-López, L. Riazuelo, J. Tardós, and J. Montiel, "Towards semantic slam using a monocular camera," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2011, pp. 1277–1284.

[16] V. Hedau, D. Hoiem, and D. Forsyth, "Recovering the spatial layout of cluttered rooms," in *IEEE International Conference on Computer vision*, 2009, pp. 1849–1856.

[17] D. Gallup, J.-M. Frahm, and M. Pollefeys, "Piecewise planar and non-planar stereo for urban scene reconstruction," in *2010 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010, pp. 1418–1425.

[18] A. Argiles, J. Civera, and L. Montesano, "Dense multi-planar scene estimation from a sparse set of images," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2011, pp. 4448–4454.

[19] B. Mičušík and J. Košecká, "Multi-view superpixel stereo in urban environments," *International journal of computer vision*, vol. 89, no. 1, pp. 106–119, 2010.

[20] P. F. Felzenszwalb and D. P. Huttenlocher, "Efficient graph-based image segmentation," *International Journal of Computer Vision*, vol. 59, no. 2, pp. 167–181, 2004.

[21] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Susstrunk, "SLIC superpixels compared to state-of-the-art superpixel methods," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 11, nov 2012.

[22] K. L. Lange, R. J. Little, and J. M. Taylor, "Robust statistical modeling using the t distribution," *Journal of the American Statistical Association*, vol. 84, no. 408, pp. 881–896, 1989.