Scalable Dense Large-Scale Mapping and Navigation

Alberto Pretto, Stefano Soatto and Emanuele Menegatti

Abstract— **This paper describes a scalable dense 3D reconstruction and navigation system suitable for real-time operation. The system represents the environment as the back-projection of a Delaunay triangulation of the omnidirectional image, estimated at each instant from two adjacent views. The cost being minimized (i.e., the reprojection error) is photometric rather than geometric, as in the majority of feature-based reconstruction and navigation systems. While temporal integration would enable more accurate reconstruction, this would carry the computational burden of handling topological changes due to occlusion phenomena. We successfully tested our system in a challenging urban scenario along a large loop using an omnidirectional camera mounted on the roof of a car.**

I. INTRODUCTION

Our goal is to build photometric and geometric models of the environment that can be used for navigation and for object-level map building. We are interested in models that have generative capabilities, so they can be used to synthesize images that can then be compared with object descriptors for localization and recognition. Therefore, rather than the ubiquitous "cloud of points" [10], [4], [5] our model consists of dense surfaces that support a radiance function. However, we also want our models to be usable for navigation, planning, obstacle avoidance and other real-time interaction tasks, so it is paramount that the model be inferred efficiently and in real-time.

We choose to represent the world surrounding the robot instantaneously as the back-projection of a Delaunay triangulation of the omnidirectional image. The nodes are salient feature points detected in one image, and the three-dimensional position of such nodes is computed instantaneously from two adjacent images by minimizing the reprojection error. Because of the large rectification artifacts of omnidirectional sensors, we choose to select point features in the original (unrectified) image, where the triangulation is computed. As a result, the boundaries of the piecewise planar segments of the surface that is reconstructed are not in general straight lines, so the surface is not a proper triangulated surface. However, we find that the errors resulting in the approximation of this surface with a triangulated one more than compensate from the sampling inhomogeneities that follow feature selection on the rectified image.

Our optimization approach is based on an efficient second-order scheme called *ESM* [14]: the topological relationships between the subdivision's vertices are exploited to iteratively divide the optimization in subproblems where only a subset of "independent" vertices are actively processed. This strategy significantly improves the efficiency of the minimization procedure.

Pretto and Menegatti are with the University of Padova, Dep. of Information Engineering (DEI), via Gradenigo 6/B, 35131 Padova, Italy alberto.pretto@dei.unipd.it

Soatto is with the Computer Science Department, University of California (UCLA), Los Angeles - CA, USA

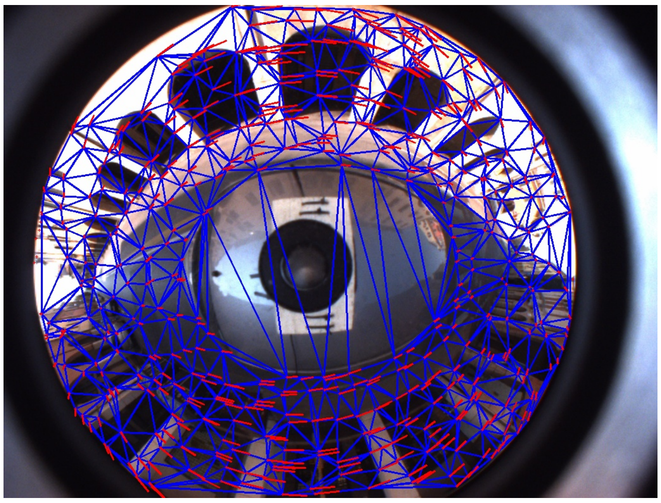


Fig. 1. An example of omnidirectional image with superimposed the Delaunay subdivision (blue segments) anchored on salient points; red segment show the inter-frame displacement of these vertices. We use a mask to suppress the car's roof and the area outside the omnidirectional mirror.

A. Related Work

Current visual SLAM (simultaneous localization and mapping) systems use perspective [10], [4], [5], [6], [21], [8], stereo [11], [7] or panoramic cameras [1], [15], looking for points [10], [4], lines [6], [8] or planar features [21] as visual landmarks. Most of them use stochastic filtering approaches such as Extended Kalman Filters (EKFs), Rao-Blackwellized Particle Filters [16] or graph-based optimization [9].
Jin *et al.* [10] and Davison *et al.* [4] proposed a feature-based SLAM approach using a single perspective camera and an EKF, where a 3D map of the features is built using the bearing only measurements from a monocular camera. A similar approach, but based on the FastSLAM framework, was presented in [5]. In [11] a high-resolution digital elevation map is built from a sequence of stereo image pairs where interest points are detected and matched between consecutive frames. A visual motion estimation algorithm is used to predict the movements, an EKF is used to estimate both the position parameters and the map. The system presented in [1] uses SIFT [12] to compute the similarity between omnidirectional images. Links between robot poses are established based on odometry and image similarity, then a relaxation algorithm

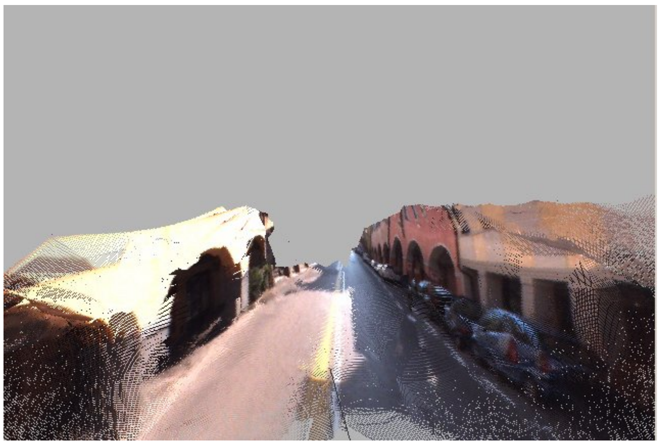


Fig. 2. A dense reconstruction of the surrounding environment obtained with the proposed method. The previous image used for this reconstruction is shown in Fig. 8(a).

is used to generate the map. In [7] a dense metric map of 3D point landmarks for large cyclic environments are built using a Rao-Blackwellised Particle Filter, where SIFT features are extracted from stereo vision and motion estimates are based on sparse optical flow. Eade *et al.* [6] presents a monocular visual SLAM approach using line features, with an efficient algorithm for selecting such landmarks. Higher level landmarks are exploited in [21], where 3D camera displacement and scene structure are computed directly from image intensity discrepancies using an efficient second-order optimization procedure for tracking planar patches. Nistér *et al.* [17] presented a robust *visual odometry* system (i.e., the Visual SLAM subproblem of estimating the robot's ego-motion using vision) based on features tracking and on the five-point algorithm, able to estimate the motion of a stereo camera or a perspective camera in large trajectory. A visual-odometry approach using omnidirectional vision is presented in [19], where the camera trajectory is estimated switching between two different trackers, one homography-based and one appearance-based, according to the distribution of the image points. Recently, Klein *et al.* [8] proposed an effective Visual SLAM approach based on edgelets tracking that exploits a key-frame-based re-localization method in order to recover form tracking failures. In [15], Micusik *et al.* proposed a framework for creating 3D maps in an urban scenario using a panoramic camera modeled as a quadrangular prismatic camera. This approach estimates camera pose by exploiting the epipolar geometry constraints, while a dense 3D map of the environment is built using a superpixel-based multi-view stero method. Cornelis *et al.* [2] integrate in a real-time 3D reconstruction framework an object recognition module in order to deals with the dynamic nature of an urban scenario.

II. DELAUNAY-BASED TRIANGLE MESH SURFACE

Triangulated meshes are commonplace in geometric surface modeling because of their approximation properties as well as the availability of efficient algorithms to perform "geometry processing" operations such as smoothing, interpolation and sub-division. Given a piecewise smooth surface $S \subset \mathbb{R}^3$ that represents the scene, we can approximate it with a triangular mesh $M(S)$ consisting of a pair $(\mathbf{K}, \mathbf{V})$ where $\mathbf{K}$ is a simplicial complex that determines the topology of the mesh and $\mathbf{V}$ is a set of n 3D vertices $\mathbf{V} = \{\mathbf{V}_1, \ldots, \mathbf{V}_n\}$ that defines the shape in $\mathbb{R}^3$.
Assuming that the surface S is completely visible from the camera (or equivalently that there are no occlusions between two adjacent time instants), every 3D vertex $\mathbf{V}_i$ is projected onto the image plane at location $\mathbf{v}_i = \mathring{\Pi}(\mathbf{V}_i)$, $i = \{1, \ldots, n\}$, with $\mathring{\Pi}$ a general projection function. Let us denote the 3D vertex by $\mathbf{V}_i$ and its neighbors with indices $\{j_{i,1}, \ldots, j_{i,n_i}\}$: under the given assumptions, the topology of the projected vertices remains the same, i.e. $\mathbf{v}_i = \mathring{\Pi}(\mathbf{V}_i)$ has as neighbors the projections of the 3D vertices defined by the same indices $\{j_{i,1}, \ldots, j_{i,n_i}\}$. Moreover, 3D points that lie in a facet defined by 3 vertices, will be projected in a triangle defined by the 2D projections of these 3 vertices.

To construct the mesh, we first select a number n of

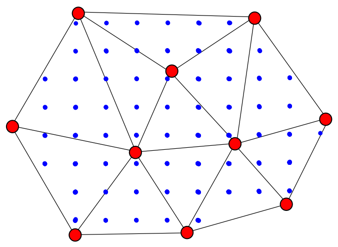


Fig. 3. An example of Delaunay triangulation: red circles are the subdivision's vertices, blue circles the sample points used in the optimization.

salient point features and associate each of them to a vertex $\{\mathbf{v}_1, \ldots, \mathbf{v}_n\}$. In our implementation we have chosen an (approximately) affine co-variant detector [20], although a number of other detectors could be used [22]. In order to maintain a spatial distribution of vertices that is as close as possible to uniform, we partition the images into regular tiles and select a minimum number of point within each tile. Given the set $\{\mathbf{v}_1, \ldots, \mathbf{v}_n\}$, we then define a subdivision of the (un-rectified) image plane into into simplices (triangles in the planar case) each representing the projection of a spatial surface element. We use a Delaunay triangulation technique, that provides a subdivision that maximizes the minimum angle between any two edges in the resulting graph. This avoids "skinny triangles" artifacts. In Fig. 3 we show an example of Delaunay triangulation, where the vertices are depicted with red circles, in Fig 1 we show an omnidirectional image with the extracted features along with the resulting Delaunay triangulation. Building the Delaunay subdivision directly in the omnidirectional image (see Fig. 3) also avoids the discontinuities introduced when unwarping the omnidirectional image into a 360^o panoramic image. We use an image mask in order to avoid searching features outside the area where the omnidirectional mirror is imaged or on in the car's roof.

In this first step of the algorithm, we extract also a set of m images sample points $\{\mathbf{s}_1, \dots, \mathbf{s}_m\}$ that lie inside the triangles (blue circles in Fig. 1), along with the topological information related to the Delaunay subdivision, i.e. the 3 indices $k_{i,1}, k_{i,2}, k_{i,3}$, $i = 1, \dots, m$, which define the vertices of the triangles that the samples belong to. These sample points (possibly, all the image points) will be used in the optimization procedure in order to match the deformation of the triangular patches between consecutive frames.

III. SURFACE TRACKING

Camera motion induces a diffeomorphic deformation of the domain of the image away from occluded regions. This deformation depends both on the motion of the camera and on the three-dimensional shape of the scene. Since for sufficiently small inter-frame motion (sufficiently fast temporal sampling), occluded regions are small, we neglect them in constructing an instantaneous estimate of the three-dimensional layout of the scene. Naturally, this is only possible in an instantaneous inference scenario: Establishing correspondence over time would force us to handle occlusions explicitly, with the obvious computational burden that follows. The only "memory" we enforce is the use of the best estimate of 3D shape from the previous time instant as the initialization of the current iterative procedure.
The iterative procedure we employ is a gradient-based scheme to estimate the position in 3D space of the nodes of the triangulated mesh, along with camera motion parameters, by minimizing the norm of the *reprojection error*. The reprojection error is the difference between the measured images and the images generated by the current estimate of the model. We call $I_t : \mathbb{R}^2 \to \mathbb{R};\ \mathbf{x} \mapsto I_t(\mathbf{x})$ the *previous* image, and similarly $I_{t+1}(\mathbf{x})$ the *current* image). Their relation, assuming Lambertian reflection and constant illumination [23], is given by

$$I_{t+1}(w(\mathbf{x})) = I_t(\mathbf{x}), \quad \mathbf{x} = \pi(\mathbf{X}), \quad \mathbf{X} \in S \subset \mathbb{R}^3 \tag{1}$$

where $\pi(\mathbf{X})$ is the projection of a point on a visible portion of the unknown scene S, and $w : \mathbb{R}^2 \to \mathbb{R}^2$, restricted to the co-visible region, is a diffeomorphism [24] given by

$$w(\mathbf{x}) = \pi\left(g_t \pi_S^{-1}(\mathbf{x})\right). \tag{2}$$

Here $\pi : \mathbb{R}^3 \to \mathbb{R}^2$ is the projection map that compounds the effects of a central projection and the deformation map induced by the omnidirectional mirror, described in the next section. Here π_S^{-1} is the inverse-projection map, that depends on the (unknown) 3D geometry of the scene S, which is the subject of inference, along with $g_t \in SE(3)$, the instantaneous motion of the camera. The reprojection error is simply

$$\phi(g_t, S) = \int_D \left(I_{t+1}(w(\mathbf{x})) - I_t(\mathbf{x})\right)^2 d\mathbf{x} \quad \text{subject to (2)} \tag{3}$$

for the case of the L^2 norm, where D is the domain of the image. One could use other norms, such as L^1, as an alternative [25]. In the next subsections we examine the two unknowns g_t, S and their structure in more detail.

A. Omnidirectional Camera Model

The projection map π describe above is usually taken to be a canonical (central) perspective projection, either onto a plane, or onto an hemisphere. If the points $\mathbf{X} \in S \subset \mathbb{R}^3$ are represented in coordinates relative to the camera reference frame (with the optical center as the origin and the optical axis aligned with the third coordinate axis), these are given by $\bar{\mathbf{x}} = \mathbf{X}/X_3 \in \mathbb{P}^2$ and $\bar{\mathbf{x}} = \mathbf{X}/\|\mathbf{X}\| \in \mathbb{S}^2$ respectively, where $\bar{\mathbf{x}}$ denotes the homogeneous (projective) coordinates. In the case of an omnidirectional camera, we assume that there exists a map $\Pi : D \subset \mathbb{S}^2 \to \mathbb{R}^2$ from the image domain, which can be modeled as a subset of the omnidirectional sphere, to the plane. This map is invertible, and can be estimated and compensated for as part of a calibration procedure, as described in [18].

More in general, when the coordinatization of the world S is relative to a reference other than the camera reference frame, we describe the transformation between the world and the camera frame with $g \in SE(3)$, represented in homogeneous coordinates as a 4×4 matrix $\mathbf{G}$ as customary [13], with rotation component $\mathbf{R} \in SO(3)$ and translation $\mathbf{T} \in \mathbb{R}^3$. In the next section we describe the constraints imposed on the motion parameters by the vehicle kinematics.

B. ICR-based Motion Model

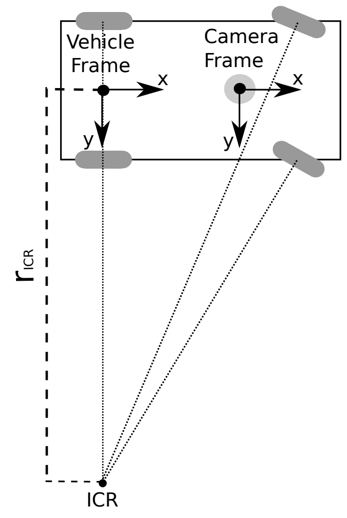


Fig. 4. Ackermann steering geometry.

In this work, we assume that the vehicle moves on a planar ground plane. In case of planar motion, the car's kinematics can be described with the Ackermann steering geometry. In this model, the instantaneous zero-motion lines of each wheel meets at a single point, the *Instantaneous Center of Rotation* (ICR) (Fig. 4). We can approximate this model assuming that the vehicle rotates around a fixed ICR for a discrete time (the shutter interval of the camera). In this way, the motion is composed by a discrete number of rotations around different ICRs.
Let us define the position of the ICR in the y-axis in the car's reference frame as r_{ICR} (the *radius* of curvature,

see Fig. 4). Since scale is unobservable with bearing-only measurements, we normalize it using the norm of translation as the unit, which we will later re-scale using vehicle's speed measurements. By doing so, we parametrize the rigid-body motion with only one parameter, the radius r_{ICR}. We can therefore restrict the instantaneous motion of the camera g_t to be represented in homogeneous coordinates by the 4×4 matrix

$$\mathbf{G}\langle r_{ICR}\rangle = \begin{bmatrix} cos(\alpha) & -sin(\alpha) & 0 & cos\left(\frac{\alpha}{2}\right) \\ sin(\alpha) & cos(\alpha) & 0 & sin\left(\frac{\alpha}{2}\right) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4}$$

where $\alpha = 2 \cdot asin\left(\frac{1}{2 \cdot r_{ICR}}\right)$. Finally, we indicate the rigid-body transformation from the camera reference frame to the vehicle's with $\mathbf{B}_{VC} \in SE(3)$ (see Fig. 4).

C. Tracker Parameterization

As we have seen, camera motion is described by the radius r_{ICR}, and structure is described by the vertices of the Delaunay subdivision along with their depth. Let $\mathbf{s}'_i \in D$ be a sample point and let $\mathbf{v}'_{k_{i,1}}, \mathbf{v}'_{k_{i,2}}, \mathbf{v}'_{k_{i,3}} \in \mathbb{S}^2$ be the three vertices that define the facet in which the sample lies. If the vertices' depths $\lambda_{k_{i,1}}, \lambda_{k_{i,2}}, \lambda_{k_{i,3}}$ are known, the depth of $\mathbf{s}'_i$ can be computed observing that the determinant of the following matrix must be equal to zero:

$$\begin{vmatrix} \mathbf{s}'_i(x) & \mathbf{s}'_i(y) & \mathbf{s}'_i(z) & 1/\lambda(\mathbf{s}'_i) \\ \mathbf{v}'_{k_{i,1}}(x) & \mathbf{v}'_{k_{i,1}}(y) & \mathbf{v}'_{k_{i,1}}(z) & 1/\lambda_{k_{i,1}} \\ \mathbf{v}'_{k_{i,2}}(x) & \mathbf{v}'_{k_{i,2}}(y) & \mathbf{v}'_{k_{i,2}}(z) & 1/\lambda_{k_{i,2}} \\ \mathbf{v}'_{k_{i,3}}(x) & \mathbf{v}'_{k_{i,3}}(y) & \mathbf{v}'_{k_{i,3}}(z) & 1/\lambda_{k_{i,3}} \end{vmatrix} = 0 \tag{5}$$

The depth $\lambda(\mathbf{s}'_i)$ is then given by the following formula:

$$\lambda(\mathbf{s}'_i) = \frac{|\mathbf{M}_4|}{\mathbf{s}'_i(x)\ |\mathbf{M}_1|\ -\ \mathbf{s}'_i(y)\ |\mathbf{M}_2|\ +\ \mathbf{s}'_i(z)|\mathbf{M}_3|} \tag{6}$$

where $|\mathbf{M}_i|$, $i = 1, \dots, 4$ are the minors of the matrix defined in Eq. 5 obtained removing the first row and the $i - th$ column.

In our optimization procedure, we use as motion parameter $\rho \in \mathbb{R}$ the inverse of the radius r_{ICR}, $\rho \triangleq 1/r_{ICR}$. This parametrization improves the stability of the system, enabling moreover to represent explicitly a pure translational motion (setting $\rho = 0$ it means $r_{ICR} \to \infty$). The update rule for the motion parameter is given by:

$$\widehat{\rho} \leftarrow \widehat{\rho} + \widetilde{\rho} \tag{7}$$

where $\widehat{\rho}$ represents the estimate and $\widetilde{\rho}$ an increment to be found.

We represent the structure parameters with the vertices' inverse depths $\zeta_i \triangleq 1/\lambda_i$, $i = \{1, \dots, n\}$. In order to enforce the cheirality constraint, similarly to [21], we also parameterize ζ_i as $\zeta_i = \zeta_i(\xi) = e^{\xi_i}$, with $\xi_i \in \mathbb{R}$. This provides the update rule:

$$\widehat{\zeta_i} \leftarrow \widehat{\zeta_i} \cdot \zeta_i(\widetilde{\xi}) = \widehat{\zeta_i} \cdot e^{\widetilde{\xi_i}} \tag{8}$$

where as usual $\widehat{\zeta_i}$ represents the estimate and $\widetilde{\xi}$ an increment.

D. Estimating the Range Map

We refer the world frame to the camera frame at the time instant t when the previous image I_t was captured. In this section we make the reprojection error (2) explicit with respect to the triangular subdivision represented by the n vertices $\{\mathbf{v}^*_1, \dots, \mathbf{v}^*_n\}$. The image itself is quantized into a discrete set of m points $\{\mathbf{s}^*_1, \dots, \mathbf{s}^*_m\}$. For each vertex $\mathbf{v}^*_i$ in I_t, the corresponding vertex in I_{t+1} is given by the following sequence of transformations:

$$\begin{aligned} &\mathbf{v}^*_i \xrightarrow{\Pi^{-1}} \mathbf{v}^{*\prime}_i \xrightarrow{\lambda^*_i} \mathbf{V}^*_i \xrightarrow{\mathbf{B}_{VC}} \mathbf{V}^{*,car}_i \rightarrow \\ &\xrightarrow{\mathbf{G}\langle r_{ICR}\rangle} \mathbf{V}^{car}_i \xrightarrow{\mathbf{B}^{-1}_{VC}} \mathbf{V}_i \xrightarrow{\lambda^{-1}_i} \mathbf{v}'_i \xrightarrow{\Pi} \mathbf{v}_i \end{aligned} \tag{9}$$

The transformation Π^{-1} maps a vertex from the (unwarped) image plane onto the corresponding normalized coordinate on the unit sphere; multiplication by λ yields the 3D coordinates, then transformed to the vehicle's reference frame by $\mathbf{B}_{VC}$. This reference frame moves with a rigid-body motion (Eq. 4). The chain is then reversed to yield the projection of the original point onto the new image plane. For the sample points $\{\mathbf{s}^*_1, \dots, \mathbf{s}^*_m\}$, the sequence of transformations is the same with the depth computed using Eq. 6. As expected from (2), the projection of a vertex from the previous image to the current image depends on both depth, and camera motion. The projection of a sample point depends on the motion parameter and the depths of the *three* vertices that define the triangle that it belongs to. Given the parametrization presented in Sec. III-C, our optimization procedure aims to find the parameters $\rho \in \mathbb{R}$ and $\xi \in \mathbb{R}^n$, $\xi = [\xi_1, \dots, \xi_n]'$ that minimize:

$$\begin{aligned} \phi(\rho, \xi) = \frac{1}{2}\sum_{i=1}^{n} \left[I_{t+1}(\mathbf{v}_i) - I_t(\mathbf{v}^*_i)\right]^2 + \\ + \frac{1}{2}\sum_{j=1}^{m} \left[I_{t+1}(\mathbf{s}_j) - I_t(\mathbf{s}^*_j)\right]^2 \end{aligned} \tag{10}$$

where, remembering Eq. 7,8 and 9:

$$\mathbf{v}_i = \Pi\left\{\lambda^{-1}_i \mathbf{B}^{-1}_{VC} \mathbf{G}\left\langle \frac{1}{\widehat{\rho} + \widetilde{\rho}} \right\rangle \mathbf{B}_{VC} \left[\frac{1}{\widehat{\zeta_i}\zeta_i(\widetilde{\xi})} \Pi^{-1}(v^*_i)\right]\right\} \tag{11}$$

The computation of $\mathbf{s}_j$ is slightly more complex, due to the fact that its depth depends on the parameters $\xi_{k_{j,1}}, \xi_{k_{j,2}}, \xi_{k_{j,3}}$, where $k_{j,1}, k_{j,2}, k_{j,3}$, $j = 1, \dots, m$, that represent the vertices of the triangle bounding the point. Setting $\theta = (\rho, \xi)$ and defining $\mathbf{d}(\theta)$ as the objective function to minimize, during every update step we seek the minimizer

$$\theta^0 \ = \ \text{argmin}_\theta \ \frac{1}{2} \parallel \mathbf{d}(\theta) \parallel \tag{12}$$

which can be approximated up to second-order efficiently following [14]:

$$\mathbf{d}(\theta) \simeq \mathbf{d}(\mathbf{0}) + \frac{1}{2}\left(\mathbf{J}(\theta) + \mathbf{J}(\mathbf{0})\right)\theta \tag{13}$$

where $\mathbf{J}(\theta)$ is the Jacobian of $\mathbf{d}(\theta)$ computed in θ. Under certain conditions, $\mathbf{J}(\theta)\theta$ can be calculated without knowing the value of θ [14]. Setting $\mathbf{d}(\theta) = 0$, we can compute θ^0 as:

$$\theta^0 = -\left(\left(\frac{\mathbf{J}_{I_t} + \mathbf{J}_{I_{t+1}}}{2}\right)\breve{\mathbf{J}}(\mathbf{0})\right)^{\dagger} = -\mathring{\mathbf{J}}^{\dagger} \qquad (14)$$

where $\dagger$ denotes the pseudoinverse, $\mathbf{J}_{I_t}$ and $\mathbf{J}_{I_{t+1}}$ are the Jacobians computed in the images and $\breve{\mathbf{J}}(\mathbf{0})$ is the Jacobian of the sequence of transformations defined in Eq. 9, computed in $\mathbf{0}$ using the chain rule. The Jacobian $\mathbf{J}_{I_t}$ can be calculated only once for every previous image.
The standard optimization procedure is as follow:

1) Initialize Eq. 10, in our case we initialize the system with $\widehat{\rho} = 0$ and $\widehat{\zeta}_{\mathbf{i}} = [0, \ldots, 0]'$.
2) Calculate the Jacobians $\mathbf{J}_{I_t}$, $\mathbf{J}_{I_{t+1}}$ and $\breve{\mathbf{J}}(\mathbf{0})$ and find θ^0 by solving Eq. 14.
3) Update $\widehat{\rho}$ and $\widehat{\zeta}_{\mathbf{i}}$ as shown in Eq. 7 and Eq. 8.
4) End when the increment is smaller than a threshold, otherwise go to 2).

E. Implementation

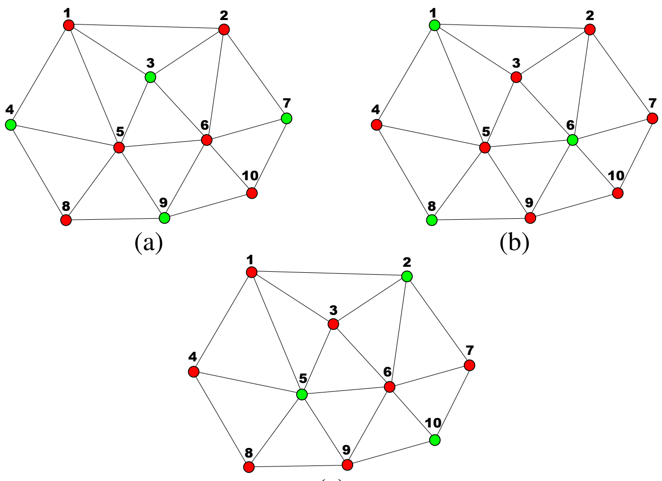


Fig. 5. Example of three iteration of the proposed optimization technique. The topology of the Delaunay triangulation is exploited to iteratively divide the optimization into smaller subproblems: in the iteration defined by (a) only vertices 3,4,7,9 are taken into account in the optimization; in (b) only vertices 1,6,8 and in (c) only 2,5,10. In three consecutive iterations, all the 10 vertices are chosen at least once.

In our system we do not solve the full optimization of Sec. III-D in one go. Instead, we iteratively solve for each triangular surface, enabling a more efficient computation of the pseudoinverse defined in Eq. 14. For each iteration, we only consider a set of "independent" vertices, keeping the rest of the structure fixed. The term "independent" refers to the fact that these vertices are directly connected with vertices that are kept fixed in that iteration. An example of this procedure is shown in Fig. 5. Given a Delaunay triangulation, in the first iteration (Fig. 5(a)) we only update vertices 3,4,7,9 (the "active" vertices) while keeping the others fixed ("inactive"). In the second and third iterations (Fig. 5(b),(c)), the set of "active" vertices changes. After 3 iterations, all 10 vertices have been active at least once.
In order to effectively select vertices during the iterations, we sort the vertices in a priority queue; vertices that are not "active" in a given iteration have their priority upped for the next iterations.
The complexity of each iteration (i.e., the number of parameters) is on average $1/4$ of the original size of the full problem, which causes a considerable speedup since the calculation of the pseudoinverse of Eq. 14 has at least quadratic complexity.

A results of the proposed tracking procedure in a omnidirectional image is shown in Fig. 1, where Delaunay subdivision is depicted with blue segments and tracked vertices are depicted with red segments.

1) Efficient Pseudoinverse Computation: From Eq. 14, we need to compute the usual Moore-Penrose pseudoinverse :

$$\mathring{\mathbf{J}}^{\dagger} = (\mathring{\mathbf{J}}^{\mathsf{T}} \cdot \mathring{\mathbf{J}})^{-1} \cdot \mathring{\mathbf{J}}^{\mathsf{T}} \qquad (15)$$

where $\mathring{\mathbf{J}} \in \mathbb{R}^{(n+m)\times(n+1)}$, with n the number of vertices and m the number of samples, is the Jacobian of the whole system. In our implementation, the Jacobian $\mathring{\mathbf{J}}$ with respect to the "active" set of parameters is smaller, i.e. $\mathring{\mathbf{J}} \in \mathbb{R}^{(\bar{n}+m)\times(\bar{n}+1)}$, with $\bar{n}$ the number of "active" vertices. Usually $n/\bar{n} \approx 4$. Within each iteration, only sample points within an active element are updated, so the Jacobian is sparse, as only two elements per row of $\mathring{\mathbf{J}}$ are non-zero. Thus $\mathring{\mathbf{J}}^{\mathsf{T}} \cdot \mathring{\mathbf{J}}$ can be computed in linear time. This results in a diagonal matrix with the elements of the first row and first column not equal to zero. Using Gauss-Jordan elimination with pivoting in order to improve the numerical stability, one can compute the inverse of this matrix in $O(n^2)$. The last matrix multiplication in Eq. 15, thanks to the sparse nature of the Jacobian $\mathring{\mathbf{J}}$, also has quadratic complexity.

2) Coarse-to-fine Optimization Strategy: In order to cope with large displacements between consecutive frames, we adopt a multi-scale strategy running the optimization in an l-levels Gaussian pyramid representation of the previous and the current images, and using the estimated parameters as initial guess for the lower level.

As we have already pointed out, we do not establish correspondence across more than 2 frames, as this would significantly increase computational complexity, although of course this would be beneficial in terms of increasing the effective baseline and therefore the robustness and accuracy of the motion estimates, and therefore of the structure estimates. The benefit of an instantaneous representation is that it results in an optimization that is easily scalable and can be implemented in real-time.

IV. EXPERIMENTAL RESULTS

We tested our system using a car equipped with an omnidirectional camera installed on the vehicle's roof (Fig. 6). We collect data moving at 30-40 Km/h in a 1300 meters loop inside a challenging urban scenario (Fig. 7 (a)). The omnidirectional camera is composed by an 1032×778 Firewire-b camera and a hyperbolic mirror, and it was calibrated using the OCamCalib Matlab toolbox [18]. We collected a sequence of 2000 images with related timestamps at a frame

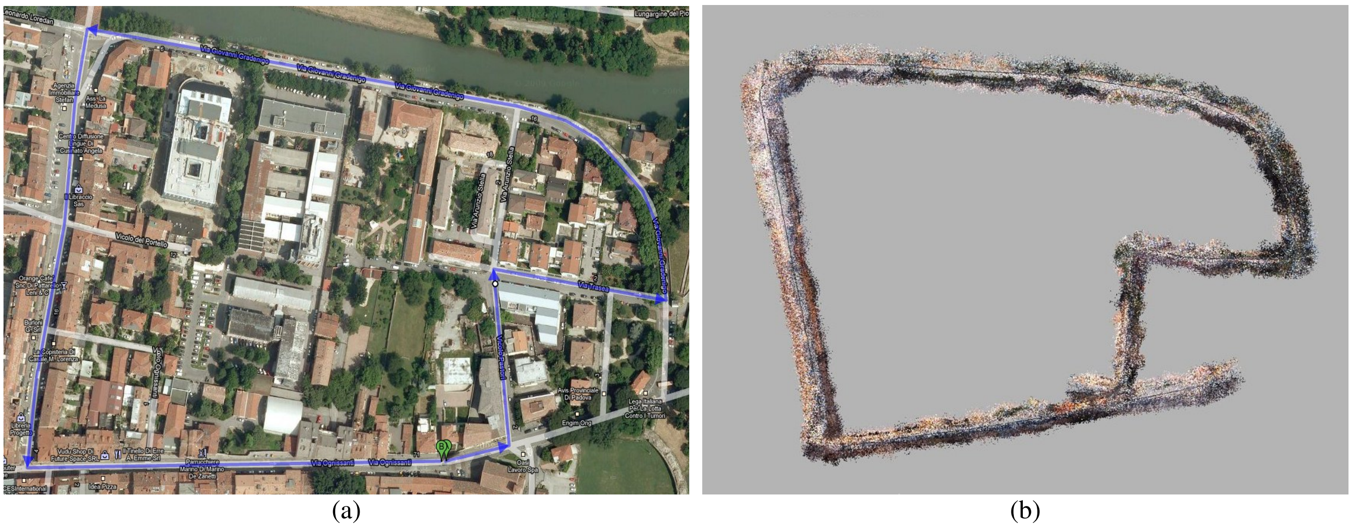


Fig. 7. In (a) a satellite image of the urban segment where the experiments were performed [source: Google Maps]. A light blue curve shows the 1300-meters path. In (b) the estimated path is shown along with the estimated 3D positions of the mesh's vertices.

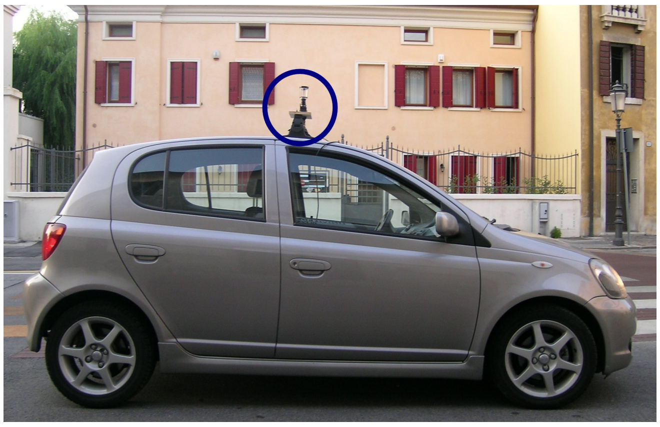


Fig. 6. The vehicle used in the experiments equipped with the omnidirectional installed on the roof (in the blue circle).

rate of 10 Hz; moreover, we collected also the vehicle speed, using the OBD-II (*On-Board Diagnostics*) interface, with a 1 Hz frequency.

A. Ego-Motion Estimation

We infer the global car ego-motion composing the single rigid-body motions estimated between every two consecutive frames with the proposed tracking framework, and recover the scale factor using the vehicle's speed. In Fig. 7 (b) we show the estimated ego-motion for the followed path (depicted in light blue in the satellite image of Fig. 7 (a)). In Fig. 7 (b) we also show the estimated 3D positions of the mesh's vertices.
The trajectory is well estimated with a moderate drift, despite the dataset proposed many challenging situations, among which:

- Very low frame rate (a frame rate of 10 Hz implies a displacement greater than 1 meter between consecutive frames at 40 Km/h);
- Large photometric variability due to light-shadow transitions;
- Large number of images with very sparse textures (blank walls, open spaces in the upper part of Fig. 7 (a)).

Moreover, part of the overall error in the motion estimation is due to inaccuracies in the vehicle's speed: we verify for example that, after arresting the car, the speed readings from the OBD-II takes a few seconds to converge to a reading of 0.

B. Dense 3D Scene Reconstruction

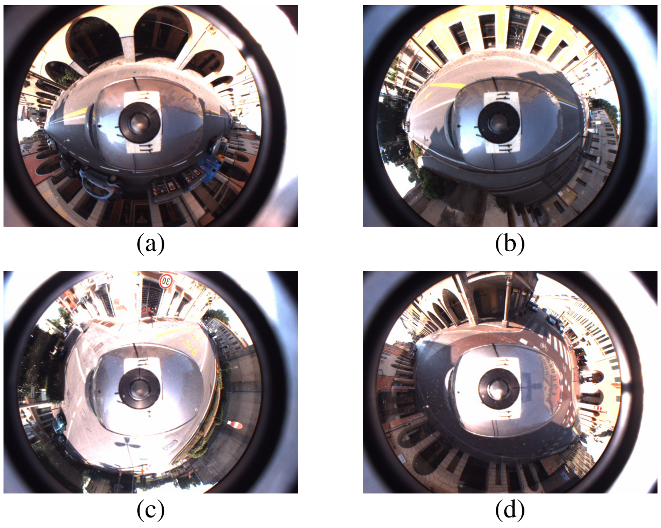
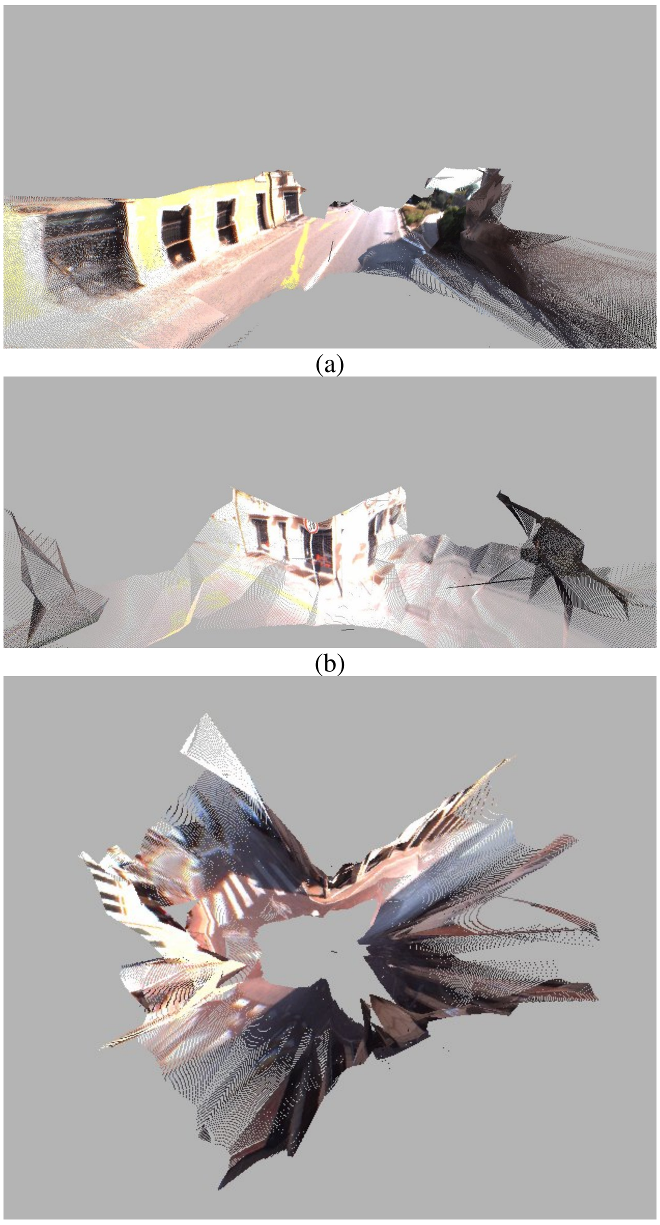


Fig. 8. Some images of our dataset. Dense 3D reconstruction results for these images are reported in Fig. 2 and Fig 9.

We obtain satisfactory results in the dense 3D reconstruction task: estimated surfaces are usually close to the real ones (e.g., Fig. 2) also in case of sharp 3D features as corners of


(a)

(b)

(c)

Fig. 9. Some results of the dense 3D reconstruction of the scene obtained with the proposed method. In (a) and (b) the first-person views of the reconstructions (the previous images used for these reconstruction are shown in Fig. 8 (b) and (c), respectively); in (c) a bird's eye view of a reconstruction (previous image is shown in Fig. 8(d)).

buildings (e.g., in Fig. 9(b) a corner of a building, in Fig. 9(c) a bird's eye view of a crossroads). Previous images for these reconstruction are shown in Fig. 8.

While the reconstruction is not of photographic rendering quality, we emphasize that our goal is to use this reconstruction in autonomous robots for motion planning, navigation and obstacle-avoidance purposes, for which high computational speed and low computational latency are more important that the overall visual quality of the reconstruction.

C. Efficiency Considerations

The whole approach (features and sample point extraction, Delaunay subdivision and optimization procedure) take 2 seconds for each frame on a 2 GHz Core 2 Linux-based Laptop PC. We use on average 500 vertices and 6500 sample points for every frame. Unfortunately, there is a severe overhead on the optimization procedure due to the low frame rate: the large displacement between consecutive frames (usually, around 1 meter) imposes to use 4 levels in the coarse-to-fine optimization strategy (see Sec. III-E.2). Moreover, every level required up to 50 iterations to correctly converge. In further experiments, we verified that using an higher frame rate (say 30 Hz), the optimization correctly converges using only 2 levels and around 20 iterations. Providing the optimization procedure with a CUDA implementation [3], the system can be easily run in real time on a standard PC.

V. CONCLUSIONS AND FUTURE WORKS

In this paper, we tackle Visual SLAM by estimating a dense approximation of the instantaneous range map. We estimate both the motion of the camera and the shape of the scene, encoded by the position of the vertices of a mesh, by minimizing the reprojection error, that uses the entirety of the image, not just the local regions around salient structures. We enable real-time operation by scheduling the choice of active nodes at each step of an iterative optimization. Temporal consistency is only enforced by using the best previous estimate as initialization for the current optimization, but otherwise the entire structure of the environment previously estimated is not used in the calculation of the current optimization. This has obvious shortcomings in terms of accuracy and robustness, but it enables us to maintain the topology of the reconstructed surface simple (by discarding occlusions) and therefore makes our approach suitable for real-time operation.

We plan to improve our system enabling surface tracking over a longer sequence of frames and integrating the proposed technique with a graph-based SLAM framework in order to achieve loop-closure detection and global optimization of the estimated motion and structure. We also plan to implement the optimization procedure using CUDA libraries. Moreover, other future works include the generalization of the piecewise smooth surface assumption, enabling the system to track multiple surfaces selected on the basis of the occlusions detected in the images.

REFERENCES

[1] Henrik Andreasson, Tom Duckett, and Achim Lilienthal. Mini-slam: Minimalistic visual slam in large-scale environments based on a new interpretation of image similarity. In *Proc. of the 2007 IEEE International Conference on Robotics and Automation (ICRA)*, 2007.

[2] N. Cornelis, B. Leibe, K. Cornelis, and L. Van Gool. 3d urban scene modeling integrating recognition and reconstruction. *International Journal of Computer Vision*, 78:121 – 141, July 2008.

[3] Nvidia Corporation. Nvidia cuda library home page. http://www.nvidia.com/object/cuda_home_new.html.

[4] Andrew J. Davison, Ian D. Reid, Nicholas D. Molton, and Olivier Stasse. Monoslam: Real-time single camera slam. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(6):1–16, 2007.

[5] E. Eade and T. Drummond. Scalable monocular slam. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 1, pages 469–476, June 2006.

[6] E. D. Eade and T. W. Drummond. Edge landmarks in monocular slam. In *British Machine Vision Conference (BMVC)*, volume 1, pages 469–476, 2006.

[7] R. Elinas, P. Sim and J. Little. slam: Stereo vision slam using the rao-blackwellised particle filter and a novel mixture proposal distribution. In *Proc. of the 2006 IEEE International Conference on Robotics and Automation (ICRA)*, 2006.

[8] Klein G. and D. Murray. Improving the agility of keyframe-based slam. In *In Proc. European Conference on Computer Vision (ECCV, Marseille)*, 2008.

[9] G. Grisetti, C. Stachniss, S. Grzonka, and W. Burgard. A tree parameterization for efficiently computing maximum likelihood maps using gradient descent. In *Robotics: Science and Systems (RSS)*, Atlanta, GA, USA, 2007.

[10] H. Jin, P. Favaro, and S. Soatto. A semi-direct approach to structure from motion. *The Visual Computer*, 19:1–18, 2003.

[11] Il-Kyun Jung and Simon Lacroix. High resolution terrain mapping using low altitude aerial stereo imagery. In *ICCV '03: Proceedings of the Ninth IEEE International Conference on Computer Vision*, page 946, Washington, DC, USA, 2003. IEEE Computer Society.

[12] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[13] Y. Ma, S. Soatto, J. Kosecka, and S. Sastry. *An Invitation to 3D Vision*. Springer Verlag, 2004.

[14] Ezio Malis. Improving vision-based control using efficient second-order minimization techniques. In *IEEE International Conference on Robotics and Automation*, New Orleans, USA, April 2004.

[15] B. Micusik and J. Kosecka. Piecewise planar city modeling from street view panoramic sequences. In *IEEE conference on Computer Vision and Pattern Recognition*, 2009.

[16] Michael Montemerlo and Sebastian Thrun. *FastSLAM: A Scalable Method for the Simultaneous Localization and Mapping Problem in Robotics*. Springer-Verlag Berlin and Heidelberg GmbH & Co. K, January 2007.

[17] D. Nistér, O. Naroditsky, and J. Bergen. Visual odometry. In *Proc. of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 2004.

[18] D. Scaramuzza. Omnidirectional vision: from calibration to robot motion estimation. PhD Thesis, 2008.

[19] D. Scaramuzza and R. Siegwart. Appearance-guided monocular omnidirectional visual odometry for outdoor ground vehicles. *IEEE Transactions on Robotics*, 28 (2), October 2008.

[20] J. Shi and C. Tomasi. Good features to track. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 593–600, 1994.

[21] G. Silveira, E. Malis, and P. Rives. An efficient direct method for improving visual SLAM. In *Proc. of the 2007 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4090–4095, Italy, 2007.

[22] S. Soatto. Towards a mathematical theory of visual information. In *in preparation*, 2010.

[23] S. Soatto, A. J. Yezzi, and H. Jin. Tales of shape and radiance in multiview stereo. In *Intl. Conf. on Comp. Vision*, pages 974–981, October 2003.

[24] G. Sundaramoorthi, P. Petersen, V. S. Varadarajan, and S. Soatto. On the set of images modulo viewpoint and contrast changes. June 2009.

[25] A. Vedaldi, G. Guidi, and S. Soatto. Joint data alignment up to (lossy) transformations. June 2008.