2011 IEEE International Conference on Robotics and Automation
Shanghai International Conference Center
May 9-13, 2011, Shanghai, China

Omnidirectional Dense Large-Scale Mapping and Navigation Based on Meaningful Triangulation

Alberto Pretto, Emanuele Menegatti and Enrico Pagello

Abstract— **In this work, we propose a robust and efficient method to build dense 3D maps, using only the images grabbed by an omnidirectional camera. The map contains exhaustive information about both the structure and the appearance of the environment and it is well suited also for large scale environments.**
We start from the assumption that the surrounding environment (the scene) forms a piecewise smooth surface represented by a triangle mesh. Our system is able to infer, without any odometry information, the structure of the environment along with the ego-motion of the camera by performing a robust tracking of the projection of this surface in the omnidirectional image. The key idea is to use a guess of the triangle mesh subdivision based on a constrained Delaunay triangulation built according to a set of point features and edgelet features extracted from the image. In such a way, we take into account both the corners and the edges of the scene imaged by the camera, constrained by the topology of the triangulation in order to improve the stability of the tracking process. Both motion and structure parameters are estimated using a direct method inside an optimization framework, taking into account the topology of the subdivision in a robust and efficient way.
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

A consistent estimation of the 3D structure of the environment with a reliable localization within the built map is a fundamental requirement for providing a mobile robot with autonomous navigation capabilities. Using vision to achieve this task offers the benefit of providing plenty of information (geometric and photometric) relevant to analyze the perceived scene. This task is commonly referred as the *Visual SLAM* (*Simultaneous Localization and Mapping*) problem, and it has received a great attention within the robotics and vision communities.
Currently, most of the Visual SLAM approaches are based on points features (among others, [8], [2], [3]): features are detected and tracked between consecutive frames, the motion of the camera and the 3D locations of the tracked points are hence estimated concurrently using mainly probabilistic estimation techniques. Point features tracking techniques are indeed powerful tools for estimating the ego-motion of a moving robot using only vision. Despite that, a map of 3D points does not provide much useful information about the environment: usually, mapped points are sparse and provide the robot with incomplete information about the structure and the appearance of the environment.

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

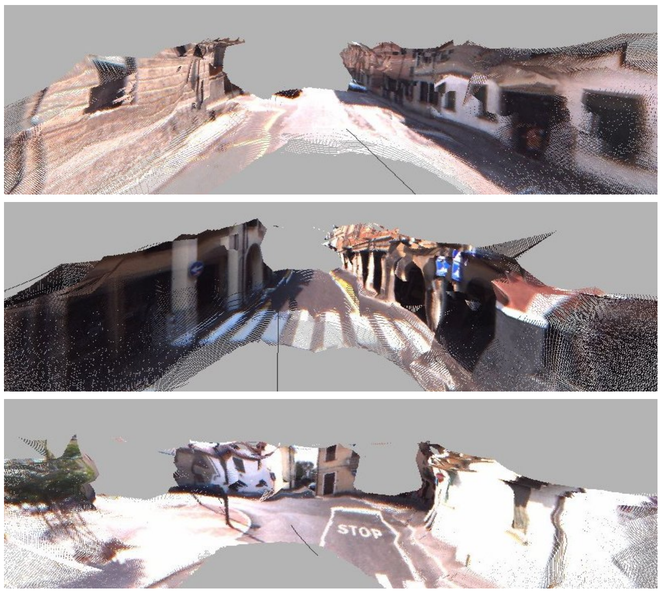


Fig. 1. Some results of the dense 3D reconstructions of the scene obtained with the proposed method. (the reference images used for these reconstructions are shown in Fig. 7).

In this work we describe an efficient method that uses only images to estimate the *dense* 3D structure of the environment along with the pose of the camera, by exploiting a simple but powerful assumption: all the objects that compose the scene can be represented with a set of 3D piecewise smooth surfaces projected on the image plane. We enforce this assumption by considering the surrounding scene as represented by a single surface. A natural approximation of a piecewise smooth surfaces is given by a triangle mesh: the perspective projection of the triangular facets generates a subdivision of the image into triangles. Since we don't know the 3D structure of the environment, and hence its approximation with a triangle mesh, we propose to generate a hypothesis of this subdivision given by the constrained Delaunay triangulation of a set of point features and edgelet features. In a previous work [15], we used only point features: the main problem with that approach was the correct representation of buildings' edges when the Delaunay triangulation created a 2D triangle over a 3D edge. The current approach instead modifies the triangulation by adding constrained edges where edgelet features are detected.
Taking into account the subdivision's topology, we compute simultaneously the 3D displacement of the camera and the dense scene structure (i.e., the depths of 3D points that are

projected onto 2D vertices of the triangulation) as an optimization problem where 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. The depths of all points that lie in a facet can be therefore computed in a closed form given the depths of the three facet's vertices. This enable us to use in the optimization a large number of image points (possibly all), where the number of parameters depends only on the chosen subdivision (i.e., the number of vertices).

Our optimization approach is based on an efficient second-order minimization (*ESM*) procedure [12]: the topological relationships between the subdivision's vertices are exploited to iteratively divide the optimization into subproblems where only a subset of "independent" vertices are taken into account. This strategy improves noticeably the efficiency of the minimization procedure.

We implemented our system using an omnidirectional camera: with a complete view of the surroundings in one shot, it is possible to obtain a complete reconstruction of the environment by only tracking between two consecutive frames.

The main contributions of this paper are:

- A proper way to deal with the dense 3D reconstruction problem based on a 'guess' of the scene's surface projection.
- A robust and efficient iterative procedure that exploits the topology of the subdivision to infer simultaneously the motion and the structure of the environment.

A. Related Works

Jin *et al.* [8] and Davison *et al.* [2] proposed a feature-based SLAM approach using a single perspective camera and *EKF* (Extended Kalman Filter), where a 3D map of the features is built using the bearing only information provided by the camera. A similar approach, but based on the FastSLAM framework, was presented in [3]. In [9] a high resolution digital elevation maps is built from a sequence of stereovision image pairs where interest points are detected and matched between consecutive frames. A visual motion estimation algorithm is used to predict the movements, an extended Kalman filter is used to estimate both the position parameters and the map. In [5] a dense metric map of 3D point landmarks for large cyclic environments is built using the Rao-Blackwellised Particle Filter, where SIFT features are extracted from stereo vision and motion estimates are based on sparse optical flow. Eade and Drummond [4] present a monocular visual SLAM approach using line features, where an efficient algorithm for selecting such landmarks is defined. Higher level landmarks are exploited in [20], where 3D camera displacement and the scene structure are computed directly from image intensity discrepancies using an efficient second-order optimization procedure for tracking planar patches. Nistér *et al.* [14] 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 a large trajectory. A visual-odometry approach using omnidirectional vision is presented in [18], 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. In [22] is presented an omnidirectional vision based SLAM that exploits epipolar constraints in order to estimate the camera rotation, while camera position is recovered using the computed 3D structure. Recently, Klein and Murray [6] proposed an effective Visual SLAM approach based on edgelets tracking that exploits a key-frame-based re-localization method in order to recover from tracking failures. In [13], 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 poses exploiting the epipolar geometry constraints, while a dense 3D map of the environment is built using a superpixel-based multi-view stereo method. Cornelis *et al.* [1] integrate in a real-time 3D reconstruction framework an object recognition module in order to deals with the dynamic nature of an urban scenario. In a very recent work, Lhuillier [10] proposed a new error model for central cameras, exploited to build 3D dense maps of the environment using a high resolution catadioptric photo camera. Similarly to our approach, Lhuillier builds a constrained Delaunay subdivision to represent the triangular mesh.

II. SURFACE REPRESENTATION BASED ON MEANINGFUL TRIANGULATION

Modeling objects with a set of piecewise smooth surfaces generally represents a good and often necessary approximation in order to deal with dense 3D reconstruction of the scene. One of the common way to describe a smooth surface in a discrete fashion is to replace it with a triangle mesh: increasing the number of the triangles used in the representation, it is possible to improve the quality of the approximation. In our system, we model the surrounding environment as a single, piecewise smooth surface approximated with a triangle mesh.

Given a piecewise smooth surface S that represents the scene, we can approximate it with a triangle mesh $M(S)$, that is 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 $\mathbf{V} = \{\mathbf{V}_1, \dots, \mathbf{V}_n\}$ that defines the shape in $\mathbb{R}^3$. This triangle mesh $M(S)$ represent the dense environment's structure that we are looking for.

Assuming that the surface can be completely imaged by the camera (hence, it is implicitly assumed that there is no occlusions in the scene), every 3D vertex $\mathbf{V}_i$ is projected on the image plane as a 2D point $\mathbf{v}_i = \mathring{\Pi}(\mathbf{V}_i)$, $i = \{1, \dots, n\}$, with $\mathring{\Pi}$ a general projection function. Let us denote the 3D vertex $\mathbf{V}_i$ to have as neighbors the 3D vertices defined by the indexes $\{j_{i,1}, \dots, j_{i,n_i}\}$: under the given assumptions, the topology of the projected vertices remains the same, i.e. $\mathbf{v}_i = \mathring{\Pi}(\mathbf{V}_i)$ has neighbors that are the projections of the 3D vertices defined by the same indexes $\{j_{i,1}, \ldots, j_{i,n_i}\}$. Moreover, 3D points that lie in a facet defined by 3 vertices, will be projected within a triangle defined by the 2D projections of these 3 vertices.

The idea we exploit in our system is to guess a "good"

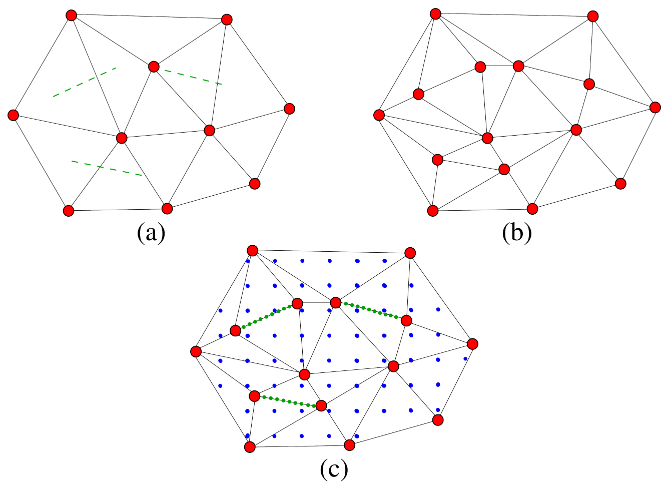


Fig. 2. An example of triangulation. (a) Delaunay triangulation obtained using only point features (the dashed green lines represent the selected edgelet features); (b) The constrained Delaunay triangulation obtained by adding the edgelets; (c) The points used in the optimization process: red circles represents the subdivision's vertices, green circles the edge pixels (*edgels*) and blue circles the inner pixels.

approximation of the projection of the triangle mesh $(\mathbf{K}, \mathbf{V})$ in the current image, and to recover the 3D structure by tracking the projection's deformation between frames.

A critical issue when representing a surface with a triangle mesh is to model sharp features as corners or sharp curves: in order to obtain a better approximation, some vertices should lie close to the 3D location of the corners, and some edges of the triangulation should lie close to the location of the sharp curves. The projections on the image plane of these vertices and edges should hence lie close to the projections of the sharp features. Looking through the image for corner features and assigning them as vertices projection is a natural and reasonable choice. We select n' vertices $\{\mathbf{v}_1, \ldots, \mathbf{v}_{n'}\}$ (for clarity, we will omit the term projection) using the Shi and Tomasi corner detector [19]. In order to obtain an uniform distribution of the vertices, we partition the images into regular tiles looking for a certain number of corner features within each tile.

Given the set $\{\mathbf{v}_1, \ldots, \mathbf{v}_{n'}\}$, we need therefore to define a subdivision into simplices (triangles in the planar case) that represents the image projections of the 3D facets. We use the Delaunay triangulation technique, which provides a subdivision that maximizes the minimum angle between any two edges in the resulting graph. This avoids "skinny triangles" artifacts. In Fig. 2 (a) is shown an example of Delaunay triangulation, where the vertices are depicted with red circles.

In the next stage the triangulation is modified by adding a set of constrained edges in order to deal with sharp 3D features. We first extract a set of edgelet features (short segments of lines in the omnidirectional image) performing a modified Canny edge extraction that breaks edges at points of high curvature, and splitting long edges into shorter segments, similarly to [6]. We partition the images into regular tiles looking for a certain number of segments within each tile, avoiding crosses between edgelets. For each edgelet (dashed green lines in Fig. 2 (a)) we add to the triangulation two new vertices representing the end points of the segment (Fig. 2 (b)). The resulting tessellation is a constrained Delaunay triangulation, that preserves as possible the "Delaunay structure" of the triangulation [1] (for details, see [7]). We define as $n = n' + n''$ the total number of vertices in the triangulation, where n'' is the number of added edgelet's end points. In Fig 3 an omnidirectional image with the extracted point features and edgelet features (red segments) along with the built constrained Delaunay triangulation. We

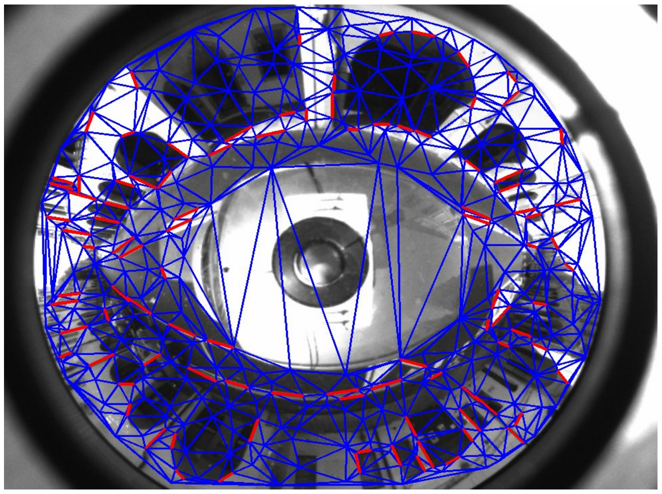


Fig. 3. An example of an omnidirectional image depicting the proposed constrained Delaunay subdivision's. Blue segments represents the links between vertices (point features), red segments are the detected edgelets added to the triangulation.We use an image mask in order to avoid searching for features in the car's roof or outside the area where the omnidirectional mirror is imaged.

build the subdivision directly in the omnidirectional image: this enables us to avoid the discontinuities introduced from unwarping the omnidirectional image into a 360-*degrees* panoramic image, at a cost of a less accurate triangles' approximation (i.e., horizontal lines turns into curves in the omndirectional image).

We finally extract a set of m images sample points $\{\mathbf{s}_1, \ldots, \mathbf{s}_m\}$ (or *inner points*) that lie inside the triangles (blue circles in Fig. 2 (c)), and a set of h *edgels* $\{\mathbf{e}_1, \ldots, \mathbf{e}_h\}$ (pixels that compose the edgelets, green circles in Fig. 2 (c)), along with the topological information related to the Delaunay subdivision (e.g., for the sample points, the 3 indexes $k_{i,1}, k_{i,2}, k_{i,3}$, $i = 1, \ldots, m$, which define the vertices of the triangle that the samples belong to). Sample points and

[1]We build the constrained Delaunay triangulation using the implementation provided by the Triangulation Template Library (TTL) http://www.simula.no/ogl/ttl/ttl_doc/html/index.html

edgels will be used in the optimization procedure in order to match the deformation of the triangular patches between consecutive frames.

III. THE SURFACE TRACKING FRAMEWORK

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 3D layout of the scene.

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 $\mathcal{I}^* : \mathbb{R}^2 \to \mathbb{R};\ \mathbf{x} \mapsto \mathcal{I}^*(\mathbf{x})$ the *reference* image, and similarly $\mathcal{I}(\mathbf{x})$ the *current* image. Their relation, assuming Lambertian reflection and constant illumination [21], is given by

$$\mathcal{I}(w(\mathbf{x})) = \mathcal{I}^*(\mathbf{x}), \quad \mathbf{x} = \pi(\mathbf{X}), \quad \mathbf{X} \in S \subset \mathbb{R}^3 \tag{1}$$

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 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 $g_t \in \mathbb{SE}(3)$, the instantaneous motion of the camera. The reprojection error is simply

$$\phi(g_t, S) = \int_D \left(\mathcal{I}(w(\mathbf{x})) - \mathcal{I}(\mathbf{x})\right)^2 d\mathbf{x} \quad \text{subject to (2)} \tag{3}$$

for the case of the L^2 norm, where D is the domain of the image. 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 described above is usually taken to be a canonical (central) perspective projection, either onto a plane, or onto a 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 [16].

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 $g \in \mathbb{SE}(3)$, represented in homogeneous coordinates as a 4×4 matrix $\mathbf{G}$ as customary [11], with rotation component $\mathbf{R} \in \mathbb{SO}(3)$ 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

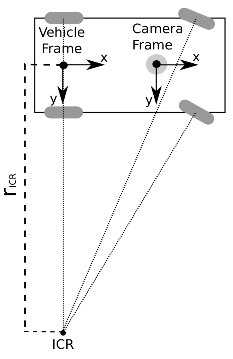


Fig. 4. Ackermann steering geometry.

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, as done in [17]. In this model, at every instant, all of wheels' zero motion lines meet at a single point, called *Instantaneous Center of Rotation* (ICR) (Fig. 4). We can approximate this model by assuming that the vehicle rotates around a fixed ICR for a discrete time (the frame period 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 of the car's reference frame as r_{ICR} (the *radius* of curvature, see Fig. 4). Since with a bearing only sensor such as an omnidirectional camera the scale factor is unknown, we can fix the displacement between two poses, in our case we set $\|\mathbf{T}\| = 1$. In this way, it is possible to parametrize the rigid-body motion with only one parameter, the radius r_{ICR}. We can therefore define $\mathbf{G}\langle r_{ICR}\rangle \in \mathbb{SE}(3)$ as:

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

where $\alpha = 2 \cdot asin\left(\frac{1}{2 \cdot r_{ICR}}\right)$. We recover the scale factor using the vehicle's speed readings along with the frame's timestamps.

Let us finally define the rigid-body transformation $\mathbf{B}_{VC} \in \mathbb{SE}(3)$ that represents the coordinate transformation from the camera frame to the vehicle frame (see Fig. 4).

C. Tracker Parameterization

As we have described, in our system the camera motion is described by the radius r_{ICR}, while the structure is described by the vertices of the Delaunay subdivision along with their depths: the topology of the subdivision enables us to recover the depth of every image point in closed form. Let $\mathbf{s}_i \in \mathbb{S}^2$ 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 by observing that the determinant of the following matrix must be equal to zero:

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

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 by removing the first row and the $i-th$ column. The depths of the edgels $\{\mathbf{e}_1, \dots, \mathbf{e}_h\}$ can be computed in an easier way, since they depend on only two vertices. i.e. the edgelets' end points.

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 us 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 represents 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 [20], 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. Tracking the Surface

Without loss of generality, we can assume the world frame to be the one from which the camera grabbed the reference image $\mathcal{I}^*$, while the current image $\mathcal{I}$ is taken from a different pose. We start to use the notation $\mathbf{p}^*$ to represent a point in the frame of the reference image, and $\mathbf{p}$ for a point in the frame of the current image.

In Sec. II we have described how to extract from the (*reference*) image the set of n vertices $\{\mathbf{v}_1^*, \dots, \mathbf{v}_n^*\}$ along with the topology information of the triangular subdivision that represents a "guess" of the projection of the triangular mesh in the image. We also collect a set of h edgels $\{\mathbf{e}_1, \dots, \mathbf{e}_h\}$ that represent the edgelets' pixels and a set of m points $\{\mathbf{s}_1^*, \dots, \mathbf{s}_m^*\}$ that lie inside the triangles. For each vertex $\mathbf{v}_i^*$ in $\mathcal{I}^*$, the corresponding vertex in $\mathcal{I}$ is given by the following sequence of transformations:

$$\mathbf{v}_i^* \xrightarrow{\Pi^{-1}\ -1-} \mathbf{v}_i^{*\prime} \xrightarrow{\lambda_i^*\ -2-} \mathbf{V}_i^* \xrightarrow{\mathbf{B}_{VC}\ -3-} \mathbf{V}_i^{*,car} \rightarrow \tag{9}$$
$$\xrightarrow{\mathbf{G}\langle r_{ICR}\rangle\ -4-} \mathbf{V}_i^{car} \xrightarrow{\mathbf{B}_{VC}^{-1}\ -5-} \mathbf{V}_i \xrightarrow{\lambda_i^{-1}\ -6-} \mathbf{v}_i^{\prime} \xrightarrow{\Pi\ -7-} \mathbf{v}_i$$

The transformation -1- maps the image point of the vertex to the corresponding normalized image point in the unit sphere; -2- multiplies the normalized image point with the depth in order to obtain the 3D vertex; -3- is the coordinate transformation from the reference camera frame to the vehicle frame; -4- is the rigid-body motion performed by the car (Eq. 4); -5- is the coordinate transformation from the vehicle frame to the current camera frame; -6- represents the 3D vertex in the unit sphere and -7- projects $\mathbf{v}_i^{\prime}$ in the current frame. λ_i^{-1} in -6-is just L2-norm of the 3D vertex $\mathbf{V}_i$. The sequence of transformations is also the same for the edgels $\{\mathbf{e}_1^*, \dots, \mathbf{e}_h^*\}$ and for the sample points $\{\mathbf{s}_1^*, \dots, \mathbf{s}_m^*\}$, where the depth used in the transformation -2- is computed by exploiting the topological information provided by the triangulation (e.g., using Eq. 6 in the case of the sample points). It's important to notice that the projection of a vertex from the reference image to the current image depends on both the motion parameter and its depth; the projection of an edgel depends on the motion and the depths of *two* vertices (the edgelet's end points), while the projection of a sample points depends on the motion and the depths of *three* vertices. The role of the edgels and the sample points is hence very important due to the fact they enforce the rigidity constraint of the scene during the optimization.

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:

$$\phi(\rho, \xi) = \frac{1}{2}\sum_{i=1}^{n} \left[\mathcal{I}\left\{\mathbf{v}_i\right\} - \mathcal{I}^*\left\{\mathbf{v}_i^*\right\}\right]^2 + \tag{10}$$
$$\frac{1}{2}\sum_{i=1}^{m} \left[\mathcal{I}\left\{\mathbf{s}_i\right\} - \mathcal{I}^*\left\{\mathbf{s}_i^*\right\}\right]^2 + \frac{1}{2}\sum_{i=1}^{h} \left[\mathcal{I}\left\{\mathbf{e}_i\right\} - \mathcal{I}^*\left\{\mathbf{e}_i^*\right\}\right]^2$$

where, remembering Eq. 7,8 and 9:

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

The computation of $\mathbf{e}_i$ and $\mathbf{s}_i$ is slightly more complex, due to the fact that their depths depend on two ore three vertices. Setting $\theta = (\rho, \xi)$ and defining $\mathbf{d}(\theta)$ as the cost function to minimize, during every update step we have to find the optimal value such that:

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

It can be shown that an efficient second-order approximation of $\mathbf{d}(\theta)$ is:

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

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 [12]. Setting $\mathbf{d}(\theta) = 0$, we can compute θ^0 as:

$$\theta^0 = -\left(\left(\frac{\mathbf{J}_{\mathcal{I}^*} + \mathbf{J}_{\mathcal{I}}}{2}\right)\breve{\mathbf{J}}(\mathbf{0})\right)^+ = -\mathring{\mathbf{J}}^+ \qquad (14)$$

where $+$ defines the pseudoinverse, $\mathbf{J}_{\mathcal{I}^*}$ and $\mathbf{J}_{\mathcal{I}}$ 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, in a similar way as done in [20]. The Jacobian $\mathbf{J}_{\mathcal{I}^*}$ can be calculated only once for every reference image.

The standard optimization procedure is as follow:

1) Initialize Eq. 10, in our case we initialize the system with $\widehat{\rho} \approx 0$ and $\widehat{\zeta_{\mathbf{i}}} \approx [0, \ldots, 0]'$ (i.e., a pure translation motion and all vertices at infinity).
2) Calculate the Jacobians $\mathbf{J}_{\mathcal{I}^*}$, $\mathbf{J}_{\mathcal{I}}$ and $\breve{\mathbf{J}}(\mathbf{0})$ and find θ^0 solving Eq. 14.
3) Update $\widehat{\rho}$ and $\widehat{\zeta_{\mathbf{i}}}$ as shown in Eq. 7 and Eq. 8.
4) Stop the iteration if the increments becomes lower than a threshold, otherwise restart from point (2)

E. Implementation

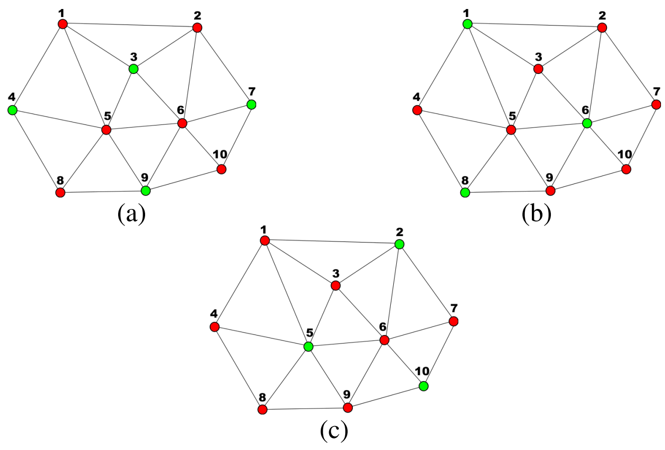


Fig. 5. Example of three iterations of the proposed optimization technique. The topology of the Delaunay triangulation is exploited to iteratively divide the optimization into smaller subproblems: in the iteration defined by (a) only vertices 3,4,7,9 are taken into account in the optimization; in (b) 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 don't cope with the full optimization system defined in Sec. III-D: we propose instead to iteratively solve inside the optimization framework smaller subproblems defined by the surface's topology. This strategy improves noticeably the efficiency of the minimization procedure, enabling moreover an efficient computation of the pseudoinverse defined in Eq. 14. For each iteration of the optimization, instead of updating all the structure parameters, we only take into account and update (using the procedure described above) a subset of independent structure parameters related with a set of "independent" vertices, keeping unchanged the other structure parameters. The term "independent" refers to the fact that these vertices are directly connected with vertices that will not be update in that iteration. An example of this procedure is shown in Fig. 5. Given a (constrained) Delaunay triangulation, in the first iteration (Fig. 5 (a)) we take into account only the vertices 3,4,7,9 (here called "active" vertices): all the vertices directly connected to these vertices (here called "inactive" vertices) are not updated in the iteration. In the second and third iterations (Fig. 5 (b),(c)), the set of "active" vertices changes. After 3 iterations, all the 10 vertices are chosen at least once.

In order to effectively chose all the vertices during the iterations of the minimization procedure, we order the vertices in a priority queue used along with the triangulation topology to chose the subset of "active" vertices. Vertices that are not switched to the "active" state for some iterations, obtains a higher priority (i.e., this increases their probability to be chosen in the next iteration).

The sizes of the optimization subproblems (i.e., the number of parameters) are on average $1/4$ of the original size of the full problem: this is a remarkable result, since a direct computation of the pseudoinverse of Eq. 14 has in the general case cubic complexity on the number of parameters used.

1) Efficient Pseudoinverse Computation: We compute the pseudoinverse by exploiting the sparsity of the Jacobian, enabling an efficient $O(\bar{n}^2)$ computation, with $\bar{n}$ the number of "active" vertices taken into account in the iteration (as said before, usually $n/\bar{n} \approx 4$). From Eq. 14, in the general case we need to compute the following Moore-Penrose pseudoinverse :

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

where $\mathring{\mathbf{J}} \in \mathbb{R}^{(n+m+h)\times(n+1)}$ (n the number of vertices, h the number of edgels and m the number of samples) is the Jacobian of the whole system. With the proposed strategy, the Jacobian $\mathring{\mathbf{J}}$ becomes smaller, i.e. $\mathring{\mathbf{J}} \in \mathbb{R}^{(\bar{n}+m+h)\times(\bar{n}+1)}$. Moreover inside such a subproblems, edgels and samples points depend on the motion parameter and *only one* vertex. This increase the sparsity of the Jacobian: for every row of $\mathring{\mathbf{J}}$, only two elements are not equal to zero. It's easy to compute $\mathring{\mathbf{J}}^\mathsf{T} \cdot \mathring{\mathbf{J}}$ in linear time: this results in a matrix with only the elements of the first row, the first column and the diagonal not equal to zero. Using the Gauss-Jordan elimination method with pivoting in order to improve the numerical stability, enables us to compute the inverse of this matrix in $O(\bar{n}^2)$. The last matrix multiplication in Eq. 15, thanks to the sparse nature of the Jacobian $\mathring{\mathbf{J}}$, takes quadratic time as well.

2) Coarse-to-fine Optimization Strategy: In order to cope with large displacements between consecutive frames, we adopt a multi-scale strategy. We run the optimization proposed above starting from the top of a l-levels Gaussian pyramid representation of images, and using the estimated parameters as initial guess for the lower level (and lower scale) images inside the pyramid.

In this implementation, we track the surface only between two consecutive frames: we plan to improve our approach by enabling surface tracking over a longer sequence of frames. It is important to note that our system is easily scalable by just increasing the number of vertices used in the surface representation in order to improve the quality of the reconstruction, and increasing the number of sample points and edgels in order to improve the stability and the convergence rate of the optimization process.

IV. EXPERIMENTAL RESULTS

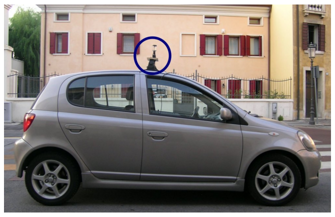


Fig. 6. The vehicle used in the experiments equipped with the omnidirectional camera installed on the roof (in the blue circle).

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 30-45 Km/h in an 1400 meters loop inside a challenging urban scenario (Fig. 8 (a)). The path is a real loop in which the starting point (i.e. the first image of the sequence) is the same as the end point (i.e. the last frame of the sequence). As you can see in the bottom right corner of Fig. 8 (a), Google's **B** placeholder is overlapped onto the **A** placeholder. 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 [16]. We collected a sequence of 1800 images with related timestamps at a frame rate of 10 Hz, moreover we collected also the vehicle speed, using the OBD-II (*On-Board Diagnostics*) interface, with an 1 Hz frequency. The dataset gives rise to many challenges, including:

- Low frame rate (a frame rate of 10 Hz implies a displacement greater than 1 meter between consecutive frames at 40 Km/h);
- Great number of images with very low brightness, due to the presence of many buildings and trees closed to the street;
- Great number of images with a side that contains very sparse features.

A. Ego-Motion Estimation

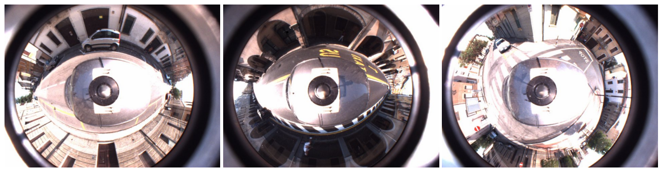


Fig. 7. Some images of our dataset. Dense 3D reconstruction results for these images are reported in Fig. 1.

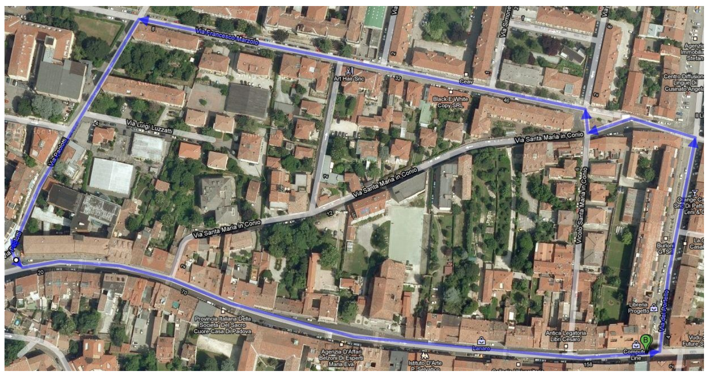
We infer the global car ego-motion composing the single rigid-body motions estimated between every two consecutive frames with the proposed tracking framework, and recover the scale factor using the vehicle's speed. In Fig. 8 (b) we show the estimated ego-motion for the followed loop (the corresponding ground-truth is depicted in light blue in the satellite image of Fig. 8 (a)). As one can see, the system is not using any loop closure detection and correction system, so the starting point is not precisely overlapped with the end point, but the accumulated drift in 1.400 meters is less then 40 meters, which is a great result if you consider this is obtained by vision only without any odometry correction. In Fig. 8 (b), we also show the cloud of the estimated 3D positions in the real world of the Delaunay triangulation's vertices with their correct RGB values.
Again, we wish to stress that the trajectory is well estimated, almost reaching the correct solution of reconstructing a close loop, despite not using any global optimization or loop-closure detection technique. Moreover, part of the overall error in the motion estimation is due to inaccuracies in the vehicle's speed. In fact, we verified that the OBD system is very imprecise for example when after stopping the car, the speed readings coming from the OBD-II interface take a few seconds to converge to 0.

B. Dense 3D Scene Reconstruction

We obtain satisfying results also with the dense 3D reconstruction task: estimated surfaces are usually close to the real ones (e.g., Fig. 1) also in the case of sharp 3D features such as corners of buildings (reference images for these reconstructions are shown in Fig. 7).
It is important to note that these results are obtained by tracking only between the last two consecutive frames: all previous information is always discarded.
Tracking over a longer sequence of frames will enable a more accurate surface reconstruction and a more reliable ego-motion estimation.

C. Efficiency Considerations

The whole approach (features and sample point extraction, constrained 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) makes it necessary 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 when using a higher frame rate (e.g., 30 Hz), as a result of the reduced baseline, the optimization correctly converges using only 2 levels and around 20 iterations. We defer to exploit this fact to future work, along with a longer sequence of images involved in the tracking process. Providing the optimization procedure with a CUDA implementation, the system can be easily run in real time on a standard PC.


(a)

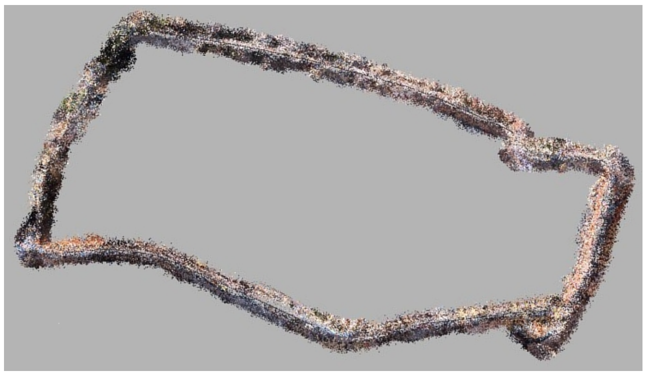

(b)

Fig. 8. In (a) a satellite image of the urban scenario where the experiments were performed [source: Google Maps]. The light blue line depicts the followed 1400 meters path. In (b) the estimated path along with the estimated 3D positions of the mesh's vertices.

V. CONCLUSIONS AND FUTURE WORKS

In this paper, we have tackled the Visual SLAM problem by proposing an efficient method that enables dense structure reconstruction and navigation in large-scale environments. Starting from the assumption that the surrounding scene forms a piecewise smooth surface represented by a triangle mesh, we proposed to track the surface's projection between frames using a "guess" of this projection based on a constrained Delaunay triangulation built taking into account both corner and edgelet features extracted form the images. The 3D structure along with the camera motion is then recovered inside an efficient optimization procedure that exploits the topology of the subdivision.

Experiments performed in a challenging large scale urban scenario show the effectiveness of our system in both ego-motion estimation and dense 3D reconstruction tasks.

We plan to improve our system by 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.

ACKNOWLEDGMENT

We would like to thank prof. Stefano Soatto for the comments and fruitful discussions on this work. This research was supported by a grant of the University of Padova.

REFERENCES

[1] 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.

[2] 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.

[3] 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.

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

[5] P. Elinas, R. Sim, and J. Little. Sigma-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.

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

[7] Ø. Hjelle and M. Dæhlen. *Triangulations and Applications*. Mathematics and Visualization. Springer, 2006.

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

[9] 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.

[10] M. Lhuillier. A generic error model and its application to automatic 3d modeling of scenes using a catadioptric camera. *International Journal of Computer Vision*, 91(2):175–199, 2011.

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

[12] 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.

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

[14] 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.

[15] A. Pretto, S. Soatto, and E. Menegatti. Scalable dense large-scale mapping and navigation. In *Proceddings of: Workshop on Omnidirectional Robot Vision (ICRA 2010).*, 2010.

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

[17] D. Scaramuzza, F. Fraundorfer, M. Pollefeys, and R. Siegwart. Absolute scale in structure from motion from a single vehicle mounted camera by exploiting nonholonomic constraints. In *Proceddings of: IEEE International Conference on Computer Vision (ICCV)*, 2009.

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

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

[20] 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.

[21] 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.

[22] J-P Tardif, Y. Pavlidis, and K. Daniilidis. Monocular visual odometry in urban environments using an omnidirectional camera. In *Proceddings of: International Conference on Intelligent Robots and Systems (IROS)*, 2009.