Machine Vision and Applications (2018) 29:1041–1068
https://doi.org/10.1007/s00138-018-0951-x

ORIGINAL PAPER

Localization of 3D objects using model-constrained SLAM

Angelique Loesch[1] · Steve Bourgeois[1] · Vincent Gay-Bellile[1] · Olivier Gomez[1] · Michel Dhome[2]

Received: 3 April 2017 / Revised: 6 June 2018 / Accepted: 6 June 2018 / Published online: 30 June 2018
© Springer-Verlag GmbH Germany, part of Springer Nature 2018

Abstract
Accurate and real-time camera localization relative to an object is needed for high-quality Augmented Reality applications. However, static object tracking is not an easy task in an industrial context where objects may be textured or not, have sharp edges or occluding contours, be relatively small or too large to be entirely observable from one point of view. This paper presents a localization solution built on a keyframe-based SLAM algorithm. This solution only requires a video stream of a 2D camera (color or grayscale) and the prior knowledge of a 3D mesh model of the object to localize (also named object of interest in this document). The 3D model provides an absolute constraint that drastically reduces the SLAM drift. It is based on 3D-oriented contour points called edgelets, dynamically extracted from the model using Analysis-by-Synthesis on the graphics hardware. This model constraint is then expressed through two different formalisms in the SLAM optimization process. The dynamic edgelet generation ensures the genericity of our tracking method, since it allows to localize polyhedral and curved objects. The proposed solution is easy to deploy, requiring no manual intervention on the model, and runs in real time on HD video streams. It is thus perfectly adapted for high-quality Augmented Reality experiences. Videos are available as supplementary material.

Keywords Simultaneous localization and mapping · Constrained bundle adjustment · Occluding contours · Memory consumption · Real time · Augmented Reality

1 Introduction

Applications such as quality control, automation of complex tasks or maintenance support with Augmented Reality (AR) could greatly benefit from visual tracking of 3D objects [7]. However, this technology is underexploited due to the difficulty of providing deployment easiness, localization quality and genericity simultaneously. Most existing solutions indeed involve a complex or an expensive deployment of motion capture sensors, or require human supervision to simplify the 3D model [33]. And finally, most tracking solutions are restricted to textured or polyhedral objects to achieved an accurate camera pose estimation [4,38].

Tracking any object is a challenging task due to the large variety of object forms and appearances. Industrial objects may indeed have sharp edges, or occluding contours that correspond to non-static and viewpoint-dependent edges. They may also be textured or textureless. Moreover, some applications require to take large amplitude motions as well as object occlusions into account, tasks that are not always dealt with common model-based tracking methods. These approaches indeed exploit 3D features extracted from a 3D model, which are matched with 2D features in the image of a video stream [15]. However, the accuracy and robustness of the camera localization depend on the visibility of the object as well as on the motion of the camera.

Electronic supplementary material The online version of this article (https://doi.org/10.1007/s00138-018-0951-x) contains supplementary material, which is available to authorized users.

✉ Angelique Loesch
angelique.loesch@cea.fr

Steve Bourgeois
steve.bourgeois@cea.fr

Vincent Gay-Bellile
vincent.gay-bellile@cea.fr

Olivier Gomez
olivier.gomez2@cea.fr

Michel Dhome
michel.dhome@univ-bpclermont.fr

1 CEA, LIST, Laboratory of Vision and Content Engineering, F-91191 Gif-sur-Yvette, France

2 Pascal Institute, Blaise Pascal University, Clermont-Ferrand, France

To better constrain the localization when the object is static, recent solutions rely on environment features that are reconstructed online, in addition to the model ones. These approaches combine simultaneous localization and mapping (SLAM) and model-based tracking solutions by using constraints from the 3D model of the object of interest. This model can indeed be used to constrain the SLAM initialization [2] or its optimization process [23,31,35]. Constraining SLAM algorithms with a 3D model results in a drift-free localization. However, such approaches are not generic since they are only adapted for textured or polyhedral objects. Furthermore, using the 3D model to constrain the optimization process may generate high memory consumption and limit the optimization to a temporal window of few cameras.

In this paper, we propose a solution that fulfills the requirements concerning deployment easiness, localization quality and genericity. This solution, based on a visual keyframe-based constrained SLAM, only exploits a camera and a geometric 3D model of the static object of interest. A classical camera is indeed preferred over an RGBD sensor, since the latter imposes limits on the volume, the reflectiveness or the absorptiveness of the object, and the lighting conditions. A geometric 3D model is also preferred over a textured model since textures may hardly be considered as stable in time (deterioration, marks, etc.) and may vary for one manufactured object. Furthermore, textured 3D models are currently not widely spread. Contrarily to previous methods, the presented approach does not need 3D model simplification to bring out sharp edges. It deals with polyhedral and curved objects by extracting sharp, occluding contours and silhouette from a model rendered on GPU and is real time, accurate and robust to occlusion or sudden motion.

2 Related work

Several visual localization methods exist in the state of the art. This section focuses especially on approaches adapted to object tracking like model-based tracker solutions that only exploit the knowledge of the 3D object model and constrained SLAM (C-SLAM) algorithms for which the camera evolves in a partially unknown environment with the 3D model as unique a priori.

2.1 Model-based tracking methods

Camera localization relative to an object of interest may be achieved thanks to model-based tracking methods. These solutions rely on a two-step process. First, 3D features provided by a 3D model of the object of interest are matched with their corresponding 2D features in the image. Then, the camera pose that minimizes the re-projection of these 3D features with respect to their 2D counterparts is estimated. With

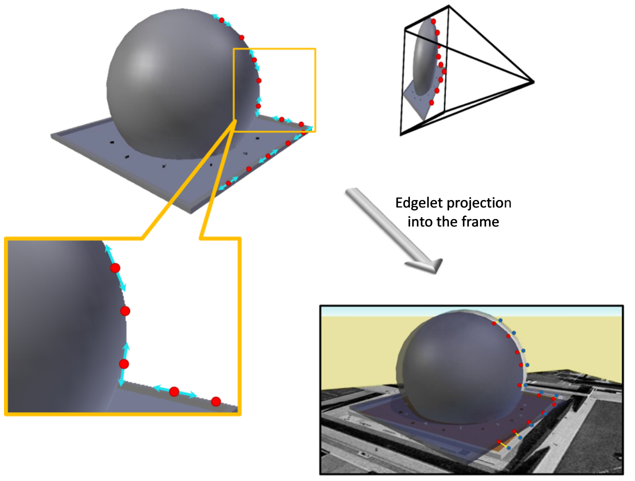

Fig. 1 Edgelet representation on the 3D model of a french monument called the Geode. Edgelets (in red) correspond to 3D- oriented surface points that may result in contour points (in blue) in the 2D image. They have the 3D orientation of the sharp edge or the occluding contour to which they belong (blue arrows) (color figure online)

a textureless 3D model, these 3D features are 3D-oriented surface points on the model, which may result in contour points in the image. These surface points corresponding to infinitesimal edges are 3D edge points with the orientation of the model edge to which they belong as presented in Fig. 1. We refer to them as edgelets throughout the paper.

The main difficulties of model-based tracking using such features are encountered during the 3D/2D matching step. Visual features such as 2D contour points are indeed not adapted to local discrimination (difficulty of distinguishing an edge point from another). To constrain the 3D/2D matching, the camera motion has to be small between two frames which limits the solution robustness against fast and sudden displacements. Besides, edgelets may depend on the camera point of view (silhouette, auto-occluding contours, etc.) and cannot be extracted from sharp edges of the model as edgelets of the Geode's dome in Fig. 1. In early works [4,39], to bypass these issues, some methods have only focused on polyhedral object tracking or limited the camera motion to small displacements.

However, polyhedral object tracking solutions are not generic approaches since they cannot deal with curved objects. When the object of interest is indeed polyhedral, surface points that generate contours in the image mostly come from sharp edges of the model. These edgelets are thus independent of the camera point of view and can be precomputed. They are identified by a simple threshold on the dihedral angle between the model facets. However, the dihedral angle criteria are not relevant when dealing with manufactured objects with sharp edges rounded into fillets. These criteria can be applied if the camera/object distance is long enough to visualize fillets as sharp edges. However, if the camera is close to the object, no edgelet will be detected. A simplification of object models may resolve this problem by transforming round edges into sharp edges and allowing to be independent with respect to the observation distance, but the simplification degree generally needs to be adjusted by an expert.

Besides, experts are often needed to manipulate the sensor during the object tracking in order to control the camera motion. Small amplitude displacements between images are indeed needed to facilitate 3D/2D matching between edgelets and 2D contour points and to maintain good robustness and stability. The 3D/2D matching step is usually performed by projecting each edgelet extracted from the model and by searching locally around this projection for a 2D image contour with a similar orientation. Without prediction, this projection is performed on the current image with the pose estimated on the previous one. Nevertheless, a small research area is crucial for low discriminant features such as edges to avoid matching errors and to keep real-time performances.

Among the state of the art, we can distinguish two families of approaches that try to tackle these model-based tracking limitations. The first family tries to improve the genericity, while the other one tries to improve the robustness to large displacements and sudden motions.

The first family of approaches try to improve the genericity of the tracking solution by considering occluding contours in addition to sharp edges. Some model-based trackers locally approximate the 3D model surface using parametric models based on curvature radius [18] or quadrics [28]. These parameterizations have the advantage of being differentiable and thus easily integrable in a cost function to optimize. However, for very complex objects, the number of quadrics or curvature radii may quickly increase and complicate the model parametrization. To enable computationally tractable tracking, the number of quadrics or curvature radii has to be limited which results in an approximation of object surfaces for complex objects and thus a less accurate tracking. Furthermore, these parametric models do not take into account the observation distance. For example, the curvature radius is computed from the mesh geometry and may not correspond to the object curvature in the image that is relative to the camera/object distance. The curvature can be smooth if the object of interest is far from the camera, but more complex if the observation distance is small and more details appear.

Other solutions exploit Analysis-by-Synthesis to identify the edgelets of an object model, with the use of normal and depth maps from synthetic model renders [27,29,41]. These methods are based on non photo-realistic renders to be independent of illumination conditions [10]. They have the advantage of being efficient on any object, since they consider the observation distance and exploit the model without simplification. However, the resulting edgelets are only locally valid, since they are not parametrized according to the object surface. Thus, they have to be re-estimated at each image of the sequence. This process is time-consuming due to the data transfer between GPU and CPU, especially when the geometric model has many faces or when the image has a high resolution.

Other methods aim to lift the low amplitude motion constraint, in order to increase the tracking robustness to sudden motion. A first solution is to use particle filters [13] to eliminate this motion constraint by generating an important number of hypotheses for the current pose. Each of them is weighted based on the re-projection error between sharp edges and their closest edge in the image. This approach offers a robust tracking solution with respect to fast motions and occlusions. However, its computing cost is not compatible with a real-time execution for complex objects.

Another strategy to increase the displacement between two frames is to predict the camera pose on the current image. One possibility is to use an external sensor such as an IMU [1]. It is also possible to exploit image features that are more discriminant than edgelets, e.g., keypoints, to compute a first estimation of the camera displacement between the previous image and the current one. This estimation is then refined with the model edges. In [29,38], keypoints extracted on the objects of interest are used. Since these approaches require a textured object that is always visible in the image (not occluded), an alternative solution is to exploit the object environment by extracting keypoints on the whole image as proposed by constrained SLAM solutions.

2.2 Constrained SLAM solutions

Model-based tracking methods only exploit the 2D contour points of the object in the image. However, in a static scene, other features from the environment can provide additional constraints. SLAM algorithms are based on these features to estimate the camera pose but suffer from drift of the scale factor and do not suit to object tracking. Constrained SLAM approaches use both multi-view geometry constraints from the scene and geometric constraints imposed by the 3D model of the object to localize the camera according to the object of interest. In this section, a first part describes keyframe-based SLAM methods that only rely on the multi-view geometry constraints to estimate the camera pose. A second part presents these solutions whose the optimization process is constrained to the object 3D model in order to not suffer from drift of the scale factor.

2.2.1 Keyframe-based SLAM

Keyframe-based SLAM algorithms [14,24] are iterative processes that reconstruct online 3D primitives from the environment with respect to 2D observations and estimate the camera pose for each image thanks to 2D/3D correspondences between the previously reconstructed primitives and those extracted from the images. The reconstructed map is improved with new primitives at each keyframe. Since such incremental approach is prone to error accumulation, a possible solution is to use a nonlinear optimization process called bundle adjustment. It locally refines camera poses chosen on a temporal or spatial window and 3D feature points by minimizing their re-projection errors [37]. Standard SLAM algorithms [14,24] are often used to estimate the localization of a camera in an unknown environment. However, they are not well adapted for object tracking. With monocular solutions, camera poses are indeed expressed in an arbitrary coordinate frame and with an arbitrary scale that is subject to drift over time. The use of the 3D model as a constraint has then been studied. It can indeed be used to constrain the SLAM initialization [2] by providing a pose obtained thanks to a model-based tracker. Nevertheless, although the coordinate frame is specified and an initial scale is provided to the SLAM with the model constraint, this solution is still prone to error accumulation when the camera is getting away from the initialization pose. Thus, the idea to use constrained bundle adjustments that minimize simultaneously the multi-view geometry and a constraint term based on the 3D model has been developed.

2.2.2 Constrained bundle adjustment

Since standard SLAM algorithms are not appropriate for object tracking, C-SLAM methods propose to include constraints provided by the model of the static object of interest in their optimization process. This constraint integration allows to express the SLAM reconstruction into the object-frame with the correct scale and to prevent the drift of SLAM algorithms throughout the entire tracking sequence.

Model constraints exploit an a priori partial knowledge of the scene geometry, usually a 3D model, and can be defined through different metric or re-projection formalisms. They can be expressed with a metric error [31] corresponding to the 3D distance between 3D features reconstructed by SLAM and the model of the object. However, combining this error with a pixel one in the constrained bundle adjustment requires an adaptive weight to deal with heterogeneous measurements.

Model constraints can also be expressed through a re-projection error of 3D features reconstructed by SLAM and projected on the model, while some of their degrees of freedom (DoF) are fixed during the optimization [21,36]. The constraint can furthermore be defined with a re-projection error between 3D features extracted from the model and their associated image observations. These features are considered as an absolute information and are fixed in the bundle adjustment process. They may come from a 3D point cloud [23] or, when objects of interest are textureless, correspond to 3D-oriented points (edgelets) from edges of a 3D model [35] as model-based tracker solutions. This latter method is called in this article edgelet-constrained SLAM (EC-SLAM). However, although the approach of [35] is real time and accurate, it reaches its limits for curved objects since only sharp edges are exploited. Furthermore, this approach using a re-projection error in its optimization process is not compact in memory due to contour images and the set of edgelets that have to be stored. Memory consumption grows with the duration and the resolution of the video and limit the local bundle adjustment to a spatial window. Besides, this edgelet constraint is time-consuming since the 2D/3D correspondences have to be re-estimated at each constrained bundle adjustment on all the optimized keyframes.

We propose in this paper an EC-SLAM that tracks polyhedral and curved objects with robustness and accuracy. Edgelets are dynamically generated using Analysis-by-Synthesis and integrated as a model constraint into a C-SLAM algorithm. Moreover, we propose different formalisms of this constraint to reduce the memory consumption of the optimization process.

Notations Matrices are designated in this paper by sans-serif capital font such as M and vectors by bold font such as $\mathbf{v}$ or $\mathbf{V}$. The projection matrix P associated with a camera is given by $\mathsf{P} = \mathsf{K}\mathsf{R}^\top(\mathsf{I}_3|-\mathbf{t})$, where K is the matrix of intrinsic parameters and $(\mathsf{R}, \mathbf{t})$ the extrinsic ones. $\mathbf{X}$ corresponds to all the optimized parameters in a bundle adjustment: the extrinsic parameters of the optimized cameras $\{\mathsf{R}_j, \mathbf{t}_j\}_{j=1}^{S}$ and the 3D point positions $\{\mathbf{Q}_i\}_{i=1}^{N_Q}$. $\bar{\mathbf{X}}$ corresponds to the vector concatenating all the optimized pose parameters $\bar{\mathbf{X}}_j = \{\mathsf{R}_j, \mathbf{t}_j\}_{j=1}^{S}$.

3 Overview and contributions

This paper is an extension of previous works [19,20]. It proposes a constrained SLAM solution for generic object tracking that deals with large amplitudes and occlusions.

Our EC-SLAM framework shown in Fig. 2 is built on keyframe-based SLAM as [35] and divided into a localization and a mapping threads running in parallel. In the tracking thread, the camera pose is estimated at each frame thanks to a matching algorithm that establishes 2D/3D correspondences, and the decision to create a keyframe is made. The mapping thread is performed in parallel of the localization process when a keyframe is detected. Generally, it is decomposed into two important steps, the triangulation that extends the 3D map with new 3D features, and the refinement of camera poses and 3D feature positions through a bundle adjustment constrained to the 3D model on a local window of keyframes. Our framework is, however, slightly different to answer the genericity and robustness issues.

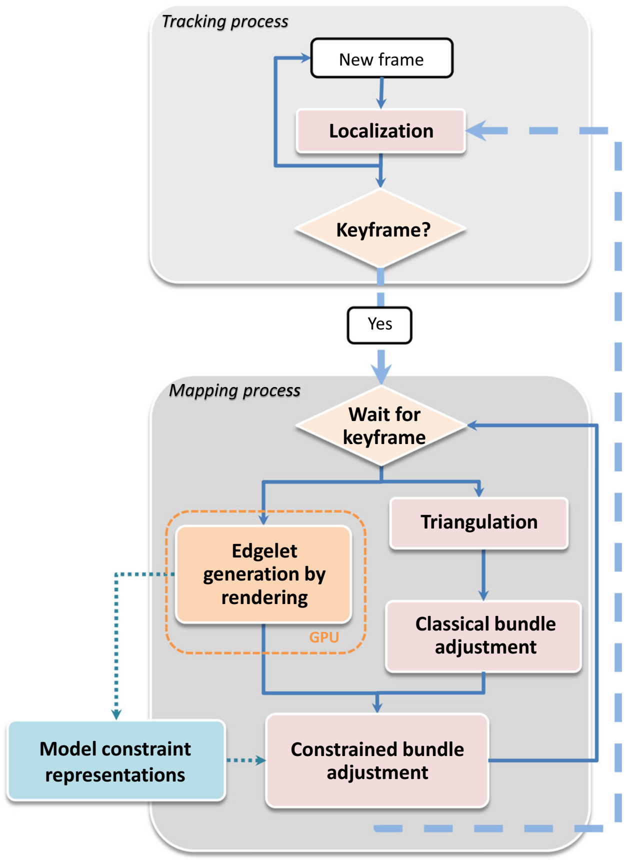
Tracking process
New frame
Localization
Keyframe?
Yes
Mapping process
Wait for keyframe
Triangulation
Edgelet generation by rendering
GPU
Classical bundle adjustment
Model constraint representations
Constrained bundle adjustment

Fig. 2 Our EC-SLAM framework with a four-step mapping process: triangulation, model constraint formalism, classical and constrained bundle adjustment. Edgelet generation by rendering is performed in parallel on GPU before the model constraint representation

3.1 Dynamic edgelet extraction

We represent the model constraint with a constellation of visible edgelets for the current keyframe point of view. In [35], the model constraint is provided by static edgelets extracted offline on sharp edges of the 3D model. A visibility test is then performed to only get the subset of visible edgelets for each keyframe optimized in the constrained bundle adjustment. However, the visible edgelet constellation may not well constrain the camera pose or the matching step since the number of edgelets varies according to the point of view, a uniform distribution is not guaranteed and the risk of false association is not dealt. Moreover, this solution being limited to polyhedral objects, we propose a different use of the 3D model by extracting dynamically edgelets with Analysis-by-Synthesis.

In Sect. 4, our edgelet extraction by rendering the 3D model of the object of interest is detailed. We propose a solution allowing to deal with curved objects and whose the sampling strategy provides an edgelet constellation adapted to the matching and pose estimation. Nevertheless, replacing the precomputed edgelets of an EC-SLAM as [35] with a constellation generated by rendering implies modifications in the SLAM framework. In fact, since the edgelets generated from the silhouette and occluding contours of the 3D model are viewpoint dependent, they should be estimated online. Moreover, our edgelet generation process induces an additional computing cost that depends on various factors, such as the image resolution used during the rendering, the performance and current workload of GPU (it can also be used to display some virtual content on a screen for Augmented Reality applications). The presented edgelet generation is then achieved in the mapping thread on GPU in parallel of other processes to prevent its computational cost from slowing down the tracking process and to maintain real-time performances. The output of this generation is finally a set of edgelets that are evenly distributed and non-ambiguous for the matching and the pose estimation steps.

3.2 Model constraint representations

The dynamic edgelets are exploited to refine the camera pose and the SLAM reconstruction in the constrained bundle adjustment as presented in Fig. 2. A first representation of the model constraint can be proposed through a re-projection formalism presented in Sect. 3.2.1, although it has an important impact on the memory consumption of the optimization process. An other formalism of the model constraint is then proposed to reduce the memory footprint of our method thanks to the output of a model-based tracker. The edgelet constellation is in that case used to refine the model-based tracker outputs as presented in Sect. 3.2.2.

3.2.1 Re-projection formalism

A first possible model constraint formalism takes directly advantage of edgelet information through a re-projection error $E(\bar{\mathbf{X}})$ similar to the one proposed by Tamaazousti et al. [35], where our dynamic edgelets are associated with 2D contour points in the image.

In that case, the model constraint is expressed as follows:

$$E(\bar{\mathbf{X}}) = \sum_{j=1}^{S} H(\bar{\mathbf{X}}_j, \bar{\mathbf{m}}_j), \tag{1}$$

with $H(\bar{\mathbf{X}}_j, \bar{\mathbf{m}}_j)$ the edgelet re-projection error

$$H(\bar{\mathbf{X}}_j, \bar{\mathbf{m}}_j) = \sum_{i=1}^{N_M} (\mathbf{n}_{i,j}.(\mathbf{m}_{i,j} - \mathsf{P}_j \mathbf{M}_i))^2. \tag{2}$$

It measures the orthogonal distance between the projection of edgelets $\mathbf{M}_i$ and their corresponding edge $\mathbf{m}_{i,j}$ concatenated in the vector $\bar{\mathbf{m}}_j$, for the jth keyframe. $\mathbf{n}_{i,j}$ is the normal to the projection of the edgelet direction. This re-projection formalism will be evaluated in Sect. 6.

However, with this constraint representation, edgelet constellations have to be stored for all the optimized keyframes. This storage increases the memory footprint of the optimization process, which grows with the duration and the resolution of the video. Besides, this formalism is time-consuming since the 2D/3D correspondences have to be re-estimated at each constrained bundle adjustment on all the optimized keyframes. Two other representations of the model constraint are proposed in this article to manage this issue.

3.2.2 Hybrid model/trajectory constraint formalisms

These other model constraint formalisms are presented in details in Sect. 5. Both correspond to hybrid model/trajectory constraints that do not exploit the dynamic edgelet constellations directly in the error expression as with the re-projection formalism. They instead use the outputs of a model-based tracker (named model-based poses as opposite of the SLAM poses). These constraints, which we define as pose errors, are also more compact in memory, but expressed in pixels to be homogeneous with respect to the environment constraint (see Sect. 3.3) proposed by the SLAM in the constrained bundle adjustment.

The output model-based poses used in both formalisms are the optimal camera poses with respect to the object model that best align the dynamic edgelets and their observations in the image. Since the 2D-associated contour points that correspond to edgelets are initially unknown, the camera pose and these correspondences are defined as the parameters that minimize the model-based error given by Eq. (2) whose vector representation is as follows:

$$H(\bar{\mathbf{X}}_j, \bar{\mathbf{m}}_j) = h^\top(\bar{\mathbf{X}}_j, \bar{\mathbf{m}}_j) h(\bar{\mathbf{X}}_j, \bar{\mathbf{m}}_j), \quad (3)$$

with $h^\top(\bar{\mathbf{X}}_j, \bar{\mathbf{m}}_j) = [(\mathbf{n}_{0,j}.(\mathbf{m}_{0,j} - \mathsf{P}_j \mathbf{M}_0)) \cdots (\mathbf{n}_{N,j}.(\mathbf{m}_{N,j} - \mathsf{P}_j \mathbf{M}_N))]$. This minimization problem is usually solved by alternating the estimation of the observations $\bar{\mathbf{m}}_j$ and the estimation of the pose parameters $\bar{\mathbf{X}}_j$. The $\bar{\mathbf{m}}_j$ are estimated by projecting the edgelets with the current pose parameters and matching them to the nearest contour with a similar orientation. The pose parameters are then refined by minimizing $H(\bar{\mathbf{X}}_j, \bar{\mathbf{m}}_j)$ with a Levenberg–Marquardt algorithm [16].

To ensure a good convergence, such an algorithm requires an initial estimation of $\bar{\mathbf{X}}_j$ close to the solution and a configuration of edgelets/observations that constrains the 6 DoF of the camera. In our context, the first assumption is verified since the optimization process uses the SLAM pose as initial guess. On the contrary, the second assumption cannot be ensured. It is then preferable to keep the poorly constrained DoF to their initial value during the optimization process. This can be achieved by using a truncated Levenberg–Marquardt [6]. This approach relies on a different approximation of the Hessian matrix's cost function than the usual Levenberg–Marquardt. While the Hessian matrix of $H(\bar{\mathbf{X}}_j, \bar{\mathbf{m}}_j)$ is approximated with $\mathsf{W}_j \approx \mathsf{J}_j{}^\top \mathsf{J}_j$ in classical Levenberg–Marquardt [30], it is approximated by $\mathsf{W}_j = TSVD(\mathsf{J}_j{}^\top \mathsf{J}_j)$ in a truncated Levenberg–Marquardt, where $TSVD(\mathsf{A})$ represents the truncated SVD of the matrix A.

The resulting orientation and position parameters $\bar{\mathbf{X}}^c_j = \{\mathsf{R}^c_j, \mathbf{t}^c_j\}$ of the optimal model-based pose are stored to be used as a constraint for this keyframe over the rest of the tracking. The associated projection matrix is denoted P^c_j. The vector $\bar{\mathbf{X}}^c$ represents the set of model-based pose parameters $\bar{\mathbf{X}}^c_j = \{\mathsf{R}^c_j, \mathbf{t}^c_j\}$ for all the keyframes optimized in the constrained bundle adjustment. The 2D contour points associated with edgelets on keyframe j with the optimal pose $(\mathsf{R}^c_j|\mathbf{t}^c_j)$, are concatenated in the vector $\bar{\mathbf{m}}^c_j$. Finally, we define $\mathsf{W}^c_j = \mathsf{J}^{c\top}_j \mathsf{J}^c_j$ as the approximation of $H''(\bar{\mathbf{X}}^c_j, \bar{\mathbf{m}}_j)$.

3.3 Optimization process

Since our model constraint may present inaccuracies especially if the object is occluded, creating incorrect 2D/3D associations or erroneous model-based poses, optimizing simultaneously as [35] the model and the multi-view constraints could result in a deterioration of the multi-view geometry relationships, which may cause localization failures. Then, our approach consists in first optimized the environment constraint only. We determine the optimal solution of the multi-view relationships for all the optimized keyframes. These keyframes are selected via a covisibility graph [25]. To obtain this optimal solution, a classical bundle adjustment without any model constraint is thus performed as presented in Fig. 2. The cost function $G(\mathbf{X})$ to minimize is as follows:

$$G(\mathbf{X}) = \sum_{i=1}^{N_Q} \sum_{j \in L_i} d^2(\mathbf{q}_{i,j}, \mathsf{P}_j \mathbf{Q}_i). \quad (4)$$

The re-projection error d of Eq. (4) is the euclidean distance between $\mathbf{q}_{i,j}$ the 2D observation of the ith 3D point $\mathbf{Q}_i$ in the jth keyframe and the projection of this 3D point. L_i is the set of the keyframe indexes observing $\mathbf{Q}_i$.

Then, the model and multi-view constraints are combined in a constrained bundle adjustment. Since multi-view constraint is less robust but more accurate than the model constraint, we use a fusion strategy, which ensures that the degradation of the multi-view constraint remains small. To achieve this task, a constrained bundle adjustment framework with a barrier term as [17] is chosen. Thus, the cost function of our constrained bundle adjustment combining multi-view and model constraints is given by:

$$F(\mathbf{X}) = \frac{\omega}{e_t - G(\mathbf{X})} + E(\bar{\mathbf{X}}), \tag{5}$$

where $E(\bar{\mathbf{X}})$ is the model constraint re-projection error or the two formalisms described in next Sect. 5 using the optimal model-based poses presented in Sect. 3.2.1. e_t is a threshold which is slightly greater than the squared re-projection error obtained after minimizing Eq. (4). The fraction $\frac{\omega}{e_t - G(\mathbf{X})}$ with $\omega > 0$ corresponds to a regularization term that prevents the degradation of the multi-view relationships estimated after the classical bundle adjustment.

We can notice that in practice, with the re-projection formalism proposed by Tamaazousti et al. [35] and defined by Eq. (1), our dynamic edgelets can also be directly integrated in the constraint bundle adjustment with a classical linear cost function. Then, this latter will be only composed of the edgelet constraint and the multi-view relationships between the images. Both terms will be expressed in pixels and optimized simultaneously. This classical linear cost function will be given by:

$$F(\mathbf{X}) = G(\mathbf{X}) + E(\bar{\mathbf{X}}), \tag{6}$$

with $G(\mathbf{X})$ the environment constraint presented in Eq. (4) and $E(\bar{\mathbf{X}})$ the re-projection formalism of the model constraint. Moreover, the classical bundle adjustment only optimizing the multi-view geometry relationships becomes optional.

3.4 Roadmap and contributions

Edgelet generation is detailed in Sect. 4. Additional contributions over [19,20] include the edgelet orientation definition in Sect. 4.1.3 and the improvement of the CPU/GPU transfer in Sect. 4.1.5. Furthermore, these edgelet generation steps are evaluated in Sect. 6. Both hybrid model/trajectory constraints are described in Sect. 5. Our EC-SLAM solution is evaluated on several polyhedral and curved objects and compared with other tracking solutions on synthetic and real data to assess its genericity, robustness and accuracy. All the evaluations of Sect. 6 are performed with the new edgelet orientation and data transfer, since they allow more accuracy in the camera pose estimation and better performances.

Moreover, additional contributions over [19,20] concern the evaluation of our EC-SLAM framework robustness when the model constraint is inaccurate, and the evaluation of our solution tested on public datasets. Section 7 is also a contribution of this paper where we present an Augmented Reality application involving a static curved object with movable parts.

4 Dynamic edgelet extraction

Image contours (except those from textures) are sharp brightness changes that are likely to correspond to discontinuities in depth or surface orientation. Furthermore, the detection of these edges is influenced by the observation distance and the image resolution as mentioned in Sect. 2. Tracking solutions based on Analysis-by-Synthesis are adapted to identify 3D surface points that are likely to generate contours from depth and surface orientation discontinuities by relying on renders of 3D models that take into account of the observation distance. For computational performance, these surface points are sampled into a set of edgelets prior to the matching process. However, in most tracking solutions, the importance of the sampling is neglected. Although it has a major impact on the matching of edgelets with 2D image contours and on the pose estimated from these correspondences, most solutions rely on a basic constant-step sampling.

In this section, the introduced edgelet extraction provides a constellation of edgelets adapted to both matching and pose estimation.

4.1 Edgelet generation

Similarly to [41], our virtual rendering process relies on an image space approach. This choice is motivated by the large amount of faces (hundred of thousands) of 3D models used in our applications that will result in costly rendering times for object space approach [8,32].

Our solution aims to estimate for each pixel from the rendered model image, both its probability of being a contour and its probability of being matched with its corresponding contour in a real image.

In the following, let us consider each pixel of the rendered image of the model as a random variable X_i, with $X_i = 1$ if the pixel is a contour, 0 otherwise. We denote Y_i the random variable that represents the 2D direction of a contour pixel X_i, with $Y_i \in \{North, North - East, North - West, West\}$. $L(X_i, Y_i)$ (respectively, $R(X_i, Y_i)$) is the function that returns the neighbor of a pixel X_i located on the left (respectively, right) of the axis defined by the direction Y_i, and $N(X_i, Y_i)$ (respectively, $P(X_i, Y_i)$) correspond to the function that returns the next (respectively, previous) neighbor of X_i along the direction Y_i. These last two functions will be used in Sect. 4.1.3 only. These notations are illustrated in Fig. 3. Finally $Nmap(X_i)$ (respectively, $Dmap(X_i)$) is the function that returns the normal (respectively, the depth) of the pixel X_i.

4.1.1 Edgelet probability

This first step of our edgelet generation estimates for each pixel of the rendered image, its probability of being a con-

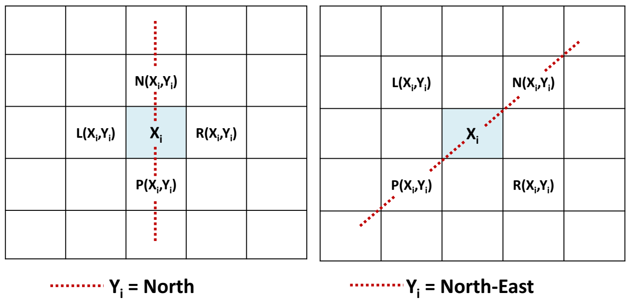

Fig. 3 Each pixel of the rendered image of the model corresponds to the random variable X_i. $L(X_i, Y_i)$ and $R(X_i, Y_i)$ are the neighbor pixels on the left and right of the axis defined by the direction Y_i. $N(X_i, Y_i)$ and $P(X_i, Y_i)$ are the neighbor pixels along the direction Y_i. On the left example with $Y_i = Nord$ and on the right with $Y_i = Nord - Est$

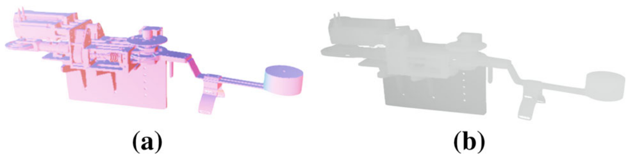

Fig. 4 Normal (left) and depth (right) maps of a rendered orthosis 3D model for a given camera point of view

tour. Usually, two types of 2D contours are distinguished: crease edges and silhouettes. On one hand, crease edges are relative to the sharp edges of the 3D model and appear as discontinuities in the normal map ($Nmap$) (see Fig. 4a). On the other hand, silhouettes correspond to the outline of the object or the auto-occluding contours and appear as discontinuities in the depth map ($Dmap$) (see Fig. 4b). In both cases, the orientation of a 2D contour point corresponds to the 2D direction of the discontinuity in the image space.

Therefore, we define the probability of X_i to be a crease edge with direction Y_i as:

$$\mathbb{P}_{crease}(X_i|Y_i) = \max(1, crease(X_i, Y_i)), \quad (7)$$

with the angular discontinuity measure:

$$crease(X_i, Y_i) = \lambda * (1 - Nmap(L(X_i, Y_i)) \cdot Nmap(R(X_i, Y_i))), \quad (8)$$

where $\lambda = (1 - cos(angleMax))^{-1}$ is the normalization factor defined to reach an intensity of 1 for an angular amplitude of $angleMax$. In our experiments, we use $angleMax = 30°$.

In a similar way, the probability of X_i to belong to the silhouette with an orientation Y_i is given by:

$$\mathbb{P}_{silhouette}(X_i|Y_i) = \max(1, silhouette(X_i, Y_i)), \quad (9)$$

where $silhouette$ is the depth discontinuity measure defined as follows:

$$silhouette(X_i, Y_i) = \beta * \text{Laplacian}(Dmap, X_i, Y_i). \quad (10)$$

Laplacian is the 1D Laplacian oriented with respect to direction Y_i, and β is a weight factor. Since the discontinuity is normalized with respect to the observation distance, the parameter β does not depend on the dimension of the scene. Therefore, β can be easily interpreted as the minimal depth discontinuity, expressed as a ratio of the observation distance, that provides a silhouette contour with a probability of 1. In our experiments, we use $\beta = 1/(0.03 * Dmap(X_i))$.

Consequently, the probability of being a contour is defined as:

$$\mathbb{P}_{contour}(X_i) = \max_{Y_j}(\max(\mathbb{P}_{crease}(X_i|Y_j), \mathbb{P}_{silhouette}(X_i|Y_j))) \quad (11)$$

and the 2D direction associated with X_i is defined as:

$$Dir_{2D}(X_i) = \arg\max_{Y_j}(\max(\mathbb{P}_{crease}(X_i|Y_j), \mathbb{P}_{silhouette}(X_i|Y_j))). \quad (12)$$

4.1.2 Matching probability

This second step estimates the probability that each virtual contour is matched to its corresponding contour in the real image. Usually, during the matching process, the edgelets are associated with the nearest contour located along the normal of the contour. Consequently, the probability of correctly matching an edgelet can be assessed from the number of contours encountered within this 1D neighborhood.

Therefore, we propose to compute this probability from the probability map estimated at the previous step. In the following, let us consider X_n to be the pixels of the 1D neighborhood of a pixel X_i still along the normal of the contour. And let N be the random variable representing the number of contours within the neighborhood. We define the probability of a pixel to be both a contour and correctly matched in a real image according to $\mathbb{E}(N)$, the expected value of N, as:

$$\mathbb{P}_{match}(X_i) = \frac{\mathbb{P}_{contour}(X_i)}{1 + \mathbb{E}(N)}. \quad (13)$$

This matching probability is represented in Fig. 5 for each edgelet detected on the rendered orthosis model.

4.1.3 3D edgelet orientation

The edgelet orientation is useful in the 2D/3D matching step to define a research area in the contour image and to select

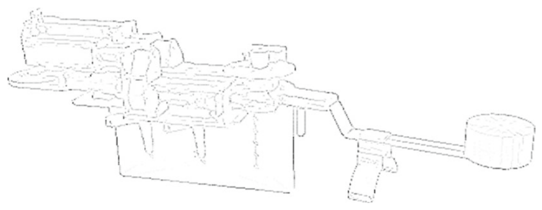

Fig. 5 Matching map of the rendered orthosis 3D model for a given camera point of view. The darker the pixel, the likelier the edgelet is to match to a 2D contour point

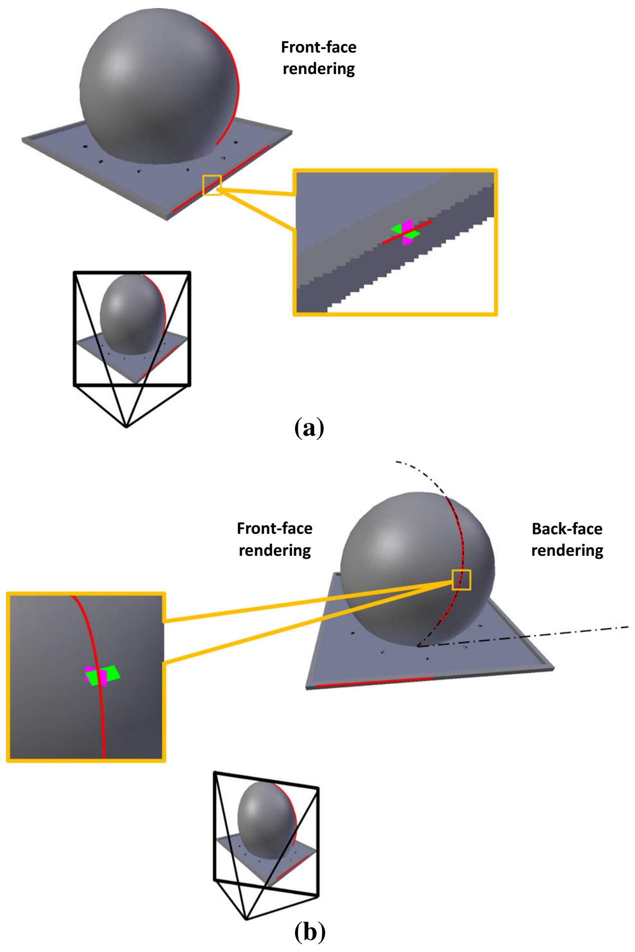

Fig. 6 Parametric model of the 3D orientation of edgelets. These last ones belong to the intersection of two planes (pink and green) representing locally the model surface. The 3D edgelet orientation corresponds to the direction of the intersection defined thanks to the two plane normals. **a** If edgelets (red) are extracted from sharp edges, the two planes forming the intersection are on the front-face rendering of the object model (here the french monument the Geode). **b** If they are extracting from occluding contours, one of the plane is located on the back-face rendering

the 2D contour point with a similar orientation. In theory, the edgelet and the edge to which it belongs have the same orientation. This orientation can be easily determined from the 3D positions of the two adjacent contour points $N(X_i, Y_i)$ and $P(X_i, Y_i)$ in the edgelet probability map, as exploited in our previous works. However, the rasterization step of the rendering process creates aliasing and damages the orientation estimation. The erroneous edgelet orientation is then able to disturb the 2D/3D matching process and decreases the tracking accuracy. In order to avoid a lack of accuracy on the 3D orientation due to the image resolution and the aliasing effect, we propose to locally define the edgelet orientation by a parametric model. We propose a solution in the image space based on the hypothesis that our 3D model mesh is orientable 2-manifold. As shown in Fig. 6, it is indeed possible to define edgelets from crease edges, or silhouettes (including auto-occluding contours) as the intersection of two planes. The idea is to identify these two planes whose intersection forms an edge that locally approximates the 3D model contour.

Each pixel X_i that might be an edgelet belongs to the intersection between two planes according to our definition. That is why an analysis of pixels in the adjacent neighborhood of X_i in the direction of the contour normal is performed, to establish which of them can be assimilated to the planes creating the contour. The selection process of the two pixels differs if X_i is a crease edge or a silhouette. On one hand, if the contour pixel X_i is part of a crease edge, the plane pixels are chosen by taking the adjacent pixels from both sides of X_i according to the direction $Dir_{2D}(X_i)$. Then, the 3D direction of the plane intersection and by extension the 3D direction of the contour pixel X_i is given by the following cross product:

$$\begin{aligned} &Dir_{3D}(X_i) \\ &\quad = Nmap(L(X_i, Dir_{2D}(X_i))) \\ &\qquad \times Nmap(R(X_i, Dir_{2D}(X_i))), \end{aligned} \tag{14}$$

the plane equations being estimated thanks to the normal map $Nmap$ of the rendered image of the model. On the other hand, if the contour point X_i belongs to the silhouette, the selection process of the two pixels has to take into account that the object has auto-occluding contours. Indeed, one of the planes creating the edge where X_i stands is not visible on the front-face rendering. To visualize this occluded plane, a back-face rendering is achieved. The first neighbor pixel belonging to the hidden plane corresponds to the pixel X_i in terms of 2D position but in the back-face rendering. The second plane pixel is choosing in the front-face rendering as for the crease edges. Its selection depends on its distance to the camera position $dist_{cam}$. Thus, 3D direction of the plane intersection and the 3D direction of the contour pixel X_i is defined as:

$$\begin{aligned} &Dir_{3D}(X_i) \\ &\quad = Nmap_{back}(X_i) \times Nmap(C(X_i, Dir_{2D}(X_i))), \end{aligned} \tag{15}$$

with $Nmap_{back}$ the function that returns the back-face rendering normal of the pixel X_i. $C(X_i, Dir_{2D}(X_i))$ is the

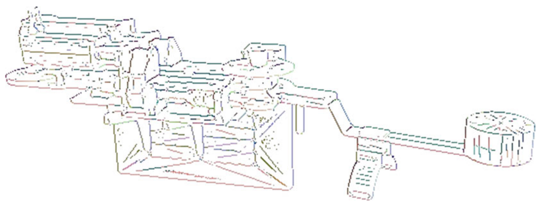

Fig. 7 Orientation map of the rendered orthosis 3D model for a given camera point of view. The 3D orientation is expressed through the three channels of the rendered image, i.e., x-axis orientation values are expressed through the R channel, y-axis orientation values through the G channel and z-axis orientation values through the B channel (color figure online)

neighbor pixel in the front-face rendering the closest of the camera. An orientation map of the rendered orthosis model is shown in Fig. 7.

4.1.4 Edgelet sampling

An edgelet map $\mathbb{P}_{match}$ (see Fig. 5), which maps each pixel to its probability of being a correctly matched edgelet to its corresponding contour, has been estimated previously. This edgelet map is then exploited by the sampling step to provide a set of edgelets relevant to the matching and pose estimation processes.

On one side, to be consistent to the matching process, this set of edgelets must maximize the expectation of the matching success. On the other side, to be suitable to the pose estimation process, the matches estimated from this set of edgelets must constrain the 6 DoF of the camera pose. Particularly, a set of matches that is unevenly distributed in the image, in terms of both position and contour orientation, is not a relevant configuration for pose estimation.

Our sampling process (see Fig. 8) relies on a division of both the 2D position and 2D orientation spaces of the edgelets projections. The 2D position space is divided into a regular grid of $N \times N$ buckets, and each bucket is divided into 4 angular sectors. Then, a set of edgelets is sampled for each angular sector of each spatial bucket, in order to obtain an evenly distribution in 2D space. The sampling itself is achieved with respect to a sampling probability affected to each edgelet. To define this sampling probability, let $\{Z_i\}$ be the set of edgelets of an angular sector of a spatial bucket. The probability $\mathbb{P}_{sampling}$ of an edgelet $Z \in \{Z_i\}$ is defined as follows:

$$\mathbb{P}_{sampling}(Z) = \frac{\mathbb{P}_{match}(Z)}{\sum_i \mathbb{P}_{match}(Z_i)}. \tag{16}$$

With this sampling strategy, inside an angular sector of a spatial bucket, edgelets with a high matching probability $\mathbb{P}_{match}$ are more likely to be sampled. Moreover, the random nature of the sampling prevents the local agglomeration of edgelets.

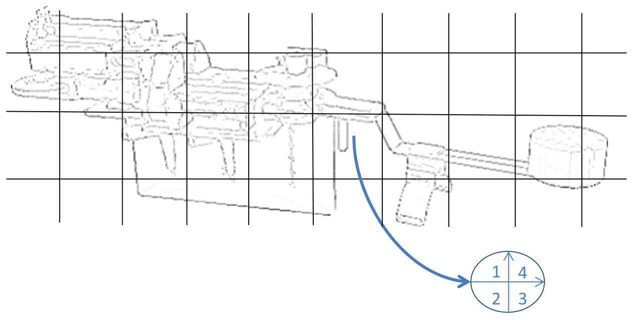

Fig. 8 Edgelet sampling method. The rendered image is divided into buckets, also divided into four angular sectors to obtain an edgelet set evenly distributed in terms of 2D position and orientation

4.1.5 Implementation details

While the edgelet sampling step is achieved on CPU, the edgelet generation step is performed on GPU. Our implementation relies on multiple render passes with GLSL shaders. The first pass computes the depth map $Dmap$ and the normal map $Nmap$. The second and third one compute the back-face normal map $Nmap_{back}$ and the 3D orientation Dir_{3D}. The fourth one computes the edgelet map $\mathbb{P}_{contour}$ and the 2D direction Dir_{2D}. The last one computes the matching probability map $\mathbb{P}_{match}$. Since the sampling is achieved on CPU, the textures containing all the edgelet information have to be transferred to CPU memory. However in each map, most of the pixels are not relevant and correspond to the background or the model surfaces instead of edgelet data. That is why, instead of transferring the entire textures asynchronously by GPU as in earlier works, a filter is proposed as a preprocessing in order to only transfer the edgelet pixels to CPU. This new filter is a render pass based on a geometry shader and a transform feedback process.

A set of vertices is generated on GPU through a vertex buffer object (VBO) in order to optimize computation times. This set contains as many vertices as pixels in one texture. The geometry shader reduces the set size by storing only the vertices representing edgelet pixels with their 2D coordinates as attributes. Moreover, the geometry shader adds more attributes to each edgelet vertex, as 2D and 3D positions, normal and depth values, 3D and 2D orientations and matching probability values of each edgelet. Then, the vertices are saved in an other VBO thanks to the transform feedback process with an interleaved data mode, in order to simplify the edgelet reading on CPU. The new set of vertices is organized on GPU similarly as the sequential reading on CPU and the reading performance is optimal. Since all the edgelet information are stored contiguously, edgelets with their attributes can be indeed read in a sequential way. In this filter step, the rasterization process coming usually after the geometry shader in a render pass is deleted since it is time-consuming and not relevant in our case. The vertices containing the edgelet data are then directly transferred to CPU in an asynchronous way after the geometry shader execution, with the help of pixel buffer object (PBO).

Moreover, to reduce the amount of data to transfer, the probability is encoded on an unsigned byte (the value 255 corresponding to 1).

A performance comparison between these transfer strategy will be given in Sect. 6.2.3.

5 Model constraint representations

Edgelets extracted from the 3D model on GPU, as described in the previous section, are exploited to expressed the model constraint of our EC-SLAM algorithm. They can be directly used in the model constraint representation as proposed in the re-projection formalism presented in Sect. 3.2.1. However, this model constraint representation is memory-consuming with the edgelet and contour image storage and time-consuming with the necessity to re-estimate the 2D/3D matches at each constrained bundle adjustment and on all the optimized keyframes. We propose instead to dedicated the edgelet extraction to the optimization of the model-based poses that are exploited into both formalisms of our model constraint in the present section.

We describe two slightly different approaches that use the 3D model to estimate model-based poses as presented in Sect. 3.2.2. These model-based poses are exploited as hybrid model/trajectory constraints in the constrained bundle adjustment in order to decrease the memory footprint of the optimization process. However, using the outputs of a model-based tracker as constraints implies to deal with heterogeneous error terms between the environment constraint and the model one. We then propose two hybrid model/trajectory constraints to combine the benefits of both kinds of constraints and keep error terms homogeneous.

Different pose distances are introduced in this section, which measure the deviation of pose parameters from $\bar{\mathbf{X}}^c$ by their impact on the model constraint.

5.1 Hybrid model/trajectory constraint formalisms

We propose to use a difference of pixel errors as pose distance. More precisely, it consists in measuring the difference between the re-projection error (see Eq. (2)) corresponding to a keyframe pose $(\mathsf{R}_j|\mathbf{t}_j)$ and the one corresponding to the model-based pose with the parameters $\bar{\mathbf{X}}_j^c$ associated with the same jth keyframe. The corresponding distance is then defined by the following equation:

$$E(\bar{\mathbf{X}}) = \sum_{j=1}^{S} H(\bar{\mathbf{X}}_j, \bar{\mathbf{m}}_j^c) - \sum_{j=1}^{S} H(\bar{\mathbf{X}}_j^c, \bar{\mathbf{m}}_j^c), \qquad (17)$$

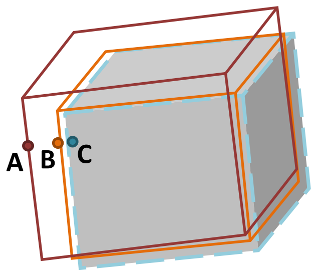

Fig. 9 Point A (respectively, B) is an edgelet extracted from a cube model and projected according to a SLAM keyframe pose in red (respectively, according to the model-based pose in orange). Point C is the 2D contour point associated with the edgelet B. The error $E(\bar{\mathbf{X}})$ of Eq. (18) can be interpreted as the difference $d_{ortho}(A, C) - d_{ortho}(B, C)$, with d_{ortho} the orthogonal distance

Notice that to ensure the pixel error to be minimal at the pose $(\mathsf{R}_j^c|\mathbf{t}_j^c)$, the first term of Eq. (17) is evaluated by using the 2D contour points $\bar{\mathbf{m}}_j^c$ associated with edgelets generated at keyframe j during the model-based pose refinement process. Thus, it is not necessary to re-estimate the 2D/3D associations. Moreover, $(\mathsf{R}_j^c|\mathbf{t}_j^c)$ is the optimal pose with respect to the object model, which best aligns the edgelets and their observations in the image (see Sect. 3.2.2). Consequently, we consider that there is no pose with a re-projection error inferior to the model-based pose $(\mathsf{R}_j^c|\mathbf{t}_j^c)$ and our hybrid error $E(\bar{\mathbf{X}})$ is then always positive. We define the vector representation of this error as follows:

$$E(\bar{\mathbf{X}}) = H(\bar{\mathbf{X}}, \bar{\mathbf{m}}^c) - H(\bar{\mathbf{X}}^c, \bar{\mathbf{m}}^c), \qquad (18)$$

with $\bar{\mathbf{m}}^c$ the vector concatenating the 2D contour points associated with the parameters $\bar{\mathbf{X}}^c$ for all the edgelets and for all the optimized keyframes. An illustration of this hybrid error for a given keyframe and a given edgelet is represented in Fig. 9. With this definition, we ensure that $E(\bar{\mathbf{X}})$ is minimal when the model constraint $H(\bar{\mathbf{X}}, \bar{\mathbf{m}}^c)$ is minimal (*i.e.,* at the pose parameters $\bar{\mathbf{X}}^c$). Besides, the pose defined by the parameters $\bar{\mathbf{X}}^c$, the 2D contour points $\bar{\mathbf{m}}^c$, and consequently the pixel error associated with the parameters $\bar{\mathbf{X}}^c$ are constant. Since the 2D contour points $\bar{\mathbf{m}}^c$ are no longer re-estimated, it is not required to store the contour images. Furthermore, contrary to the re-projection formalism described in Sect. 3.2.1, the edgelet/contour matching step is not necessary anymore, which allows to reduce computation time. Only the vector of the 2D contour points $\bar{\mathbf{m}}^c$ and, in case of dynamic edgelets, the edgelet constellation are memorized for each keyframe. Compared with re-projection formalism, it implies a large reduction in memory footprint and computation cost. However, this memory consumption can remain important if the number of edgelets and keyframes is high. In the following subsections, we introduce two approximations of $E(\bar{\mathbf{X}})$ that reduce the memory footprint.

5.2 First approximation

Thanks to the refinement processing performed with a Levenberg–Marquardt algorithm as presented in Sect. 3.2.2, the model-based tracker has converged to a local minimum. The first approximation of the constraint relies then on the hypothesis that the local minimum also corresponds to the global one. Thus, it is reasonable to consider that the distance between the projection of the edgelets with respect to the parameters $\bar{\mathbf{X}}^c$ and their corresponding 2D contour points is negligible. In Fig. 9, this assumption corresponds to approximate $d_{ortho}(A, C) - d_{ortho}(B, C)$ by $d_{ortho}(A, B)$.

Under such hypothesis, the 2D contour points $\bar{\mathbf{m}}^c_j$ of Eq. (17) are replaced by the projection of their corresponding 3D edgelets with respect to the parameters $\bar{\mathbf{X}}^c_j$. The resulting approximation of $E(\bar{\mathbf{X}})$ is given by:

$$E(\bar{\mathbf{X}}) = \sum_{j=1}^{S} H(\bar{\mathbf{X}}_j, \bar{\mathbf{M}}^c_j), \tag{19}$$

where $\bar{\mathbf{M}}^c_j = \{\mathsf{P}^c_j \mathbf{M}_i\}_{i=1}^{N_M}$ is the vector concatenating edgelets $\mathbf{M}_i$ projected according to P^c_j.

With this approach, it is not necessary to keep the 2D-associated contours in memory, since only the model-based pose parameters are exploited. In the case of polyhedral object, the memory footprint of the resulting constrained bundle adjustment is very small, since only a set of edgelets shared by all the keyframes has to be stored as proposed by Tamaazousti et al. [35]. However, the memory consumption is higher with dynamic edgelets extracted from occluding contours as presented in Sect. 4, since they depend on the point of view. They are generated online and stored for each keyframe. That is why a second approximation of Eq. (18) is proposed to reduce memory consumption.

5.3 Second approximation

This second approximation relies on the hypothesis that the poses of the EC-SLAM keyframes are located in the neighborhood of their corresponding model-based poses. This approximation is valid in practice since the model-based poses are obtained by refining the SLAM poses with Eq. (2). Under this hypothesis, Eq. (18) can be approximated with a second-order Taylor expansion around the model-based poses $\bar{\mathbf{X}}^c$:

$$E(\bar{\mathbf{X}}) \approx E(\bar{\mathbf{X}}^c) + E'(\bar{\mathbf{X}}^c)\delta + \frac{1}{2}\delta^\top E''(\bar{\mathbf{X}}^c)\delta, \tag{20}$$

The SLAM and model-based poses being 4×4 matrices and belonging to the Lie $\mathbb{SE}(3)$ group [3,5], δ corresponds to the projection of the SLAM poses into the tangent space of the model-based poses:

$$\delta = log\left(\begin{pmatrix} R^c & \mathbf{t}^c \\ 0_{1\times3} & 1 \end{pmatrix}^{-1} \begin{pmatrix} R^\top & \mathbf{t} \\ 0_{1\times3} & 1 \end{pmatrix}\right). \tag{21}$$

$\delta \in \mathbb{R}^6$ with $\delta[3\ldots5] = \mathbf{t}^c - \mathbf{t}$ and $\delta[0\ldots2]$ defined as follows:

$$\delta[0\ldots2] = \frac{\theta}{2\sin(\theta)}((\mathsf{R}^{c-1}\mathsf{R}) - (\mathsf{R}^{c-1}\mathsf{R})^T), \tag{22}$$

with $\theta = arccos(\frac{trace(\mathsf{R}^{c-1}\mathsf{R})-1}{2})$. $\bar{\mathbf{X}}^c$ is obtained after the model-based refinement and corresponds to the minima of the model-based cost functions (Eq. (2)) for all the optimized keyframes. Thus, $H'(\bar{\mathbf{X}}^c, \bar{\mathbf{m}}^c)$ (Eq. (3)) is null. Consequently, the first derivative $E'(\bar{\mathbf{X}}^c) = H'(\bar{\mathbf{X}}^c, \bar{\mathbf{m}}^c) = 0$. Besides, the second derivative $E''(\bar{\mathbf{X}}^c) = H''(\bar{\mathbf{X}}^c, \bar{\mathbf{m}}^c)$ can be approximated by the $6S \times 6S$ diagonal matrix W^c with S the number of optimized keyframes. The diagonal term of this matrix is $\mathsf{W}^c_j = \mathsf{J}^{c\top}_j \mathsf{J}^c_j$, $j \in [0\ldots S]$, which corresponds to the approximation of $H''(\bar{\mathbf{X}}^c_j, \bar{\mathbf{m}}_j)$ introduced in Sect. 3.2.2. This matrix is estimated during the model-based refinement step for the keyframe j. W^c_j is determined only once per keyframe and stored in addition to the optimal pose parameters $\bar{\mathbf{X}}^c_j$. The hybrid error defined in Eq. (20) becomes:

$$E(\bar{\mathbf{X}}) = E(\bar{\mathbf{X}}^c) + \frac{1}{2}\delta^\top \mathsf{W}^c \delta \tag{23}$$

Since $E(\bar{\mathbf{X}}^c) = 0$ according to Eq. (18), the hybrid error definition becomes:

$$E(\bar{\mathbf{X}}) = \frac{1}{2}\delta^\top \mathsf{W}^c \delta \tag{24}$$

This definition of our hybrid constraint presents several advantages. W^c converts the difference between the pose parameters to an error in pixels and makes this error homogeneous with the multi-view term of the constrained bundle adjustment (Eq. (5)). It also allows not to store contour images for each keyframe contrary to Eq. (2). Moreover, neither the edgelets, nor their associated observations intervene, allowing not to store any of them. Only the model-based pose $\{\mathsf{R}^c_j, \mathbf{t}^c_j\}$ and the 6×6 matrix W^c_j are exploited in the optimization process for each keyframe.

This hybrid constraint results in a bundle adjustment with a memory footprint invariant to the video resolution and the model complexity.

Table 1 Acronyms of the different SLAM algorithms exploited in the experimental results

EC-SLAM **Edgelet constrained** SLAM			
SEC-SLAM **Static** EC-SLAM	**D**EC-SLAM **Dynamic** EC-SLAM		
rSEC-SLAM	**r**DEC-SLAM	**rh**DEC-SLAM	**ph**DEC-SLAM
SEC-SLAM with **re-projection** error	DEC-SLAM with **re-projection** error	DEC-SLAM with **hybrid re-projection** error	DEC-SLAM with **hybrid pose** error

6 Experimental results

In this entire section, the EC-SLAM of [35] exploiting static edgelets extracted from sharp edges of 3D models will be called static edgelet constrained SLAM (SEC-SLAM). Since this method only uses a re-projection error as presented in Eq. (1), the letter "r" will be added to its acronym to represent this re-projection formalism: rSEC-SLAM. On the contrary, our SLAM solutions exploiting dynamic edgelets extracted from sharp and occluding contours (see Sect. 4) will be called dynamic edgelet-constrained SLAM (DEC-SLAM). The approach using the re-projection error will be named rDEC-SLAM, the letter "r" still representing the re-projection formalism in the acronym. If the first hybrid formalism is used (see Eq. (19)), our solution will be called rhDEC-SLAM (the letter "h" representing the hybrid formalism et the letter "r" representing re-projection error of the first approximation). Finally, our DEC-SLAM exploiting the second hybrid representation (see Eq. (24)) will be named phDEC-SLAM (the letter "p" representing the difference of poses of the second approximation in the hybrid formalism). Table 1 resumes these acronyms.

We first evaluate in Sect. 6.2 the different contributions of this paper against state-of-the-art methods: the 3D orientation of edgelets, their sampling and their transfer, the robustness of our optimization process against erroneous model constraint, the robustness of our method to sudden motions against the robustness of a model-based tracker, its localization accuracy and genericity on polyhedral and curved object. For this evaluation, the EC-SLAM framework which is used exploits dynamic edgelets and the second hybrid constraint presented in Sect. 5.3. Results are only presented with the phDEC-SLAM, since the choice of the constraint formalism has no impact for the evaluation of these contribution bricks of our solution framework.

Secondly, we evaluate in Sect. 6.3 the different model constraint formalisms described in Sects. 3.2.1, 5.2 and 5.3 in terms of computation time, memory consumption and accuracy. Our DEC-SLAM is also compared on these criteria with the solution rSEC-SLAM of [35] exploiting static edgelets. Finally, in Sect. 6.4, our DEC-SLAM using the three model constraint formalisms is evaluated on two public benchmarks: CoRBS [40] and ICL-NUIM [9].

Evaluations are performed on synthetic and real data with objects that have different natures to assess the genericity of the proposed solution. In the experiments, we use a laptop with an Intel (R) Core (TM) i7-4800MQ CPU @ 2.70 GHz processor and an NVIDIA GeForce GT 730M graphics hardware.

6.1 Synthetic data presentation

The synthetic sequences used for the experimental results are generated with a resolution of 640×480 and are composed of a thousand frames. Only the tracked object is replaced across all sequences. The camera trajectory and the object environment are the same. The distance between the camera and the object do not exceed 8 m, and the object of interest is not always entirely in the camera field of view. The object environment is composed of 4 walls covered with brick texture and an untextured ground. Several objects, with a volume of about 4 m^3, have been tested:

- A torus (Fig. 11a), a simple curved object with a 13K face model.
- A dragon (Fig. 11b) that presents both sharp edges and occluding contours. The associated model has 100K faces.
- A part of a robotic exoskeleton arm, referred to as an orthosis (Fig. 11c), with many sharp edges. The 3D model of this object has 264K faces.
- A dwarf (Fig. 14a), a curved object with a 8K face model.
- A bypass (Fig. 19a), a curved object from the chemical industry, mainly composed of pipes. Its CAD model used during the tracking has 152K faces. The bypass environment is slightly different from the others. In some case (see Sect. 6.2.4), the bypass can be occluded by an orthosis in order to allow robustness evaluation as shown in Fig. 13.

The EC-SLAM initialization for the synthetic sequences is obtained by the ground truth pose.

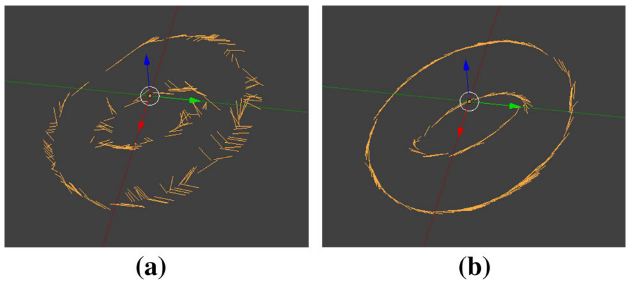

Fig. 10 Representation of 3D edgelet orientations of the torus rendered model for a given camera point of view with simple orientation on the left and with our proposed orientation on the right

Quantitative results are obtained by measuring the difference between the ground truth and the estimated camera positions. These localization errors are expressed as a percentage of the camera/object distance. Another measure used in these experimental results is an orientation error expressed in radians between the camera orientation and the ground truth one. A mean 2D error expressed in pixels is also computed and corresponds to the mean 2D distance between edgelets projected according to the estimated camera poses and edgelets projected according to the ground truth camera poses.

6.2 Evaluation of our DEC-SLAM framework

Evaluations of our DEC-SLAM framework through phDEC-SLAM solution concern the edgelet orientation definition (presented in Sect. 4.1.3), the sampling strategy (Sect. 4.1.4), the transfer time between GPU and CPU of edgelet information (Sect. 4.1.5), and the exploitation of a cost function with a barrier term as [17] in our constrained bundle adjustment to deal with erroneous constraint (Sect. 3). A first comparison with a model-based tracker algorithm similar to [41] is also performed to demonstrate the robustness and the stability of our solution when sudden motions occur. A second comparison with the rSEC-SLAM of [35] is achieved in order to evaluate the advantage of using dynamic edgelets instead of static ones (that are extracted offline and on sharp edges only). Finally, our DEC-SLAM algorithm is tested on several real sequences tracking different kinds of objects, to assess its genericity.

6.2.1 Edgelet orientation comparison

The 3D edgelet orientation described in Sect. 4.1.3 is compared with a simple orientation definition based on the 3D positions of two adjacent contour points of the rendered model in the image space.

The edgelet orientation is used in two processes of our DEC-SLAM algorithm: the 2D/3D matching step when edgelets are associated with image contours and then the camera pose estimation based on the minimization of the orthogonal distance between projected 3D edgelets and their associated 2D contours (for the re-projection formalism or the optimization of the output model-based poses for the hybrid constraint representations). Thus, the 3D orientations are first evaluated in a qualitative way; then, their impact on the rate of 2D/3D matches and on the accuracy of the camera position is evaluated.

Figure 10 shows the 3D orientations obtained for a simple object as the torus. Figure 10a corresponds to the simple 3D orientation, whereas Fig. 10b represents our 3D edgelet orientation for a given point of view. With our method, the 3D directions seem to correspond to the expected ones. On the contrary, with the simple definition, the 3D orientation is not well estimated because of the aliasing effect from the rasterization step of the rendering process.

Figure 11 presents the impact of our new 3D orientation definition. The evaluation is performed on the torus (Fig. 11a), the dragon (Fig. 11b) and the orthosis (Fig. 11c) sequences in order to confirm the genericity of our method. Our DEC-SLAM is compared with one with simple 3D edgelet orientation. The impact on the 2D/3D matching step is shown in Fig. 11d, e and f representing the rate of edgelets associated with image contours. Indeed, the accuracy of edgelet direction estimations has a direct impact on the matching between projected edgelets and image contours with a similar orientation. That is, the probability grows as the accuracy of the orientation increases. The 2D/3D matching rate is presented according to the number of extracted edgelets for each keyframe. For each sequence, the 2D/3D matching mean rate is higher for our solution with 82.2 against 45.0% for the torus sequence, 64.2 against 46.3% for the dragon sequence and 63.1 against 41.4% for the orthosis sequence. The gap between the two methods is more important when the object is curved, the aliasing effect being more significant.

Finally, the accuracy of our solution exploiting the new 3D edgelet orientation definition is compared with a DEC-SLAM with a simple edgelet orientation. It is evaluated through a position error. The quantitative results are presented in Fig. 11g, h and i. On the torus sequence, for both methods, the position error is important. The optimization process is indeed less constrained due to the object symmetry. However, our 3D edgelet orientation results in more accurate camera positions since the mean position error is around 9%, whereas the solution with the simple orientation definition has a mean position error of more than 16%. For the dragon sequence and the orthosis one, the tracking is also more accurate with the new 3D orientation. The mean position error is around 0.5% for the dragon sequence using the proposed 3D orientation definition, against 1.5% with the simple method. For the orthosis sequence, the object of inter-

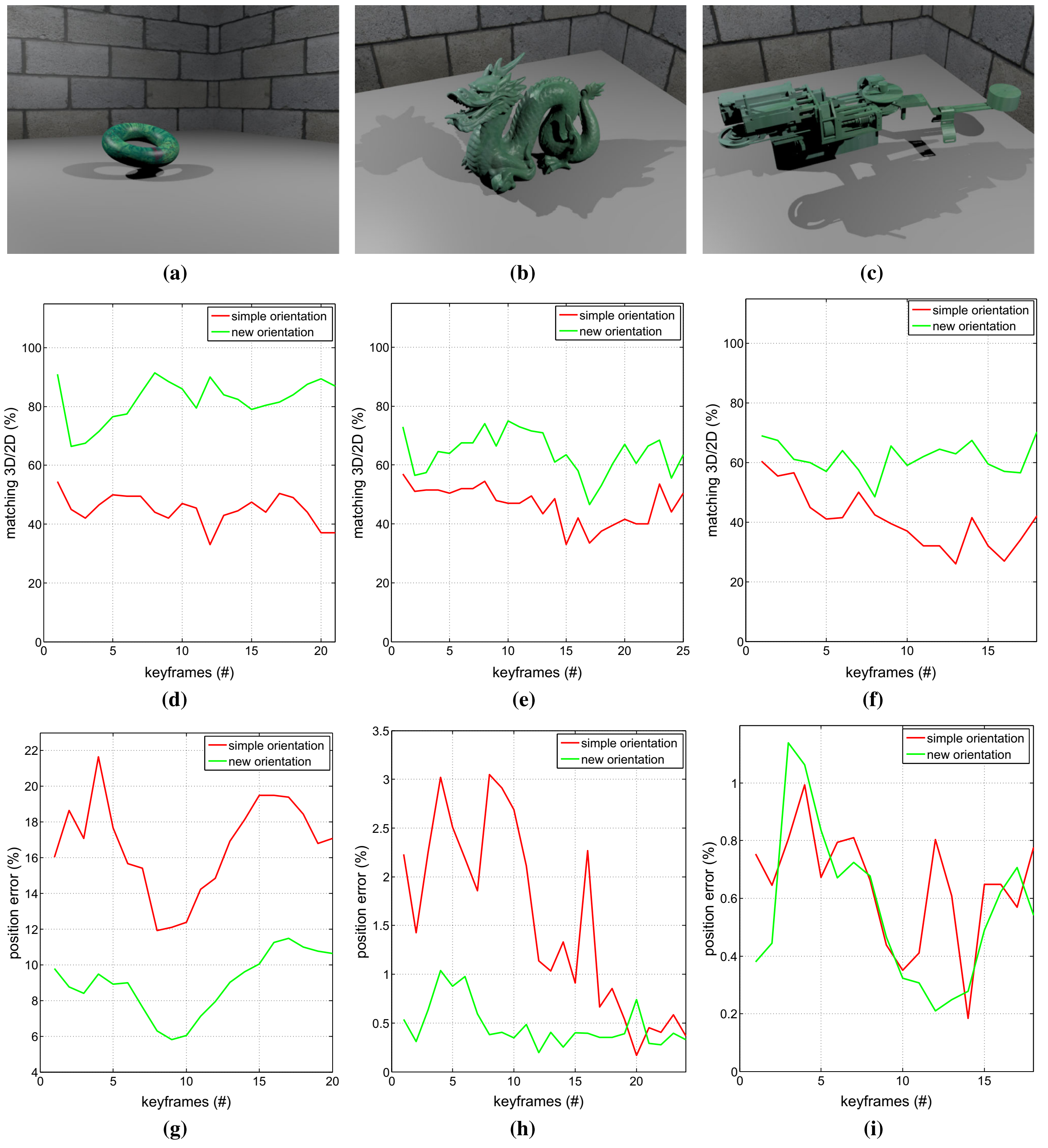

Fig. 11 Comparison of DEC-SLAM with simple 3D orientation definition and the solution we proposed that uses dynamic generated edgelets with new orientations. In green our solution exploiting the presented edgelet generation and in red the other method (color figure online)

est is polyhedral and the aliasing effect is less perceptible. Thus, both methods are accurate with a mean position error $< 1\%$.

The DEC-SLAM with the proposed 3D orientation allows a higher 2D/3D matching rate and a higher or similar accuracy than the one with simple edgelet orientation.

6.2.2 Sampling strategy comparison

The proposed sampling strategy described in Sect. 4.1.4 is compared with a random sampling. This evaluation is performed on the orthosis sequence where the proposed solution is ran a hundred times with the two sampling strategies (400 edgelets are used). The mean errors over the 100 trials and

Table 2 Localization errors (%) with different sampling strategies

	Min (%)	Max (%)	Mean (%)	SD (%)
Random sampling	0.9228	19.0420	8.4383	8.4943
Proposed sampling	0.5788	10.3170	3.5466	3.4458

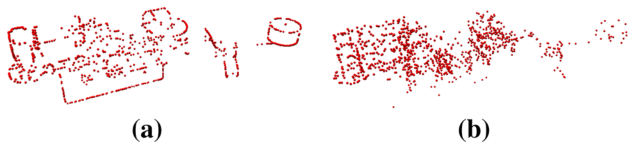

Fig. 12 Set of edgelets extracted by using our sampling method on the left and with a random sampling solution on the right

Table 3 Computation time for the edgelet data transfer between GPU and CPU. The comparison is made between a texture transfer and the proposed method

	SD sequence (ms)		HD sequence (ms)	
	Mean	SD	Mean	SD
Texture transfer	3.55	0.17	17.07	3.75
Proposed solution	0.56	0.10	1.03	0.23

over all the images of the orthosis sequence are estimated, as well as the min/max errors and the standard deviation (SD). Results are reported in Table 2. Our proposed sampling better constrains the matching and the pose estimation, halving the localization errors. Edgelets resulting of the different sampling methods are also shown in Fig. 12. Our sampling allows a more even distribution.

6.2.3 Edgelet data transfer time comparison

Our method for transferring edgelets from GPU to CPU, detailed in Sect. 4.1.5, is compared with a simple asynchronous texture transfer using pixel buffer object.

In the presented approach, the transfer begins with the synchronization between GPU and CPU for the data reading access. Then, the filtering pass follows, storing only the edgelet data. The transfer ends with the reading of the VBO containing edgelet information. With the simple texture transfer method, the transfer also begins with the synchronization between GPU and CPU. However, there is no filter performed and the transfer ends when all the textures containing edgelet information are read.

Table 3 presents the mean and standard deviation (SD) of transfer and reading times on the whole keyframes of the dragon sequence. Computation time is expressed in milliseconds. To evaluate the impact of our optimization, transfer computation time is evaluating on a standard-definition (SD) sequence with a 640 × 480 resolution and a high-definition (HD) sequence with a 1280 × 960 resolution.

Table 3 shows an important time saving since transfer time between the simple texture transfer and our proposed solution decreases over 84% for the SD video and 94% for the HD one. Moreover, we can see that the simple texture transfer approach is almost five times more time-consuming between the SD and the HD sequence, whereas our proposed transfer method is only almost twice more time-consuming. Edgelet transfer time with the simple texture transfer method depends directly on the sequence definition since all the pixels of all the textures are transferred to CPU. On the contrary, our method proposes to only transfer the edgelet pixels through an optimized array. Thus, the presented approach runs faster and is less affected by the sequence resolution.

6.2.4 Robustness to erroneous model constraint

In this experiment, the robustness of our solution exploiting inaccurate constraints is evaluated. Particularly, we seek to compare our constrained bundle adjustment using a cost function with a barrier term as [17] and an other one using a classical linear cost function optimizing simultaneously the environment and the model constraints as in [35]. The hybrid constraint may indeed be erroneous, especially if the output of the model-based tracker is inaccurate when the object is occluded or small in the image.

The experiment consists in occluding an object of interest to disturb the model-based tracking and obtain erroneous model-based poses for the constrained bundle adjustment. Figure 13 shows the position errors and 2D errors with our phDEC-SLAM approach and with a phDEC-SLAM constrained with a classical linear cost function. The two object trackers are compared on a synthetic sequence where a bypass is momentarily occluded by an orthosis as shown in Fig. 13a and b. When the bypass is occluded, the model-based pose can reach a position error of almost 22%. However, this erroneous constraint does not degrade the camera pose since after the optimization process exploiting Lhuillier cost function, the mean position error is $< 1\%$ (see Fig. 13c) and the mean 2D error is 1.56 pixels. However, with the use of a classical cost function, the localization is less accurate when the model-based pose is erroneous. The mean localization error is more than 2%, and the mean 2D error is around 2 pixels.

6.2.5 Comparison to a model-based tracking

We compare our DEC-SLAM with model-based tracking in terms of robustness against large displacements. The latter relies on edgelets extracted by using the proposed solution described in Sect. 4. For the matching step, the pose estimated on the previous image is used to project each edgelet and the nearest 2D image contour with a similar orientation is

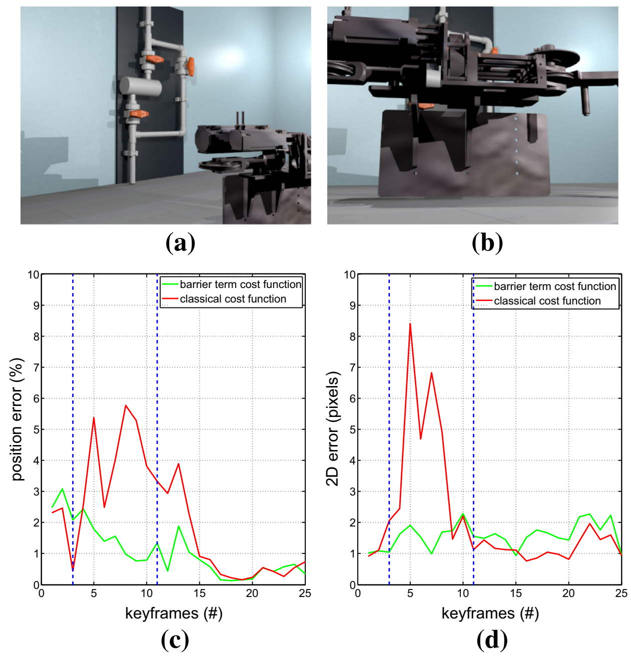

Fig. 13 Comparison between our cost function with a barrier term as [17] and classical linear cost function on a synthetic sequence. The bypass is the object of interest and is occluded by an orthosis in a part of the video. **a**, **b** Some frames of this sequence. **c** The position errors of our phDEC-SLAM with a classical cost function (red) or the cost function with a barrier term (green) in the constrained bundle process. The blue dots correspond to the interval where the bypass is occluded and the model-based poses are erroneous. **d** The 2D errors of both methods (color figure online)

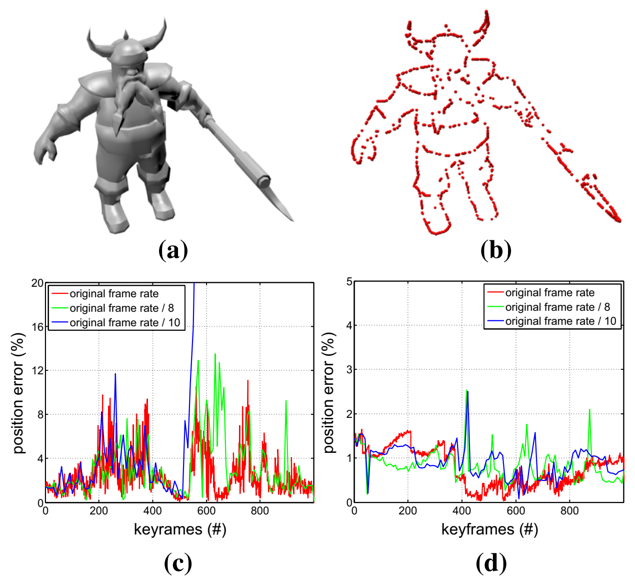

Fig. 14 Comparison of a model-based tracker and our DEC-SLAM solution on the dwarf sequence. **a** Illustration of the dwarf. **b** Edgelets extracted by using the solution described in Sect. 4 for a given camera pose. Localization errors resulting from model-based tracking (**c**) and from the proposed DEC-SLAM (**d**) on the dwarf sequence with the original frame rate (red), a frame rate divided by 8 (green) and by 10 (blue) (color figure online)

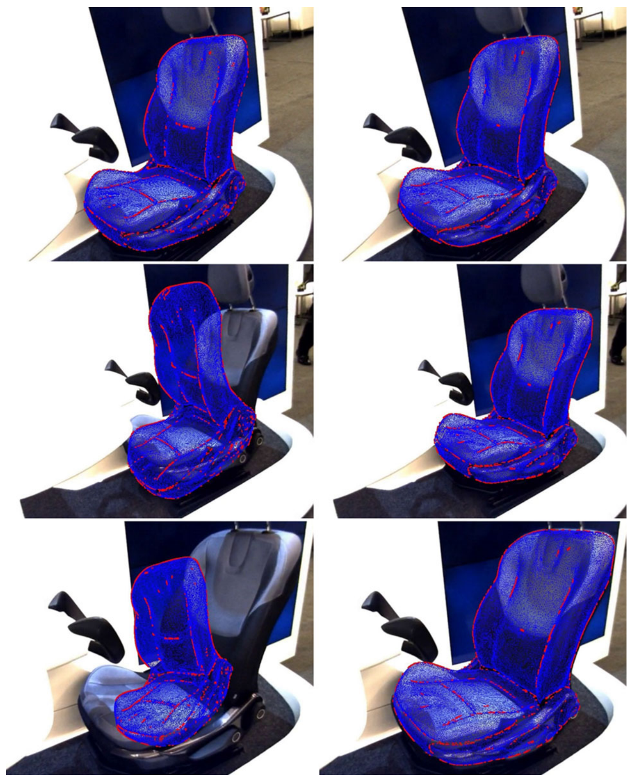

Fig. 15 Comparison of the proposed solution (right) with model-based tracking (left) on the car seat sequence where sudden motions occur

selected as its correspondent. The comparison is made on the dwarf synthetic sequence described in Sect. 6.1. The two algorithms are evaluated by varying the frame rate of the camera on the dwarf sequence. The travel speed of the camera is unchanged.

Three scenarios are tested: the original frame rate and frame rates divided by 8 and by 10. Figure 14c and d represents the position errors for the model-based tracking and the proposed solution, respectively. The DEC-SLAM succeeds on the three scenarios. The mean error stays in the interval [0.78%, 0.92%] and is almost constant for all frame rates. On the other hand, model-based tracking fails on the sequence with the lowest frame rate due to too large-amplitude motions. The mean position error increases with the displacement amplitude, and its values are 2.41 and 6.65% for the sequences with the original frame rate and with a frame rate divided by 8, respectively. Moreover, the standard deviation is important (2.76 vs 0.33% for our solution) due to tracking instabilities (jitter). The proposed solution is thus more accurate and more robust to large displacements than model-based tracking.

This robustness comparison is also made on a real sequence, as shown in Fig. 15. The tracked object corresponds to a car seat that is a curved object whose 3D model has 79K faces.

Thanks to the SLAM prediction based on keypoints, the generated edgelets at each keyframe are extracted from an accurate pose contrary to the model-based approach, which uses the pose estimated on the previous frame. Moreover, the pose predicted by the SLAM also facilitates the 3D/2D matching step, especially when large-amplitude motion occurs. Thus, our solution results in a stable (jitter-free) localization, while model-based tracking presents some localization instabilities and fails to track when the motion amplitude is large as illustrated in Fig. 15. Additional data are given in the first part of Online Resource 1, with a comparison video on the car seat sequence.

6.2.6 Comparison to rSEC-SLAM

We compare our proposed solution to the rSEC-SLAM algorithm described in [35] in terms of localization accuracy and tracking genericity. The dragon and the orthosis sequences are used for the evaluation. Both objects have sharp edges, but they are not predominant for the dragon. Figure 16a and b represents position errors on the two sequences. For the dragon sequence, the proposed solution is 5 times more accurate than the one proposed by Tamaazousti et al. [35]. In fact, the edgelets obtained with the dihedral criteria are not evenly distributed on the object, whereas the dynamic edgelets generated with the solution we proposed (see Sect. 4) better represents the silhouette and object contours. For the orthosis sequence, the original model and a simplified version are used. With the proposed solution, the localization errors are almost the same whatever the model complexity. On the other hand, the localization accuracy of the rSEC-SLAM is affected by the model complexity. Mean errors values are of 1.18 and 2.36% with the simplified and the original model, respectively. In fact, without model simplification, edgelet detection with the dihedral criteria results in a noisy constellation. The proposed solution does not require any model simplification to generate an evenly distributed edgelet constellation and thus to achieve accurate tracking.

Figure 17a and b illustrates on a real bypass sequence the same issue with a set of 2000 edgelets dynamically generated with the solution described in Sect. 4 and extracted by using a dihedral criteria, respectively. A constellation of edgelets obtained via the dihedral criteria is not evenly distributed, whereas the edgelets obtained with the proposed solution are distributed on the whole object surface, including the pipes.

The accuracy of EC-SLAM is directly affected by the quality of the edgelet constellation as seen in Fig. 17c and d on the real bypass sequence. The DEC-SLAM is more accurate than the rSEC-SLAM proposed in [35] since their edgelets constellation poorly constraints camera poses. Additional data are given in the second part of Online Resource 1, with a comparison video on the bypass sequence.

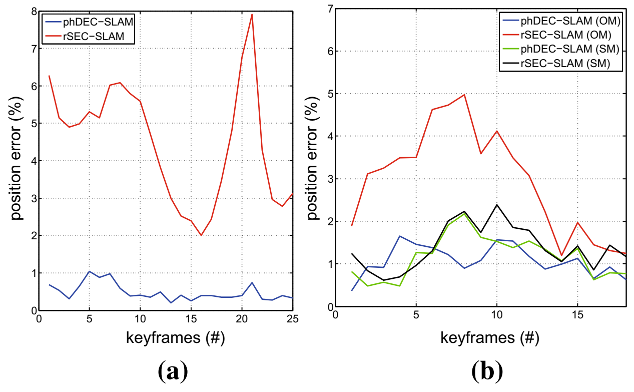

Fig. 16 Comparison of the rSEC-SLAM of [35] and our phDEC-SLAM, on the dragon (**a**) and the orthosis (**b**) sequences. Localization errors for the orthosis sequence are measured with the use of a simplified model (SM) and the original one (OM)

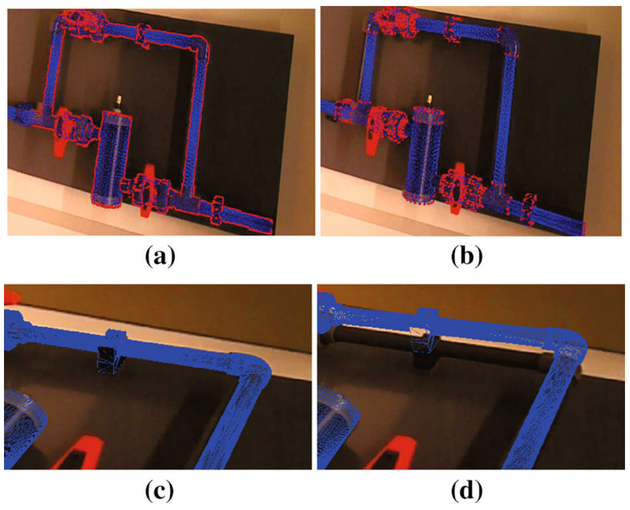

Fig. 17 Comparison between DEC-SLAM (left) and the rSEC-SLAM of [35] (right) on the bypass sequence. **a** Dynamic generated edgelets for that point of view. **b** Static edgelets generated with a dihedral criteria. **c**, **d** Model re-projections on a bypass portion

6.2.7 Tracking genericity evaluation

To finally assess the genericity of our DEC-SLAM method, experiments are carried on several real sequences with different kinds of objects. All of our proposed formalisms result in similar accuracy and robustness. Then, Fig. 26 shows 3D model projections on the real scene according to the estimated camera poses with phDEC-SLAM solution only.

Our proposed solution has been successfully tested with objects for which a 3D model is available, for example, with a sport car (Fig. 26g), a bypass from the chemical industry

Table 4 Computation time in milliseconds of the mapping process for the rhDEC-SLAM, phDEC-SLAM and rSEC-SLAM on the sport car sequence. It is given for 3, 10 and 30 optimized keyframes

	Polyhedral object (sport car)		
	rSEC-SLAM [35]	rhDEC-SLAM	phDEC-SLAM
3 Optimized keyframes	47.9	27.9	23.6
10 Optimized keyframes	130.5	77.9	76.2
30 Optimized keyframes	216.6	203.9	188.7

Table 5 Computation time in milliseconds of the mapping process for the rDEC-SLAM, rhDEC-SLAM and phDEC-SLAM on the dragon sequence. It is given for 3, 10 and 30 optimized keyframes

	Curved object (dragon statue)		
	rDEC-SLAM	rhDEC-SLAM	phDEC-SLAM
3 Optimized keyframes	97.8	94.8	94.1
10 Optimized keyframes	172.1	170.6	130.3
30 Optimized keyframes	385.3	330.0	261.3

(Fig. 17) and a seat from a car OEM (Fig. 15). The genericity of our solution is also demonstrated by tracking objects like a metal statue of dragon (Fig. 26d), and a Raving Rabbid (Fig. 26a) using 3D models reconstructed by photogrammetry. Industrial objects like a real-sized car (Fig. 26j), a car cylinder head (Fig. 26m), or an other kind of orthosis (Fig. 26p) with a known 3D model have been tracked successfully.

These objects might be polyhedral (the sport car, the cylinder head or the orthosis) or curved (the Raving Rabbid, the dragon or the car seat), but they also can present sharp and occluding contours at the same time (the bypass or the real-sized car). Some are textured (the dragon), while others are absolutely textureless (the Raving Rabbid).

Our solution is robust to occlusions, thanks to the proposed DEC-SLAM framework, which guarantees the multi-view relationships to be well estimated. Additional data are given in the third part of Online Resource 1, with a compilation video that shows our tracking solution on several polyhedral and curved objects.

6.3 EC-SLAM comparison

The different formalisms of the presented solution are compared in terms of computational cost, memory consumption and accuracy in this section.

6.3.1 Computation time evaluation

In this subsection, the different model constraint formalisms integrated in our DEC-SLAM framework are evaluated in terms of computational costs. Since these solutions differ only by their bundle adjustment, this experiment consists in comparing the processing time required by the mapping process described in Sect. 3. The mapping process is achieved once per new keyframe; then, we compare the median processing time of these executions. Also, since the computation time depends on the number of optimized keyframes, the different solutions are compared for a set of 3, 10 and 30 optimized keyframes.

This experiment is achieved on two real sequences recorded with a HD resolution (1280×1024). The two objects used are a metal statue of dragon (see Fig. 26d) with a 3D model reconstructed by photogrammetry and a sport car (see Fig. 26g), whose the 3D model is available. The SLAM initialization on these real videos is performed manually and followed by a contour refinement process.

The sport car illustrates the performances for a polyhedral object, while the dragon sequence illustrates the performances for a curved object. For the car sequence, the performances of our rhDEC-SLAM and phDEC-SLAM are compared with the performances of rSEC-SLAM of [35]. For the dragon sequence, the rDEC-SLAM, rhDEC-SLAM and phDEC-SLAM are compared in terms of computational cost.

The computation time of the mapping processes is presented in Tables 4 and 5. The classical bundle adjustment optimizing the multi-view constraint and the dynamic edgelet generation (and optimal output model-based poses for both hybrid constraint representations) are running in parallel, to optimize the computation time. Moreover, even if rDEC-SLAM and rSEC-SLAM have not the model-based pose refinement step in their framework (contrary to the rhDEC-SLAM and phDEC-SLAM), the 2D/3D associations between edgelets and their 2D counterparts have to be re-estimated at each constrained bundle adjustment on all the optimized keyframes. This process participates in slowing down its execution. Computation time is globally more important with the tracking of curved objects. It is due to the edgelet generation process, which has to be performed online at the last keyframe since the occluding contours depend on the camera point of

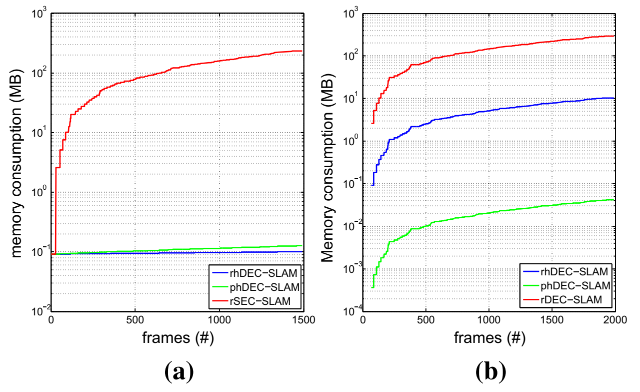

Fig. 18 The memory footprint associated with the car sequence (**a**) and the dragon one (**b**) is drastically decreased with the use of the rhDEC-SLAM and phDEC-SLAM (see Sect. 5) compared with rSEC-SLAM of [35] or our rDEC-SLAM. The memory consumption is expressed in MB with a logarithmic scale on the y axis

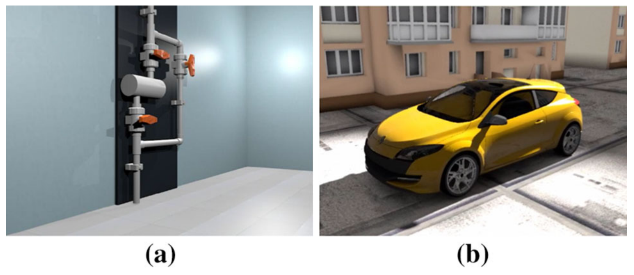

Fig. 19 Images from the bypass (**a**) and sedan car (**b**) synthetic sequences

view (see Sect. 3.1). On the contrary, the edgelet extraction is performed offline only once for polyhedral objects. Tables 4 and 5 show that the exploitation of any hybrid constraint representation allows to reduce computation time. Compared with the use of re-projection formalism, the computation time decreases around 44% for the sport car sequence and around 8% on the dragon one even if a model-based pose refinement is performed in addition to the constrained bundle adjustment with the use of hybrid constraint (with 3 and 10 keyframes in the bundle adjustment). The computation time decreases around 10 and 23% with the use of 30 keyframes.

6.3.2 Memory consumption comparison

Optimizing a SLAM reconstruction with a bundle adjustment constrained to a 3D model implies the storage of additional data for each keyframe. In this section, the memory footprint of these data is evaluated, depending on the use of the chosen model constraint formalism and the edgelet generation process (static or dynamic). These evaluations are conducted on the two real sequences introduced in Sect. 6.3.1. The edgelet extraction is performed offline for the car sequence and online for the dragon statue. Our rhDEC-SLAM and phDEC-SLAM are compared with [35] for the sport car and with our rDEC-SLAM for the dragon. The results are summarized in Fig. 18a and b.

As explained in Sect. 3.2, rDEC-SLAM or rSEC-SLAM require to store edgelets and contour images for each keyframe optimized in the constrained bundle adjustment. Consequently, the memory footprint increases every time a new keyframe is created. The impact is even greater as the video resolution (for curved objects) and the number of edgelets are high. For the real sequences, the number of edgelets projected and associated with 2D contour points is set to 2000, and 94 (respectively, 113) keyframes are created over the car sequence (respectively, dragon sequence).

Thus, as shown in Fig. 18a and b, the memory footprint for rSEC-SLAM is near 235 MB at the end of the sport car sequence and more than 292 MB for the rDEC-SLAM at the end of the dragon one. The latter sequence requires more memory since edgelets extracted from occluding contours are stored for each keyframe, which is not required with polyhedral objects. The memory footprint drastically decreases with the use of a hybrid constraint formalisms, since contour images are not required anymore in the constrained bundle adjustment. Pose parameters of the model-based tracker outputs are henceforth stored instead. In addition to the pose parameters storage, our rhDEC-SLAM requires to save sets of edgelets. Since they are extracted only once with polyhedral object, the memory consumption is limited to 0.1 MB at the end of the car sequence. However, generating and accumulating sets of edgelets at each keyframe for curved objects increase the memory footprint to 10 MB at the end of the dragon sequence. Finally the phDEC-SLAM allows a memory footprint close to 0.12 MB (respectively, 0.04 MB) at the end of the sport car sequence (respectively, dragon sequence), only pose parameters and matrices W^c of the model-based tracker outputs (see Sect. 5.3) being stored. We can notice that with polyhedral objects, the memory consumption is higher for the phDEC-SLAM (0.12 MB) than for the rhDEC-SLAM (0.10 MB). Indeed, in addition to the pose parameters stored for both hybrid constraint representations, for the rhDEC-SLAM, a unique set of edgelets is stored for the whole sequence, whereas for the phDEC-SLAM, matrices W_j^c are stored at each keyframe creation.

The latter allows our memory consumption to be invariant to the number of edgelets and their generation, invariant to the resolution of the tracking video. Thus, if polyhedral (respectively, curved) objects are tracked, the rhDEC-SLAM (respectively, phDEC-SLAM) is more suitable. Additional data are given in the fourth part of Online Resource 1, with a comparison video on the sport car and dragon sequences.

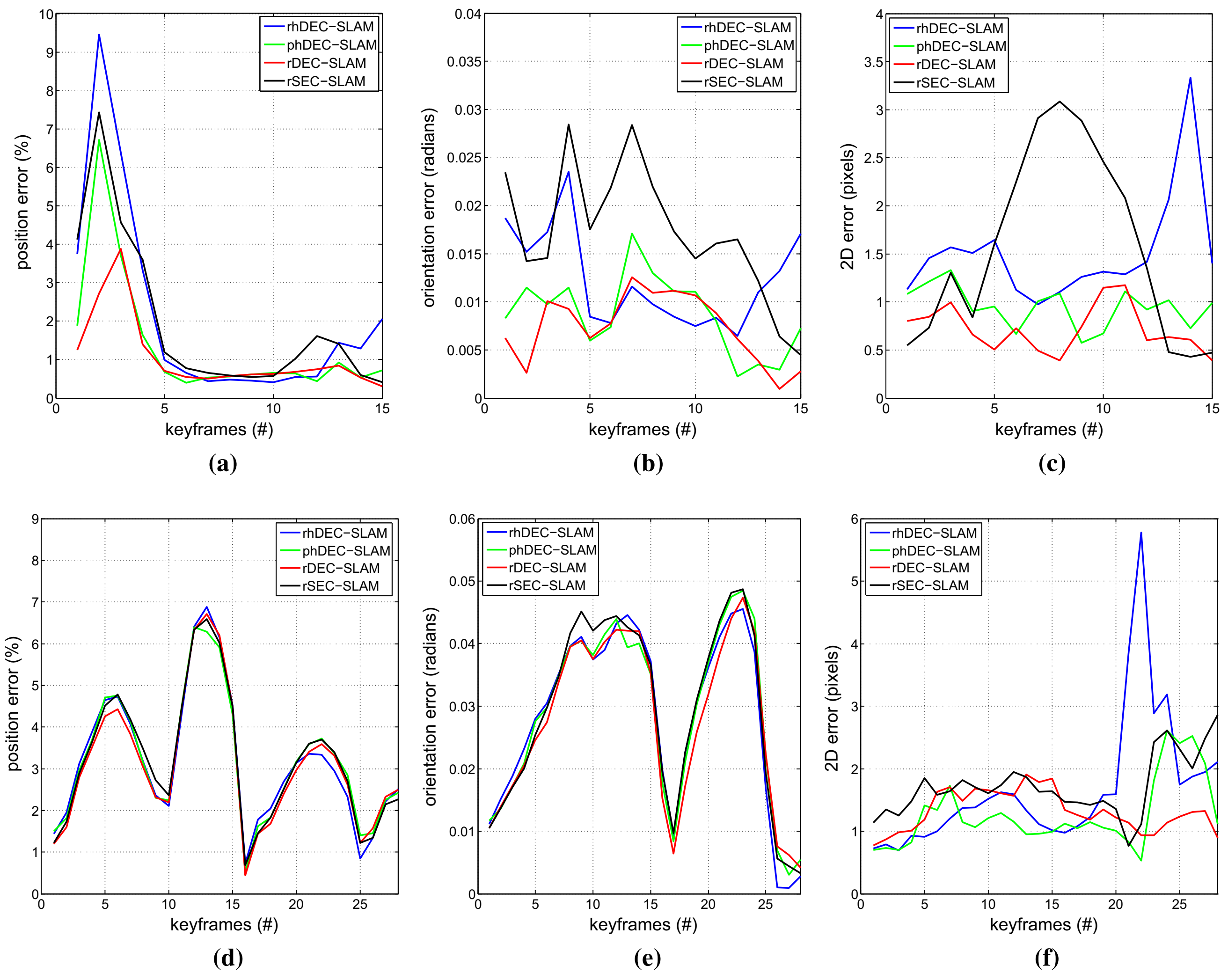

Fig. 20 Accuracy comparison between static edgelet cSLAM [35] (black), dynamic edgelet cSLAM (red) and the first hybrid (blue) and the second (green) cSLAM on the bypass (top) and sedan car (bottom) sequences (color figure online)

6.3.3 Accuracy evaluation

Our proposed DEC-SLAM algorithm is evaluated in terms of accuracy and robustness on polyhedral and curved objects and also compared with rSEC-SLAM described in [35].

The accuracy of our DEC-SLAM solution is evaluated on two synthetic sequences. The first one corresponds to the bypass sequence (see Fig. 19a) presented in Sect. 6.1. The second sequence has a sedan car as object of interest. Its 3D model has 50K faces. The sedan car environment is composed of city buildings as seen in Fig. 19b.

Figure 20a, b and c presents the localization error in position and orientation, and the 2D error on the bypass sequence. Figure 20d, e and f presents the same errors on the sedan car sequence.

The rDEC-SLAM and the phDEC-SLAM provide similar accuracies on both polyhedral and curved objects, contrary to the rhDEC-SLAM that presents some lacks of accuracy. On the sedan sequence, the median pixel error is 1.38 for the rhDEC-SLAM, 1.12 for the phDEC-SLAM, 1.28 for the rDEC-SLAM and 1.63 for the rSEC-SLAM of [35]. On the bypass sequence, the median pixel error is 1.39 for the rhDEC-SLAM, 0.99 for the phDEC-SLAM, 0.91 for the rDEC-SLAM and 0.66 for the rSEC-SLAM of [35]. This latter is less accurate when the object of interest is curved. Edgelets obtained with the dihedral criteria are not evenly distributed on the object, whereas the dynamic edgelets generated with the proposed solution better represent the silhouette and object contours.

6.4 Evaluation on public datasets

To assess the robustness and accuracy of our DEC-SLAM approach, we run our method on two public datasets. Even if these benchmarks are more adapted for RGBD solutions, EC-SLAM are able to localize accurately the camera without depth information. The SLAM initialization on these sequences is obtained by the first ground truth pose.

6.4.1 CoRBS dataset

Our DEC-SLAM solution is evaluated and compared on the Comprehensive RGBD Benchmark for SLAM (CoRBS) dataset that proposes real sequences with different objects of interest. It provides the real depth and RGB frames thanks to the use of a Kinect v2, with a ground truth trajectory of the camera and a ground truth 3D model of the scene. The CoRBS dataset is composed of 5 sequences named kt0, kt1, kt2, kt3 and kt4 for each object of interest (a desk and a wooden human manikin). An overview of these two objects with their reconstructed 3D models is presented in Fig. 21. This benchmark is interesting to evaluate our DEC-SLAM with the different model constraint formalisms, since the tracked objects are polyhedral and complex like the desk, or curved as the wooden human manikin. The sequences have a 640 × 480 definition and a frequency of 30Hz.

We evaluate our rDEC-SLAM, rhDEC-SLAM and phDEC-SLAM solutions and compare it to RGBD C-SLAM [22] (only the kt0 sequence is available for this approach) and rSEC-SLAM [35]. Table 6 describes the absolute trajectory error (ATE) inspired by [34] and expressed through the root-mean-square-error metric (RMSE). This error quantifies the accuracy of the entire trajectory for all the five desk sequences. The rDEC-SLAM solution presents the best results in terms of accuracy for the first three sequences and is very close of the most accurate approach on the last two sequences. For the kt0 one, RGBD C-SLAM has the same RMSE of 0.013 m as our rDEC-SLAM. However, the other model constraints formalisms integrated into our DEC-SLAM solution provide also accurate localization with a trajectory error around 0.020 m for this sequence. On this polyhedral object, rSEC-SLAM of [35] reaches good performances, especially for the kt1 and kt3 sequences where it gets the smallest trajectory error. Our rhDEC-SLAM and phDEC-SLAM also track with accuracy the desk with similar localization error. The first one has the best result for the kt1 and kt4 sequences, the second one following just near by. This latter has the best RMSE for the kt2 sequences.

Table 7 describes the localization error for all the five human manikin sequences. The object of interest is a curved one. Then, rSEC-SLAM [35] and RGBD C-SLAM are less accurate, few sharp edges existing on the 3D model. Edgelets extracted by using the dihedral criteria are not well distributed on the model and do not well constrain the camera pose estimation. The rDEC-SLAM approach obtains the best trajectory errors except for the kt1 sequence where the phDEC-SLAM precedes it with a RMSE of 0.146 m.

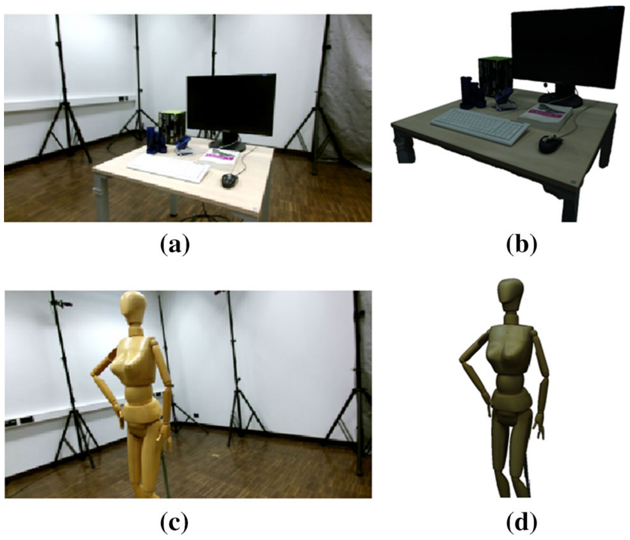

Fig. 21 CoRBS sequences [40]. Frames from the different sequences where the desk and human manikin are tracked (left). Their associated reconstructed 3D model (right)

Table 6 Trajectory error for the CoRBS Desk sequences [40]

Desk sequence					
Error (m)	kt0	kt1	kt2	kt3	kt4
RGBD C-SLAM [22]					
RMSE	**0.013**	–	–	–	–
SD	0.005	–	–	–	–
Min	–	–	–	–	–
Max	0.073	–	–	–	–
rSEC-SLAM [35]					
RMSE	0.020	**0.010**	0.601	**0.287**	0.354
SD	0.006	0.003	0.238	0.126	0.256
Min	0.006	0.001	0.047	0.053	0.017
Max	0.048	0.026	1.011	0.841	1.244
rDEC-SLAM					
RMSE	**0.013**	**0.010**	**0.600**	0.294	0.360
SD	0.005	0.004	0.238	0.128	0.257
Min	0.003	0.001	0.042	0.061	0.017
Max	0.041	0.025	1.007	0.858	1.246
rhDEC-SLAM					
RMSE	0.020	**0.010**	0.603	0.288	**0.351**
SD	0.006	0.004	0.238	0.126	0.257
Min	0.006	0.001	0.053	0.055	0.016
Max	0.049	0.025	1.016	0.854	1.260
phDEC-SLAM					
RMSE	0.021	0.011	**0.600**	0.293	0.358
SD	0.012	0.004	0.237	0.129	0.254
Min	0.001	0.001	0.042	0.066	0.026
Max	0.063	0.030	1.003	0.854	1.222

Bold indicates the best RMSE (root-mean-square-error metric) is reached compared with the other SLAM methods

Table 7 Trajectory error for the CoRBS Human manikin sequences [40]

Human sequence					
Error (m)	kt0	kt1	kt2	kt3	kt4
RGBD C-SLAM [22]					
RMSE	0.036	–	–	–	–
SD	0.017	–	–	–	–
Min	–	–	–	–	–
Max	0.208	–	–	–	–
rSEC-SLAM [35]					
RMSE	0.032	0.223	0.077	0.740	0.147
SD	0.014	0.120	0.047	0.349	0.074
Min	0.002	0.015	0.006	0.052	0.017
Max	0.067	0.619	0.218	1.572	0.472
rDEC-SLAM					
RMSE	**0.019**	0.158	**0.027**	**0.510**	**0.086**
SD	0.008	0.087	0.011	0.252	0.048
Min	0.001	0.016	0.003	0.024	0.005
Max	0.055	0.440	0.059	1.282	0.248
rhDEC-SLAM					
RMSE	0.046	0.150	0.062	0.760	0.272
SD	0.021	0.072	0.032	0.308	0.076
Min	0.003	0.026	0.032	0.141	0.130
Max	0.095	0.317	0.174	1.524	0.475
phDEC-SLAM					
RMSE	0.030	**0.146**	0.029	0.516	0.127
SD	0.013	0.073	0.012	0.252	0.074
Min	0.003	0.011	0.002	0.043	0.012
Max	0.068	0.290	0.070	1.310	0.431

Bold indicates the best RMSE (root-mean-square-error metric) is reached compared with the other SLAM methods

The proposed DEC-SLAM method obtains accurate results on CoRBS sequences with all the model constraint formalisms, the objects of interest being polyhedral or curved.

6.4.2 ICL-NUIM dataset

The Imperial College London and National University of Ireland Maynooth (ICL-NUIM) dataset [9] is a benchmark created for a different context than object tracking evaluation. The scene indeed corresponds to a living room as seen in Fig. 22a, a much larger object of interest than the ones we exploit previously. However, even if this dataset is out of our context and our DEC-SLAM is not necessarily well adapted for this kind of scene when the object of interest is too large for the camera field of view, it is able to run on these sequences. The 3D model of the scene is indeed available (Fig. 22b). The ICL-NUIM sequences have a 640 × 480 resolution and are

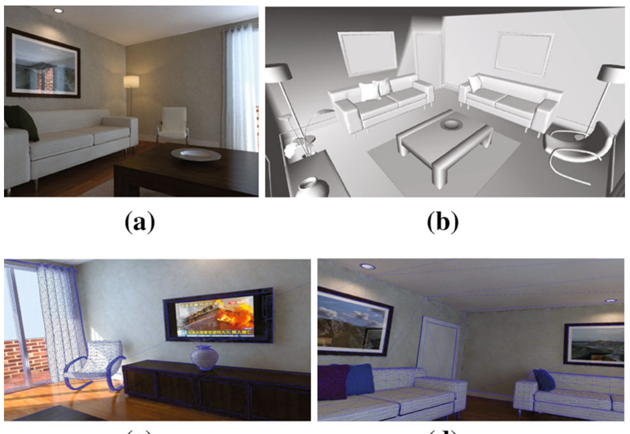

Fig. 22 ICL-NUIM sequences [9]. **a** A frame from the kt1 sequence. **b** The 3D model of the living room used as a constraint in our solution. **c, d** 3D model projected on the kt1 and kt2 sequences with the rDEC-SLAM

recorded at 30 Hz. Images are synthetic but with real-world lightning conditions. The sequences are also noisy according to an RGB noise model to simulate the one providing of real camera.

Table 8 describes the trajectory accuracy expressed with the RMSE metric of [34] as for the CoRBS dataset, for the living room sequences kt1 and kt2. We compare our rDEC-SLAM, rhDEC-SLAM and phDEC-SLAM to the rSEC-SLAM of [35] and RGBD SLAM approaches such as DVO [12], FOVIS [11], ICP of KinectFusion [26] and finally, RGBD C-SLAM [22].

Trajectory errors for the kt0 and kt3 living room sequences are not presented in this paper. Our DEC-SLAM solution does not exploit RGBD frames and is not able to estimate the camera pose when this latter is looking at a planar textureless region of the scene (walls) as it happens with these both sequences (see Fig. 23).

However, accurate results are obtained for the kt1 and kt2 sequences with our DEC-SLAM and the different model constraint formalisms. On kt1, the rDEC-SLAM and the phDEC-SLAM have the second best trajectory error with a RMSE of 0.018 m behind ICP solution [26] (RMSE equal to 0.005 m). On kt2, the rDEC-SLAM has the best trajectory error with the ICP approach and the rSEC-SLAM [35] that corresponds to a RMSE equal to 0.010 m. Our phDEC-SLAM as the second best trajectory accuracy with a RMSE of 0.011 m.

Even if we do not exploit depth information, our DEC-SLAM approach presents similar accurate results compared with [26] and superior results to [11,12,22]. Moreover, the mean 2D error of our DEC-SLAM on both sequences is around 4 pixels, which is small enough for RA application. The 3D model of the scene is globally well re-projected on the images as seen in Fig. 22c and d.

Table 8 Trajectory error for the kt1 and kt2 ICL-NUIM sequences [9]

Error (m)	kt1	kt2
DVO [12]		
RMSE	0.125	0.473
SD	0.037	0.175
Min	0.051	0.137
Max	0.200	0.834
FOVIS [11]		
RMSE	1.868	1.495
SD	0.871	0.504
Min	0.333	0.270
Max	3.039	2.773
ICP [26]		
RMSE	**0.005**	**0.010**
SD	0.002	0.004
Min	0.001	0.004
Max	0.011	0.015
RGBD C-SLAM [22]		
RMSE	0.025	0.023
SD	0.015	0.011
Min	–	–
Max	0.087	0.093
rSEC-SLAM [35]		
RMSE	0.024	**0.010**
SD	0.011	0.004
Min	0.003	0.001
Max	0.089	0.025
rDEC-SLAM		
RMSE	0.018	**0.010**
SD	0.013	0.004
Min	0.000	0.001
Max	0.094	0.025
rhDEC-SLAM		
RMSE	0.068	0.048
SD	0.038	0.017
Min	0.001	0.005
Max	0.157	0.077
phDEC-SLAM		
RMSE	0.018	0.011
SD	0.012	0.005
Min	0.001	0.001
Max	0.091	0.034

Bold indicates the best RMSE (root-mean-square-error metric) is reached compared with the other SLAM methods

Fig. 23 Our DEC-SLAM is not able to estimate camera poses when any model information or environment keypoints are available to constrain our method. Without depth information, our solution fails when the camera is looking at planar textureless regions. **a**, **b** Frames from the ICL-NUIM kt0 sequence [9]. The tracking is lost when only walls are visible

7 Application

DEC-SLAM is useful to Augmented Reality (AR) applications for maintenance support, automation of complex tasks or other quality controls. Our method is able to localize complex static objects, but also objects with simple dynamic parts thanks to dynamic edgelet generation (see Sect. 4). Thus, more complex scenarios can be proposed to guide people with AR applications. In the industry settings, objects of interest may be composed of several parts, curved or not. For example, in the petrochemical industry and in a maintenance support scenario, a valve may be open or closed during the sequence, or in a part assembly scenario, elements may be piled up successively. In these cases, the objects of interest have different states. To give suitable information, the AR application has then to localize these object parts according to the camera pose and their current state.

7.1 Automatic object part state estimation

s

In order to estimate the state of a movable object part, the idea is to exploit 3D contours extracted from the 3D model as presented in Sect. 4.

However, it is not enough to only compare the rate of 2D/3D matching between projected edgelets and image contours for each potential state to estimate the current one.

Projected edgelets and 2D image contours are not perfectly aligned when the model is rendered in the current state (see Fig. 24). The camera pose may be inaccurate, and the 3D model may not be identical to the object of interest, particularly when it is created by photogrammetry. The edgelet extraction and the 2D contour detection may also be inaccurate. The object state estimation is then not obvious. In order to correctly estimate which edgelets from the different rendered models correspond to the current state, and thus to propose a robust state estimation, these contours alignments issues have to be taken into account.

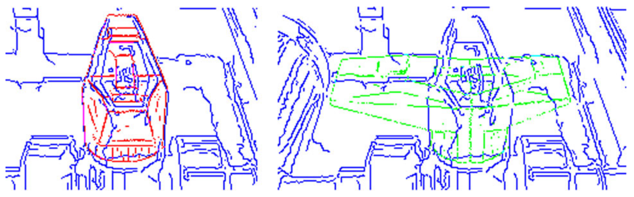
(a) (b)

Fig. 24 Superimposition between projected edgelets from the current state (red) and 2D image contours (blue) on the left, and between projected edgelets from a false potential state (green) and 2D contours on the right. To estimate the accurate state of the valve, false-positive association as between edgelets from the potential state and the static part of the object have to be considered (color figure online)

Table 9 Weights for the state estimation

	Other potential states		
2D/3D matching	Same orientation	No match	Wrong orientation
One potential state			
Same orientation	0	+	+
No match	–	0	+
Wrong orientation	–	–	0

Our state estimation approach is based on a score for each potential state (open, closed or half open, for example, in the case of a valve), computed as a weighted sum of 2D/3D matches between edgelets and 2D contours. The highest score defines the potential state as the accurate current state. Since some associations must be irrelevant and disturb the state estimation, our method does not seek to match an edgelet to a contour pixel but the opposite. It aims to associate each 2D contour pixel to edgelets extracted from the different rendered models based on the potential states. In a region of interest (ROI), 2D contours of the static part as well as the movable one are analyzed in order to help the state estimation. The 2D/3D matching step is performed between a contour pixel and edgelets from all the potential model renders in the ROI. The resulting association states are defined through a weight presented in Table 9 that follows.

Three different states can result from pixel/edgelet associations. The first possibility is that no match is found. The remaining two cases happen in the event of successful matching, depending on whether or not the orientation between the edgelet and the contour agrees. The weight according to a 2D/3D association between the given pixel and an edgelet from a potential state depends on the association of this same pixel with edgelets from the other potential states. The score of each potential state corresponds to the sum of all the 2D/3D correspondences weighted according to their status for every 2D contour pixels. Thus, a pixel with a 2D/3D association having the same status for each potential state is not relevant to estimate whether a state is actually the accurate current state. Its weight is then null. A 2D/3D correspondence with the same orientation between a pixel and a projected edgelet will have a positive weight if it exists only for one potential state. It will have a negative weight for the other potential states where it does not exist. Table 9 also shows that if a noisy contour pixel has no match for a potential state but has a 2D/3D association with a wrong orientation on the other potential states, this correspondence is relevant and a positive weight will be given for this no match.

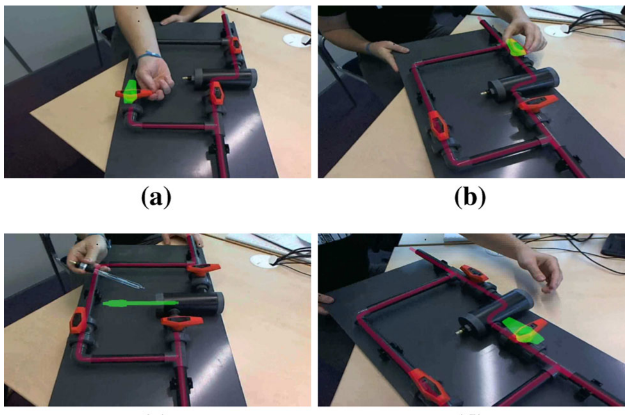
(a) (b) (c) (d)

Fig. 25 Maintenance support on a bypass system with AR effect displays

7.2 Integration into an AR application

In an AR application, state estimation is performed during a live sequence in addition to the other tasks as the camera localization and the display of AR effects. It is also important to have a real-time running application for user comfort.

To display virtual information in the AR application, the camera pose has to be known. This camera localization is also in our case needed to estimate the state of the movable parts. The AR application is running on a multi-thread architecture with a thread for the localization process corresponding to our DEC-SLAM, an other one for the display and a last one for the state estimation.

The presented application aims to guide a user in order to replace a pH sensor. The sensor is integrated into a bypass system. This device allows to deviate the liquid contained into the pipes in order to change the pH sensor without leak. The maintenance support scenario is as follows:

- Open the bypass in order to let the liquid circulate.
- Close the entrance of the main pipe to deviate the fluid.
- Close the pipe exit.
- Change the pH sensor located in the main pipe.
- Open the main pipe exit.
- Open the main pipe entrance.
- Close the bypass.

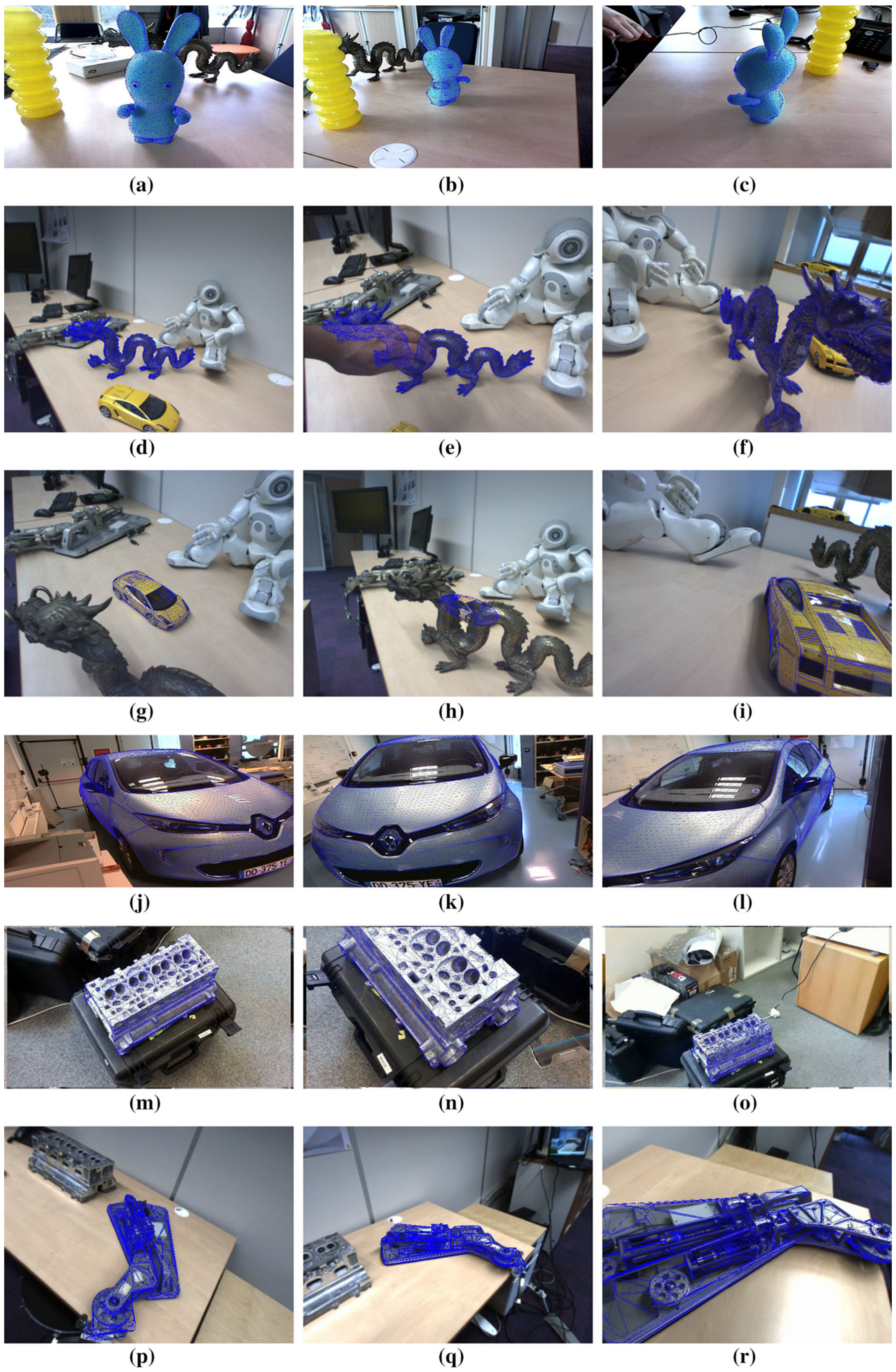

Fig. 26 Accurate localization results on a Raving Rabbid, a dragon, a sport car, a real-sized car, a car cylinder head, and an orthosis with our phDEC-SLAM. Results are similar with rDEC-SLAM and rhDEC-SLAM. The accuracy can be appreciated by the projection of the 3D blue models on the object of interest in the images (color figure online)

The application has to detect the bypass state (open/close), validate automatically the different steps of the scenario and show the next move to the user. The application user can see through a tablet screen different AR effects like the fluid circulation according to the valve state in red, or actions to do on the valves and the pH sensor in green (see Fig. 25). Additional data are given in the fifth part of Online Resource 1.

8 Conclusion and perspectives

In this article, a real-time solution for camera localization relative to polyhedral and curved industrial object is proposed. A keyframe-based SLAM algorithm is presented with a model constraint improving the tracking accuracy.

This constraint is enforced through the integration of dynamic edgelets extracted by rendering on the graphic hardware. They are sampled to guarantee homogeneous spatial and angular distributions and to prevent the inclusion of ambiguous edgelets in matching steps of the DEC-SLAM algorithm. Two different formalisms in addition to the reprojection error representation are proposed to integrate the model constraint in the optimization process. They correspond to hybrid model/trajectory constraint expressions that exploit the output of a model-based tracker, optimized according to the dynamic edgelets. They reduce the memory footprint of the DEC-SLAM while achieving similar accuracy. Our solution has been tested on several polyhedral and curved objects to attest its genericity. Experimental results demonstrate that our solution is as robust and accurate as rSEC-SLAM [35] exploiting static edgelets extracted offline from sharp model edges and using a re-projection error formalism. In addition, the hybrid constraint representations are able to drastically reduce the memory consumption and the computation time. Our experiments demonstrate high accuracy that enables convincing Augmented Reality applications (Fig. 26).

As further work, we aim to focus on larger object to track as power plants or pipelines, or more rooms with available 3D models as with the ICL-NUIM dataset. Dealing with big objects may be indeed a challenging task. Whereas our DEC-SLAM solution proposes an accurate localization with local optimization when tracked objects are relatively small, it is not necessarily the case when objects are too large for the camera field of view. Potentially significant drifts over time may occur because of not well constrained DoF. Localization error may then accumulate itself, needing local and global corrections. However, with the use of our hybrid constraint formalisms light in memory consumption and with the use of a model-based tracking of the large object of interest, a global optimization of the trajectory may be possible when localization error occurs.

Acknowledgements This work was partly funded by the french research program FUI through the projects NASIMA and SEEMAKE. The authors would also like to thank their project partners Diotasoft and Faurecia for providing the car seat sequence.

References

1. Bleser, G., Stricker, D.: Advanced tracking through efficient image processing and visual-inertial sensor fusion. Comput. Graph. **3**(1), 59–72 (2009)
2. Bleser, G., Wuest, H., Strieker, D.: Online camera pose estimation in partially known and dynamic scenes. In: International Symposium on Mixed and Augmented Reality (2006)
3. Drummond, T.: Lie groups, lie algebras, projective geometry and optimization for 3D geometry, engineering and computer vision. https://www.dropbox.com/s/5y3tvypzps59s29/3DGeometry.pdf?dl=0 (2014). Accessed 26 June 2018
4. Drummond, T., Cipolla, R.: Real-time visual tracking of complex structures. PAMI **24**(7), 932–946 (2002)
5. Eade, E.: Lie groups for computer vision. http://ethaneade.com (2014). Accessed 26 June 2018
6. Finsterle, S., Kowalsky, M.: A truncated Levenberg Marquardt algorithm for the calibration of highly parameterized nonlinear models. Comput. Geosci. **37**(6), 731–738 (2011)
7. Gay-Bellile, G., Bourgeois, S., Tamaazousti, M., Naudet-Collette, S., Knodel, S.: A mobile markerless augmented reality system for the automotive field. In: International Symposium on Mixed and Augmented Reality Workshop (2012)
8. Hajagos, B., Szcsi, L., Csbfalvi, B.: Fast silhouette and crease edge synthesis with geometry shaders. In: Spring Conference on Computer Graphics (2013)
9. Handa, A., Whelan, T., McDonald, J., Davison, A.: A benchmark for RGB-D visual odometry, 3D reconstruction and SLAM. In: International Conference on Robotics and Automation. Hong Kong, China (2014)
10. Hertzmann, A.: Introduction to 3D non-photorealistic rendering: Silhouettes and outlines. In: Special Interest Group on Computer GRAPHics and Interactive Techniques (1999)
11. Huang, A.S., Bachrach, A., Henry, P., Krainin, M., Maturana, D., Fox, D., Roy, N.: Visual odometry and mapping for autonomous flight using an RGB-D camera. In: International Symposium on Robotics Research (2011)
12. Kerl, C., Sturm, J., Cremers, D.: Robust odometry estimation for RGB-D cameras. In: International Conference on Robotics and Automation (2013)
13. Klein, G., Murray, D.: Full-3D edge tracking with a particle filter. In: British Machine Vision Conference (2006)
14. Klein, G., Murray, D.: Parallel tracking and mapping for small AR workspaces. In: International Symposium on Mixed and Augmented Reality (2007)
15. Lepetit, V., Fua, P.: Monocular model-based 3D tracking of rigid objects: a survey. Found. Trends Comput. Graph. Vis. **1**(1), 1–89 (2006)
16. Levenberg, K.: A method for the solution of certain non-linear problems in least squares. Q. Appl. Math. **2**(2), 164–168 (1944)
17. Lhuillier, M.: Incremental fusion of structure-from-motion and GPS using constrained bundle adjustments. Pattern Anal. Mach. Intell. **34**(12), 2489–2495 (2012)
18. Li, G., Tsin, Y., Genc, Y.: Exploiting occluding contours for real-time 3D tracking: A unified approach. In: International Conference on Computer Vision (2007)
19. Loesch, A., Bourgeois, S., Gay-Bellile, V., Dhome, M.: Generic edgelet-based tracking of 3D objects in real-time. In: Intelligent RObots and Systems (2015)

20. Loesch, A., Bourgeois, S., Gay-Bellile, V., Dhome, M.: A hybrid structure/trajectory constraint for visual slam. In: 3D Vision (2016)
21. Lothe, P., Bourgeois, S., Dekeyser, F., Royer, E., Dhome, M.: Towards geographical referencing of monocular slam reconstruction using 3D city models: application to real-time accurate vision-based localization. In: Computer Vision and Pattern Recognition (2009)
22. Melbouci, K., Collette, S.N., Gay-Bellile, V., Ait-aider, O., Dhome, M.: Model based RGBD SLAM. In: International Conference on Image Processing (2016)
23. Middelberg, S., Sattler, T., Untzelmann, O., Kobbelt, L.: Scalable 6-DOF localization on mobile devices. In: European Conference on Computer Vision (2014)
24. Mouragnon, E., Lhuillier, M., Dhome, M., Dekeyser, F., Sayd, P.: Real time localization and 3D reconstruction. In: Computer Vision and Pattern Recognition (2006)
25. Mur-Artal, R., Montiel, J.M.M., Tards, J.D.: ORB-SLAM: a versatile and accurate monocular SLAM system. Trans. Robot. **31**(5), 1147–1163 (2015)
26. Newcombe, R.A., Izadi, S., Hilliges, O., Molyneaux, D., Kim, D., Davison, A.J., Fitzgibbon, A.: Kinectfusion: Real-time dense surface mapping and tracking. In: International Symposium on Mixed and Augmented Reality (2011)
27. Nienhaus, M., Doellner, J.: Edge-enhancement–an algorithm for real-time non-photorealistic rendering. J. WSCG **11**(2), 1–3 (2003)
28. Oikawa, M.A., Taketomi, T., Yamamoto, G., Fujisawa, M., Amano, T., Miyazaki, J., Kato, H.: Local quadrics surface approximation for real-time tracking of textureless 3D rigid curved objects. In: Symposium on Virtual and Augmented Reality (2012)
29. Petit, A., Marchand, E., Kanani, K.: Combining complementary edge, point and color cues in model-based tracking for highly dynamic scenes. In: International Conference on Robotics and Automation (2014)
30. Press, W.H. Teukolsky, S.A., Vetterling, W.T., Flannery, B.P. Numerical Recipes: The Art of Scientific Computing, 3rd edn, pp. 282–283. Cambridge University Press, Cambridge (2007)
31. Ramadasan, D., Chevaldonne, M., Chateau, T.: Dcslam: A dynamically constrained real-time SLAM. In: International Conference on Image Processing (2015)
32. Raskar, R.: Hardware support for non-photorealistic rendering. In: Special Interest Group on computer GRAPHics and Interactive Techniques Workshop on Graphics hardware (2001)
33. Stanimirovic, D., Damasky, N., Webel, S., Koriath, D., Spillner, A., Kurz, D.: [Poster] A mobile augmented reality system to assist auto mechanics. In: International Symposium on Mixed and Augmented Reality (2014)
34. Sturm, J., Engelhard, N., Endres, F., Burgard, W., Cremers, D.: A benchmark for the evaluation of RGB-D SLAM systems. In: Intelligent Robots and Systems (2012)
35. Tamaazousti, M., Gay-Bellile, V., Collette, S.N., Bourgeois, S., Dhome, M.: Real-time accurate localization in a partially known environment: application to augmented reality on textureless 3D objects. In: International Symposium on Mixed and Augmented Reality Workshop (2011)
36. Tamaazousti, M., Gay-Bellile, V., Naudet-Collette, S., Bourgeois, S., Dhome, M.: Nonlinear refinement of structure from motion reconstruction by taking advantage of a partial knowledge of the environment. In: Conference on Computer Vision and Pattern Recognition (2011)
37. Triggs, B., McLauchlan, P., Hartley, R., Fitzgibbon, A.: Bundle adjustment—a modern synthesis. In: International Conference on Computer Vision (2000)
38. Vacchetti, L., Lepetit, V., Fua, P.: Combining edge and texture information for real-time accurate 3D camera tracking. In: International Symposium on Mixed and Augmented Reality (2004)
39. Vacchetti, L., Lepetit, V., Fua, P.: Stable real-time 3D tracking using online and offline information. Pattern Anal. Mach. Intell. **26**(10), 1385–1391 (2004)
40. Wasenmuller, O., Meyer, M., Stricker, D.: CoRBS: comprehensive RGB-D benchmark for SLAM using Kinect v2. In: Winter Conference on Applications of Computer Vision. http://corbs.dfki.uni-kl.de/ (2016). Accessed 26 June 2018
41. Wuest, H., Wientapper, F., Stricker, D.: Adaptable model-based tracking using analysis-by-synthesis techniques. In: Computer Analysis of Images and Patterns (2007)

Angelique Loesch received PhD degree in computer science from the University Blaise Pascal, Clermont-Ferrand, France. She did her research in partnership with the Pascal Institute under the supervision of Professor Michel Dhome and Doctors Steve Bourgeois and Vincent Gay-Bellile. She is currently a permanent researcher at CEA LIST. Her main research interests include simultaneous localization and mapping, 3D object tracking, as well as visual perception with deep learning (detection, tracking, classification, recognition).

Steve Bourgeois received PhD degree in computer vision from the University Blaise Pascal, Clermont-Ferrand, France, in 2006 under the supervision of Professor Michel Dhome. He is currently a permanent researcher at CEA LIST where he works on industrial projects and supervises PhD students. His main research interests include simultaneous localization and mapping, 3D object tracking and their applications in augmented reality and robotics.

Vincent Gay-Bellile received PhD degree in computer science from the University Blaise Pascal, Clermont-Ferrand, France, in 2008. He did his research in partnership with the LASMEA and LSVE laboratory under the supervision of Professors Jean-Thierry Lapresté, Adrien Bartoli and Doctor Patrick Sayd. He is currently a permanent researcher at CEA LIST. His main research interests include structure from motion for rigid scenes, real-time vision-based localization and reconstruction (SLAM) for autonomous system.

Olivier Gomez received a masters degree in computer science from the University of Burgundy, Dijon, France. He is currently a permanent engineer/researcher at CEA LIST. He focuses on hardware specification and concerns about real-time, memory-efficient code mainly for applications using vision-based localization and reconstruction (vSLAM).

Michel Dhome aged 60 years, is a senior researcher of the French National Council for Scientific Research (CNRS). From 2005 to 2011, he was the head of the Laboratory of the Sciences and Materials for Electronics, and for Automatic (LASMEA) of Clermont-Ferrand. From 2012 to 2016, he was the head of a new laboratory « Pascal Institute » of Clermont-Ferrand (union of three old laboratories—300 people). He is actually the head of the laboratory of excellence IMobS3, research program on "Innovative Mobility: Smart and Sustainable Solutions", involving seven laboratories working in the mechanics, automatics, electronics, computer sciences and process engineering. He is also the head of FACTOLAB which is a common laboratory between IMobS3 and the French manufactory of tires MICHELIN. His main scientific contributions concern essentially the domain of the « Artificial Perception », with the « Robotics » as main field of application. His activity can be classified according to the following four topics: - Auto-calibration of video cameras; - Complex 3D objects alignment on video sequences; - Real-time tracking of textured objects; - Artificial vision approaches for the autonomous vehicles guidance. He has published more than 200 scientific publications and has 6 patents.