2013 IEEE/RSJ International Conference on
Intelligent Robots and Systems (IROS)
November 3-7, 2013. Tokyo, Japan

On unifying key-frame and voxel-based dense visual SLAM at large scales

Maxime Meilland and Andrew I. Comport

Abstract— **This paper proposes an approach to real-time dense localisation and mapping that aims at unifying two different representations commonly used to define dense models. On one hand, much research has looked at 3D dense model representations using voxel grids in 3D. On the other hand, image-based key-frame representations for dense environment mapping have been developed. Both techniques have their relative advantages and disadvantages which will be analysed in this paper. In particular each representation's space-size requirements, their effective resolution, the computation efficiency, their accuracy and robustness will be compared. This paper then proposes a new model which unifies various concepts and exhibits the main advantages of each approach within a common framework. One of the main results of the proposed approach is its ability to perform large scale reconstruction accurately at the scale of mapping a building.**

I. Introduction

The problem of dense real-time localisation and mapping within complex environments is a challenging problem for a wide range of applications ranging from robotics to augmented reality. The present work is undertaken as part of a French DGA Rapid project named Fraudo which requires dense localisation and mapping in real-time so as to allow path planning for a mobile robot to traverse uneven ground and surfaces autonomously. In this context it is important to perceive the 3D surfaces of the environment in real-time and for this dense techniques are the best approach. Amongst state-of-the-art dense SLAM approaches 3D pose tracking is usually performed in a similar manner, however, the underlying 3D representation of the dense model differs. The goal of this paper is therefore to develop an efficient, accurate and robust *dense visual model* for localisation and mapping that unifies various aspects of existing approaches.

More precisely, this article aims to investigate the problem of real-time dense localisation and mapping from an RGB-D camera. In the last five years many different approaches have emerged to tackle this challenging problem. Mostly, research in real-time tracking and mapping has focused on feature-based techniques [1], [2], [3], [4]. Dense fusion techniques have previously been studied in the context of photometric rendering and off-line structure from motion [5]. Several techniques take the approach of minimising an error based on a voxelized 3D model [6] (model-based). More recent approaches minimise both photometric and geometric errors directly using key-frames (image-based) in the sensor space [7], [8]. The aim of this paper is to compare voxel and key-frame approaches to highlight the main differences and advantages. An improved approach will then be presented which exhibits better properties.

Although the main focus of the paper will not be on the 3D pose estimation technique, it will be advocated that direct pose estimation from photometric and depth images performs better [8] and subsequently this type of pose estimation will be considered common to all approaches (even if in many articles feature-based or purely geometric approaches are used). Instead this paper will focus on the underlying 3D model representation that is used to stock both geometry and photometric information of a large scale scene. Large scale scene reconstruction is relatively little studied in the literature due to the added complexity of managing large amounts of sensor information consistently. Nevertheless, several approaches have been proposed including [7], which is based on a trajectory of key-frames, [9] which proposes a graph of spherical RDB-D key-frames and, more recently, [10] and [11] who propose to extend volumetric approaches to large scale by using a moving volume.

A. Volumetric 3D Modelling

Classically, volumetric reconstructions were first proposed in the computer graphics literature [12], [13]. The aim is to extend the nice regular properties of 2D image grids to 3D voxel space. These approaches therefore provide a good formalism that allows for the easy development of mathematical filters and tools for manipulating voxels. Volumetric approaches typically use a Signed Distance Function (SDF) [6] to store a distance to a surface in a discretised voxel grid, representing the world geometry.

One advantage of this representation is its ease and efficiency in defining and applying mathematical operators. Another advantage is that it implicitly models self occlusions from different parts of the scene. Finally, it allows to reduce noise in structure measurements in a global 3D reference frame. Computationally, most modern graphics are cards also designed to be compatible.

Initially, volumetric approaches were used on a pre-acquired set of images and full bundle adjustment (BA) of all camera poses was performed to minimise drift. On the other hand, incremental approaches such as SLAM have been heavily studied in the robotics literature and more recently these approaches have been used to perform incremental real-time dense mapping [14], [10], [11]. The effects of incremental voxel integration are only beginning to be studied through these real-time approaches. SLAM approaches aim for real-time computation but are also subject to a much larger amount of drift over time. This is often compensated by loop closure approaches and/or reduced by performing local sliding window BA [15]. Unfortunately, voxel-based approaches have not been designed with incremental drift in mind and performing loop-closure on a volume is a complex task that would require redefining the underlying structure.

One main aspect of volumetric models is that, in general, they represent *all* space with voxels, whether the space is occupied or not. Take as a simple example the Signed Distance

M. Meilland and A. I. Comport are with CNRS-I3S Laboratory, University of Nice Sophia Antipolis surname@i3s.unice.fr

Function (SDF), which is used to represent occupied surfaces within a $1m^3$ volume. Consider the memory requirements for a $512 \times 512 \times 512 \times 64$bits SDF, where each voxel contains a 16bit signed distance measurement, a 16bit weight and a 32bit RGBA colour value therefore requiring roughly 1GB of memory. In the literature these space requirements have been reduced by using octrees to reduce the resolution of the grid in empty space. Recently, this approach has been adapted to real-time dense mapping in [16] and [17].

A disadvantage of this approach is that the knowledge about the camera positions, used to acquire the map, are usually completely discarded (i.e. it is independent of the sensor trajectory) and the volumetric approach is unable to easily capture the non-linear variation of the image resolution, which depends on a particular camera path. Whilst a co-variance can be stored for each voxel, the associated uncertainty becomes linearised in 3D space and the uncertainties of the sensor measurements are approximated. Finally volumetric approaches will consume too much memory to represent high image resolutions (super-resolution [8]) across all space (as opposed to locally around nodes in a graph).

B. Image-based Key-frame Modelling

Image-based key-frame models have been popular in the robotics literature, due to their locally accurate representation, their capability to handle incremental drift and readjust a more compact set of poses using classic loop-closure constraints in real-time [18]. Early real-time key-frame approaches based on features [19], [20], [21]. The first real-time dense visual odometry approach was published in [7] but key-frames corresponded to real-images and no structure or photometric information was fused. Recently, a key-frame approach has been published, which fuses simultaneously geometry and photometry onto a set of super-resolution key-frames [8] whereby the closest key-frame in the graph is often used to perform localisation. Subsequently key-frame approaches also have great noise reduction properties.

Another advantage of key-frame approaches is their memory efficiency when the viewing path is the same as the acquisition trajectory. Contrary to volumetric approaches, these models encode the raw sensor data (and sensor uncertainty) along with the trajectory associated with the image acquisition, whereas volumetric approaches discard this information. One further result of maintaining a reference trajectory is that it directly encodes the camera resolution from these vantage points since the resolution of the map is a function of the path taken to acquire it. The resolution of volumetric approaches aims to cover all vantage points equally. In terms of comparing memory efficiency, consider the extreme boundary case of a single floating point RGB-D key-frame of dimensions $640 \times 480 \times 64bits$. It contains 16 bits for each depth, 16 bits for each weight, 32 bits for each RGBA colour and the parameters for the pose of the key-frame can be considered negligible. In this case the key-frame image only requires 2.34 MB to represent a local subset of viewpoints through novel-view synthesis (depending on the scene structure). Furthermore, this representation can be accessed in constant time, independently of the number of key-frames in the graph.

Whilst volumetric approaches benefit from efficient computation due to a regular (non-scattered) grid pattern, key-frame approaches are also able to perform computation efficiently by exploiting a 2D grid pattern in the image space.

As will be seen later, one disadvantage of key-frame approaches is that the current camera image does not always correspond to the same field of view as the closest key-frame and subsequently only have a partial overlap. This can degrade the localisation robustness since only a small area of the current image is registered. This can be solved by more sophisticated reference images such as spherical panoramas with depth [22]. However, this model is not easily applicable for commonly used sensors with limited field of view.

C. An approach to Unifying

In this paper, a new approach will be proposed that aims to combine the advantages of volumetric and key-frame based approaches. From the outset, it is clear that key-frame approaches encode additional information about the acquisition phase. In particular, the acquisition trajectory is known and raw sensor data at set resolutions are stored. This raw data can easily be transformed into a volumetric representation to exhibit the properties of volumetric approaches. Nevertheless if we discard the acquisition data from the model then it will be lost completely. As such, the proposed model will maintain an underlying key-frame representation that can easily be transformed into a volumetric 3D model if required.

The aim of the new approach will therefore be to develop a multi-key-frame approach that can easily handle self occlusions and that can fuse multiple sources of data as volumetric approaches do. A simple approach to solve this problem is to simultaneously use multiple key-frames extracted from the graph, however, this would require warping the current image several times, increasing the computational cost. To avoid the redundant warping of the current image onto several overlapping reference frames, a new forward composition approach is proposed by developing a computationally efficient warp and blending function. This function will take multiple reference images and combine them efficiently in real-time to create a single predicted frame. As will be demonstrated, this allows more efficient registration with respect to only one *synthesised* reference key-frame.

Returning to the small example on space requirements, volumetric approaches account for the set of all 3D points implicitly in the structure (up to a given resolution). When compared to the same resolution as a single key-frame, the voxel model requires adding a 3rd dimension to the image size. For example, a key-frame image of size 512^2 becomes a volume of dimension 512^3 to represent the same resolution for that view. Of course, a single key-frame does not cover arbitrary viewpoints as a volumetric model does and it is necessary to determine how many key-frames are needed. Unfortunately this is highly dependant on the scene structure.

In the ideal case for key-frame models, the camera observes either a convex scene or a concave object. In that case the equivalent number of images to the voxel grid corresponds to the 6 images of a cube observing the scene or the object at the same resolution as the voxel grid (assuming an orthographic camera projection model). The key-frame

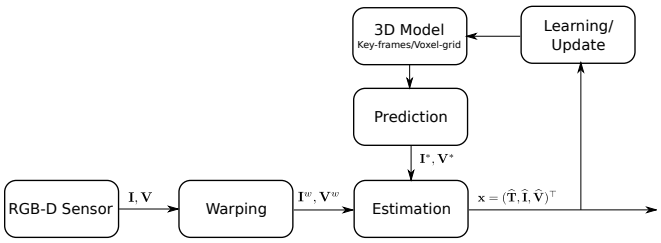

Fig. 1. Unified image and model-based system. For both volumetric and key-frame based representations, tracking and mapping can be decomposed in four major blocks: warping, prediction, estimation and model update

model requires $512^2 \times 6$ images plus 6 poses to represent the scene, while, volumetric approaches require 512^3 voxels minus any compression achieved with octrees. It is clear that if complex non-concave objects or non-convex scenes need to be mapped, key-frame based approaches will require an increasing number of reference views to account for the self occlusions of the scene. However, in the proposed approach, key-frames are to be chosen carefully to encapsulate the concave and convex elements of the scene, allowing a compact and efficient representation to be found.

II. Dense real-time tracking and mapping

A. Introduction

Consider a calibrated RGB-D sensor with a colour brightness function $\mathbf{I} : \Omega \times \mathbb{R}^+ \to \mathbb{R}^+; (\mathbf{p}) \mapsto \mathbf{I}(\mathbf{p}, t)$ and a depth function $\mathbf{D} : \Omega \times \mathbb{R}^+ \to \mathbb{R}^+; (\mathbf{p}, t) \mapsto \mathbf{D}(\mathbf{p}, t)$, where $\Omega = [1, n] \times [1, m] \subset \mathbb{R}^2$, $\mathbf{P} = (\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_{nm})^\top \in \mathbb{R}^{\mathrm{mn}\times 2} \subset \Omega$ are pixel locations within the image acquired at time t, and $n \times m$ is the dimension of the sensor's images. It is convenient to consider the set of measurements in vector form such that $\mathbf{I}(\mathbf{P}, t) \in \mathbb{R}^{+\mathrm{nm}\times 1}$ and $\mathbf{D}(\mathbf{P}, t) \in \mathbb{R}^{+\mathrm{nm}\times 1}$. Note that t and $\mathbf{P}$ may be omitted for clarity.

$\mathbf{V} = (\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_{nm})^\top \in \mathbb{R}^{\mathrm{mn}\times 3}$ is defined as the matrix of 3D vertices related to the surface according to the following point-depth back-projection:

$$\mathbf{v}_i = \mathbf{K}^{-1}\overline{\mathbf{p}}_i \mathbf{D}(\mathbf{p}_i), \tag{1}$$

where $\mathbf{K} \in \mathbb{R}^{3\times 3}$ is the intrinsic camera matrix and $\overline{\mathbf{p}}_i$ are homogeneous pixels coordinates. $\mathbf{V}$ will be taken to be a 3D vertex function $\mathbf{V} : \Omega \times \mathbb{R}^+ \to \mathbb{R}^3; (\mathbf{p}, t) \mapsto \mathbf{V}(\mathbf{p}, t)$.

The set $\boldsymbol{\mathcal{S}} = \{\mathbf{I}, \mathbf{V}, \mathbf{N}, \mathbf{C}\}$ is defined to be a 3D textured surface. The uncertainties $\mathbf{C} \in \mathbb{R}^{\mathrm{nm}}$ are computed for each vertex according to the depth sensor model of [23]. The normals $\mathbf{N} \in \mathbb{R}^{3\times \mathrm{nm}}$ are computed for each vertex using a local cross product on the image grid. Again, the surface normals and the uncertainties will be considered as functions such that $\mathbf{N} : \Omega \times \mathbb{R}^+ \to \mathbb{R}^3; (\mathbf{p}, t) \mapsto \mathbf{V}(\mathbf{p}, t)$ and $\mathbf{C} : \Omega \times \mathbb{R}^+ \to \mathbb{R}; (\mathbf{p}, t) \mapsto \mathbf{C}(\mathbf{p}, t)$.

A typical localisation/reconstruction pipeline is shown in Figure 1. For both volumetric and key-frame based approaches, the incremental reconstruction pipeline contains the following blocks: frame prediction, warping, estimation and model update. Here frame-to-frame pose estimation is made between a predicted view and the live frame so that this step is the same for both approaches. The only blocks that differ are the view prediction and model update.

B. Pose estimation

Now consider the predicted reference surface $\boldsymbol{\mathcal{S}}^* = \{\mathbf{I}^*, \mathbf{V}^*, \mathbf{N}^*, \mathbf{C}^*\}$ to be a view prediction of the current surface $\boldsymbol{\mathcal{S}} = \{\mathbf{I}, \mathbf{V}, \mathbf{N}, \mathbf{C}\}$. The goal is to find the unknown motion parameters $\mathbf{x} \in \mathbb{R}^6$ between $\boldsymbol{\mathcal{S}}^*$ and $\boldsymbol{\mathcal{S}}$ defined as:

$$\mathbf{x} = \int_0^1 (\boldsymbol{\omega}, \boldsymbol{\upsilon}) dt \in se(3), \tag{2}$$

which is the integral of a constant velocity twist which produces a pose $\mathbf{T}$, where $\mathbf{T} = (\mathbf{R}, \mathbf{t}) \in \mathbb{SE}(3)$. $\mathbf{R} \in \mathbb{SO}(3)$ is a rotation matrix and $\mathbf{t} \in \mathbb{R}(3)$ a translation vector. The pose and the twist are related via the exponential map as $\mathbf{T} = e^{[\mathbf{x}]_\wedge}$ with the operator $[.]_\wedge$ as:

$$[\mathbf{x}]_\wedge = \begin{bmatrix} [\boldsymbol{\omega}]_\times & \boldsymbol{\upsilon} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}, \tag{3}$$

where $[.]_\times$ represents the skew symmetric matrix operator.

An improved version of the bi-objective direct iterative closest point (ICP) approach of [24] is employed, which simultaneously minimises a photometric error along with a geometric error between the surfaces $\boldsymbol{\mathcal{S}}^*$ and $\boldsymbol{\mathcal{S}}$ such that

$$\mathbf{e}(\mathbf{x}) = \begin{bmatrix} \mathbf{I}\left(w(\widehat{\mathbf{T}}\mathbf{T}(\mathbf{x}), \mathbf{V}^*)\right) - \mathbf{I}^*(\mathbf{P}^*) \\ \widehat{\mathbf{R}}\mathbf{R}(\mathbf{x})\mathbf{N}^{*\top}\left(\mathbf{V}^\top - \boldsymbol{\Pi}\widehat{\mathbf{T}}\mathbf{T}(\mathbf{x})\mathbf{V}^{*\top}\right)^\top \end{bmatrix}, \tag{4}$$

where the first row of equation (4) is the photometric term and the second row is a point-to-plane ICP error with projective data association. The function $w(\cdot)$ is the inverse warping function described in Section III-B.

This non-linear error is iteratively minimised using a Gauss-Newton approach such that:

$$\mathbf{x} = -(\mathbf{J}^T\mathbf{W}\mathbf{J})^{-1}\mathbf{J}^T\mathbf{W}\mathbf{e}, \tag{5}$$

where $\mathbf{J}^T = \begin{bmatrix} \mathbf{J}_{esm} & \mathbf{N}^{*\top}\mathbf{J}_{icp} \end{bmatrix}$.

$\mathbf{J}$ contains the stacked Jacobian matrices of the errors of equation (4). $\mathbf{J}_{esm}$ is the Jacobian matrix of the photometric term computed using the efficient second order (ESM) approach [25] and $\mathbf{J}_{icp}$ is the standard ICP Jacobian matrix.

$\mathbf{W}$ is a diagonal weighting matrix of dimensions $2nm \times 2nm$ obtained by M-estimation [26], with Huber's influence function. Since the photometric error is influenced by the geometric error, the matrix $\mathbf{W}$ is computed such that:

$$\mathbf{W} = \begin{bmatrix} \mathbf{W}_{esm}\mathbf{W}_{icp} & \mathbf{0} \\ \mathbf{0} & \mathbf{W}_{icp}. \end{bmatrix} \tag{6}$$

This weighting scheme efficiently handles geometric occlusions, such as dynamic moving objects. Note that the photometric errors (and subsequently the weights) are coupled with the geometry while the geometric error is independent of the photometry (only for projective light RGB-D sensors).

The pose estimate $\widehat{\mathbf{T}}$ is finally updated using a homogeneous update until convergence as $\widehat{\mathbf{T}} \leftarrow \widehat{\mathbf{T}}\mathbf{T}(\mathbf{x})$.

III. Synthesising a novel view from the model

The main difference between volumetric and key-frame approaches appears whenever it is necessary to interact and use the 3D model data. Even within each of these approaches several errors may be minimised, primarily depending on a

forward or inverse warping function. These warping functions will be elaborated here and a new warping function will be proposed for combining several key-frames together efficiently, evidently solving the problem of partial overlap and occlusion (between current and key-frame images).

A. Geometric warping

The geometric warping function $\overline{\mathbf{P}}^{w\top} = w(\mathbf{T}, \mathbf{K}, \mathbf{V})$, transforms the vertices $\mathbf{V}$ with the pose $\mathbf{T}$ and projects the transformed vertices on the image plane by $\mathbf{P}^{w\top} = \mathbf{M}\overline{\mathbf{V}}^{\top}$, where the perspective projection matrix $\mathbf{M}$ is defined such that $\mathbf{M} = \mathbf{K}\mathbf{\Pi}\mathbf{T}$. $\mathbf{\Pi} = [\mathbf{1}, \mathbf{0}] \in \mathbb{R}^{3\times 4}$ projects the 4×4 pose matrix onto the 3×4 space and $\mathbf{P}^{w\top}$ is normalised by dividing by its third co-ordinate to obtain $\overline{\mathbf{P}}^{w\top}$.

B. Inverse warping / texture projection

Inverse warping is the equivalent of texture projection in computer graphics. It basically consists in projecting the current image/texture onto the world geometry. It is often referred as *inverse warping* since the transformation from the source intensities to the destination space is performed by projecting the destination vertices (reference image) to the source space (current image) and then interpolating the intensity values on a regular grid as:

$$\mathbf{I}^*(\mathbf{P}^*) = \mathbf{I}(w(\overline{\mathbf{T}}, \mathbf{K}, \mathbf{V}^*)). \tag{7}$$

As it can be seen in equation (7), the current intensity warping only depends on the reference vertices $\mathbf{V}^*$. This allows to perform computationally efficient grid interpolation of the surface and obtain a correspondence between the reference and the current intensities. One major drawback of this approach is that occlusions are not handled, since several vertices may project onto the same pixel, leading to inconsistencies in the warped image. This is, however, the fastest way to transfer image intensities between frames and its Jacobian can be pre-computed.

C. Forward warping / rasterisation

Standard rasterisation is one of the easiest ways to render novel views whilst handling occlusions via z-buffering. The pipeline is this time straightforward (source to destination), since the world geometry (triangles) is directly projected and interpolated at the screen pixels (destination) yet the colours are interpolated in the reference texture. The main difference is that this requires a scattered interpolation which is computationally more expensive. The intensity warping can be defined such that

$$\mathbf{I}^*(\mathbf{P}^*) = \mathbf{I}\Big(\Gamma\big(\mathbf{P}^*, \mathbf{E}, w(\mathbf{V}, \overline{\mathbf{T}}^{-1}, \mathbf{K}^*)\big)\Big) \tag{8}$$

where $\mathbf{E} \in \mathbb{N}^{J\times 3}$ contains $J \times 3$ indices of each triangle (computed in the image as described in Section V-A.1), and Γ is the rasterisation function that interpolates the scattered vertices projected at the destination pixel locations $\mathbf{P}^*$. Rasterisation is directly implemented in hardware on GPUs and allows to efficiently perform scattered interpolation on a very large amount of triangles. With abuse of notation, the function

$$\boldsymbol{\mathcal{S}}^* = \boldsymbol{\mathcal{S}}\Big(\Gamma\big(\mathbf{P}^*, \mathbf{E}, w(\mathbf{V}, \overline{\mathbf{T}}^{-1}, \mathbf{K}^*)\big)\Big) \tag{9}$$

will be a synthesis of the surface $\boldsymbol{\mathcal{S}}$ at position $\mathbf{T}$, where $\boldsymbol{\mathcal{S}}^*$ is the new predicted surface which contains new intensities, vertices, normals and uncertainties.

D. Warping comparisons

Forward warping (rasterisation) benefits from many hardware accelerated features that cannot be done efficiently with inverse warping (texture projection). In particular, mipmapping, anisotropic filtering (which prevents aliasing) and occlusions (via z-buffering). Even if the scattered interpolation can be highly optimised (see Section. V-A.1), the computational cost is less efficient than inverse warping, mainly due to triangulation and scattered interpolation of the surface.

On the other hand, inverse warping is only valid for small viewpoint changes where occlusions and perceptual aliasing can be neglected. In order to take advantage of hardware rasterisation, it is proposed here to use forward warping for view prediction and model update steps, which only have to be performed once per frame. Dense iterative registration, which requires multiple warps per frame, can use a fast inverse warping between the predicted frame and the live frame which are assumed to be close.

E. Multi-key-frame fusion

Supposing that an initial estimation of the current camera pose $\widehat{\mathbf{T}}$ is available (*i.e.* the last estimated pose), the surface prediction is performed by first extracting close key-frames from the graph. A simple criteria is used to select the M closest key-frames to the current frame based on the distance along the graph. This avoids choosing key-frames which are close geometrically but have not been connected visually during the acquisition (i.e. on the other side of a wall). Further connections could be made by performing loop closure and bundle adjustment on the entire key-frame graph.

Each key-frame is then rasterised and blended into a virtual frame at the predicted camera viewpoint such that

$$\boldsymbol{\mathcal{S}}^* = \sum_{i=1}^{M} f\Big(\boldsymbol{\mathcal{S}}\Big(\Gamma\big(\mathbf{P}^*, \mathbf{E}, w(\mathbf{V}_i, \widehat{\mathbf{T}}^{-1}\mathbf{T}_i, \mathbf{K}^*)\big)\Big)\Big), \tag{10}$$

where $f(\boldsymbol{\mathcal{S}}(\cdot))$ is a blending function, that correctly fuses the synthesised images. In order to detect occlusions between the rendered surfaces before blending a Mahalanobis distance is computed between the candidate vertices to test their mutual dependency. Let us consider the sets $\{\mathbf{v}_a^*, \mathbf{c}_a^*\}$ and $\{\mathbf{v}_b^*, \mathbf{c}_b^*\}$ to be two vertices and uncertainties candidates for a new vertex $\mathbf{v}^*$. The distance is defined by

$$d_M = \mathbf{e}_3^\top(\mathbf{v}_a^* - \mathbf{v}_b^*)^\top(\mathbf{c}_a^* + \mathbf{c}_b^*)^{-1}\mathbf{e}_3^\top(\mathbf{v}_a^* - \mathbf{v}_b^*). \tag{11}$$

The blended vertex $\mathbf{v}^*$ is obtained using a Chi-square test such that:

$$\mathbf{v}^* = \begin{cases} \frac{(\mathbf{v}_a^*\mathbf{c}_a^{*-1} + \mathbf{v}_b^*\mathbf{c}_b^{*-1})}{(\mathbf{c}_a^{*-1} + \mathbf{c}_b^{*-1})} & \text{if} \quad d_M < 3.841 \\ \mathbf{v}_a^* & \text{elseif} \quad \mathbf{e}_3^\top\mathbf{v}_a^* < \mathbf{e}_3^\top\mathbf{v}_b^* \\ \mathbf{v}_b^* & \text{else} \end{cases} \tag{12}$$

where 3.841 corresponds to 5% of error tolerance.

The intensities of the virtual image $\mathbf{I}^*$ are also obtained by blending the reference key-frames. The weighting function proposed in [8] is used if the dependence test of (12)

succeeds, otherwise the final intensity is assigned to the closest vertex. This function weights the relative resolution between the predicted image and the closest key-frames giving larger weights to images with closer viewpoints.

IV. Model update

This step consists in updating the model using the sensor pose estimation and the current image data. For volumetric approaches such as [14], this is achieved by integrating the new measurements (depths and intensities) into the TSDF through a weighted running average. For moving TSDF variants [11], [10], the volume can also be shifted if the camera pose exceeds some movement threshold.

For key-frame approaches such as [4], new measurements can be integrated into nearby frames by warping the current data using the estimated pose, and a new key-frame is generated if the camera pose exceeds some movement threshold.

In the proposed approach, the current surface is rasterised onto the M key-frames that were used for the pose estimation. The integration proposed in [8] is employed to correctly fuse the depth-maps and intensities, allowing to reduce the noise contained in the raw depth images. A simple criterion for managing new key-frame generation is used in addition to a classic threshold on the camera motion. This criterion consists in monitoring the amount of occluded pixels between the current frame and the next frame prediction (which is computed according to the current pose estimate) using the test of equation (12). A new key-frame is therefore added and connected to the M closest frames in the graph when the amount of occluded pixels in the image is greater than a defined threshold. This allows to dynamically scale the number of key-frames with the scene complexity.

V. Experiments

A. Real-time implementation

The proposed approach was implemented and optimised for modern GPUs using the OpenCL library with OpenGL interoperability. This allows to use the OpenGL hardware pipeline for surface rasterisation with GLSL shaders and the more flexible OpenCL kernels for iterative registration with inverse warping. Extensive use of texture memory has been made to ensure fast random memory access with hardware bilinear interpolation. The iterative registration technique of Section II-B is performed with a coarse to fine multi-resolution approach as detailed in [7]. Since the RGB-D sensor usually provides noisy depth measurements, a bilateral filter is applied to remove noise whilst preserving discontinuities. The filtered depth-map is only used for pose estimation, while the raw depth-map is used for depth integration to preserve details in the integration process. The test platform used for experiments is a standard desktop PC running Ubuntu 12.10 with an Intel Core i7 2700k and a 2GB nVidia GeForce GTX670. Table I shows the average computing time of each step of the algorithm along with its standard deviation. The first step (initialisation) consists in loading the current RGB-D frame in GPU memory, performing bilateral filtering and computing the Gaussian pyramids of the images. The second step is the iterative pose estimation described in Section II-B. The third step is the model update of Section IV

TABLE I
Algorithm computing time

	Average time (ms)	Standard dev. (ms)
Initialisation	12.90	2.49
Pose estimation	11.78	2.06
Map update/fusion	2.83	1.55
View Prediction	3.14	1.38
Total	30.63	3.08

which consists in updating the key-frames. The last step is the next view prediction performed using the warping of Section III-C and the fusion of Section III-E.

1) Efficient triangulation on the GPU: Since the vertices of a key-frame are stored onto a pixel grid, one of the fastest way to rasterise a new surface is to use triangle strips. Triangle strips are a kind of primitive that specifies a series of connected triangles, sharing vertices, allowing for more efficient memory usage. However, such a surface does not take into account depth discontinuities, as well as missing values in the reference depth-map, which will create artefacts in the synthesis. One approach to handle this is to perform a simple 2D triangulation on the CPU using valid depth measurements and send the obtained indices to the GPU before drawing. This has some limitations:

- Since the depth-map is directly integrated in GPU memory, it must be sent to the CPU before triangulating;
- The amount of triangle indices that have to be uploaded to the GPU after triangulation is generally enormous (up to 1843200 indices for a 640×480 surface).

To avoid performing this slow GPU/CPU memory transfer each time a reference view needs to be triangulated (*i.e.* multiple times at frame-rate), it is proposed here to perform the "triangulation" on the fly using geometry shaders. Geometry shaders are an optional part of the graphics rendering pipeline which is applied after primitive assembly. They allow to simultaneously access all the vertices of a primitive and can be used to discard or generate new primitives (only a few).

In this case, the entire set of reference vertices are drawn as points and sent to the vertex shader stage, which just sends the vertices positions to the geometry shader. The input of the geometry shader is then a vertex and the output is a triangle strip with a maximum of 4 vertices (2 triangles). The 3 neighbouring values of the current vertex are read via texture fetch and the triangle strip is emitted only if the following constraints are respected:

- All 4 depth values must be in a valid range (*i.e.* > 0).
- All triangle strip edge lengths must be smaller than a threshold to avoid depth discontinuities on the surface.

The standard rasterisation pipeline is then applied with texture interpolation in screen space. Note that the rendered vertices and uncertainties are simultaneously interpolated and stored into RGBA floating point textures, using multiple render targets (MRT). Table II compares the computation time required for triangulating and rendering a 640×480 surface, containing 606404 triangles, using a naive CPU triangulation and the proposed approach. It can be seen that the full GPU triangulation is almost 14x faster than the naive CPU approach. Another advantage of triangulating on the fly is the memory footprint, since no temporary buffer of memory is used to store the triangles elements.

TABLE II

COMPARISON OF THE COMPUTATIONAL TIME BETWEEN A NAIVE CPU TRIANGULATION AND THE PROPOSED GPU TRIANGULATION

	CPU+GPU (ms)	Proposed (ms)
Depth-map read-back	0.80	none
2D Triangulation	10.50	none
Elements update	3.56	none
Rasterisation	0.77	1.13
Total	15.64	**1.13**

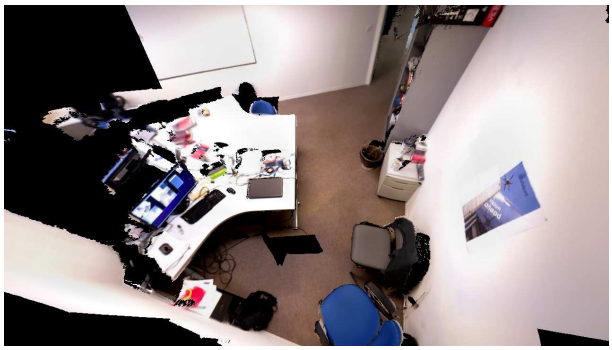

Fig. 2. Bird eye view of a reconstructed office rendered in real-time using the proposed multi-key-frame fusion approach.

2) Reduction: To solve the linear system from equation (5), it is necessary to compute $\mathbf{H} = \mathbf{J}^T\mathbf{J}$ and $\mathbf{g} = \mathbf{J}^T\mathbf{e}$. This is a large reduction problem since $\mathbf{J}$ is of dimensions $2nm \times 6$ and $\mathbf{e}$ is of dimensions $2nm \times 1$. Hence, $\mathbf{H}$ is of dimensions 6×6 and $\mathbf{g}$ is of dimensions 6×1. First note that the matrix $\mathbf{H}$ is symmetric, thus only $21 + 6 = 27$ sums have to be computed for $\mathbf{H}$ and $\mathbf{g}$. The final reduction can be interpreted as summing the $2nm$ rows of a $2nm \times 27$ matrix into a 1×27 vector that can be read back to the CPU before computing the pose update of equation (5). Practically, in order to save GPU cycles, only a partial reduction is performed on GPU using local memory, until a size of $k \times 27$, with $k << 2nm$. The final sum is then performed on the CPU and the system is inverted using Cholesky factorisation.

To further improve speed, SIMD instructions are used to perform 4 single precision instructions per cycle, reducing the problem from a $2nm \times 27$ to a $2nm \times 7$ reduction.

In the optimised implementation, the $2mn \times 27$ matrix is not explicitly constructed and stored in global memory. Instead the summation is performed in parallel as new error and Jacobian values are computed for each pixel. This allows to perform one iteration of the registration in only one kernel call without read/write operations via intermediate buffers.

B. Results

1) Quantitative results: The proposed approach has been evaluated on the RGB-D benchmark of [27]. Four selected sequences are reported on tables III, IV and V, which respectively represent the Absolute Trajectory Error (ATE), the Relative frame-to-frame Error (RPE) and the relative Root Mean Square Error per second (RMSE) corresponding to the drift per second. Three approaches have been compared: the Kintinuous volumetric approach of [11] with the ICP+RGB-D error metric, an image-based approach using only the Closest Key-frame in the Graph (CKF), and the proposed approach using a maximum of 5 View Predictions (VP).

TABLE III

RELATIVE ROOT MEAN SQUARE ERROR OF DRIFT IN METERS/SECOND (RMSE).

Sequence	VP (m/s)	CKF (m/s)	Kintu. (m/s)
fr1/desk	**0.0259**	0.0721	0.0393
fr2/desk	**0.0147**	0.0161	0.0208
fr1/room	**0.0351**	0.0502	0.0622
fr2/large_no_loop	**0.0695**	0.1529	0.1795

TABLE IV

ABSOLUTE TRAJECTORY ERROR IN METERS (ATE).

Sequence	VP (proposed)		CKF		Kintu.	
	Median	Max	Median	Max	Median	Max
fr1/desk	**0.018**	**0.066**	0.044	0.131	0.069	0.234
fr2/desk	**0.093**	**0.116**	0.099	0.130	0.119	0.362
fr1/room	**0.144**	**0.339**	0.201	0.430	0.158	0.421
fr2/large_no_loop	**0.187**	**0.317**	0.228	0.437	0.256	0.878

For the ATE and RPE, the median translation errors and the maximum translation errors are reported. It can be seen that for each error metric, both image-based approaches (VP and CKF) outperform the volumetric approach in all sequences. The proposed approach using frame prediction also performs better than the closest key-frame approach: reducing the amount of occlusions between the reference and the current view allows a better minimisation of the error in sensor space and reduces the maximum pose error and drift.

2) Qualitative results: The dense tracking and mapping algorithm has been successfully tested on a number of real scenes in real-time, ranging from small workspaces to large scale buildings. Figure 3 TOP shows a 3D reconstruction representing an entire floor of a building, reconstructed in real-time from a 100 meters trajectory using a hand-held RGB-D camera (Asus Xtion). The trajectory begins in the left alcove and the camera is moved across corridors and rooms, until the last alcove on the right. The environment contains many texture-less and geometrically unconstrained areas (corridors). Thanks to the bi-objective error minimisation, which combines a photometric minimisation and a geometric minimisation, the proposed approach is always able to robustly estimate an accurate pose. Figure 3 BOTTOM shows a side view of the final point cloud of Figure 3 TOP and demonstrates the very low drift of the approach since the entire reconstruction remains flat (floor). The final map, is only represented by 67 key-frames resulting in a point cloud of $20,369,566$ 3D points.

Figure 2 shows a synthesised global view of an office rendered using the proposed multi-key-frame 3D fusion approach using all the images of the graph. The rendered image is near photo-realistic which demonstrates the accuracy of the reconstruction pipeline. More detailed results can be found in the accompanying video (also available at: http://youtu.be/9p5IAkgh_U4).

TABLE V

RELATIVE FRAME-TO-FRAME POSE ERROR IN METERS (RPE).

Sequence	VP (proposed)		CKF		Kintu.	
	Median	Max	Median	Max	Median	Max
fr1/desk	**0.0055**	**0.0390**	0.0059	0.1899	0.0056	0.0655
fr2/desk	**0.0015**	**0.0098**	0.0018	0.0171	0.0025	0.0108
fr1/room	**0.0041**	**0.0275**	0.0046	0.1693	0.0045	0.0892
fr2/large_no_loop	**0.0075**	0.1620	0.0093	0.3324	0.0087	**0.1101**

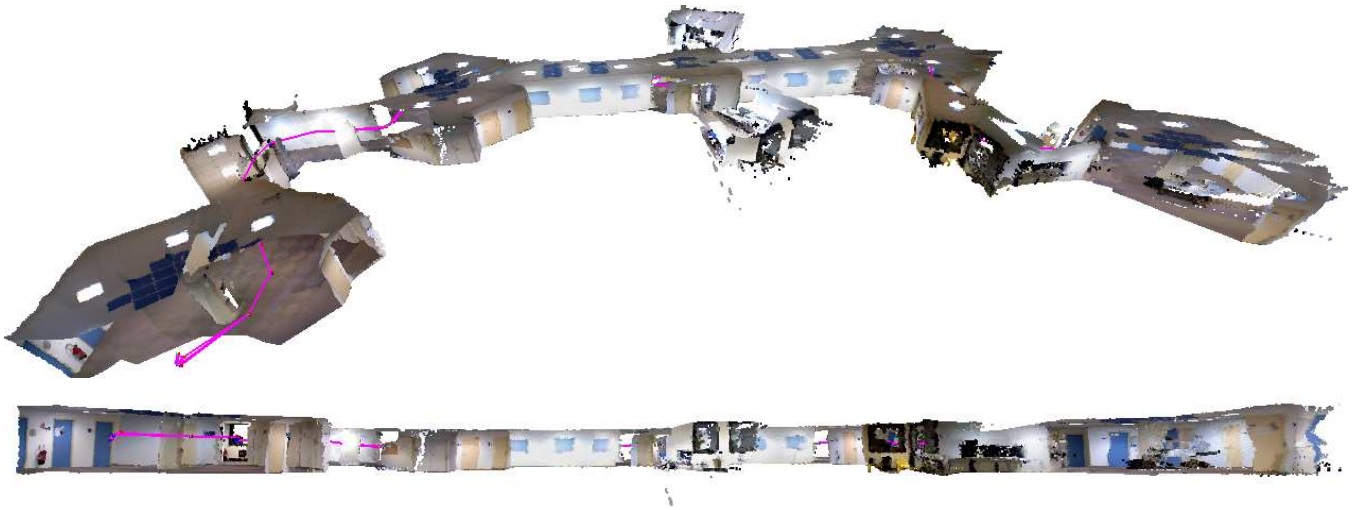

Fig. 3. Dense reconstruction of an entire floor obtained in real-time from a 100 meters trajectory containing 67 key-frames. The the graph of key-frames is superimposed on the images. TOP: a bird eye view of the reconstruction. BOTTOM: Side view of the reconstruction

VI. Conclusion

In conclusion this paper has investigated two different representations commonly used for dense real-time localisation and mapping. The advantages and disadvantages of each approach have been demonstrated and a model has been proposed to unify these approaches. The proposed model is based on an efficient hybrid intensity and point-to-plane ICP formulation. Multiple reference images stored in a graph have been used to provide efficient occlusion handling to a multi-key-frame approach. In that respect a fast surface triangulation/rasterisation method has been developed on the GPU and it has been shown to use relatively low memory and be able to map large scale buildings densely in real-time.

Future works will aim at detecting large loop closures using appearance-based techniques and correcting drift using pose-graph optimisation.

References

[1] P. Henry, M. Krainin, E. Herbst, X. Ren, and D. Fox, "Rgb-d mapping: Using depth cameras for dense 3d modeling of indoor environments," in *Int. Symp. on Experimental Robotics*, 2010.

[2] N. Engelhard, F. Endres, J. Hess, J. Sturm, and W. Burgard, "Real-time 3d visual slam with a hand-held rgb-d camera," in *RGB-D Work. on 3D Perception in Robotics*, 2011.

[3] J. Sturm, K. Konolige, C. Stachniss, and W. Burgard, "3d pose estimation, tracking and model learning of articulated objects from dense depth video using projected texture stereo," in *RSS Work. on RGB-D: Advanced Reasoning with Depth Cameras*, 2010.

[4] J. Stückler and S. Behnke, "Model learning and real-time tracking using multi-resolution surfel maps," in *AAAI Conf. on Artificial Intelligence*, 2012.

[5] Y. Furukawa and J. Ponce, "Accurate, dense, and robust multiview stereopsis," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 2010.

[6] B. Curless and M. Levoy, "A volumetric method for building complex models from range images," in *Annual Conf. on Computer graphics and interactive techniques*, 1996.

[7] A. I. Comport, E. Malis, and P. Rives, "Accurate quadrifocal tracking for robust 3d visual odometry," in *IEEE Int. Conf. on Robotics and Automation*, 2007.

[8] M. Meilland and A. I. Comport, "Super-resolution 3d tracking and mapping," in *IEEE Int. Conf. on Robotics and Automation*, 2013.

[9] M. Meilland, A. I. Comport, and P. Rives, "A spherical robot-centered representation for urban navigation," in *IEEE Int. Conf. on Intelligent Robots and Systems*, 2010.

[10] H. Roth and M. V. J. McDonald, "Moving volume kinectfusion," in *British Machine Vision Conf.*, 2012.

[11] T. Whelan, H. Johannsson, M. Kaess, J. Leonard, and J. McDonald, "Robust real-time visual odometry for dense RGB-D mapping," in *IEEE Int. Conf. on Robotics and Automation*, 2013.

[12] C. Connolly, "Cumulative generation of octree models from range data," in *IEEE Int. Conf. on Robotics and Automation*, 1984.

[13] C.-H. Chien, Y. B. Sim, and J. K. Aggarwal, "Generation of volume/surface octree from range data," in *Int. Conf. Conf. on Computer Vision and Pattern Recognition*, 1988.

[14] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohli, J. Shotton, S. Hodges, and A. Fitzgibbon, "Kinectfusion: Real-time dense surface mapping and tracking," in *Int. symposium on mixed and augmented reality*, 2011.

[15] C. Mei, G. Sibley, M. Cummins, P. Newman, and I. Reid, "A constant time efficient stereo slam system," in *British Machine Vision Conf.*, 2009.

[16] M. Zeng, F. Zhao, J. Zheng, and X. Liu, "A memory-efficient kinectfusion using octree," in *Int. Conf. on Computational Visual Media*, 2012.

[17] D. Damen, A. Gee, W. Mayol-Cuevas, and A. Calway, "Egocentric real-time workspace monitoring using an rgb-d camera," in *Int. Conf. on Intelligent Robots and Systems*, 2012.

[18] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "g2o: A General Framework for Graph Optimization," in *IEEE Int. Conf. on Robotics and Automation*, Shanghai, China, 2011.

[19] D. Nistér, O. Naroditsky, and J. Bergen, "Visual odometry," in *IEEE Int. Conf. on Computer Vision and Pattern Recognition*, vol. 1, July 2004, pp. 652–659.

[20] E. Mouragnon, M. Lhuillier, M. Dhome, F. Dekeyser, and P. Sayd, "Real time localization and 3d reconstruction," in *IEEE Int. Conf. on Computer Vision and Pattern Recognition*, 2006.

[21] K. Konolige and M. Agrawal, "Frameslam: from bundle adjustment to realtime visual mapping," *IEEE Trans. on Robotics*, vol. 24, no. 5, pp. 1066–1077, October 2008.

[22] M. Meilland, A. I. Comport, and P. Rives, "Dense visual mapping of large scale environments for real-time localisation," in *IEEE Int. Conf. on Intelligent Robots and Systems*, 2011.

[23] K. Khoshelham and S. O. Elberink, "Accuracy and resolution of kinect depth data for indoor mapping applications," *Sensors*, 2012.

[24] T. Tykkala, C. Audras, and A. I. Comport, "Direct iterative closest point for real-time visual odometry," in *ICCV Work. on Computer Vision in Vehicle Technology*, 2011.

[25] E. Malis, "Improving vision-based control using efficient second-order minimization techniques," in *IEEE Int. Conf. on Robotics and Automation*, may 2004, pp. 1843–1848.

[26] Z. Zhang, "Parameter Estimation Techniques: A Tutorial with Application to Conic Fitting," INRIA, Tech. Rep. RR-2676, Oct. 1995.

[27] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of rgb-d slam systems," in *Int. Conf. on Intelligent Robot Systems*, 2012.