Deformation-based Loop Closure for Large Scale Dense RGB-D SLAM

Thomas Whelan[1], Michael Kaess[2], John J. Leonard[2] and John McDonald[1]

***Abstract*— In this paper we present a system for capturing large scale dense maps in an online setting with a low cost RGB-D sensor. Central to this work is the use of an "as-rigid-as-possible" space deformation for efficient dense map correction in a pose graph optimisation framework. By combining pose graph optimisation with non-rigid deformation of a dense map we are able to obtain highly accurate dense maps over large scale trajectories that are both locally and globally consistent. With low latency in mind we derive an incremental method for deformation graph construction, allowing multi-million point maps to be captured over hundreds of metres in real-time. We provide benchmark results on a well established RGB-D SLAM dataset demonstrating the accuracy of the system and also provide a number of our own datasets which cover a wide range of environments, both indoors, outdoors and across multiple floors.**

I. INTRODUCTION

Simultaneous Localisation and Mapping (SLAM) is a key problem in the area of robotics that has been the focus of an enormous research effort for over twenty years. A robot's ability to create a map of an unknown environment and know its position within that map is crucial for intelligent autonomous operation. 2D SLAM, which allows a robot to map and localise on a plane, has always been a large focus in the robotics community, however in recent years 3D mapping has become a more extensively studied problem.

The release of the Microsoft Kinect and other RGB-D sensors has caused a surge in 3D perception research in the past few years. Previous to this, devices such as time of flight (TOF), stereo vision and 3D LIDAR sensors were required for 3D perception. The low cost of the Kinect sensor coupled with its high quality sensing capabilities has proven to be an attractive alternative to previous more expensive 3D sensing platforms. As a result of this many visual SLAM and 3D reconstruction systems relying purely on RGB-D sensing have been created in recent times.

One of the most notable RGB-D 3D reconstruction systems of recent times is KinectFusion [1], which enables dense volumetric modeling of a static scene in real-time at sub-centimetre resolution, although restricted to a fixed region in space. In previous work we proposed the Kintinuous system [2], which allows dense volumetric modeling over an extended area by virtually translating the volumetric model as the sensor moves. However this is an open-loop process which inevitably suffers from unbounded drift.

[1]T. Whelan and J. McDonald are with the Department of Computer Science, National University of Ireland Maynooth, Co. Kildare, Ireland. thomas.j.whelan at nuim.ie

[2]M. Kaess and J. J. Leonard are with Computer Science and Artificial Intelligence Laboratory (CSAIL), Massachusetts Institute of Technology (MIT), Cambridge, MA 02139, USA.

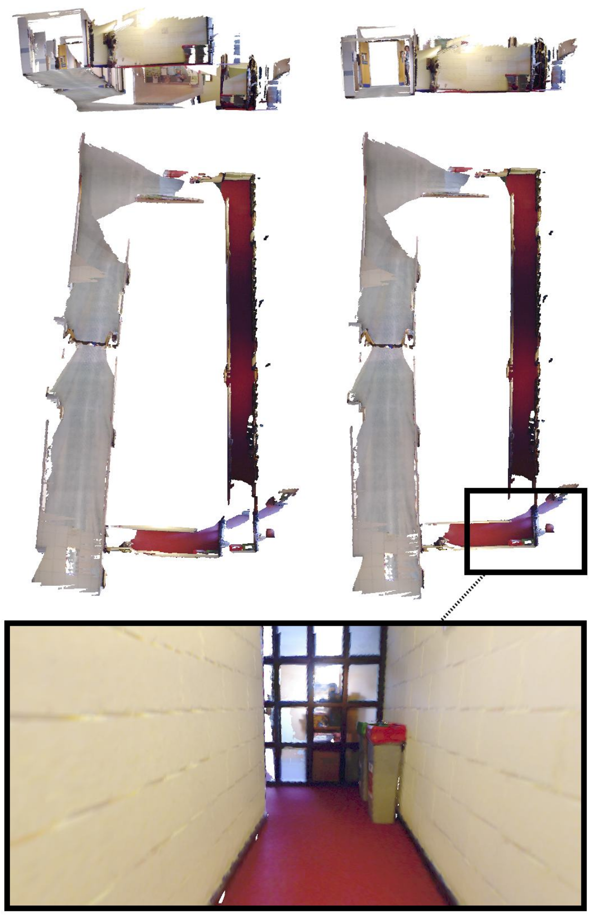

Fig. 1. Loop closed 49.6m camera trajectory containing approximately 1.3 million vertices at a resolution of 1.38cm; Effective volume mapping of 17,795m^3. The inset shows map consistency at the point of loop closure.

In this work we present a method for dealing with loop closures in a volume shifting-based mapping system like Kintinuous which takes advantage of camera pose graph optimisation and non-rigid space deformation. The result is a visual SLAM system which captures high fidelity dense maps in real-time with the local reconstruction quality of KinectFusion and the advantages of global consistency given by camera pose graph optimisation. We present quantitative results on the widely used Freiburg RGB-D benchmark [3] and also a number of datasets demonstrating the quality and scale at which the system can function.

II. RELATED WORK

One of the first approaches to RGB-D SLAM was that of Henry *et al.*, who used visual feature matching in conjunction

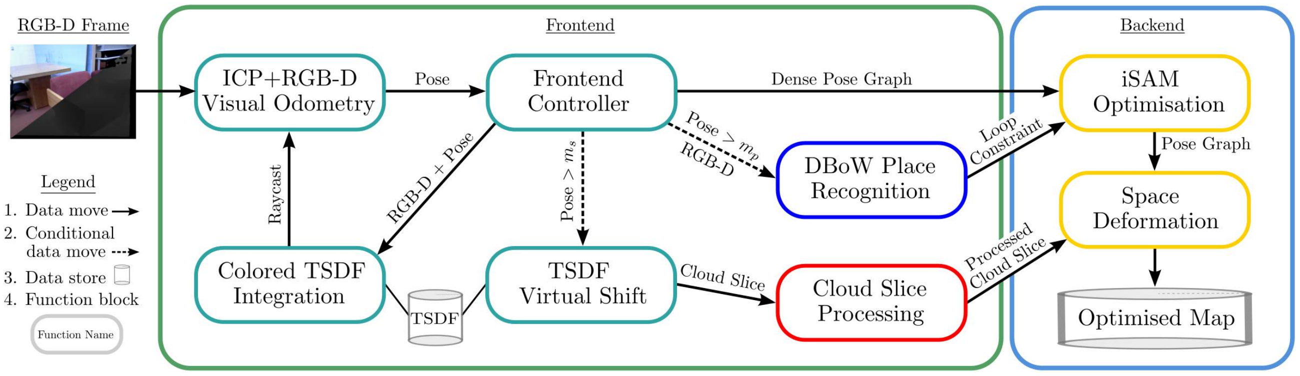

Fig. 2. System architecture diagram. Differently colored function blocks are executing asynchronously in separate CPU threads. The m_s quantity denotes the volume shifting threshold and m_p denotes the place recognition movement threshold.

with Generalised Iterative Closest Point (GICP) to build a pose graph and subsequently build an optimised surfel map of the environment [4]. Working in an offline manner, they contrast the use of pose graph optimisation versus sparse bundle adjustment (SBA) to minimise feature reprojection errors in a strictly rigid transformation framework. Similar work by Huang *et al.* computes a map by SBA as a post-processing step, by minimising rigid reprojection errors [5]. Recent work by Hu *et al.* and Lee *et al.* also attempts to minimise rigid reprojection error for map correction after optimisation [6], [7].

In the RGB-D SLAM system of Endres *et al.* visual features are used for camera pose estimation and global consistency is achieved using pose graph optimisation [8]. The map is represented by probabilistically reprojecting all point measurements into an octree-based volumetric map, provided by the OctoMap framework [9]. OctoMap has the advantage of taking measurement uncertainty into account, being space efficient and implicitly representing free and occupied space. However like most voxel representations integration of measurements (by raycasting) and non-rigid transformations are computationally expensive to perform.

The GPSlam algorithm of Pirker *et al.* uses sparse visual features in combination with a dense volumetric occupancy grid for the modeling of large environments [10]. Sliding window bundle adjustment is used with visual place recognition in a pose graph optimisation framework. Upon loop closure the occupancy grid is "morphed" into a globally consistent grid using a weighted average of the log-odds perceptions of each camera for each voxel. Audras *et al.* estimate a warping function using both geometric and photometric information for pose estimation but do not make use of a pose graph and rely on rigid reprojection to produce a 3D map reconstruction [11]. An octree-based multi-resolution surfel map representation is used by Stückler and Behnke which registers surfel maps for pose estimation and relies on pose graph optimisation for global consistency [12]. A globally consistent map is computed by fusing key views after graph optimisation has completed.

An independent method of extended KinectFusion was presented by Roth and Vona [13]. However no method for recovering the map is provided. Recent work by Zeng *et al.* replaces the explicit voxel representation of KinectFusion with an octree representation, which allows volumetric mapping of areas up to 8m×8m×8m area [14]. A drawback of this approach is that it increases the amount of possible drift within the volume and no method for correcting such drift is provided.

Many of the above techniques are capable of producing impressive globally consistent maps, however most are either unable to operate in real-time, efficiently incorporate large non-rigid updates to the map or provide an up-to-date optimised representation of the map at runtime. In contrast to this, the system we present operates in real-time, provides a means of efficiently updating the existing map with a non-rigid map deformation to reflect an optimised pose graph and preserves dense high quality small scale features. In the remainder of this paper we describe our approach followed by a set of quantitative and qualitative experimental results.

III. ARCHITECTURE

Our SLAM system is made up of two main components, the frontend (for camera tracking and surface extraction) and the backend (for pose graph and map optimisation). A detailed system architecture diagram is shown in Figure 2.

A. Frontend

The frontend used in our system is the dense Kintinuous mapping system [15]. The Kintinuous system is based on the KinectFusion system of Newcombe *et al.* [1]. KinectFusion is a GPU-based real-time dense mapping system that integrates all depth measurements into a volumetric data structure, known as the truncated signed distance function (TSDF) [16]. Camera pose estimation is then carried out via dense ICP between the current depth frame and a raycasted surface prediction from the TSDF. Kintinuous implements this functionally but in contrast to KinectFusion also allows the region in space which is being reconstructed to move with the camera trajectory. We use the robust visual odometry variant of Kintinuous that utilises both dense photometric and geometric information in camera pose estimation (henceforth referred to as ICP+RGB-D odometry) [15]. As discussed in our previous work on the volume shifting approach of Kintinuous [2], as the TSDF volume moves, the region of the surface that leaves the volume is extracted in point cloud form. Hence along a camera trajectory there exists a stream of "cloud slices". Each cloud slice has an associated camera pose; the pose of the camera at the time of the slice's creation. This relationship is shown in Figure 3 and is one of importance as we later expand on in Section IV-C. The final component which lies in the frontend is a visual place recognition module relying on the DBoW place recognition system [17], which detects visual loop closures and computes appropriate relative camera pose constraints.

B. Backend

We propose a novel optimisation backend for deformation-based dense SLAM, comprised of incremental pose graph optimisation coupled with incremental non-rigid dense map optimisation. We use iSAM [18] to optimise a dense every-frame pose graph according to loop closure constraints provided by our place recognition module. The optimised dense camera trajectory is then used in conjunction with matched visual features to constrain a non-rigid space deformation of the map. We adapt the embedded deformation approach of Sumner *et al.* [19] to apply it to large scale dense maps captured with the Kintinuous frontend and derive efficient incremental methods to prepare the map for deformation.

IV. APPROACH

In this section we provide a detailed description of each component involved in our SLAM pipeline including pose graph representation, place recognition and loop closure, deformation graph construction and map optimisation.

A. Pose Graph

In contrast to a number of existing SLAM systems a dense every-frame pose graph is used, as opposed to a keyframe-based pose graph. Given the robustness of the camera pose estimation coupled with the resolution of the reconstructed surface we choose to maintain as much information as possible for map optimisation. A camera pose P_i is composed of a rotation $P_{i_\mathbf{R}} \in \mathbb{SO}(3)$, a translation $P_{i_\mathbf{t}} \in \mathbb{R}^3$ and timestamp i. A camera pose P_i is estimated for every processed frame. Some camera poses also have an associated cloud slice as shown in Figure 3 where the relationship between pose P_γ and cloud slice $\mathcal{C}_j$ is shown. This provides a useful association between camera poses and the extracted surface, capturing both temporal and spatial proximity. We define $\mathcal{C}_{j_P}$ to be the pose associated with cloud slice $\mathcal{C}_j$.

Referring to our previous work on dense visual odometry [15], we can approximate the constraint uncertainty with the Hessian as $\Sigma = (\mathbf{J}^\top \mathbf{J})^{-1}$, where $\mathbf{J}$ is the measurement Jacobian. We also experimented using uniform (identity) covariances for every constraint ($\Sigma = \mathbf{I}$).

B. Place Recognition

We use Speeded Up Robust Feature (SURF) descriptors with the bag-of-words-based DBoW loop detector for place recognition [17]. Adding every RGB-D frame to the place recognition system is non-optimal, therefore we utilise a movement metric sensitive to both rotation and translation which indicates when to add a new frame to the place recognition system. Defining $\mathbf{r}(\mathbf{R}) : \mathbb{SO}(3) \rightarrow \mathbb{R}^3$ to provide the rotation vector form of some rotation matrix $\mathbf{R}$, we compute a hybrid movement distance between two poses a and b that compounds both translation and rotation into a single quantity as:

$$m_{ab} = \left\| \mathbf{r}(P_{a_\mathbf{R}}^{-1} P_{b_\mathbf{R}}) \right\|_2 + \left\| P_{a_\mathbf{t}} - P_{b_\mathbf{t}} \right\|_2 \tag{1}$$

For each frame we evaluate the movement distance between the current frame pose and the pose of the last frame added to the place recognition system according to Equation 1. If this metric is above some threshold m_p, a new frame is added. Empirically we found $m_p = 0.3$ provides good performance.

Defining the image space domain as $\Omega \subset \mathbb{N}^2$, an RGB-D frame I_i is composed of an RGB image $I_{i_\mathbf{rgb}} : \Omega \rightarrow \mathbb{R}^3$, a depth image $I_{i_\mathbf{d}} : \Omega \rightarrow \mathbb{R}$ and a timestamp i. Upon receiving a new RGB-D frame I_i the place recognition module first computes a set of SURF keypoints and associated descriptors $S_i \in \Omega \times \mathbb{R}^{64}$ for that frame. These features are cached in memory for future queries. The depth image $I_{i_\mathbf{d}}$ is also cached, however to ensure low memory usage it is compressed on-the-fly using lossless compression [20]. Following this, the existing bag-of-words descriptor database is queried. If a match is found the SURF keypoints and descriptors S_m and depth data $I_{m_\mathbf{d}}$ (on-the-fly decompressed) for the matched image are retrieved for constraint computation. A number of validation steps are performed to minimise the chance of false positives. Overall we choose very high threshold parameters to prevent any false place recognitions in our experiments. They are as follows:

1) SURF Correspondence Threshold: Given S_i and S_m we find correspondences by a k-nearest neighbour search in the SURF descriptor space. We use the Fast Library for Approximate Nearest Neighbors (FLANN) to perform this search and populate a set of valid correspondences $V \in \Omega \times \Omega$, thresholding matches using an L_2-norm between descriptors in $\mathbb{R}^{64}$. We discard the loop closure candidate if $|V|$ is less than some threshold; a value of 35 has been found to provide adequate performance in our experiments.

2) RANSAC Transformation Estimation: Given V and $I_{m_\mathbf{d}}$, we first attempt to approximate a 6-DOF relative transformation between the camera poses of frames i and m using a RANSAC-based 3-point algorithm [21]. Given a calibrated camera intrinsics matrix $\mathbf{K}$, depth image $I_{m_\mathbf{d}}$ and keypoint location $\mathbf{p} \in \Omega$, we can compute the 3D back-projection $\mathbf{p}_w = I_{m_\mathbf{d}}(\mathbf{p})\mathbf{K}^{-1}(\mathbf{p}|1)^\top$, where $\mathbf{p}_w \in \mathbb{R}^3$. Each matching keypoint in V is back-projected from image m to a 3D point, transformed according to the current RANSAC model and reprojected into the image plane of frame i (using standard perspective projection onto an image plane) where the reprojection error quantified by the L_2-norm in $\mathbb{R}^2$ is used for outlier detection. Empirically we chose a maximum reprojection error of 2.0 for inliers. If the percentage of inliers for the RANSAC estimation is below 25% the loop closure is discarded. Otherwise, we refine the estimated

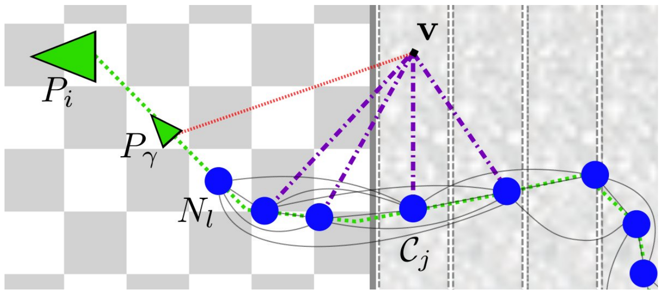

Fig. 3. Two-dimensional example showing the current position of the TSDF shifting volume as a checkerboard pattern and the previously extracted cloud slices as textured columns. Also shown is the dense pose graph as small green points as well as a pose P_γ which caused a volume shift. The association between P_γ and the extracted cloud slice is shown with a dotted red line. A $k = 4$ connected sequential deformation graph is also shown, demonstrating the back-traversal vertex association algorithm on a random vertex $\mathbf{v}$.

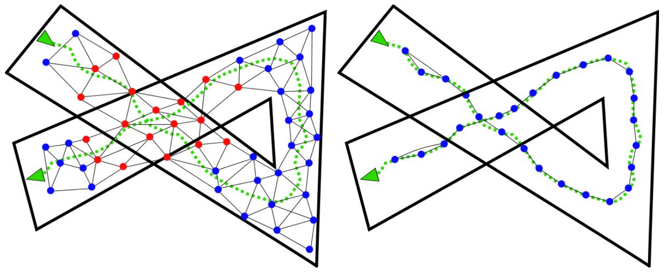

Fig. 4. Two-dimensional example of deformation graph construction. On the left a spatially-constrained graph is constructed over a pre-loop closure map suffering from significant drift. The nodes highlighted in red are connected to nodes which belong in potentially completely unrelated areas of the map. On the right our incremental sampling and connectivity strategy is shown (two-nearest neighbours for simplicity) which samples and connects nodes along the dense pose graph, preventing unrelated areas of the map being connected by the deformation graph.

transformation by minimising all inlier feature reprojection errors in a Levenberg-Marquardt optimisation.

3) Point Cloud ICP: At this point only candidate loop closures with strong geometrically consistent visual feature correspondences remain. As a final step we perform a non-linear ICP step between $I_{i_\mathbf{d}}$ and $I_{m_\mathbf{d}}$. Firstly we back-project each point in both depth images to produce two point clouds. In order to speed up the computation, we carry out a uniform downsampling of each point cloud in $\mathbb{R}^3$ using a voxel grid filter. Finally, using the RANSAC approximate transformation estimate as an initial guess, we iteratively minimise nearest neighbour correspondence distances between the two point clouds using a Levenberg-Marquardt optimisation. We accept the final refined transformation if the mean L_2^2-norm of all correspondence errors is below a threshold. Typically we found a threshold of 0.01 to provide good results.

Once a loop closure candidate has passed all of the described tests, the relative transformation constraint between the two camera poses is added to the dense pose graph maintained by the iSAM module. Section IV-D describes how this constraint is used to update the map.

C. Space Deformation

Our approach to non-rigid space deformation of the map is based on the embedded deformation approach of Sumner *et al.* [19]. Their system allows deformation of open triangular meshes and point clouds; no connectivity information is required as is the case with many deformation algorithms [22], [23]. Exploiting this characteristic, Chen *et al.* applied embedded deformation to automatic skeletonised rigging and real-time animation of arbitrary objects in their KinÊtre system [24]. Next we describe our adaptation of Sumner *et al.*'s work to apply to large scale dense maps with a focus on automatic incremental deformation graph construction.

1) Deformation Graph: Sumner *et al.* propose the use of a deformation graph to facilitate space deformation of a set of vertices. A deformation graph is composed of nodes and edges spread across the surface to be deformed. Each node N_l has an associated position $N_{l_\mathbf{g}} \in \mathbb{R}^3$ and set of neighbouring nodes $\mathcal{N}(N_l)$. The neighbours of each node are what make up the edges of the graph. Each node also stores an affine transformation in the form of a 3×3 matrix $N_{l_\mathbf{R}}$ and a 3×1 vector $N_{l_\mathbf{t}}$, initialised by default to the identity and $(0, 0, 0)^\top$ respectively. The effect of this affine transformation on any vertex which that node influences is centered at the node's position $N_{l_\mathbf{g}}$.

2) Incremental Graph Construction: The original work of Sumner *et al.* relies on a uniform sampling of the vertices in $\mathbb{R}^3$ to construct the deformation graph [19]. Chen *et al.* substitute this with a method that uses a 5D orientation-aware sampling strategy based on the Mahalanobis distance between surface points in order to prevent links in the graph between physically unrelated areas of the model [24]. Neither strategy is appropriate in a dense mapping context as drift in odometry estimation before loop detection may cause unrelated areas of the map to completely overlap in space. This issue also arises in determining connectivity of the graph. Applying sampling and connectivity strategies that are only spatially aware can result in links between completely unrelated areas of the map, as shown in Figure 4. For this reason we derive a sampling and connectivity strategy that exploits the dense camera pose graph for deformation graph construction and connection. The method is computationally efficient and incremental, enabling real-time execution. Our sampling strategy is listed in Algorithm 1.

We connect deformation graph nodes returned by our sampling strategy in a sequential manner, following the temporal order of the pose graph itself. That is to say our set of graph nodes N is ordered. We sequentially connect nodes up to a value k. We use $k = 4$ in all of our experiments. For example, a node l will be connected to nodes $(l\pm1, l\pm2)$. We show $k = 2$ connectivity in Figure 4. Note the connectivity of end nodes which maintains k-connectivity.

3) Incremental Vertex Weighting: Each vertex $\mathbf{v}$ has a set of influencing nodes in the deformation graph $\mathcal{N}(\mathbf{v})$. The deformed position of a vertex is given by [19]:

$$\hat{\mathbf{v}} = \sum_{k \in \mathcal{N}(\mathbf{v})} w_k(\mathbf{v}) \left[N_{k_\mathbf{R}}(\mathbf{v} - N_{k_\mathbf{g}}) + N_{k_\mathbf{g}} + N_{k_\mathbf{t}} \right] \qquad (2)$$

where $w_k(\mathbf{v})$ is defined as (all k summing to 1):

$$w_k(\mathbf{v}) = (1 - \left\| \mathbf{v} - N_{k_\mathbf{g}} \right\|_2 / d_{max})^2 \qquad (3)$$

Algorithm 1: Incremental Deformation Node Sampling

```
Input: $P$ dense camera pose graph
       $i$ pose id of last added node
       $d_p$ pose sampling rate
Output: $N$ set of deformation graph nodes
do
    $l \leftarrow |N|$
    if $l = 0$ then
        $N_{l_\mathbf{g}} \leftarrow P_{0_\mathbf{t}}$
        $l \leftarrow l + 1$
        $i \leftarrow 0$
    $P_{last} \leftarrow P_i$
    for $i$ to $|P|$ do
        if $\left\|P_{i_\mathbf{t}} - P_{last_\mathbf{t}}\right\|_2 > d_p$ then
            $N_{l_\mathbf{g}} \leftarrow P_{i_\mathbf{t}}$
            $l \leftarrow l + 1$
            $P_{last} \leftarrow P_i$
end
```

Here d_{max} is the Euclidean distance to the $k+1$-nearest node of $\mathbf{v}$. In previous work based on this technique the sets $\mathcal{N}(\mathbf{v})$ for each vertex are computed in batch using a k-nearest neighbour technique. Again, being based on spatial constraints alone this method fails in the example shown in Figure 4. To overcome this issue we derive an algorithm that assigns nearest neighbour nodes to each vertex using a greedy back-traversal of the sampled pose graph nodes.

Algorithm 2: Back-Traversal Vertex Association

```
Input: $C$ cloud slices
       $N$ set of deformation graph nodes
       $b_p$ number of poses to traverse back
Output: $\mathcal{N}(\mathbf{v})$ for each $\mathbf{v}$
do
    foreach $C_j$ do
        foreach $\mathbf{v} \in C_j$ do
            $l \leftarrow \text{binary\_search\_closest}(C_{j_P}, N)$
            $N' \leftarrow \emptyset$
            $n \leftarrow 0$
            for $i \leftarrow 0$ to $b_p$ do
                $N'_n \leftarrow N_l$
                $n \leftarrow n + 1$
                $l \leftarrow l - 1$
            $\text{sort\_by\_distance}(N', \mathbf{v})$
            $\mathcal{N}(\mathbf{v}) \leftarrow N'_{1 \rightarrow k}$
end
```

Referring back to Figure 3, we recall that each pose that causes a volume shift has an associated set of vertices contained within a cloud slice. We can exploit the inverse mapping of this association to map each vertex onto a single pose in the dense pose graph. However, the associated pose is at least a distance of half the TSDF volume size away from the vertex, which is not ideal for space deformation. In order to pick sampled pose graph nodes for each vertex that are spatially and temporally optimal, we use the closest sampled pose to the associated cloud slice pose as a starting point to traverse back through the sampled pose graph nodes to populate a set of candidate nodes. From these candidates the k-nearest neighbours of the vertex are chosen. We list the algorithm for this procedure in Algorithm 2 and provide a visual example in Figure 3.

The per-vertex node weights can be computed within the back-traversal algorithm, which itself can be carried out incrementally online while the frontend volume shifting component provides new cloud slices. The ability to avoid computationally expensive batch steps for deformation graph construction and per-vertex weighting by using incremental methods is the key to allowing low latency online map optimisation at any time.

D. *Optimisation*

On acceptance of a loop closure constraint as described in Section IV-B we perform two optimisation steps, firstly on the dense pose graph and secondly on the dense vertex map. The dense pose graph optimisation provides the measurement constraints for the dense map deformation optimisation in place of user specified constraints that were necessary in Sumner *et al.*'s original approach [19]. Dense pose graph optimisation is carried out using the iSAM framework [18]. Although we are optimising a massive number of variables by using a dense every-frame pose graph, we benefit from the sparse linear algebra representation used internally in iSAM, such that execution time is reasonable.

1) Map Deformation: Sumner *et al.* define three cost functions over the deformation graph and user constraints to optimise the set of affine transformations over all graph nodes N. The first maximises rigidity in the deformation:

$$E_{rot} = \sum_l \left\|N_{l_\mathbf{R}}^\top N_{l_\mathbf{R}} - \mathbf{I}\right\|_F^2 \tag{4}$$

Where Equation 4 is the alternative Frobenius-norm form provided by Chen *et al.* [24]. The second is a regularisation term that ensures a smooth deformation across the graph:

$$E_{reg} = \sum_l \sum_{n \in \mathcal{N}(N_l)} \left\|N_{l_\mathbf{R}}(N_{n_\mathbf{g}} - N_{l_\mathbf{g}}) + N_{l_\mathbf{g}} + N_{l_\mathbf{t}} - (N_{n_\mathbf{g}} + N_{n_\mathbf{t}})\right\|_2^2 \tag{5}$$

The third is a constraint term that minimises the error on a set of user specified vertex position constraints U, where a given constraint $U_p \in \mathbb{R}^3$ and $\phi(\mathbf{v})$ is the result of applying Equation 2 to $\mathbf{v}$:

$$E_{con} = \sum_p \left\|\phi(\mathbf{v}) - U_p\right\|_2^2 \tag{6}$$

We link the optimised dense pose graph to the map deformation through the E_{con} cost function. With P being the pose graph before loop constraint integration we set P' to be the optimised pose graph returned from iSAM. We then add each of the dense camera pose translations to the deformation cost as if they were user specified vertex constraints, redefining Equation 6 as:

$$E_{con_P} = \sum_i \left\|\phi(P_{i_\mathbf{t}}) - P'_{i_\mathbf{t}}\right\|_2^2 \tag{7}$$

A dense constraint distribution across the surface obtained from this parameterisation aids in constraining both surface translation and orientation. However at some points the

TABLE I

ROOT MEAN SQUARED ABSOLUTE TRAJECTORY ERROR IN METRES ON EVALUATED DATASETS WITH VARIOUS UNCERTAINTY ESTIMATES.

Dataset	Estimate	
	Uniform	Hessian
fr1/room	0.083	0.078
fr1/desk2	0.088	0.075
fr3/longOffice	0.054	0.029
fr3/noStructureTextureLoop	0.037	0.031

surface orientation may not be well constrained. In order to overcome this issue we add additional vertex constraints between the unoptimised and optimised 3D back-projections of each of the matched inlier SURF keypoints detected in Section IV-B, where P_i is the camera pose of the matched loop closure frame:

$$E_{surf} = \sum_q \left\| \phi((P_{i_\mathbf{R}} V_q) + P_{i_\mathbf{t}}) - ((P'_{i_\mathbf{R}} V_q) + P'_{i_\mathbf{t}}) \right\|_2^2 \quad (8)$$

The final total cost function is defined as:

$$w_{rot} E_{rot} + w_{reg} E_{reg} + w_{con_P} E_{con_P} + w_{surf} E_{surf} \quad (9)$$

With $w_{rot} = 1$, $w_{reg} = 10$, $w_{con_P} = 100$ and $w_{surf} = 100$, we minimise this cost function using the iterative Gauss-Newton algorithm choosing weighting values in line with those used in [19]. As highlighted by Sumner *et al.*, the Jacobian matrix in this problem is sparse, enabling the use of sparse linear algebra libraries for efficient optimisation. We use the CHOLMOD library to perform sparse Cholesky factorization and efficiently solve the system [25]. We then apply the optimised deformation graph N to all vertices over all cloud slices $\mathcal{C}$ in parallel across multiple CPU threads. As described in previous work we compute an incremental mesh surface representation of the cloud slices as they are produced by the frontend [2]. We use an incremental variant of Marton *et al.*'s fast triangulation algorithm to maintain edge connectivity between cloud slices [26]. The incremental mesh can be deformed by applying the deformation graph to its vertices. In our experience an incremental mesh typically contains more minuscule holes than a batch mesh, which in path planning is functionally almost identical but less visually appealing. In all results we show the batch mesh computed over the set of optimised vertices.

V. RESULTS

We evaluate our system both quantitatively and qualitatively, demonstrating strong performance in trajectory estimation, map quality and computational performance.

A. Quantitative Performance

To evaluate camera trajectory estimation we present results on the widely used RGB-D benchmark of Sturm *et al.* [3]. We evaluated four datasets with results shown in Table I. Figure 5 shows the measured trajectory errors using the hessian uncertainty method. Our results show that regardless of the method for uncertainty used, consistent performance is achieved.

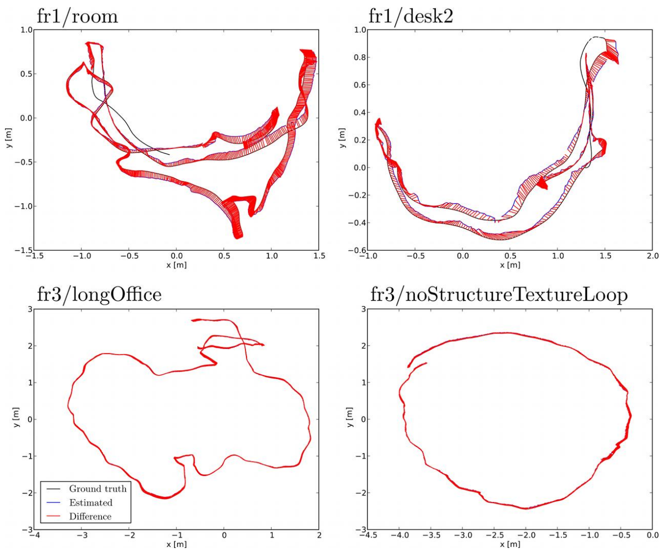

Fig. 5. Absolute trajectory errors on the four evaluated RGB-D benchmarks using the hessian uncertainty estimate.

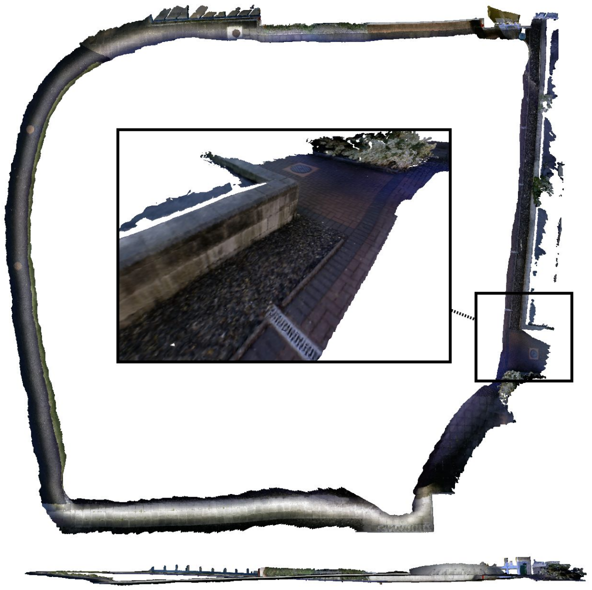

Fig. 6. Large outdoor dataset. Inset shows brickwork is clearly visible.

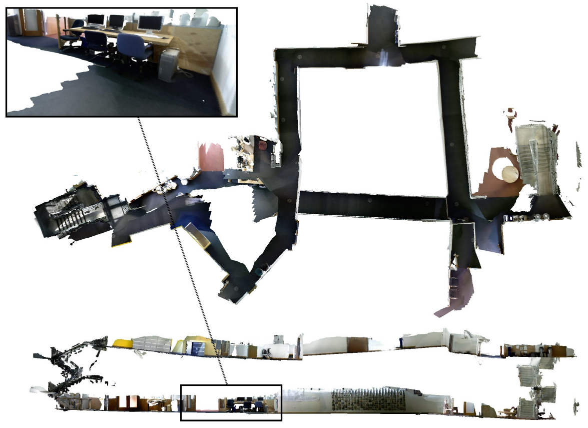

Fig. 7. Dataset composed of two floors. Inset shows everyday objects such as chairs and computers are captured in high detail.

TABLE II

STATISTICS ON FOUR HANDHELD DATASETS CAPTURED OVER A WIDE VARIETY OF ENVIRONMENTS.

Dataset	Data			
	Length	Vertices	Volume	Figure
Indoors	49.6m	1,301,593	17,795m^3	1
Outdoors	152.6m	2,460,663	28,836m^3	6
Two floors	173.9m	3,285,373	38,500m^3	7
In/outdoors	316.6m	5,161,204	60,483m^3	8

B. Qualitative Performance

We present a number of datasets collected in a handheld fashion that span a wide range of scales over static scenes. Statistics on each dataset are shown in Table II. These results demonstrate the viability of our system for use over large scale trajectories, indoors, outdoors (when absence of sunlight permits structured light-based sensing) and across multiple floors. We compute an effective volume statistic for each dataset as the number of vertices times the voxel resolution of the frontend volumetric TSDF. A volume size of 6m was used in all datasets except the "Indoors" dataset (Figure 1), where a 7m volume was used to provide a larger overlap between the TSDF volume and camera field of view. In all experiments a 512^3 voxel TSDF was used with an ASUS Xtion Pro Live RGB-D camera at a frame rate of 30Hz. We provide additional qualitative results and visualisations of the deformation process and final dense maps in a video available at `http://www.youtube.com/watch?v=MNw-GeHHSuA`. Accurate large scale dense surface reconstruction ground truth appears to be an open problem (as opposed to *trajectory* ground truth), which makes it difficult to evaluate whether a single pass with deformation approach is more or less accurate than a two pass approach that reruns and reintegrates the RGB-D data based on an optimised trajectory. The latter approach no longer benefits from explicit frame-to-model tracking, which has been shown to provide superior tracking performance [1].

C. Computational Performance

As discussed in previous work the frontend runs at camera frame rate, 30Hz [15]. As a measure of computational performance in the context of an online SLAM system we measure the latency of the system. That is, how long it takes for 1) a loop closure to be recognised when one is encountered and 2) map deformation to be completed. Table III shows execution time statistics on our test platform, a standard desktop PC running Ubuntu 12.04 with an Intel Core i7-3960X CPU at 3.30GHz, 16GB of RAM and an nVidia GeForce 680GTX GPU with 2GB of memory. The results show that the frontend scales very well, which is expected with the DBoW loop detector [17]. Both iSAM and the deformation processes scale with the size of the map and number of poses, however given the size and detail of the maps we consider the detection-to-correction map loop closure latency to be acceptable for online operation.

TABLE III

COMPUTATIONAL PERFORMANCE STATISTICS ON FOUR DATASETS.

Quantities	Datasets			
	Indoors	Outdoors	Two floors	In/outdoors
DBoW images	306	1181	1570	2626
Poses	3007	5295	12949	25377
Nodes	56	176	189	363
Vertices	1,355,059	2,801,265	3,878,785	6,356,911
Process	Timings (ms)			
Frontend	681	596	820	520
iSAM	364	1341	2676	7487
Deformation	393	918	2230	4463
Total latency	1438	2855	5726	12470

VI. CONCLUSION

In this paper we have presented a novel SLAM system that makes use of non-rigid map deformations for map correction during loop closures. With online operation in mind we have presented new methods for constructing a deformation graph incrementally in real-time and demonstrated the system's ability to non-rigidly correct multi-million vertex maps in a matter of seconds. In future work we aim to better model the dense visual odometry uncertainty, look at issues with redundancy or "over-representation" in the map and scalability above hundreds of metres.

VII. ACKNOWLEDGEMENTS

Research presented in this paper was funded by a Strategic Research Cluster grant (07/SRC/I1168) by Science Foundation Ireland under the Irish National Development Plan, the Embark Initiative of the Irish Research Council and by ONR grants N00014-10-1-0936, N00014-11-1-0688, N00014-12-1-0093, and N00014-12-10020. The authors would also like to thank Guillaume Gales and Richard H. Middleton.

REFERENCES

[1] 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 *Proc. of the 2011 10th IEEE Int. Symposium on Mixed and Augmented Reality*, ISMAR '11, (Washington, DC, USA), pp. 127–136, 2011.

[2] T. Whelan, J. McDonald, M. Kaess, M. Fallon, H. Johannsson, and J. Leonard, "Kintinuous: Spatially Extended KinectFusion," in *3rd RSS Workshop on RGB-D: Advanced Reasoning with Depth Cameras*, (Sydney, Australia), July 2012.

[3] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of RGB-D SLAM systems," in *Proc. of the Int. Conf. on Intelligent Robot Systems (IROS)*, October 2012.

[4] P. Henry, M. Krainin, E. Herbst, X. Ren, and D. Fox, "RGB-D mapping: Using Kinect-style depth cameras for dense 3D modeling of indoor environments," *The Int. Journal of Robotics Research*, 2012.

[5] A. S. Huang, A. Bachrach, P. Henry, M. Krainin, D. Maturana, D. Fox, and N. Roy, "Visual odometry and mapping for autonomous flight using an RGB-D camera," in *Int. Symposium on Robotics Research (ISRR)*, (Flagstaff, Arizona, USA), August 2011.

[6] G. Hu, S. Huang, L. Zhao, A. Alempijevic, and G. Dissanayake, "A robust RGB-D SLAM algorithm," in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pp. 1714 –1719, October 2012.

[7] D. Lee, H. Kim, and H. Myung, "GPU-based real-time RGB-D 3D SLAM," in *Ubiquitous Robots and Ambient Intelligence (URAI), 2012 9th International Conference on*, pp. 46 –48, November 2012.

[8] F. Endres, J. Hess, N. Engelhard, J. Sturm, D. Cremers, and W. Burgard, "An evaluation of the RGB-D SLAM system," in *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, (St. Paul, MA, USA), May 2012.

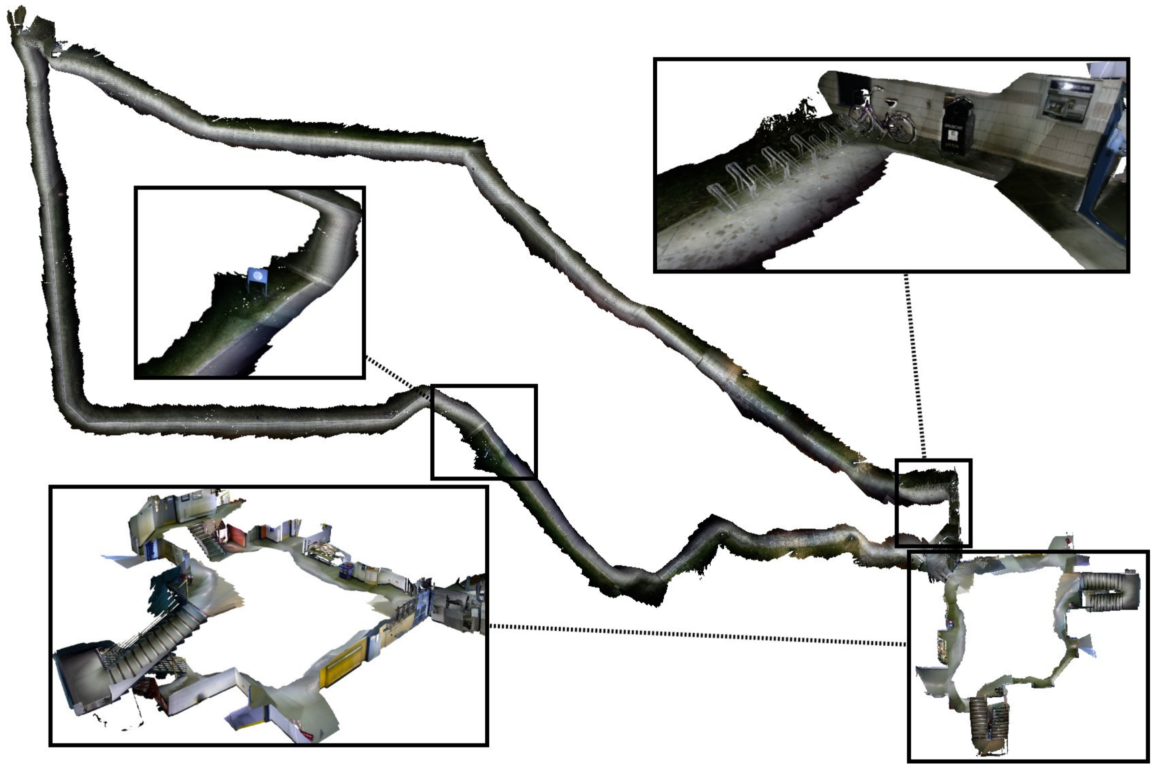

Fig. 8. Large indoor and outdoor dataset made up of over five million vertices. Insets show the high fidelity of small scale features in the map.

[9] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: An efficient probabilistic 3D mapping framework based on octrees," *Autonomous Robots*, 2013.

[10] K. Pirker, M. Rüther, G. Schweighofer, and H. Bischof, "GPSlam: Marrying sparse geometric and dense probabilistic visual mapping," in *Proc. of the British Machine Vision Conf.*, pp. 115.1–115.12, 2011.

[11] C. Audras, A. I. Comport, M. Meilland, and P. Rives, "Real-time dense RGB-D localisation and mapping," in *Australian Conf. on Robotics and Automation*, (Monash University, Australia), December 2011.

[12] J. Stückler and S. Behnke, "Integrating depth and color cues for dense multi-resolution scene mapping using RGB-D Cameras," in *Proc. of the IEEE Int. Conf. on Multisensor Fusion and Information Integration (MFI)*, (Hamburg, Germany), September 2012.

[13] H. Roth and M. Vona, "Moving volume KinectFusion," in *British Machine Vision Conf. (BMVC)*, (Surrey, UK), September 2012.

[14] M. Zeng, F. Zhao, J. Zheng, and X. Liu, "A Memory-Efficient Kinect-Fusion Using Octree," in *Computational Visual Media*, vol. 7633 of *Lecture Notes in Computer Science*, pp. 234–241, Springer, 2012.

[15] T. Whelan, H. Johannsson, M. Kaess, J. Leonard, and J. McDonald, "Robust real-time visual odometry for dense RGB-D mapping," in *IEEE Intl. Conf. on Robotics and Automation, ICRA*, (Karlsruhe, Germany), May 2013.

[16] B. Curless and M. Levoy, "A volumetric method for building complex models from range images," in *Proc. of the 23rd annual Conf. on Computer graphics and interactive techniques - SIGGRAPH '96*, (New York, New York, USA), pp. 303–312, August 1996.

[17] D. Galvez-Lopez and J. D. Tardos, "Real-time loop detection with bags of binary words," in *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pp. 51 –58, September 2011.

[18] M. Kaess, A. Ranganathan, and F. Dellaert, "iSAM: Incremental smoothing and mapping," *IEEE Transactions on Robotics (TRO)*, vol. 24, pp. 1365–1378, December 2008.

[19] R. W. Sumner, J. Schmid, and M. Pauly, "Embedded deformation for shape manipulation," in *ACM SIGGRAPH 2007 papers*, SIGGRAPH '07, (New York, NY, USA), ACM, 2007.

[20] P. Deutsch and J.-L. Gailly, "Zlib compressed data format specification version 3.3," 1996.

[21] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, vol. 24, June 1981.

[22] K. S. Karan, "Skinning characters using surface-oriented free-form deformations," in *In Graphics Interface 2000*, pp. 35–42, 2000.

[23] A. Jacobson and O. Sorkine, "Stretchable and twistable bones for skeletal shape deformation," *ACM Transactions on Graphics (proceedings of ACM SIGGRAPH ASIA)*, vol. 30, no. 6, pp. 165:1–165:8, 2011.

[24] J. Chen, S. Izadi, and A. Fitzgibbon, "KinÊtre: animating the world with the human body," in *Proceedings of the 25th annual ACM symposium on User interface software and technology*, UIST '12, (New York, NY, USA), pp. 435–444, ACM, 2012.

[25] T. A. Davis and W. W. Hager, "Modifying a sparse Cholesky factorization," *SIAM J. Matrix Anal. Appl.*, vol. 20, May 1999.

[26] Z. C. Marton, R. B. Rusu, and M. Beetz, "On fast surface reconstruction methods for large and noisy datasets," in *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, (Kobe, Japan), May 2009.