ElasticFusion: Dense SLAM Without A Pose Graph

Thomas Whelan*, Stefan Leutenegger*, Renato F. Salas-Moreno†, Ben Glocker† and Andrew J. Davison*
*Dyson Robotics Laboratory at Imperial College, Department of Computing, Imperial College London, UK
†Department of Computing, Imperial College London, UK
{t.whelan,s.leutenegger,r.salas-moreno10,b.glocker,a.davison}@imperial.ac.uk

***Abstract*—We present a novel approach to real-time dense visual SLAM. Our system is capable of capturing comprehensive dense globally consistent surfel-based maps of room scale environments explored using an RGB-D camera in an incremental online fashion, without pose graph optimisation or any post-processing steps. This is accomplished by using dense frame-to-model camera tracking and windowed surfel-based fusion coupled with frequent model refinement through non-rigid surface deformations. Our approach applies local model-to-model surface loop closure optimisations as often as possible to stay close to the mode of the map distribution, while utilising global loop closure to recover from arbitrary drift and maintain global consistency.**

I. Introduction

In dense 3D SLAM, a space is mapped by fusing the data from a moving sensor into a representation of the continuous surfaces it contains, permitting accurate viewpoint-invariant localisation as well as offering the potential for detailed semantic scene understanding. However, existing dense SLAM methods suitable for incremental, real-time operation struggle when the sensor makes movements which are both of extended duration and often criss-cross loop back on themselves. Such a trajectory is typical if a non-expert person with a handheld depth camera were to scan in a room with a loopy "painting" motion; or would also be characteristic of a robot aiming to explore and densely map an unknown environment.

SLAM algorithms have too often targeted one of two extremes; (i) either extremely loopy motion in a very small area (*e.g.* MonoSLAM [4] or KinectFusion [15]) or (ii) "corridor-like" motion on much larger scales but with fewer loop closures (*e.g.* McDonald *et al.* [13] or Whelan *et al.* [25]). In sparse feature-based SLAM, it is well understood that loopy local motion can be dealt with either via joint probabilistic filtering [3], or in-the-loop joint optimisation of poses and features (*bundle adjustment*) [11]; and that large scale loop closures can be dealt with via partitioning of the map into local maps or keyframes and applying pose graph optimisation [12]. In fact, even in sparse feature-based SLAM there have been relatively few attempts to deal with motion which is both extended and extremely loopy, such as Strasdat *et al.*'s work on double window optimisation [20].

With a dense vision frontend, the number of points matched and measured at each sensor frame is much higher than in feature-based systems (typically hundreds of thousands). This makes joint filtering or bundle adjustment local optimisation computationally infeasible. Instead, dense frontends have relied on alternation and effectively per-surface-element-independent filtering [15, 9]. However, it has been observed in the field of dense visual SLAM that the enormous weight of data serves to overpower the approximations to joint filtering which this assumes. This also raises the question as to whether it is optimal to attach a dense frontend to a sparse pose graph structure like its feature-based visual SLAM counterpart. Pose graph SLAM systems primarily focus on optimising the camera trajectory, whereas our approach (utilising a deformation graph) instead focuses on optimising the map.

Fig. 1: Comprehensive scan of an office containing over 4.5 million surfels captured in real-time.

Some examples of recent real-time dense visual SLAM systems that utilise pose graphs include that of Whelan *et al.* which parameterises a non-rigid surface deformation with an optimised pose graph to perform occasional loop closures in corridor-like trajectories [25]. This approach is known to scale well but perform poorly given locally loopy trajectories while being unable to re-use revisited areas of the map. The DVO SLAM system of Kerl *et al.* applies keyframe-based pose graph optimisation principles to a dense tracking frontend but performs no explicit map reconstruction and functions off of raw keyframes alone [10]. Meilland and Comport's work on unified keyframes utilises fused predicted 2.5D keyframes of mapped environments while employing pose graph optimisation to close large loops and align keyframes, although not creating an explicit continuous 3D surface [14]. MRSMap by Stückler and Behnke registers octree encoded surfel maps together for pose estimation. After pose graph optimisation the final map is created by merging key surfel views [21].

In our system we wish to move away from the focus on

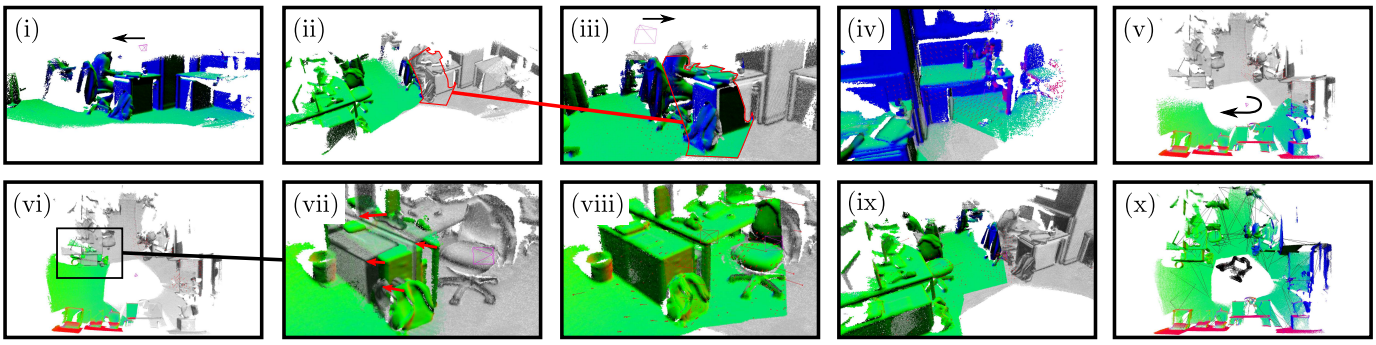


Fig. 2: Example SLAM sequence with active model coloured by surface normal overlaid on the inactive model in greyscale; (i) Initially all data is in the active model as the camera moves left; (ii) As time goes on, the area of map not seen recently is set to inactive. Note the highlighted area; (iii) The camera revisits the inactive area of the map, closing a local loop and registering the surface together. The previously highlighted inactive region then becomes active; (iv) Camera exploration continues to the right and more loops are closed; (v) Continued exploration to new areas; (vi) The camera revisits an inactive area but has drifted too far for a local loop closure; (vii) Here the misalignment is apparent, with red arrows visualising equivalent points from active to inactive; (viii) A global loop closure is triggered which aligns the active and inactive model; (ix) Exploration to the right continues as more local loop closures are made and inactive areas reactivated; (x) Final full map coloured with surface normals showing underlying deformation graph and sampled camera poses in global loop closure database.

pose graphs originally grounded in sparse methods and move towards a more map-centric approach that more elegantly fits the model-predictive characteristics of a typical dense frontend. For this reason we also put a strong emphasis on hard real-time operation in order to always be able to use surface prediction every frame for true incremental simultaneous localisation and dense mapping. This is in contrast to other dense reconstruction systems which don't strictly perform *both* tracking and mapping in real-time [18, 19]. The approach we have developed in this paper is closer to the offline dense scene reconstruction system of Zhou *et al.* than a traditional SLAM system in how it places much more emphasis on the accuracy of the reconstructed map over the estimated trajectory [27].

In our map-centric approach to dense SLAM we attempt to apply surface loop closure optimisations early and often, and therefore always stay near to the mode of the map distribution. This allows us to employ a non-rigid space deformation of the map using a sparse deformation graph embedded in the surface itself rather than a probabilistic pose graph which is rigidly transforming independent keyframes. As we show in our evaluation of the system in Section VII, this approach to dense SLAM achieves state-of-the-art performance with trajectory estimation results on par with or better than existing dense SLAM systems that utilise pose graph optimisation. We also demonstrate the capability to capture comprehensive dense scans of room scale environments involving complex loopy camera trajectories as well as more traditional "corridor-like" forward facing trajectories. At the time of writing we believe our real-time approach to be the first of its kind to; (i) use photometric and geometric frame-to-model predictive tracking in a fused surfel-based dense map; (ii) perform dense model-to-model local surface loop closures with a non-rigid space deformation and (iii) utilise a predicted surface appearance-based place recognition method to resolve global surface loop closures and hence capture globally consistent dense surfel-based maps without a pose graph.

II. Approach Overview

We adopt an architecture which is typically found in real-time dense visual SLAM systems that alternates between tracking and mapping [15, 25, 9, 8, 2, 16]. Like many dense SLAM systems ours makes significant use of GPU programming. We mainly use CUDA to implement our tracking reduction process and the OpenGL Shading Language for view prediction and map management. Our approach is grounded in estimating a dense 3D map of an environment explored with a standard RGB-D camera[1] in real-time. In the following, we summarise the key elements of our method.

1) Estimate a fused surfel-based model of the environment. This component of our method is inspired by the surfel-based fusion system of Keller *et al.* [9], with some notable differences outlined in Section III.
2) While tracking and fusing data in the area of the model most recently observed (*active* area of the model), segment older parts of the map which have not been observed in a period of time δ_t into the *inactive* area of the model (not used for tracking or data fusion).
3) Every frame, attempt to register the portion of the active model within the current estimated camera frame with the portion of the inactive model underlaid within the same frame. If registration is successful, a loop has been closed to the older inactive model and the entire model is non-rigidly deformed into place to reflect this registration. The inactive portion of the map which caused this loop closure is then reactivated to allow

[1] Such as the Microsoft Kinect or ASUS Xtion Pro Live.

tracking and surface fusion (including surfel culling) to take place between the registered areas of the map.

4) For global loop closure, add predicted views of the scene to a randomised fern encoding database [6]. Each frame, attempt to find a matching predicted view via this database. If a match is detected, register the views together and check if the registration is globally consistent with the model's geometry. If so, reflect this registration in the map with a non-rigid deformation, bringing the surface into global alignment.

Figure 2 provides a visualisation of the outlined main steps of our approach. In the following section we describe our fused map representation and method for predictive tracking.

III. Fused Predicted Tracking

Our scene representation is an unordered list of surfels $\mathcal{M}$ (similar to the representation used by Keller *et al.* [9]), where each surfel $\mathcal{M}^s$ has the following attributes; a position $\mathbf{p} \in \mathbb{R}^3$, normal $\mathbf{n} \in \mathbb{R}^3$, colour $\mathbf{c} \in \mathbb{N}^3$, weight $w \in \mathbb{R}$, radius $r \in \mathbb{R}$, initialisation timestamp t_0 and last updated timestamp t. The radius of each surfel is intended to represent the local surface area around a given point while minimising visible holes, computed as done by Salas-Moreno *et al.* [17]. Our system follows the same rules as described by Keller *et al.* for performing surfel initialisation and depth map fusion (where surfel colours follow the same moving average scheme), however when using the map for pose estimation our approach differs in two ways; (i) instead of only predicting a depth map via splatted rendering for geometric frame-to-model tracking, we additionally predict a full colour splatted rendering of the model surfels to perform photometric frame-to-model tracking; (ii) we define a time window threshold δ_t which divides $\mathcal{M}$ into surfels which are *active* and *inactive*. Only surfels which are marked as active model surfels are used for camera pose estimation and depth map fusion. A surfel in $\mathcal{M}$ is declared as inactive when the time since that surfel was last updated (*i.e.* had a raw depth measurement associated with it for fusion) is greater than δ_t. In the following, we describe our method for joint photometric and geometric pose estimation from a splatted surfel prediction.

We define the image space domain as $\Omega \subset \mathbb{N}^2$, where an RGB-D frame is composed of a depth map $\mathcal{D}$ of depth pixels $d : \Omega \to \mathbb{R}$ and a colour image $\mathcal{C}$ of colour pixels $\mathbf{c} : \Omega \to \mathbb{N}^3$. We also compute a normal map for every depth map as necessary using central difference. We define the 3D back-projection of a point $\mathbf{u} \in \Omega$ given a depth map $\mathcal{D}$ as $\mathbf{p}(\mathbf{u}, \mathcal{D}) = \mathbf{K}^{-1}\boldsymbol{u}d(\mathbf{u})$, where $\mathbf{K}$ is the camera intrinsics matrix and $\boldsymbol{u}$ the homogeneous form of $\mathbf{u}$. We also specify the perspective projection of a 3D point $\mathbf{p} = [x, y, z]^\top$ (represented in camera frame $\underrightarrow{\mathcal{F}}_C$) as $\mathbf{u} = \boldsymbol{\pi}(\mathbf{K}\mathbf{p})$, where $\boldsymbol{\pi}(\mathbf{p}) = (x/z, y/z)^\top$ denotes the dehomogenisation operation. The intensity value of a pixel $\mathbf{u} \in \Omega$ given a colour image $\mathcal{C}$ with colour $\mathbf{c}(\mathbf{u}) = [c_1, c_2, c_3]^\top$ is defined as $I(\mathbf{u}, \mathcal{C}) = (c_1 + c_2 + c_3)/3$. For each input frame at time t we estimate the global pose of the camera $\mathbf{P}_t$ (w.r.t. a global frame $\underrightarrow{\mathcal{F}}_G$) by registering the current live depth map and colour image captured by the camera with the surfel-splatted predicted depth map and colour image of the active model from the previous pose estimate. All camera poses are represented with a transformation matrix where:

$$\mathbf{P}_t = \begin{bmatrix} & \mathbf{R}_t & & \mathbf{t}_t \\ 0 & 0 & 0 & 1 \end{bmatrix} \in \mathbb{SE}_3, \tag{1}$$

with rotation $\mathbf{R}_t \in \mathbb{SO}_3$ and translation $\mathbf{t}_t \in \mathbb{R}^3$.

A. Geometric Pose Estimation

Between the current live depth map $\mathcal{D}_t^l$ and the predicted active model depth map from the last frame $\hat{\mathcal{D}}_{t-1}^a$ we aim to find the motion parameters $\boldsymbol{\xi}$ that minimise the cost over the point-to-plane error between 3D back-projected vertices:

$$E_{icp} = \sum_k \left(\left(\mathbf{v}^k - \exp(\hat{\boldsymbol{\xi}})\mathbf{T}\mathbf{v}_t^k \right) \cdot \mathbf{n}^k \right)^2, \tag{2}$$

where $\mathbf{v}_t^k$ is the back-projection of the k-th vertex in $\mathcal{D}_t^l$, $\mathbf{v}^k$ and $\mathbf{n}^k$ are the corresponding vertex and normal represented in the previous camera coordinate frame (at time step $t-1$). $\mathbf{T}$ is the current estimate of the transformation from the previous camera pose to the current one and $\exp(\hat{\boldsymbol{\xi}})$ is the matrix exponential that maps a member of the Lie algebra $\mathfrak{se}_3$ to a member of the corresponding Lie group $\mathbb{SE}_3$. Vertices are associated using projective data association [15].

B. Photometric Pose Estimation

Between the current live colour image $\mathcal{C}_t^l$ and the predicted active model colour from the last frame $\hat{\mathcal{C}}_{t-1}^a$ we aim to find the motion parameters $\boldsymbol{\xi}$ that minimise the cost over the photometric error (intensity difference) between pixels:

$$E_{rgb} = \sum_{\mathbf{u} \in \Omega} \left(I(\mathbf{u}, \mathcal{C}_t^l) - I\left(\boldsymbol{\pi}(\mathbf{K}\exp(\hat{\boldsymbol{\xi}})\mathbf{T}\mathbf{p}(\mathbf{u}, \mathcal{D}_t^l)), \hat{\mathcal{C}}_{t-1}^a \right) \right)^2, \tag{3}$$

where as above $\mathbf{T}$ is the current estimate of the transformation from the previous camera pose to the current one. Note that Equations 2 and 3 omit conversion between 3-vectors and their corresponding homogeneous 4-vectors (as needed for multiplications with $\mathbf{T}$) for simplicity of notation.

C. Joint Optimisation

At this point we wish to minimise the joint cost function:

$$E_{track} = E_{icp} + w_{rgb}E_{rgb}, \tag{4}$$

with $w_{rgb} = 0.1$ in line with related work [8, 25]. For this we use the Gauss-Newton non-linear least-squares method with a three level coarse-to-fine pyramid scheme. To solve each iteration we calculate the least-squares solution:

$$\arg\min_{\boldsymbol{\xi}} \ \|\mathbf{J}\boldsymbol{\xi} + \mathbf{r}\|_2^2, \tag{5}$$

to yield an improved camera transformation estimate:

$$\mathbf{T}' = \exp(\hat{\boldsymbol{\xi}})\mathbf{T} \tag{6}$$

$$\hat{\boldsymbol{\xi}} = \begin{bmatrix} & [\boldsymbol{\omega}]_\times & & \mathbf{x} \\ 0 & 0 & 0 & 0 \end{bmatrix}, \tag{7}$$

with $\boldsymbol{\xi} = [\boldsymbol{\omega}^\top \mathbf{x}^\top]^\top$, $\boldsymbol{\omega}, \in \mathbb{R}^3$ and $\mathbf{x} \in \mathbb{R}^3$.

Blocks of the combined measurement Jacobian $\mathbf{J}$ and residual $\mathbf{r}$ can be populated (while being weighted according to w_{rgb}) and solved with a highly parallel tree reduction in CUDA to produce a 6×6 system of normal equations which is then solved on the CPU by Cholesky decomposition to yield $\boldsymbol{\xi}$. The outcome of this process is an up to date camera pose estimate $\mathbf{P}_t = \mathbf{T}\mathbf{P}_{t-1}$ which brings the live camera data $\mathcal{D}_t^l$ and $\mathcal{C}_t^l$ into strong alignment with the current active model (and hence ready for fusion with the active surfels in $\mathcal{M}$).

IV. Deformation Graph

In order to ensure local and global surface consistency in the map we reflect successful surface loop closures in the set of surfels $\mathcal{M}$. This is carried out by non-rigidly deforming all surfels (both active and inactive) according to surface constraints provided by either of the loop closure methods later described in Sections V and VI. We adopt a space deformation approach based on the embedded deformation technique of Sumner *et al.* [23].

A deformation graph is composed of a set of nodes and edges distributed throughout the model to be deformed. Each node $\mathcal{G}^n$ has a timestamp $\mathcal{G}_{t_0}^n$, a position $\mathcal{G}_{\mathbf{g}}^n \in \mathbb{R}^3$ and set of neighbouring nodes $\mathcal{N}(\mathcal{G}^n)$. The neighbours of each node make up the (directed) edges of the graph. A graph is connected up to a neighbour count k such that $\forall n, |\mathcal{N}(\mathcal{G}^n)| = k$. We use $k = 4$ in all of our experiments. Each node also stores an affine transformation in the form of a 3×3 matrix $\mathcal{G}_{\mathbf{R}}^n$ and a 3×1 vector $\mathcal{G}_{\mathbf{t}}^n$, initialised by default to the identity and $(0,0,0)^\top$ respectively. When deforming a surface, the $\mathcal{G}_{\mathbf{R}}^n$ and $\mathcal{G}_{\mathbf{t}}^n$ parameters of each node are optimised according to surface constraints, which we later describe in Section IV-C.

In order to apply a deformation graph to the surface, each surfel $\mathcal{M}^s$ identifies a set of influencing nodes in the graph $\mathcal{I}(\mathcal{M}^s, \mathcal{G})$. The deformed position of a surfel is given by:

$$\hat{\mathcal{M}}_{\mathbf{p}}^s = \phi(\mathcal{M}^s) = \sum_{n \in \mathcal{I}(\mathcal{M}^s, \mathcal{G})} w^n(\mathcal{M}^s) \left[\mathcal{G}_{\mathbf{R}}^n(\mathcal{M}_{\mathbf{p}}^s - \mathcal{G}_{\mathbf{g}}^n) + \mathcal{G}_{\mathbf{g}}^n + \mathcal{G}_{\mathbf{t}}^n\right], \tag{8}$$

while the deformed normal of a surfel is given by:

$$\hat{\mathcal{M}}_{\mathbf{n}}^s = \sum_{n \in \mathcal{I}(\mathcal{M}^s, \mathcal{G})} w^n(\mathcal{M}^s) {\mathcal{G}_{\mathbf{R}}^{n}}^{-1^\top} \mathcal{M}_{\mathbf{n}}^s, \tag{9}$$

where $w^n(\mathcal{M}^s)$ is a scalar representing the influence node $\mathcal{G}^n$ has on surfel $\mathcal{M}^s$, summing to a total of 1 when $n = k$:

$$w^n(\mathcal{M}^s) = (1 - \left\|\mathcal{M}_{\mathbf{p}}^s - \mathcal{G}_{\mathbf{g}}^n\right\|_2 / d_{max})^2. \tag{10}$$

Here d_{max} is the Euclidean distance to the $k+1$-nearest node of $\mathcal{M}^s$. In the following we describe our method for sampling the deformation graph $\mathcal{G}$ from the set of surfels $\mathcal{M}$ along with our method for determining graph connectivity.

A. Construction

Each frame a new deformation graph for the set of surfels $\mathcal{M}$ is constructed, since it is computationally cheap and simpler than incrementally modifying an existing one. We initialise a new deformation graph $\mathcal{G}$ each frame with node positions set to surfel positions ($\mathcal{G}_{\mathbf{g}}^n = \mathcal{M}_{\mathbf{p}}^s$) and node timestamps set to surfel initialisation timestamps ($\mathcal{G}_{t_0}^n = \mathcal{M}_{t_0}^s$) sampled from $\mathcal{M}$ using systematic sampling such that $|\mathcal{G}| \ll |\mathcal{M}|$. Note that this sampling is uniformly distributed over the population, causing the spatial density of $\mathcal{G}$ to mirror that of $\mathcal{M}$. The set $\mathcal{G}$ is also ordered over n on $\mathcal{G}_{t_0}^n$ such that $\forall n, \mathcal{G}_{t_0}^n \geq \mathcal{G}_{t_0}^{n-1}, \mathcal{G}_{t_0}^{n-2}, \ldots, \mathcal{G}_{t_0}^0$. To compute the connectivity of the graph we use this initialisation time ordering of $\mathcal{G}$ to connect nodes sequentially up to the neighbour count k, defining $\mathcal{N}(\mathcal{G}^n) = \{\mathcal{G}^{n\pm1}, \mathcal{G}^{n\pm2}, \ldots, \mathcal{G}^{n\pm\frac{k}{2}}\}$. This method is computationally efficient (compared to spatial approaches [23, 1]) but more importantly prevents temporally uncorrelated areas of the surface from influencing each other (*i.e.* active and inactive areas), as shown in Figure 3. Note that in the case where $n \pm \frac{k}{2}$ is less than zero or greater than $|\mathcal{G}|$ we connect the graph either forwards or backwards from the bound. For example, $\mathcal{N}(\mathcal{G}^0) = \{\mathcal{G}^1, \mathcal{G}^2, \ldots, \mathcal{G}^k\}$ and $\mathcal{N}(\mathcal{G}^{|\mathcal{G}|}) = \{\mathcal{G}^{|\mathcal{G}|-1}, \mathcal{G}^{|\mathcal{G}|-2}, \ldots, \mathcal{G}^{|\mathcal{G}|-k}\}$. Next we describe how to apply the deformation graph to the map of surfels.

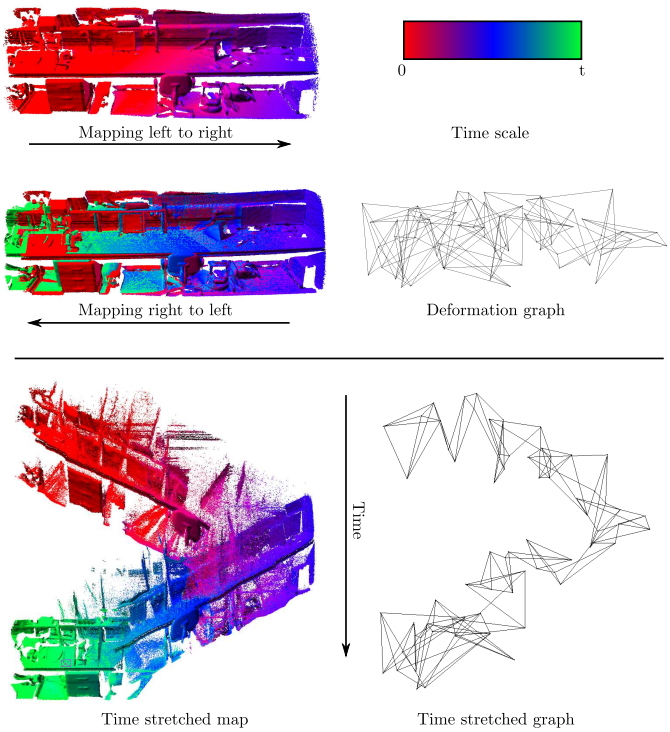


Fig. 3: Temporal deformation graph connectivity before loop closure. The top half shows a mapping sequence where the camera first maps left to right over a desk area and then back across the same area. Given the windowed fusion process it appears that the map and hence deformation graph is tangled up in itself between passes. However, observing the bottom half of the figure where the vertical dimension has been artificially stretched by the initialisation times $\mathcal{M}_{t_0}$ and $\mathcal{G}_{t_0}$ of each surfel and graph node respectively, it is clear that multiple passes of the map are disjoint and free to be aligned.

B. Application

In order to apply the deformation graph after optimisation (detailed in the next section) to update the map, the set of nodes which influence each surfel $\mathcal{M}^s$ must be determined. In tune with the method in the previous section a temporal association is chosen, similar to the approach taken by Whelan *et al.* [25]. The algorithm which implements $\mathcal{I}(\mathcal{M}^s, \mathcal{G})$ and applies the deformation graph $\mathcal{G}$ to a given surfel is listed in Algorithm 1. When each surfel is deformed, the full set of deformation nodes is searched for the node which is closest in time. The solution to this L_1-norm minimisation is actually a binary search over the set $\mathcal{G}$ as it is already ordered. From here, other nodes nearby in time are collected and the k-nearest nodes (in the Euclidean distance sense) are selected as $\mathcal{I}(\mathcal{M}^s, \mathcal{G})$. Finally the weights for each node are computed as in Equation 10 and the transformations from Equations 8 and 9 are applied. All other attributes of the updated surfel $\hat{\mathcal{M}}^s$ are copied from $\mathcal{M}^s$.

Algorithm 1: Deformation Graph Application

Input: $\mathcal{M}^s$ surfel to be deformed
$\mathcal{G}$ set of deformation nodes
α number of nodes to explore
Output: $\hat{\mathcal{M}}^s$ deformed surfel
do

```
// Find closest node in time
```
$c \leftarrow \arg\min_i \left\| \mathcal{M}^s_{t_0} - \mathcal{G}^i_{t_0} \right\|_1$
```
// Gather set of temporally nearby nodes
```
$\mathcal{I} \leftarrow \emptyset$
for $i \leftarrow -\alpha/2$ **to** $\alpha/2$ **do**
  $\mathcal{I}^{i+\alpha/2} \leftarrow c + i$
$\text{sort\_by\_euclidean\_distance}(\mathcal{I}, \mathcal{G}, \mathcal{M}^s_{\mathbf{p}})$
```
// Take closest k as influencing nodes
```
$\mathcal{I}(\mathcal{M}^s, \mathcal{G}) \leftarrow \mathcal{I}^{0 \rightarrow k-1}$
```
// Compute weights
```
$h \leftarrow 0$
$d_{max} \leftarrow \left\| \mathcal{M}^s_{\mathbf{p}} - \mathcal{G}^{\mathcal{I}^k}_{\mathbf{g}} \right\|_2$
for $n \in \mathcal{I}(\mathcal{M}^s, \mathcal{G})$ **do**
  $w^n(\mathcal{M}^s) \leftarrow (1 - \left\| \mathcal{M}^s_{\mathbf{p}} - \mathcal{G}^n_{\mathbf{g}} \right\|_2 / d_{max})^2$
  $h \leftarrow h + w^n(\mathcal{M}^s)$
```
// Apply transformations
```
$\hat{\mathcal{M}}^s_{\mathbf{p}} = \sum_{n \in \mathcal{I}(\mathcal{M}^s, \mathcal{G})} \frac{w^n(\mathcal{M}^s)}{h} \left[\mathcal{G}^n_{\mathbf{R}} (\mathcal{M}^s_{\mathbf{p}} - \mathcal{G}^n_{\mathbf{g}}) + \mathcal{G}^n_{\mathbf{g}} + \mathcal{G}^n_{\mathbf{t}} \right]$
$\hat{\mathcal{M}}^s_{\mathbf{n}} = \sum_{n \in \mathcal{I}(\mathcal{M}^s, \mathcal{G})} \frac{w^n(\mathcal{M}^s)}{h} {\mathcal{G}^n_{\mathbf{R}}}^{-1^\top} \mathcal{M}^s_{\mathbf{n}}$

C. Optimisation

Given a set of surface correspondences $\mathcal{Q}$ (later expanded upon in Sections V and VI) the parameters of the deformation graph can be optimised to reflect a surface registration in the surfel model $\mathcal{M}$. An element $\mathcal{Q}^p \in \mathcal{Q}$ is a tuple $\mathcal{Q}^p = (\mathcal{Q}^p_{\mathbf{d}}; \mathcal{Q}^p_{\mathbf{s}}; \mathcal{Q}^p_{\mathbf{d}_t}; \mathcal{Q}^p_{\mathbf{s}_t})$ which contains a pair of points representing a destination position $\mathcal{Q}^p_{\mathbf{d}} \in \mathbb{R}^3$ and a source position $\mathcal{Q}^p_{\mathbf{s}} \in \mathbb{R}^3$ which should reach the destination upon deformation. The timestamps of each point are also stored in $\mathcal{Q}^p$ as $\mathcal{Q}^p_{\mathbf{d}_t}$ and $\mathcal{Q}^p_{\mathbf{s}_t}$ respectively. We use four cost functions over the deformation graph akin to those defined by Sumner *et al.* [23]. The first maximises rigidity in the deformation:

$$E_{rot} = \sum_l \left\| {\mathcal{G}^l_{\mathbf{R}}}^\top \mathcal{G}^l_{\mathbf{R}} - \mathbf{I} \right\|^2_F, \tag{11}$$

using the Frobenius-norm. The second is a regularisation term that ensures a smooth deformation across the graph:

$$E_{reg} = \sum_l \sum_{n \in \mathcal{N}(\mathcal{G}^l)} \left\| \mathcal{G}^l_{\mathbf{R}} (\mathcal{G}^n_{\mathbf{g}} - \mathcal{G}^l_{\mathbf{g}}) + \mathcal{G}^l_{\mathbf{g}} + \mathcal{G}^l_{\mathbf{t}} - (\mathcal{G}^n_{\mathbf{g}} + \mathcal{G}^n_{\mathbf{t}}) \right\|^2_2 \tag{12}$$

The third is a constraint term that minimises the error on the set of position constraints $\mathcal{Q}$, where $\phi(\mathcal{Q}^p_{\mathbf{s}})$ is the result of applying Equation 8 to $\mathcal{Q}^p_{\mathbf{s}}$:

$$E_{con} = \sum_p \| \phi(\mathcal{Q}^p_{\mathbf{s}}) - \mathcal{Q}^p_{\mathbf{d}} \|^2_2 \tag{13}$$

Note that in order to apply Equation 8 to $\mathcal{Q}^p_{\mathbf{s}}$ we must compute $\mathcal{I}(\mathcal{Q}^p_{\mathbf{s}}, \mathcal{G})$ and subsequently $w^n(\mathcal{Q}^p_{\mathbf{s}})$. For this we use the same algorithm as described in Algorithm 1 to deform the position only, using $\mathcal{Q}^p_{\mathbf{s}}$ (inclusive of timestamp $\mathcal{Q}^p_{\mathbf{s}_t}$) in place of $\mathcal{M}^s$. In practice $\mathcal{Q}^p_{\mathbf{s}_t}$ will always be the timestamp of a surfel within the active model while $\mathcal{Q}^p_{\mathbf{d}_t}$ will be the timestamp of a surfel within the inactive model. The temporal parameterisation of the surface we are using allows multiple passes of the same surface to be non-rigidly deformed into alignment allowing mapping to continue and new data fusion into revisited areas of the map. Given this, the final cost function "pins" the inactive area of the model in place ensuring that we are always deforming the active area of the model into the inactive coordinate system:

$$E_{pin} = \sum_p \| \phi(\mathcal{Q}^p_{\mathbf{d}}) - \mathcal{Q}^p_{\mathbf{d}} \|^2_2 \tag{14}$$

As above we use Algorithm 1 to compute $\phi(\mathcal{Q}^p_{\mathbf{d}})$, using $\mathcal{Q}^p_{\mathbf{d}}$ in place of $\mathcal{M}^s$. The final total cost function is defined as:

$$E_{def} = w_{rot} E_{rot} + w_{reg} E_{reg} + w_{con} E_{con} + w_{con} E_{pin} \tag{15}$$

With $w_{rot} = 1$, $w_{reg} = 10$ and $w_{con} = 100$ (in line with related work [23, 1, 25]) we minimise this total cost with respect to $\mathcal{G}^n_{\mathbf{R}}$ and $\mathcal{G}^n_{\mathbf{t}}$ over all n using the iterative Gauss-Newton algorithm. The Jacobian matrix in this problem is sparse and as a result we use sparse Cholesky factorisation to efficiently solve the system on the CPU. From here the deformation graph $\mathcal{G}$ is uploaded to the GPU for application to the entire surfel map as described in Section IV-B.

V. Local Loop Closure

To ensure local surface consistency throughout the map our system closes many small loops with the existing map as those areas are revisited. As shown in Figure 2, we fuse into the active area of the model while gradually labeling surfels that have not been seen in a period of time δ_t as inactive. The

inactive area of the map is not used for live frame tracking and fusion until a loop is closed between the active model and inactive model, at which point the matched inactive area becomes active again. This has the advantage of continuous frame-to-model tracking and also model-to-model tracking which provides viewpoint-invariant local loop closures.

We divide the set of surfels in our map $\mathcal{M}$ into two disjoint sets Θ and Ψ, such that given the current frame timestamp t for each surfel in the map $\mathcal{M}^s \in \Theta$ if $t - \mathcal{M}^s_t < \delta_t$ and $\mathcal{M}^s \in \Psi$ if $t - \mathcal{M}^s_t \geq \delta_t$, making Θ the active set and Ψ the inactive set. In each frame if a global loop closure has not been detected (described in the following section), we attempt to compute a match between Θ and Ψ. This is done by registering the predicted surface renderings of Θ and Ψ from the latest pose estimate $\mathbf{P}_t$, denoted $\mathcal{D}^a_t, \mathcal{C}^a_t$ and $\mathcal{D}^i_t, \mathcal{C}^i_t$ respectively. This pair of model views is registered together using the same method as described in Section III. The output of this process will be a relative transformation matrix $\mathbf{H} \in \mathbb{SE}_3$ from Θ to Ψ which brings the two predicted surface renderings into alignment.

In order to check the quality of this registration and decide whether or not to carry out a deformation, we inspect the final condition of the Gauss-Newton optimisation used to align the two views. The residual cost E_{track} from Equation 4 must be sufficiently small, while the number of inlier measurements used must be above a minimum threshold. We also inspect the eigenvalues of the covariance of the system (approximated by the Hessian as $\mathbf{\Sigma} = (\mathbf{J}^\top \mathbf{J})^{-1}$) by; $\sigma_i(\mathbf{\Sigma}) < \mu$ for $i = \{1, \ldots, 6\}$, where $\sigma_i(\mathbf{\Sigma})$ is the i-th eigenvalue of $\mathbf{\Sigma}$ and μ a sufficiently conservative threshold.

If a high quality alignment has been achieved, we produce a set of surface constraints $\mathcal{Q}$ which are fed into the deformation graph optimisation described in Section IV to align the surfels in Θ with those in Ψ. To do this we also require the initialisation timestamps Ψ_{t_0} of each surfel splat used to render $\mathcal{D}^i_t$. These are rendered as $\mathcal{T}_t{}^i$ and are necessary to correctly constrain the deformation between the active model and inactive model. We uniformly sample a set of pixel coordinates $\mathcal{U} \subset \Omega$ to compute the set $\mathcal{Q}$. For each pixel $\mathbf{u} \in \mathcal{U}$ we populate a constraint:

$$\mathcal{Q}^p = ((\mathbf{H}\mathbf{P}_t)\mathbf{p}(\mathbf{u}, \mathcal{D}^a_t); \mathbf{P}_t\mathbf{p}(\mathbf{u}, \mathcal{D}^a_t); \mathcal{T}_t{}^i(\mathbf{u}); t). \tag{16}$$

After the deformation has occurred a new up to date camera pose is resolved as $\hat{\mathbf{P}}_t = \mathbf{H}\mathbf{P}_t$. At this point the set of surfels which were part of the alignment are reactivated to allow live camera tracking and fusion with the existing active surfels. An up to date prediction of the active model depth must be rendered to reflect the deformation for the depth test for inactive surfels, computed as $\tilde{\mathcal{D}}^a_t$. For each surfel $\mathcal{M}^s$:

$$\mathcal{M}^s_t = \begin{cases} t & \text{if } \boldsymbol{\pi}(\mathbf{K}\hat{\mathbf{P}}_t^{-1}\mathcal{M}^s_{\mathbf{p}}) \in \Omega \\ & \text{and } (\mathbf{K}\hat{\mathbf{P}}_t^{-1}\mathcal{M}^s_{\mathbf{p}})_z \lesssim \tilde{\mathcal{D}}^a_t(\boldsymbol{\pi}(\mathbf{K}\hat{\mathbf{P}}_t^{-1}\mathcal{M}^s_{\mathbf{p}})), \\ \mathcal{M}^s_t & \text{else.} \end{cases} \tag{17}$$

The process described in this section brings active areas of the model into strong alignment with inactive areas of the model to achieve tight local surface loop closures. In the event of the active model drifting too far from the inactive model for local alignment to converge, we resort to an appearance-based global loop closure method to bootstrap a surface deformation which realigns the active model with the underlying inactive model for tight global loop closure and surface global consistency. This is described in the following section.

VI. Global Loop Closure

We utilise the randomised fern encoding approach for appearance-based place recognition [6]. Ferns encode an RGB-D image as a string of codes made up of the values of binary tests on each of the RGB-D channels in a set of fixed pixel locations. The approach presented by Glocker *et al.* includes an automatic method for fern database management that avoids adding redundant views and non-discriminative frames. This technique has been demonstrated to perform very reliably in terms of computational performance and viewpoint recognition. Our implementation of randomised fern encoding is identical to that of Glocker *et al.* with the difference that instead of encoding and matching against raw RGB-D frames, we use predicted views of the surface map once they are aligned and fused with the live camera view. Parts of the predicted views which are devoid of any mapped surface are filled in using the live depth and colour information from the current frame.

Each frame we maintain a fern encoded frame database $\mathcal{E}$, using the same process as originally specified by Glocker *et al.* for fern encoding, frame harvesting and identification of matching fern encodings [6]. As they suggest, we use a downsampled frame size of 80×60. Each element $\mathcal{E}^i \in \mathcal{E}$ contains a number of attributes; a fern encoding string $\mathcal{E}^i_{\mathbf{f}}$, a depth map $\mathcal{E}^i_{\mathcal{D}}$, a colour image $\mathcal{E}^i_{\mathcal{C}}$, a source camera pose $\mathcal{E}^i_{\mathbf{P}}$ and an initialisation time $\mathcal{E}^i_t$. At the end of each frame we add $\hat{\mathcal{D}}^a_t$ and $\hat{\mathcal{C}}^a_t$ (predicted active model depth and colour after fusion filled in with $\mathcal{D}^l_t$ and $\mathcal{C}^l_t$) to $\mathcal{E}$ if necessary. We also query this database immediate after the initial frame-to-model tracking step to determine if there is a global loop closure required. If a matching frame $\mathcal{E}^i$ is found we perform a number of steps to potentially globally align the surfel map.

Firstly, we attempt to align the matched frame with the current model prediction. Similar to the previous section, this involves utilising the registration process outlined in Section III to bring $\mathcal{D}^a_t$ and $\mathcal{C}^a_t$ into alignment with $\mathcal{E}^i_{\mathcal{D}}$ and $\mathcal{E}^i_{\mathcal{C}}$, including inspection of the final condition of the optimisation. If successful, a relative transformation matrix $\mathbf{H} \in \mathbb{SE}_3$ which brings the current model prediction into alignment with the matching frame is resolved. From here, as in the previous section, we populate a set of surface constraints $\mathcal{Q}$ to provide as input to the deformation, where each $\mathbf{u}$ is a randomly sampled fern pixel location (lifted into full image resolution):

$$\mathcal{Q}^p = ((\mathbf{H}\mathcal{E}^i_{\mathbf{P}})\mathbf{p}(\mathbf{u}, \mathcal{D}^a_t); \mathbf{P}_t\mathbf{p}(\mathbf{u}, \mathcal{D}^a_t); \mathcal{E}^i_t; t). \tag{18}$$

Note $\mathcal{Q}^p_{\mathbf{d}}$ which incorporates the difference in the estimated point position given by the alignment and the known actual global point position given by $\mathcal{E}^i_{\mathbf{P}}$. From here, the deformation

System	fr1/desk	fr2/xyz	fr3/office	fr3/nst
DVO SLAM	0.021m	0.018m	0.035m	0.018m
RGB-D SLAM	0.023m	**0.008m**	0.032m	0.017m
MRSMap	0.043m	0.020m	0.042m	2.018m
Kintinuous	0.037m	0.029m	0.030m	0.031m
Frame-to-model	0.022m	0.014m	0.025m	0.027m
ElasticFusion	**0.020m**	0.011m	**0.017m**	**0.016m**

TABLE I: Comparison of ATE RMSE on the evaluated real world datasets of Sturm *et al.* [22].

cost from Equations 11-15 is computed and evaluated to determine if the proposed deformation is consistent with the map's geometry. We are less likely to accept unreliable fern matching triggered deformations as they operate on a much coarser scale than the local loop closure matches. If E_{con} is too small the deformation is likely not required and the loop closure is rejected (*i.e.* it should be detected and applied as a local loop closure). Otherwise, the deformation graph is optimised and the final state of the Gauss-Newton system is analysed to determine if it should be applied. If after optimisation E_{con} is sufficiently small while over all E_{def} is also small, the loop closure is accepted and the deformation graph $\mathcal{G}$ is applied to the entire set of surfels $\mathcal{M}$. At this point the current pose estimate is also updated to $\hat{\mathbf{P}}_t = \mathbf{H}\mathcal{E}^i_{\mathbf{P}}$. Unlike in the previous section the set of active and inactive surfels is not revised at this point. This is for two main reasons; (i) correct global loop closures bring the active and inactive regions of map into close enough alignment to trigger a local loop closure on the next frame and (ii) this allows the map to recover from potentially incorrect global loop closures. We also have the option of relying on the fern encoding database for global relocalisation if camera tracking ever fails (however this was not encountered in any evaluated datasets).

VII. Evaluation

We evaluate the performance of our system both quantitatively and qualitatively in terms of trajectory estimation, surface reconstruction accuracy and computational performance.

A. Trajectory Estimation

To evaluate the trajectory estimation performance of our approach we test our system on the RGB-D benchmark of Sturm *et al.* [22]. This benchmark provides synchronised ground truth poses for an RGB-D sensor moved through a scene, captured with a highly precise motion capture system. In Table I we compare our system to four other state-of-the-art RGB-D based SLAM systems; DVO SLAM [10], RGB-D SLAM [5], MRSMap [21] and Kintinuous [25]. We also provide benchmark scores for our system if all deformations are disabled and only frame-to-model tracking is used. We use the absolute trajectory (ATE) root-mean-square error metric (RMSE) in our comparison, which measures the root-mean-square of the Euclidean distances between all estimated camera poses and the ground truth poses associated by timestamp [22]. These results show that our trajectory estimation performance is on par with or better than existing state-of-the-art systems that

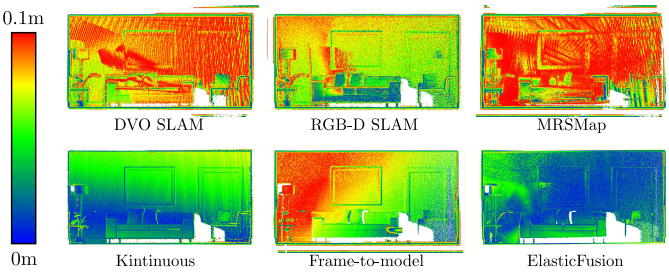


Fig. 4: Orthogonal frontal view heat maps showing reconstruction error on the kt0 dataset. Points more than 0.1m from ground truth have been removed for visualisation purposes.

System	kt0	kt1	kt2	kt3
DVO SLAM	0.104m	0.029m	0.191m	0.152m
RGB-D SLAM	0.026m	0.008m	0.018m	0.433m
MRSMap	0.204m	0.228m	0.189m	1.090m
Kintinuous	0.072m	**0.005m**	**0.010m**	0.355m
Frame-to-model	0.497m	0.009m	0.020m	0.243m
ElasticFusion	**0.009m**	0.009m	0.014m	**0.106m**

TABLE II: Comparison of ATE RMSE on the evaluated synthetic datasets of Handa *et al.* [7].

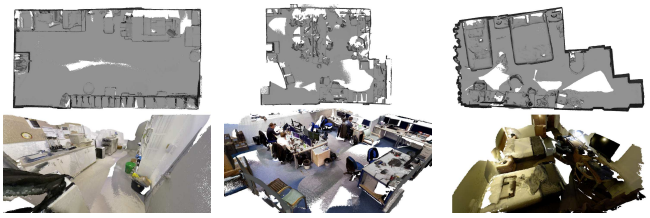
rely on a pose graph optimisation backend. Interestingly our frame-to-model only results are also comparable in performance, whereas a uniform increase in accuracy is achieved when active to inactive model deformations are used, proving their efficacy in trajectory estimation. Only on fr3/nst does a global loop closure occur. Enabling local loops alone on this dataset results in an error of 0.022m, while only enabling global loops results in an error of 0.023m.

B. Surface Estimation

We evaluate the surface reconstruction results of our approach on the ICL-NUIM dataset of Handa *et al.* [7]. This benchmark provides ground truth poses for a camera moved through a synthetic environment as well as a ground truth 3D model which can be used to evaluate surface reconstruction accuracy. We evaluate our approach on all four trajectories in the living room scene (including synthetic noise) providing surface reconstruction accuracy results in comparison to the same SLAM systems listed in Section VII-A. We also include trajectory estimation results for each dataset. Tables II and III

System	kt0	kt1	kt2	kt3
DVO SLAM	0.032m	0.061m	0.119m	0.053m
RGB-D SLAM	0.044m	0.032m	0.031m	0.167m
MRSMap	0.061m	0.140m	0.098m	0.248m
Kintinuous	0.011m	0.008m	0.009m	0.150m
Frame-to-model	0.098m	**0.007m**	0.011m	0.107m
ElasticFusion	**0.007m**	**0.007m**	**0.008m**	**0.028m**

TABLE III: Comparison of surface reconstruction accuracy results on the evaluated synthetic datasets of Handa *et al.* [7]. Quantities shown are the mean distances from each point to the nearest surface in the ground truth 3D model.



(i) (ii) (iii)

Fig. 5: Qualitative datasets; (i) A comprehensive scan of a copy room; (ii) A loopy large scan of a computer lab; (iii) A comprehensive scan of a twin bed hotel room (note that the actual room is not rectilinear). To view small details we recommend either using the digital zoom function in a PDF reader or viewing of our accompanying videos[2].

Name (Fig.)	Copy (5i)	Lab (5ii)	Hotel (5iii)	Office (1)
Frames	5490	6533	7725	5000
Surfels	4.4×10^6	3.5×10^6	4.1×10^6	4.8×10^6
Graph nodes	351	282	328	386
Fern frames	582	651	325	583
Local loops	15	13	11	17
Global loops	1	4	1	0

TABLE IV: Statistics on qualitative datasets.

summarise our trajectory estimation and surface reconstruction results. Note on kt1 the camera never revisits previously mapped portions of the map, making the frame-to-model and ElasticFusion results identical. Additionally, only the kt3 sequence triggers a global loop closure in our approach. This yields a local loop only ATE RMSE result of 0.234m and a global loop only ATE RMSE result of 0.236m. On surface reconstruction, local loops only scores 0.099m and global loops only scores 0.103m. These results show that again our trajectory estimation performance is on par with or better than existing approaches. It is also shown that our surface reconstruction results are superior to all other systems. Figure 4 shows the reconstruction error of all evaluated systems on kt0.

We also present a number of qualitative results on datasets captured in a handheld manner demonstrating system versatility. Statistics for each dataset are listed in Table IV. The Copy dataset contains a comprehensive scan of a photocopying room with many local loop closures and a global loop closure at one point to resolve global consistency. This dataset was made available courtesy of Zhou and Koltun [26]. The Lab dataset contains a very loopy trajectory around a large office environment with many global and local loop closures. The Hotel dataset follows a comprehensive scan of a non-rectilinear hotel room with many local loop closures and a single global loop closure to resolve final model consistency. Finally the Office dataset contains a comprehensive scan of a complete office with many local loop closures avoiding the need for any global loop closures for model consistency. We recommend viewing of our accompanying videos to more clearly visualise and understand the capabilities of our approach[2].

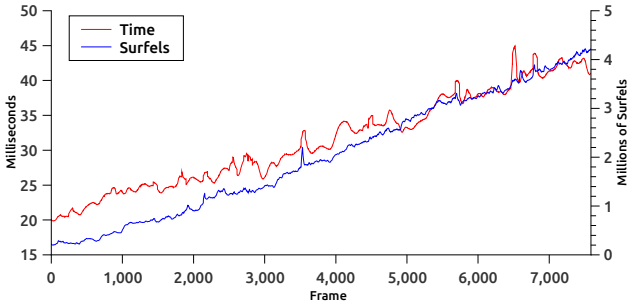


Fig. 6: Frame time vs. number of surfels on the Hotel dataset.

C. Computational Performance

To analyse the computational performance of the system we provide a plot of the average frame processing time across the Hotel sequence. The test platform was a desktop PC with an Intel Core i7-4930K CPU at 3.4GHz, 32GB of RAM and an nVidia GeForce GTX 780 Ti GPU with 3GB of memory. As shown in Figure 6 the execution time of the system increases with the number of surfels in the map, with an overall average of 31ms per frame scaling to a peak average of 45ms implying a worst case processing frequency of 22Hz. This is well within the widely accepted minimum frequencies for fused dense SLAM algorithms [24, 17, 2, 9], and as shown in our qualitative results more than adequate for real-time operation.

VIII. Conclusion

We have presented a novel approach to the problem of dense visual SLAM that performs time windowed surfel-based dense data fusion in combination with frame-to-model tracking and non-rigid deformation. Our main contribution in this paper is to show that by incorporating many small local model-to-model loop closures in conjunction with larger scale global loop closures we are able to stay close to the mode of the distribution of the map and produce globally consistent reconstructions in real-time without the use of pose graph optimisation or post-processing steps. In our evaluation we show that the use of frequent non-rigid map deformations improve both the trajectory estimate of the camera and the surface reconstruction quality. We also demonstrate the effectiveness of our approach in long scale occasionally looping camera motions and more loopy comprehensive room scanning trajectories. In future work we wish to address the problem of map scalability beyond whole rooms and also investigate the problem of dense globally consistent SLAM as $t \rightarrow \infty$.

IX. Acknowledgements

Research presented in this paper has been supported by Dyson Technology Ltd.

[2]https://youtu.be/XySrhZpODYs, https://youtu.be/-dz_VauPjEU

References

[1] J. Chen, S. Izadi, and A. Fitzgibbon. KinÊtre: animating the world with the human body. In *Proceedings of ACM Symposium on User Interface Software and Technolog (UIST)*, 2012.

[2] J. Chen, D. Bautembach, and S. Izadi. Scalable real-time volumetric surface reconstruction. In *Proceedings of SIGGRAPH*, 2013.

[3] A. J. Davison. Real-Time Simultaneous Localisation and Mapping with a Single Camera. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2003.

[4] A. J. Davison, N. D. Molton, I. Reid, and O. Stasse. MonoSLAM: Real-Time Single Camera SLAM. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 29(6):1052–1067, 2007.

[5] F. Endres, J. Hess, N. Engelhard, J. Sturm, D. Cremers, and W. Burgard. An evaluation of the RGB-D SLAM system. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2012.

[6] B. Glocker, J. Shotton, A. Criminisi, and S. Izadi. Real-Time RGB-D Camera Relocalization via Randomized Ferns for Keyframe Encoding. *IEEE Transactions on Visualization and Computer Graphics*, 21(5):571–583, 2015.

[7] A. Handa, T. Whelan, J. B. McDonald, and A. J. Davison. A Benchmark for RGB-D Visual Odometry, 3D Reconstruction and SLAM. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2014. URL http://www.doc.ic.ac.uk/~ahanda/VaFRIC/iclnuim.html.

[8] P. Henry, D. Fox, A. Bhowmik, and R. Mongia. Patch Volumes: Segmentation-based Consistent Mapping with RGB-D Cameras. In *Proc. of Joint 3DIM/3DPVT Conference (3DV)*, 2013.

[9] M. Keller, D. Lefloch, M. Lambers, S. Izadi, T. Weyrich, and A. Kolb. Real-time 3D Reconstruction in Dynamic Scenes using Point-based Fusion. In *Proc. of Joint 3DIM/3DPVT Conference (3DV)*, 2013.

[10] C. Kerl, J. Sturm, and D. Cremers. Dense visual SLAM for RGB-D cameras. In *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*, 2013.

[11] G. Klein and D. W. Murray. Parallel Tracking and Mapping for Small AR Workspaces. In *Proceedings of the International Symposium on Mixed and Augmented Reality (ISMAR)*, 2007.

[12] K. Konolige and M. Agrawal. FrameSLAM: From Bundle Adjustment to Real-Time Visual Mapping. *IEEE Transactions on Robotics (T-RO)*, 24:1066–1077, 2008.

[13] J. B. McDonald, M. Kaess, C. Cadena, J. Neira, and J. J. Leonard. Real-time 6-DOF multi-session visual SLAM over large scale environments. *Robotics and Autonomous Systems*, 61(10):1144–1158, 2013.

[14] M. Meilland and A. I. Comport. On unifying key-frame and voxel-based dense visual SLAM at large scales. In *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*, 2013.

[15] 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 *Proceedings of the International Symposium on Mixed and Augmented Reality (ISMAR)*, 2011.

[16] R. A. Newcombe, S. Lovegrove, and A. J. Davison. DTAM: Dense Tracking and Mapping in Real-Time. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2011.

[17] R. F. Salas-Moreno, B. Glocker, P. H. J. Kelly, and A. J. Davison. Dense Planar SLAM. In *Proceedings of the International Symposium on Mixed and Augmented Reality (ISMAR)*, 2014.

[18] F. Steinbrucker, C. Kerl, J. Sturm, and D. Cremers. Large-scale multi-resolution surface reconstruction from RGB-D sequences. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2013.

[19] F. Steinbrucker, J. Sturm, and D. Cremers. Volumetric 3d mapping in real-time on a CPU. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2014.

[20] H. Strasdat, A. J. Davison, J. M. M. Montiel, and K. Konolige. Double Window Optimisation for Constant Time Visual SLAM. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2011.

[21] J. Stückler and S. Behnke. Multi-resolution surfel maps for efficient dense 3d modeling and tracking. *Journal of Visual Communication and Image Representation*, 25(1): 137–147, 2014.

[22] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers. A benchmark for RGB-D SLAM evaluation. In *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*, 2012.

[23] R. W. Sumner, J. Schmid, and M. Pauly. Embedded deformation for shape manipulation. In *Proceedings of SIGGRAPH*, 2007.

[24] T. Whelan, J. B. McDonald, M. Kaess, M. Fallon, H. Johannsson, and J. J. Leonard. Kintinuous: Spatially Extended KinectFusion. In *Workshop on RGB-D: Advanced Reasoning with Depth Cameras, in conjunction with Robotics: Science and Systems*, 2012.

[25] T. Whelan, M. Kaess, H. Johannsson, M. F. Fallon, J. J. Leonard, and J. B. McDonald. Real-time large scale dense RGB-D SLAM with volumetric fusion. *International Journal of Robotics Research (IJRR)*, 34(4-5):598–626, 2015.

[26] Q. Zhou and V. Koltun. Dense scene reconstruction with points of interest. In *Proceedings of SIGGRAPH*, 2013.

[27] Q. Zhou, S. Miller, and V. Koltun. Elastic Fragments for Dense Scene Reconstruction. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2013.