ElasticFusion: Real-Time Dense SLAM and Light Source Estimation

Thomas Whelan, Renato F. Salas-Moreno, Ben Glocker, Andrew J. Davison and Stefan Leutenegger

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 and beyond 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. In the spirit of improving map quality as well as tracking accuracy and robustness, we furthermore explore a novel approach to real-time discrete light source detection. This technique is capable of detecting numerous light sources in indoor environments in real-time as a user handheld camera explores the scene. Absolutely no prior information about the scene or number of light sources is required. By making a small set of simple assumptions about the appearance properties of the scene our method can incrementally estimate both the quantity and location of multiple light sources in the environment in an online fashion. Our results demonstrate that our technique functions well in many different environments and lighting configurations. We show that this enables (a) more realistic augmented reality (AR) rendering; (b) a richer understanding of the scene beyond pure geometry and; (c) more accurate and robust photometric tracking.

Keywords: surfel fusion, camera pose estimation, dense methods, large scale, real-time, RGB-D, SLAM, GPU, light sources, reflections, specular

1 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 by Davison et al. (2007) or KinectFusion from Newcombe et al. (2011a)) or (ii) "corridor-like" motion on much larger scales but with fewer loop closures (*e.g.* the work of McDonald et al. (2013) or Whelan et al. (2015a)). In sparse feature-based SLAM, it is well understood that loopy local motion can be dealt with either via joint probabilistic filtering (Davison (2003)), or in-the-loop joint optimisation of poses and features (*bundle adjustment*) (Klein and Murray (2007)); 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 (Konolige and Agrawal (2008)). 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 the work on double window optimisation by Strasdat et al. (2011).

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 by holding the camera pose fixed during the mapping step (Newcombe et al. (2011a); Keller et al. (2013)), which has proven to be a highly efficient technique capable of running in real-time even on resource constrained mobile devices Kahler et al. (2015). It has been observed in the aforementioned dense SLAM systems that the enormous weight of the input data serves to overpower the approximations to joint filtering which per-surface-element-independent filtering assumes. This also raises the question as to whether

T. Whelan, S. Leutenegger and A. J. Davison are with the Dyson Robotics Laboratory at Imperial College, Department of Computing, Imperial College London, UK. thomas.j.whelan@mumail.ie, {s.leutenegger,a.davison}@imperial.ac.uk

R. F. Salas-Moreno and B. Glocker are with the Department of Computing, Imperial College London, UK. renato.salasmoreno10@alumni.imperial.ac.uk, b.glocker@imperial.ac.uk

This work was presented in part at Robotics: Science and Systems (RSS), Rome, Italy, July 2015 (Whelan et al. (2015b)).

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

it is optimal to attach a dense frontend to a sparse pose graph structure like its feature-based visual SLAM counterpart. Pose graph optimisation focuses on optimising the camera trajectory, leaving it unclear how to simultaneously modify the map in a consistent manner, *i.e.* actually improving its accuracy at all levels of scale.

Some examples of recent real-time dense visual SLAM systems that utilise pose graphs include that of Whelan et al. (2015a) which parameterises a non-rigid surface deformation with an optimised pose graph to perform occasional loop closures in corridor-like trajectories. 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. (2013) 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. The work from Meilland and Comport (2013) 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. MRSMap by Stückler and Behnke (2014) registers octree encoded surfel maps together for pose estimation. After pose graph optimisation the final map is created by merging key surfel views.

In our system we wish to move away from the focus on 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 (Steinbrücker et al. (2013, 2014)). Typically with volumetric SLAM systems (as opposed to point-based systems) real-time map correction is difficult due to the sheer amount of data that requires updating. Hybrid semi-real-time approaches have been developed which function in a similar vein to the parallel tracking and mapping (PTAM) system of Klein and Murray (2007), where tracking in the frontend is real-time and uninterrupted while a backend optimiser performs map correction. The work of Fioraio et al. (2015) is an example of one such system, where tracking is performed using last-k-frame subvolumes that are gradually globally aligned and fused together in a background optimisation process.

The approach we have developed in this paper is closer to the offline dense scene reconstruction system of Zhou et al. (2013) than a traditional SLAM system in how it places much more emphasis on the accuracy of the reconstructed map over the estimated trajectory. However rather than registering small incremental fragments together in an alternating global optimisation, we embed a complete deformation graph

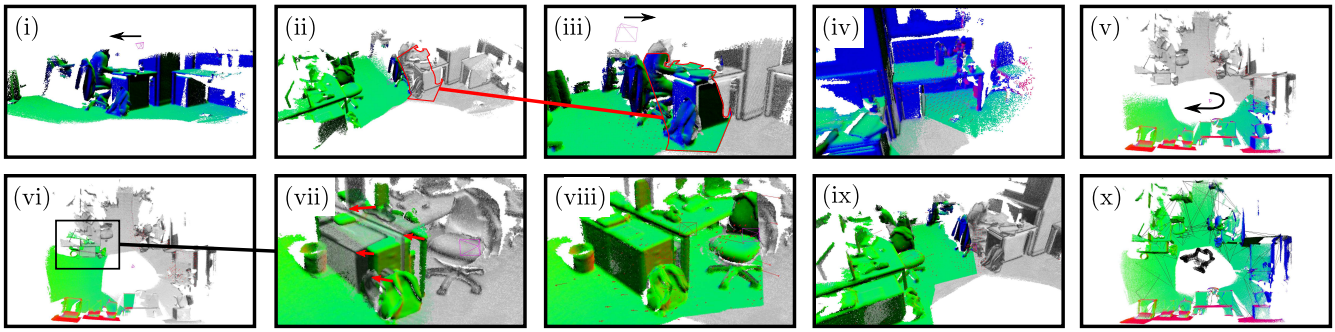


Figure 2: Example SLAM sequence with active model coloured by surface normal overlaid on the inactive model in greyscale; (i) initially all data is in the active model as the camera moves left; (ii) as time goes on, the area of map not seen recently is set to inactive. Note the highlighted area; (iii) 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.

across the full incrementally reconstructed surface and solve the global consistency problem online in real-time. Other previous works have applied deformation graphs specifically in object scanning (Weise et al. (2009)) and automatic skeleton rigging for 3D avatars (Chen et al. (2012)). There are also parallels in our approach to the real-time non-rigid reconstruction system of Newcombe et al. (2015) in how an embedded surface deformation graph is utilised for alignment rather than propagation of some camera pose update information.

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 8, 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 (first outlined in Whelan et al. (2015b)) 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. In this extended paper we provide a more in-depth description of the system and also present the novel addition of relative constraints within the space deformation, discovered to be required for more complex camera trajectories. Furthermore we build upon the predictive capabilities of the dense globally consistent maps made available with our system to develop an approach to real-time discrete light source detection, which to our knowledge has not been shown in any other similar system.

2 Approach Overview

We adopt an architecture which is typically found in real-time dense visual SLAM systems that alternates between tracking and mapping (Newcombe et al. (2011a); Whelan et al. (2015a); Keller et al. (2013); Henry et al. (2013); Chen et al. (2013); Newcombe et al. (2011b)). 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 (such as the Microsoft Kinect or ASUS Xtion Pro Live) 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. (2013), with some notable differences outlined in Section 3.

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 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 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 (Glocker et al. (2015)). 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 four main steps of our approach. An in-depth system architecture diagram is shown in Figure 3 along with a detailed caption. In the following section we describe our fused map representation and method for predictive tracking.

3 Fused Predicted Tracking

Our scene representation is an unordered list of surfels $\mathcal{M}$ (similar to the representation used by Keller et al. (2013)), 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, initialised as:

$$r = \frac{d\sqrt{2}}{f|\mathbf{n}_z|} \tag{1}$$

where d is the depth, f is the focal length of the depth camera and $\mathbf{n}_z$ the z component of the estimated normal (computed via central difference on the input depth map). Surfel weights are initialised as $w = e^{-\gamma^2/2\sigma^2}$ where γ is the normalised radial distance of the current depth measurement from the camera center and $\sigma = 0.6$ in accordance with previous work (Keller et al. (2013)). The update rules for each surfel component are detailed as follows, where the prime superscript (*e.g.* $\mathbf{p}'$) denotes the newly associated measurement for a given surfel (after live raw depth map registration), and the hat operator (*e.g.* $\hat{\mathbf{p}}$) denotes the new updated value for a given surfel at the next time step:

$$\hat{\mathbf{p}} = \frac{w\mathbf{p} + w'\mathbf{p}'}{w + w'} \tag{2}$$

$$\hat{\mathbf{n}} = \frac{w\mathbf{n} + w'\mathbf{n}'}{w + w'} \tag{3}$$

$$\hat{r} = \frac{wr + w'r'}{w + w'} \tag{4}$$

$$\hat{w} = w + w' \tag{5}$$

An equivalent update rule applies to the colour attribute unless light source estimation is being performed, which is described in Section 7.1.1. When using the map for pose estimation our approach differs to previous related work 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 RGB-D 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 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 $I(\mathbf{u}, \mathcal{C}) = \mathbf{c}(\mathbf{u})^\top \mathbf{i}$, where $\mathbf{i} = [0.114, 0.299, 0.587]^\top$. 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:

$$\mathbf{P}_t = \begin{bmatrix} & \mathbf{R}_t & & \mathbf{t}_t \\ 0 & 0 & 0 & 1 \end{bmatrix} \in \mathbb{SE}_3, \tag{6}$$

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

3.1 Geometric Pose Estimation

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

$$E_{icp} = \sum_k \left(\left(\mathbf{v}^k - \exp(\hat{\boldsymbol{\xi}}) \mathbf{T} \mathbf{v}_t^k \right) \cdot \mathbf{n}^k \right)^2, \tag{7}$$

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

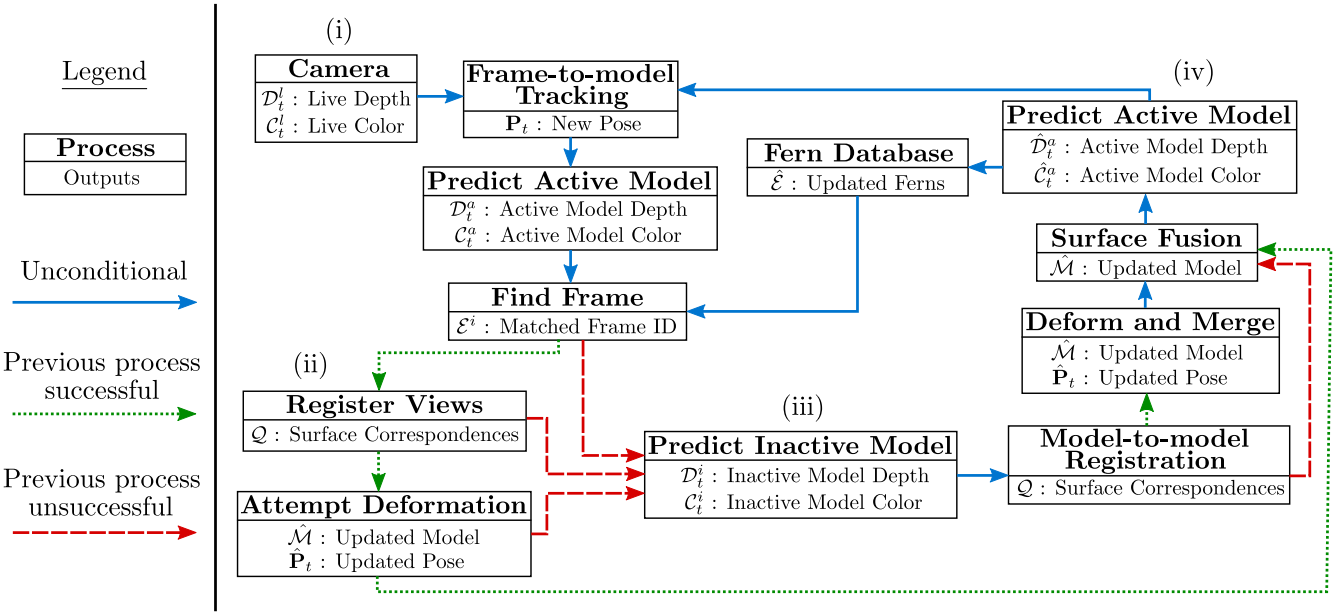


Figure 3: System architecture diagram; (i) the current live depth map $\mathcal{D}_t^l$ and colour image $\mathcal{C}_t^l$ are aligned with the predicted active model from the previous frame ($\hat{\mathcal{D}}_{t-1}^a$ and $\hat{\mathcal{C}}_{t-1}^a$). Next, a new predicted view of the active model ($\mathcal{D}_t^a$ and $\mathcal{C}_t^a$) is rendered and matched against the fern database; (ii) if a matching frame $\mathcal{E}_{id}$ is found, an attempt is made to register the matched view with the current predicted active view. If successful this yields a deformation of the model $\hat{\mathcal{M}}$ and pose update $\hat{P}_t$; (iii) failing this, the inactive model in the current view is predicted ($\mathcal{D}_t^i$ and $\mathcal{C}_t^i$) and then registered with the portion of the active model in the current view. If successful, the model is deformed to incorporate this registration, while all visible inactive points are merged with the set of active points; (iv) live camera data is fused with the latest updated model $\hat{\mathcal{M}}$ and an up to date prediction of the active model ($\hat{\mathcal{D}}_t^a$ and $\hat{\mathcal{C}}_t^a$) is rendered to track against in the next frame, while also potentially being added to the fern database $\hat{\mathcal{E}}$.

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 (Newcombe et al. (2011a)).

3.2 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:

$$E_{rgb} = \sum_{\mathbf{u}\in\Omega} \left(I(\mathbf{u}, \mathcal{C}_t^l) - I\left(\boldsymbol{\pi}(\mathbf{K}\exp(\hat{\boldsymbol{\xi}})\mathbf{T}\mathbf{p}(\mathbf{u}, \mathcal{D}_t^l)), \hat{\mathcal{C}}_{t-1}^a\right)\right)^2, \tag{8}$$

where as above $\mathbf{T}$ is the current estimate of the transformation from the previous camera pose to the current one. Note that Equations 7 and 8 omit conversion between 3-vectors and their corresponding homogeneous 4-vectors (as needed for multiplications with $\mathbf{T}$) for simplicity of notation.

3.3 Joint Optimisation

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

$$E_{track} = E_{icp} + w_{rgb}E_{rgb}, \tag{9}$$

with $w_{rgb} = 0.1$ in line with related work (Henry et al. (2013); Whelan et al. (2015a)). A visualisation of the residuals for this cost function is shown in Figure 4. To optimise this cost 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:

$$\arg\min_{\boldsymbol{\xi}} \|\mathbf{J}\boldsymbol{\xi} + \mathbf{r}\|_2^2, \tag{10}$$

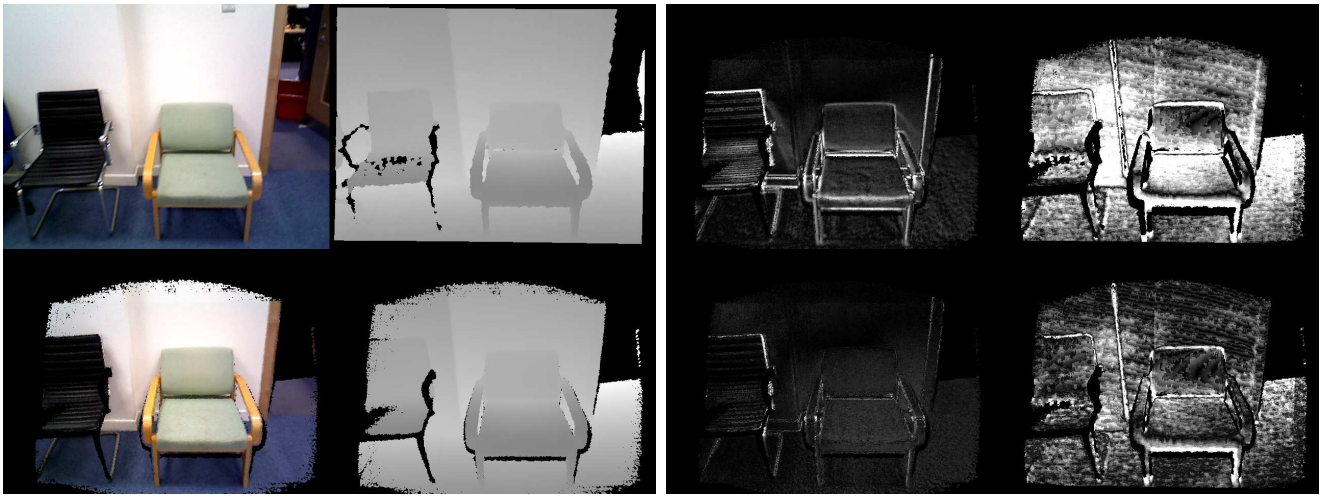
to yield an improved camera transformation estimate:

$$\mathbf{T}' = \exp(\hat{\boldsymbol{\xi}})\mathbf{T} \tag{11}$$

$$\hat{\boldsymbol{\xi}} = \begin{bmatrix} [\boldsymbol{\omega}]_\times & \mathbf{x} \\ 0\ 0\ 0 & 0 \end{bmatrix}, \tag{12}$$

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



(i) (ii)

Figure 4: Visualisation of frame-to-model tracking components; (i) shown in clockwise order is the live colour image $\mathcal{C}_t^l$, the live depth map $\mathcal{D}_t^l$, the predicted depth map $\mathcal{D}_{t-1}^a$ and the predicted colour image $\mathcal{C}_{t-1}^a$; (ii) the top row in this subfigure shows the initial residual errors for the photometric cost E_{rgb} and geometric cost E_{icp}. The bottom row shows the residual error after minimisation of the joint cost E_{track} according to Equation 9. In this particular frame the normalised initial average cost was reduced from 1.0 to 0.3.

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 $\mathcal{M}$).

4 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 5 and 6. We adopt a space deformation approach based on the embedded deformation technique of Sumner et al. (2007).

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 4.3.

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:

$$\hat{\mathcal{M}}_{\mathbf{p}}^s = \phi(\mathcal{M}^s) = \sum_{n \in \mathcal{I}(\mathcal{M}^s, \mathcal{G})} w^n(\mathcal{M}^s) \left[\mathcal{G}_{\mathbf{R}}^n(\mathcal{M}_{\mathbf{p}}^s - \mathcal{G}_{\mathbf{g}}^n) + \mathcal{G}_{\mathbf{g}}^n + \mathcal{G}_{\mathbf{t}}^n\right], \tag{13}$$

while the deformed normal of a surfel is given by:

$$\hat{\mathcal{M}}_{\mathbf{n}}^s = \sum_{n \in \mathcal{I}(\mathcal{M}^s, \mathcal{G})} w^n(\mathcal{M}^s) {\mathcal{G}_{\mathbf{R}}^n}^{-1^\top} \mathcal{M}_{\mathbf{n}}^s, \tag{14}$$

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 $n = k$:

$$w^n(\mathcal{M}^s) = (1 - \left\|\mathcal{M}_{\mathbf{p}}^s - \mathcal{G}_{\mathbf{g}}^n\right\|_2 / d_{max})^2. \tag{15}$$

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.

4.1 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

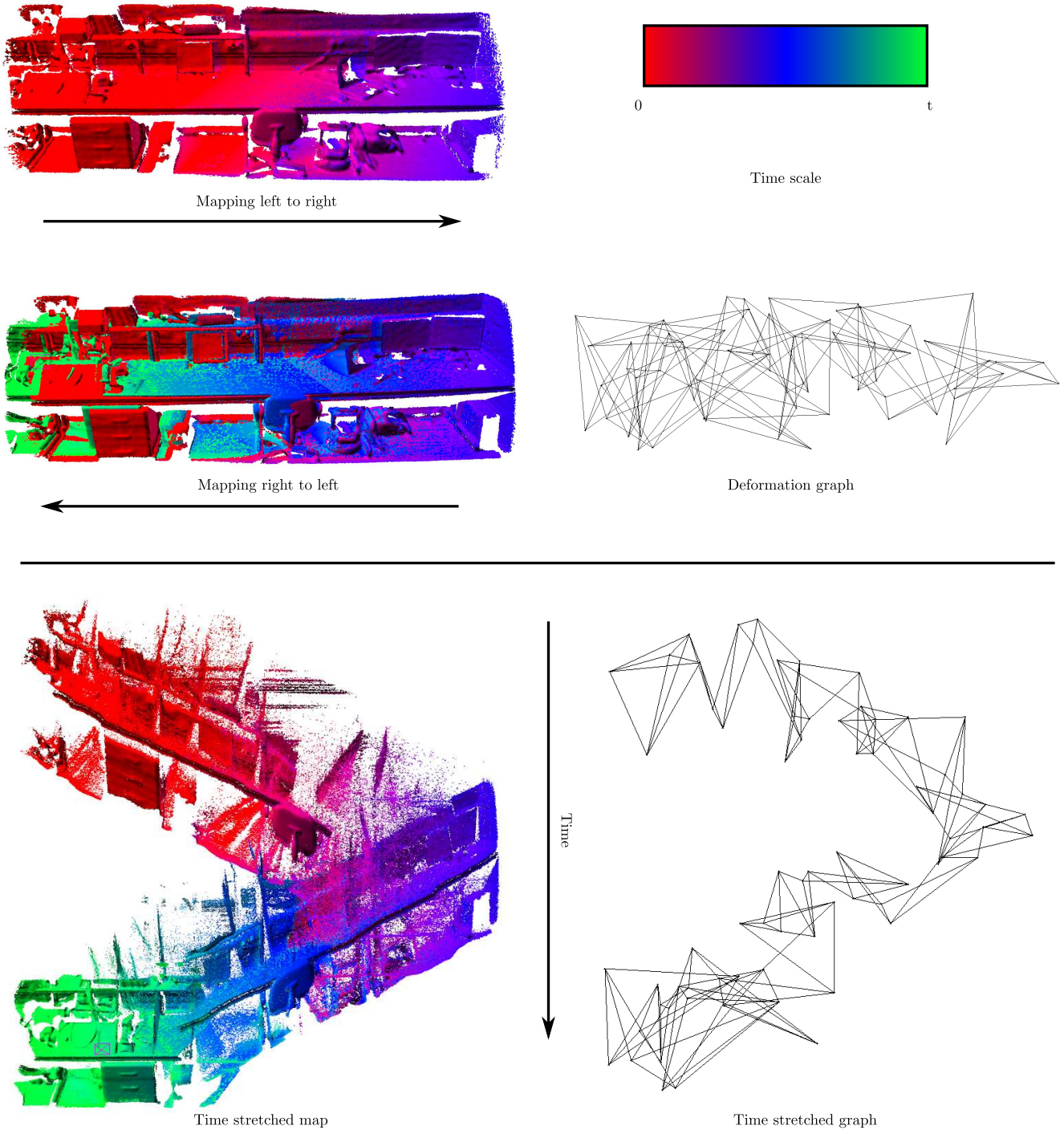


Figure 5: Temporal deformation graph connectivity before loop closure. The top half shows a mapping sequence where the camera first maps left to right over a desk area and then back across the same area. Given the windowed fusion process it appears that the map and hence deformation graph is tangled 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.

$\forall n, \mathcal{G}^n_{t_0} \geq \mathcal{G}^{n-1}_{t_0}, \mathcal{G}^{n-2}_{t_0}, \ldots, \mathcal{G}^0_{t_0}$. 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 such as Sumner et al. (2007) and Chen et al. (2012)) but more importantly prevents temporally uncorrelated areas of the surface from influencing each other (*i.e.* active and inactive areas), as shown in Figure 5. 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.

4.2 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. (2015a). 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 15 and the transformations from Equations 13 and 14 are applied. All other attributes of the updated surfel $\hat{\mathcal{M}}^s$ are copied from $\mathcal{M}^s$.

4.3 Optimisation

Given a set of surface correspondences $\mathcal{Q}$ and relative correspondences $\mathcal{R}$ (later expanded upon in Sections 5 and 6) 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 (both in the global frame $\underset{\rightarrow G}{\mathcal{F}}$) 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 five cost function summands over the deformation graph, the first maximises rigidity in the deformation:

$$E_{rot} = \sum_l \left\| {\mathcal{G}^l_{\mathbf{R}}}^\top \mathcal{G}^l_{\mathbf{R}} - \mathbf{I} \right\|^2_F, \tag{16}$$

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`
$c \leftarrow \arg\min_i \left\| \mathcal{M}^s_{t_0} - \mathcal{G}^i_{t_0} \right\|_1$
`// Get set of temporally nearby nodes`
$\mathcal{I} \leftarrow \emptyset$
for $i \leftarrow -\alpha/2$ **to** $\alpha/2$ **do**
$\mathcal{I}^{i+\alpha/2} \leftarrow c + i$
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**
$w^n(\mathcal{M}^s) \leftarrow \left(1 - \left\| \mathcal{M}^s_{\mathbf{p}} - \mathcal{G}^n_{\mathbf{g}} \right\|_2 / d_{max}\right)^2$
$h \leftarrow h + w^n(\mathcal{M}^s)$
`// Apply transformations`
$\hat{\mathcal{M}}^s_{\mathbf{p}} = \sum_{n \in \mathcal{I}(\mathcal{M}^s, \mathcal{G})} \frac{w^n(\mathcal{M}^s)}{h} \left[\mathcal{G}^n_{\mathbf{R}}(\mathcal{M}^s_{\mathbf{p}} - \mathcal{G}^n_{\mathbf{g}}) + \mathcal{G}^n_{\mathbf{g}} + \mathcal{G}^n_{\mathbf{t}}\right]$
$\hat{\mathcal{M}}^s_{\mathbf{n}} = \sum_{n \in \mathcal{I}(\mathcal{M}^s, \mathcal{G})} \frac{w^n(\mathcal{M}^s)}{h} {\mathcal{G}^n_{\mathbf{R}}}^{-1\top} \mathcal{M}^s_{\mathbf{n}}$

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

$$E_{reg} = \sum_{l} \sum_{n \in \mathcal{N}(\mathcal{G}^l)} \left\| \mathcal{G}^l_{\mathbf{R}}(\mathcal{G}^n_{\mathbf{g}} - \mathcal{G}^l_{\mathbf{g}}) + \mathcal{G}^l_{\mathbf{g}} + \mathcal{G}^l_{\mathbf{t}} - (\mathcal{G}^n_{\mathbf{g}} + \mathcal{G}^n_{\mathbf{t}}) \right\|_2^2 \tag{17}$$

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 13 to $\mathcal{Q}^p_{\mathbf{s}}$:

$$E_{con} = \sum_{p} \| \phi(\mathcal{Q}^p_{\mathbf{s}}) - \mathcal{Q}^p_{\mathbf{d}} \|_2^2 \tag{18}$$

Note that in order to apply Equation 13 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 fourth 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:

$$E_{pin} = \sum_{p} \| \phi(\mathcal{Q}^p_{\mathbf{d}}) - \mathcal{Q}^p_{\mathbf{d}} \|_2^2 \tag{19}$$

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 cost function uses the set of relative constraints $\mathcal{R}$ to prevent previous surface registrations from being pulled apart by subsequent global loop closures. An element $\mathcal{R}^p \in \mathcal{R}$ is a tuple $\mathcal{R}^p = (\mathcal{R}^p_{\mathbf{d}}; \mathcal{R}^p_{\mathbf{s}}; \mathcal{R}^p_{\mathbf{d}_t}; \mathcal{R}^p_{\mathbf{s}_t})$ which contains a pair of points (again both in the global frame $\underrightarrow{\mathcal{F}}_G$) representing a destination position $\mathcal{R}^p_{\mathbf{d}} \in \mathbb{R}^3$ and a source position $\mathcal{R}^p_{\mathbf{s}} \in \mathbb{R}^3$ whose relative distance should be minimised upon deformation:

$$E_{rel} = \sum_{p} \| \phi(\mathcal{R}^p_{\mathbf{s}}) - \phi(\mathcal{R}^p_{\mathbf{d}}) \|_2^2 \tag{20}$$

With $w_f = 1$, $w_r = 10$ and $w_c = 100$ (in line with related work (Sumner et al. (2007); Chen et al. (2012); Whelan et al. (2015a))), the combined total cost function for a local loop closure (detailed in Section 5) is defined as:

$$E_{loc} = w_f E_{rot} + w_r E_{reg} + w_c (E_{con} + E_{pin}) \tag{21}$$

and during a global loop closure (detailed in Section 6):

$$E_{glo} = w_f E_{rot} + w_r E_{reg} + w_c (E_{con} + E_{pin} + E_{rel}) \tag{22}$$

We minimise this total cost with respect to $\mathcal{G}^n_{\mathbf{R}}$ and $\mathcal{G}^n_{\mathbf{t}}$ over all n nodes during a global loop closure and over only the n nodes since the last loop closure during a local loop closure 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 surfel map as described in Section 4.2.

5 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, according to our architecture in Figure 3, 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 3. 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 cost of the Gauss-Newton tracking optimisation used to align the two views. The residual cost E_{track} from Equation 9 must be sufficiently small, while the number of inlier measurements used must be above a minimum threshold. We also inspect the eigenvalues of the relative pose 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 4 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 $\mathcal{T}^i_t$ 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:

$$\mathcal{Q}^p = ((\mathbf{H}\mathbf{P}_t)\mathbf{p}(\mathbf{u}, \mathcal{D}_t^a); \mathbf{P}_t\mathbf{p}(\mathbf{u}, \mathcal{D}_t^a); \mathcal{T}_t^i(\mathbf{u}); t). \quad (23)$$

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 $\tilde{\mathcal{D}}_t^a$. For each surfel $\mathcal{M}^s$:

$$\mathcal{M}_t^s = \begin{cases} t & \text{if } \boldsymbol{\pi}(\mathbf{K}\hat{\mathbf{P}}_t^{-1}\mathcal{M}_{\mathbf{p}}^s) \in \Omega \\ & \text{and } (\mathbf{K}\hat{\mathbf{P}}_t^{-1}\mathcal{M}_{\mathbf{p}}^s)_z \lesssim \tilde{\mathcal{D}}_t^a(\boldsymbol{\pi}(\mathbf{K}\hat{\mathbf{P}}_t^{-1}\mathcal{M}_{\mathbf{p}}^s)), \\ \mathcal{M}_t^s & \text{else.} \end{cases} \quad (24)$$

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. Note that for each successful local loop closure we cache a subsampled set of the post-deformation surface constraints $\mathcal{R} \subset \mathcal{Q}$ and convert them to relative constraints through Equation 20 in order to "stick" the registered surfaces together locally, while allowing their position to vary globally provided that their relative position does not change. The inclusion of this set of relative constraints $\mathcal{R}$ in the global loop closure cost in Equation 22 is also described in the following section.

6 Global Loop Closure

We utilise the randomised fern encoding approach of Glocker et al. (2015) for appearance-based place recognition. 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. (2015) 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. (2015) 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. (2015) for fern encoding, frame harvesting and identification of matching fern encodings. 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 $\mathcal{E}_{\mathbf{f}}^i$, a depth map $\mathcal{E}_{\mathcal{D}}^i$, a colour image $\mathcal{E}_{\mathcal{C}}^i$, a source camera pose $\mathcal{E}_{\mathbf{P}}^i$ and an initialisation time $\mathcal{E}_t^i$. At the end of each frame we add $\hat{\mathcal{D}}_t^a$ and $\hat{\mathcal{C}}_t^a$ (predicted active model depth and colour after fusion filled in with $\mathcal{D}_t^l$ and $\mathcal{C}_t^l$) 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 3 to bring $\mathcal{D}_t^a$ and $\mathcal{C}_t^a$ into alignment with $\mathcal{E}_{\mathcal{D}}^i$ and $\mathcal{E}_{\mathcal{C}}^i$, 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):

$$\mathcal{Q}^p = ((\mathbf{H}\mathcal{E}_{\mathbf{P}}^i)\mathbf{p}(\mathbf{u}, \mathcal{D}_t^a); \mathbf{P}_t\mathbf{p}(\mathbf{u}, \mathcal{D}_t^a); \mathcal{E}_t^i; t). \quad (25)$$

Note $\mathcal{Q}_{\mathbf{d}}^p$ which incorporates the difference in the estimated point position given by the alignment and the known actual global point position given by $\mathcal{E}_{\mathbf{P}}^i$. From here, the deformation cost from Equation 22 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 set of surfels $\mathcal{M}$. At this point the current pose estimate is also updated to $\hat{\mathbf{P}}_t = \mathbf{H}\mathcal{E}_{\mathbf{P}}^i$. 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).

6.1 Relative Constraints

Given that upon deformation optimisation the set of surface constraints $\mathcal{Q}$ are effectively "baked" into the map and forgotten about, it is possible (due to the temporal connectivity

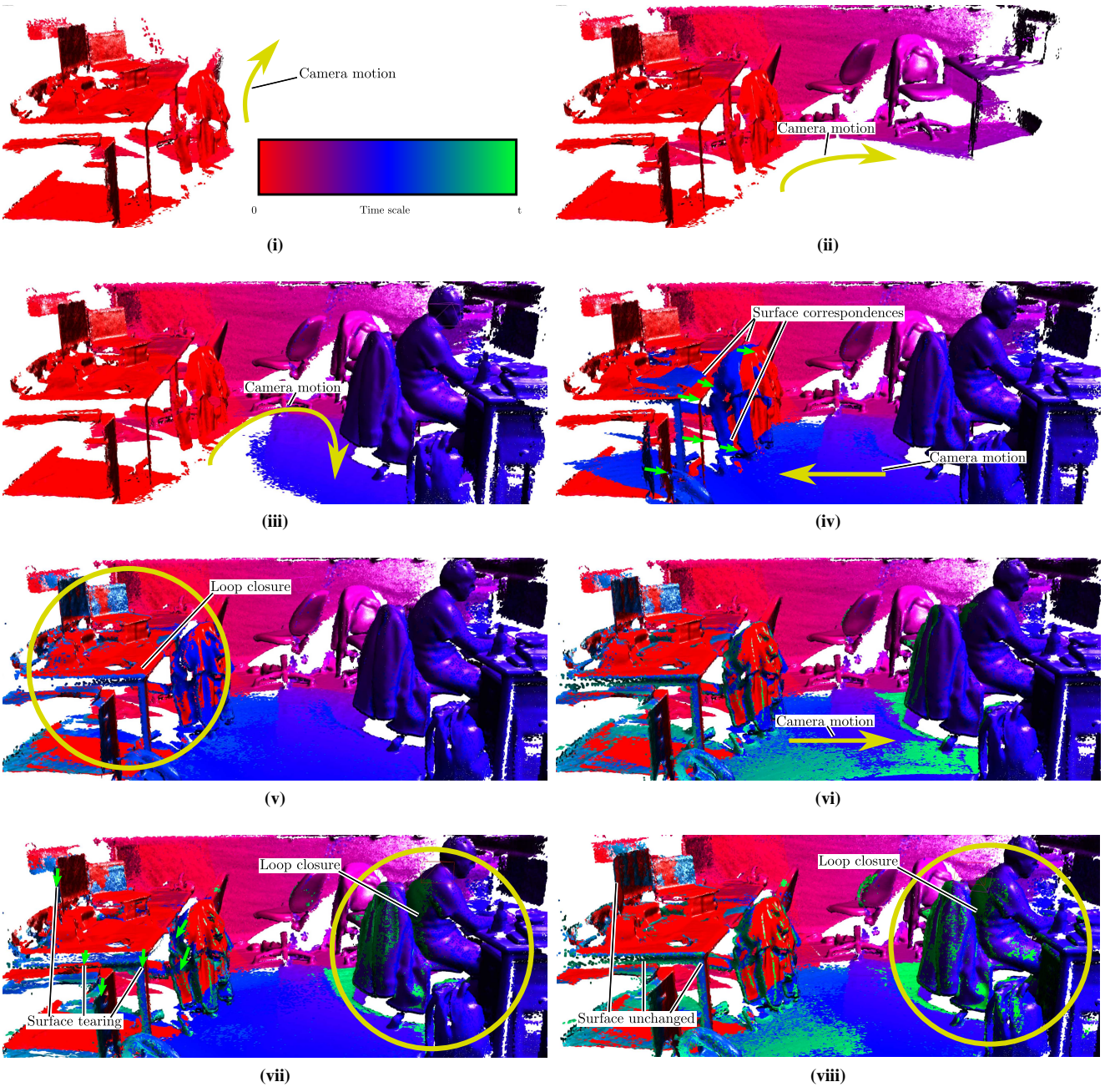


Figure 6: Relative constraint motivation. This figure shows a simple mapping sequence where relative constraints are necessary to prevent surface corruption; (i)-(iii) camera maps from the empty desk on the left towards the occupied desk on the right; (iv) the empty desk is revisited and remapped, here the surface correspondences are shown which are brought into alignment in Subfigure (v) by a loop closure; (vi) camera moves back towards the occupied desk on the right and closes another loop, however shown in Subfigure (vii) without relative constraints the previous loop closure registration is torn apart; (viii) shown finally is the same sequence except using relative constraints, preventing the previous loop closure from being torn apart.

parameterisation of the deformation graph) for previously registered areas to be torn apart during subsequent loop closures without the inclusion of relative constraints accumulated from all previous local loop closures. Without relative constraints, there is no term in the deformation optimisation which encourages previously registered areas of the map to stick together. Hence we cache a small number of relative surface constraints from each local loop closure to include in every subsequent deformation, preventing surface tearing in areas with previous loop closures. This is demonstrated in Figure 6. While this does imply that the number of constraints in the global loop closure optimisation grows continuously as more loops are closed, only a small number of relative constraints are required from the full set of surface constraints $\mathcal{Q}$ in each loop closure to maintain consistency. In practice this causes the execution time of the optimisation to only grow very slowly, where the predicted view rendering time still dominates in terms of scaling performance bottlenecks.

7 Light Source Detection

Up until now, capturing a room scale globally consistent map in real-time which can be used for fully dense prediction has been impossible. Now given this capability we can begin to reason about higher order scene attributes beyond geometry and diffuse lambertian surface appearance. In particular, discrete point light source estimation is of particular interest due to its usability in predictive tracking, path planning and real-time augmented reality effects. In order to detect a discrete set of light sources in an enviroment we exploit the predictive powers of the models produced by the described SLAM system. By using a globally consistent rich geometric map in our light source estimation process we can make many meaningful predictions and assumptions about measurements accrued from raw data. 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 utilises the surfel-based fusion system described in the previous sections. However, there are some important points in our approach to colour data fusion required for diffuse appearance estimation which we later detail in Section 7.1.

2. While camera tracking and data fusion is carried out, detect specular light source reflections off individual surfels using the estimated diffuse appearance of each surfel.

3. Aggregate each reflected ray measurement in a 3D spatially hashed voxel space in a raycast hough-like voting scheme.

4. Integrate information about the observed geometry of the scene (surfaces and freespace) into the voxel space to cull and constrain possible light source positions.

5. Extract voxels at the intersection of many reflected ray votes and scene geometry as hypothesised light sources.

6. Cluster all hypothesised light source voxels together to retrieve a set of discrete light sources.

The full pipeline that composes our method runs in real-time at camera framerate while the user is exploring the scene. This allows online estimation of numerous light sources without any offline or batch post-processing steps. In the following section we describe our underlying scene representation and method for detecting specular light source ray reflections.

7.1 Specular Reflection Detection

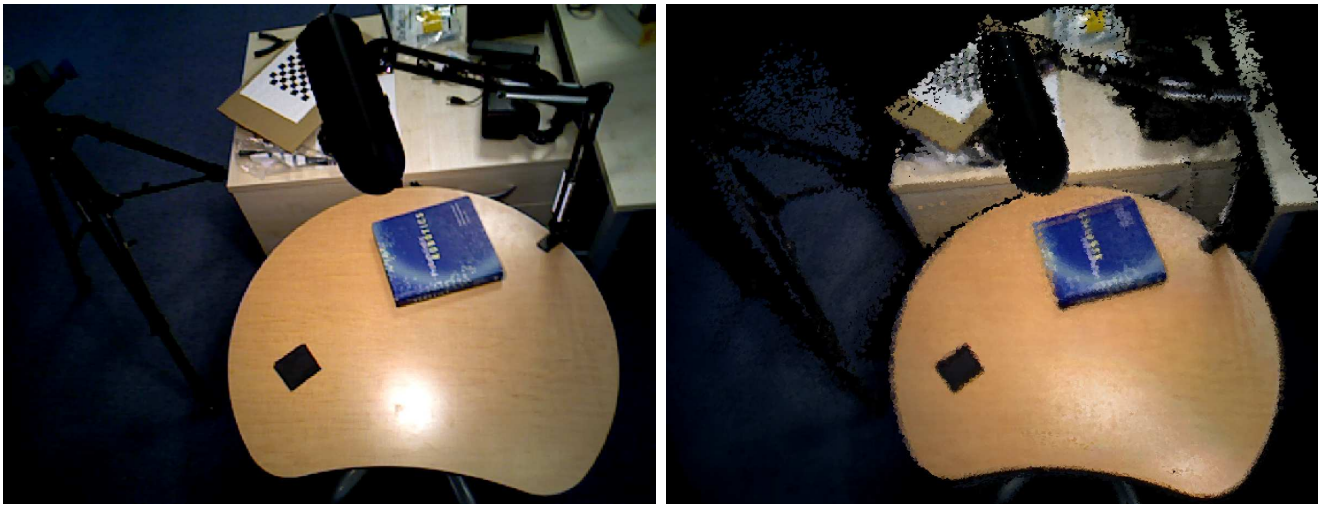
As mentioned previously, our method is grounded on the availability of a rich, fully predictive dense 3D geometric model. While many approaches exist to create such a model in real-time there are few that guarantee global consistency (Whelan et al. (2015a)). In order for reflected ray measurements to agree our model must be high quality and drift-free (as provided by our described dense SLAM system). The underlying representation in this system is a dense map of surfels. We describe our specific rules for colour data fusion for diffuse appearance estimation below. Note that we disable the automatic exposure and white balance features on the sensor prior to beginning data capture with our system (known to be an undesirable attribute of popular RGB-D sensors). This is a current limitation of our approach as there is an assumption made that the only changes in intensity observed for a given surface will be due to a light source in the scene propagating through the bidirectional reflectance distribution function of that surface. However, if the exact exposure and white balance settings for the sensor were available at the time of capture (which is not the case with current consumer depth cameras), along with the camera response function for the detector, the algorithm could easily utilise data with variable exposure and white balance.

7.1.1 Surfel-based Fusion

As mentioned in Section 3, each surfel $\mathcal{M}^s$ has a diffuse colour $\mathbf{c} \in \mathbb{R}^3$ (stored as unit RGB components). We also store for each surfel the last brightest intensity measured for that surfel $l \in \mathbb{R}$, the direction of the ray reflected by that surfel at its last brightest viewing angle $\mathbf{h} \in \mathbb{R}^3$ and the difference between the diffuse surfel intensity and the maximum measured intensity at its last brightest viewing angle $\Delta l \in \mathbb{R}$. In all cases, where applicable, the surfel attribute quantities are in the global coordinate frame.

When fusing colour information our approach differs to previous work; instead of simply fusing in a moving average fashion, we only fuse colour measurements which are strictly no brighter than the current colour value by a large margin.

To update the diffuse colour value we use the following scheme (where like in Section 3, $\mathbf{i}$ maps colour to intensity):



(i) (ii)

Figure 7: Diffuse colour surface reconstruction; (i) real RGB image of the current scene; (ii) fused diffuse colour surface reconstructed with notable diminishing of the specular reflection from the point light source in the scene.

$$\alpha = \mathbf{c} \cdot \mathbf{i} \tag{26}$$

$$\beta = \mathbf{c}' \cdot \mathbf{i} \tag{27}$$

$$\hat{\mathbf{c}} = \begin{cases} \frac{w\mathbf{c}+w'\mathbf{c}'}{w+w'} & \text{if } \beta \leq \alpha + \epsilon \\ \mathbf{c} & \text{else.} \end{cases} \tag{28}$$

An example output of this process is shown in Figure 7.

7.1.2 Ray Detection

In order to detect reflected ray measurements we compare the individual estimated diffuse surfel intensities to raw live measurement intensities of pixels associated with each surfel during data fusion. For each input frame during the data association step, we associate at most one pixel measurement with each surfel in the model. This is accomplished by splatting the surfel model as a predicted view of the scene given the most recently estimated camera pose $\mathbf{P}_t$, and associating raw depth and colour measurements with each surfel based on some data association metrics (for more details see Keller et al. (2013)). From here, we can compute the α and β diffuse and raw intensity values respectively (as listed in Equations 26 and 27). If the raw intensity value β is greater than the previous maximum intensity for that surfel l, we can update the lighting related components of the surfel as follows, where italicised values such as $\boldsymbol{p}$ denote the value of $\mathbf{p}$ in the camera coordinate frame (*i.e.* including homogenisation and dehomogenisation through multiplication with ${\mathbf{P}_t}^{-1}$) and the tilde operator denotes normalisation (*e.g.* $\tilde{\mathbf{a}} = \frac{\mathbf{a}}{||\mathbf{a}||}$):

$$\hat{l} = \beta \tag{29}$$

$$\hat{\Delta l} = \beta - \alpha \tag{30}$$

$$\hat{\mathbf{h}} = \mathbf{R}_t(\tilde{\boldsymbol{p}} - 2(\tilde{\boldsymbol{p}} \cdot \boldsymbol{n})\boldsymbol{n}) \tag{31}$$

Each frame, any surfel that had its maximum intensity value l updated triggers a new ray measurement if the new Δl value of that surfel is above some predefined threshold. This essentially implements the logic that if we observe an intensity much greater than our estimated diffuse intensity for a given surfel, we are probably measuring a bright specular reflection. In order to prevent spurious ray measurements due to misalignment errors between the raw image data and the predicted model data, we exclude large intensity change measurements in points of the predicted image that have a high local gradient in image space. Each frame the output of this process is a set of new reflected ray measurements $\mathcal{H}$. Each ray measurement has a source position component $\mathcal{H}_{\mathbf{p}}$, which is the position of the surfel the ray was observed from, and a direction component $\mathcal{H}_{\mathbf{h}}$, being the direction the reflected ray travels off the surfel. This process is visualised in Figure 8. This set of measurements is fed into our pipeline for light source estimation, which we describe in the following section.

7.2 Light Source Estimation

In order to estimate regions in space which are likely to contain light sources we integrate both the information contained within the estimated geometry of the scene and the specular reflection ray measurements discussed in the previous section. The data structure used in this procedure is a spatially hashed voxel grid $\mathcal{V}$. Using spatial hashing has the advantages of

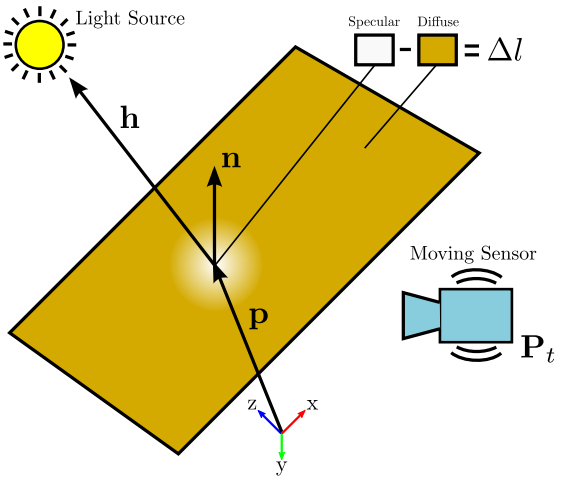


Figure 8: Specular reflection ray detection described in Section 7.1.2. This figure shows the relationship between the moving sensor, diffuse surface reconstruction and light source in terms of the vector quantities used to estimate the source ray $\mathbf{h}$ which produces a specular reflection on a surface.

being memory efficient, computationally efficient in terms of look-up and also not requiring the predefinition of the spatial extent of the voxel grid (Nießner et al. (2013)). Each voxel $\mathcal{V}^i \in \mathcal{V}$ has a number of attributes; a position $\mathcal{V}_\mathbf{p} \in \mathbb{R}^3$, an integration value $\mathcal{V}_n \in \mathbb{N}$, a freespace flag and a geometry flag. Voxels are dynamically allocated on-the-fly as they are required.

7.2.1 *Ray Measurement Integration*

Each frame we integrate the set of reflected ray measurements $\mathcal{H}$ into the voxel grid, incrementing the integration value $\mathcal{V}_n$ of each voxel the rays pass through. Rather than casting a single ray through the voxel grid a cone of rays is cast through space, generating a spread which aids in representing the inherent uncertainty in the ray measurements. Rays are cast to a fixed distance away from the source positions $\mathcal{H}_\mathbf{p}$ (although rays are not cast through scene geometry), typically no more than 3m was found to be sufficient. This does mean that we rule out the possibility of detecting light sources at infinity, however in principle we could extend our approach to include these as part of future work. Voxels are marked using a 3D variation of the Bresenham line rasterisation algorithm to ensure good coverage.

7.2.2 *Geometry Measurement Integration*

Each frame we also integrate the information contained within the reconstructed geometry of the scene. A low resolution predicted depth map rendering of the model from the current estimated camera pose is sampled and integrated into the voxel grid to mark voxels as freespace and also as "geometry" voxels. Rays from the current camera pose to each of the predicted depth map pixels are cast, where every voxel crossed along the path is marked as freespace. Freespace voxels are

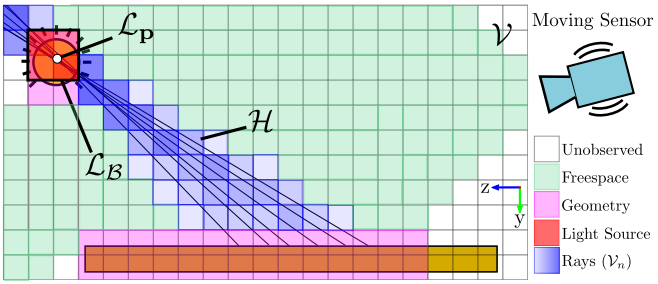


Figure 9: Ray measurement integration into the voxel grid for Figure 8. Shown is the main voxel grid $\mathcal{V}$, a number of integrated ray measurements $\mathcal{H}$ and a light source detected with centroid $\mathcal{L}_\mathbf{p}$ and bounding volume $\mathcal{L}_\mathcal{B}$.

not valid candidates for light sources and have their integration values set permanently to 0. The voxels where the actual geometry lies are marked as "geometry" voxels. If a voxel is marked as a geometry voxel and it contains an integration value above some threshold, it is marked as a potential light source voxel. Figure 9 shows a 2D example of the ray measurement integration process for the scene shown in Figure 8.

7.2.3 *Light Source Clustering*

Each newly marked potential light source voxel from the previous step is added to a second voxel grid volume which only contains voxels that are hypothesised to belong to light sources. Clusters with no observed geometry are not included in our estimation process due to the inherent ambiguity in the light source's position and extent given only bundles of ray measurements which may not accurately converge at the exact point in space from where they are emitted. In order to detect light sources in such a manner, a more comprehensive collection of ray measurements with a specific camera trajectory would be required (akin to the work of Jachnik et al. (2012)).

It is in the second volume that light source clustering occurs. We maintain a list of light sources $\mathcal{L}$, where each light source has a list of voxels making up that light source $\mathcal{L}_\mathcal{V}$, a centroid position $\mathcal{L}_\mathbf{p} \in \mathbb{R}^3$ to represent the 3D position of the actual estimated light source and a bounding volume $\mathcal{L}_\mathcal{B}$ which contains the 3D axis-aligned bounding box that contains all voxels belonging to that light source.

Firstly, for each new light source voxel measurement, we test the 6-connected neighbours of that voxel to see if an existing light source already neighbours the voxel. If so, the new light source voxel is added to the existing light source, while the matching light source's centroid and bounding volume is updated. If there are no existing neighbouring light sources at this point, a new light source is initialised with the new light source voxel as its only member.

Next we carry out a light source merging step. For all existing light sources in the scene, we identify pairs which have overlapping bounding volumes. If any of the voxels in these pairs of light sources are neighbouring in the 6-connected

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 1: Comparison of ATE RMSE on the evaluated real world datasets of Sturm et al. (2012).

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 3: Comparison of ATE RMSE on the evaluated synthetic datasets of Handa et al. (2014).

sense, we merge the two light sources together (deleting one light source and updating the list of voxels, centroid and bounding volume of the other). The process is repeated until it is no longer possible to merge any of the current light sources in this manner.

The result of this process is a set of discrete light sources, each with a spatial extent (encoded in the set of voxels clustered into each light source) and a hypothesised central point. In additional to this, we aggregate all rays which have contributed to each light source in order to estimate the source directionality (essentially the variance of the directions of all reflected rays for a given light source). This information can then be used to add rendered light sources to the scene for either aiding predictive photometric tracking or producing more convincing AR effects. We present results on both of these use cases in Section 9.

8 Dense SLAM Evaluation

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

8.1 Trajectory Estimation

To evaluate the trajectory estimation performance of our approach we test our system on the RGB-D benchmark of Sturm et al. (2012). 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 1 we compare our system to four other state-of-the-art RGB-D based SLAM systems; DVO SLAM of Kerl et al. (2013), RGB-D SLAM of Endres et al. (2012), MRSMap of Stückler and Behnke (2014) and Kintinuous of Whelan et al. (2015a). 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 (Sturm et al. (2012)). These results show that our trajectory estimation performance is on par with or better than existing state-of-the-art systems that 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.

In addition to these comparison results we have also evaluated our system on all of the static scenes in the RGB-D benchmark of Sturm et al. (2012). Our results are listed in Table 2, along with dataset statistics and discussion on the achieved performance.

8.2 Surface Estimation

We evaluate the surface reconstruction results of our approach on the ICL-NUIM dataset of Handa et al. (2014). 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 8.1. We also include trajectory estimation results for each dataset. Tables 3 and 4 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 10 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 5. 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 (2013).

Dataset	RMSE	(i) Frame Drops	(ii) Depth Fill	(iii) $\bar{\omega}$ $(^{\circ}/s)$
freiburg1_360	0.108m	1.6%	94.5%	41.600
freiburg1_desk	0.020m	3.1%	90.8%	23.327
freiburg1_desk2	0.048m	3.2%	90.1%	29.308
freiburg1_floor	—	11.8%	98.0%	15.071
freiburg1_plant	0.022m	1.3%	91.9%	27.891
freiburg1_room	0.068m	0.7%	90.6%	29.882
freiburg1_rpy	0.025m	4.0%	94.7%	50.147
freiburg1_teddy	0.083m	1.3%	93.4%	21.320
freiburg1_xyz	0.011m	0.8%	93.7%	8.920
freiburg2_360_hemisphere	—	2.2%	77.3%	20.569
freiburg2_360_kidnap	—	1.2%	73.6%	13.425
freiburg2_coke	—	2.0%	80.9%	9.432
freiburg2_desk	0.071m	2.5%	92.2%	6.338
freiburg2_dishes	—	2.2%	82.5%	9.666
freiburg2_large_no_loop	—	1.8%	82.5%	15.090
freiburg2_large_with_loop	—	2.4%	75.1%	17.211
freiburg2_metallic_sphere	—	1.7%	85.2%	10.422
freiburg2_metallic_sphere2	—	1.2%	75.2%	12.946
freiburg2_pioneer_360	—	61.6%	79.8%	12.053
freiburg2_pioneer_slam	—	52.7%	84.0%	13.379
freiburg2_pioneer_slam2	—	52.3%	90.6%	12.209
freiburg2_pioneer_slam3	—	32.3%	78.7%	12.339
freiburg2_rpy	0.015m	2.1%	78.3%	5.774
freiburg2_xyz	0.011m	1.5%	84.2%	1.716
freiburg3_cabinet	—	4.1%	97.8%	10.248
freiburg3_large_cabinet	0.099m	3.9%	85.2%	8.747
freiburg3_long_office_household	0.017m	4.9%	94.4%	10.188
freiburg3_nostructure_notexture_far	—	4.7%	99.9%	2.712
freiburg3_nostructure_notexture_near_withloop	—	3.9%	99.4%	11.241
freiburg3_nostructure_texture_far	0.074m	4.0%	99.6%	2.890
freiburg3_nostructure_texture_near_withloop	0.016m	3.5%	99.4%	7.430
freiburg3_structure_notexture_far	0.030m	3.5%	98.1%	4.000
freiburg3_structure_notexture_near	0.021m	3.7%	98.2%	6.247
freiburg3_structure_texture_far	0.013m	4.7%	97.6%	4.323
freiburg3_structure_texture_near	0.015m	4.9%	97.2%	7.677
freiburg3_teddy	0.049m	4.2%	80.3%	20.410

Table 2: ATE RMSE on the evaluated static scene datasets of Sturm et al. (2012), using per-sequence best parameters. We have included a number of statistics for each dataset which aid in the understanding of limited performance on some sequences; (i) although the capturing device provides RGB-D frames at 30Hz, a number of sequences are missing a certain amount of frames that were simply never recorded. This hinders dense tracking methods which rely on a continuous sequence of frames to satisfy the projective data association assumption made; (ii) on several sequences the capturing device was pointed at an area of the scene outside of the range of valid depths for the device (either too near or too far), resulting in a low overall percentage of valid depth pixels throughout the dataset. The depth fill is the percentage of pixels which have valid depth values across the entire sequence; (iii) a high average angular velocity (given in degrees per second) for the ground truth trajectory implies that at certain points in a sequence successive frames were very far apart in orientation, again challenging the assumption made by projective data association used in dense tracking. We have omitted the results for sequences on which our system failed because of a loss of the camera pose estimate due these three phenomena.

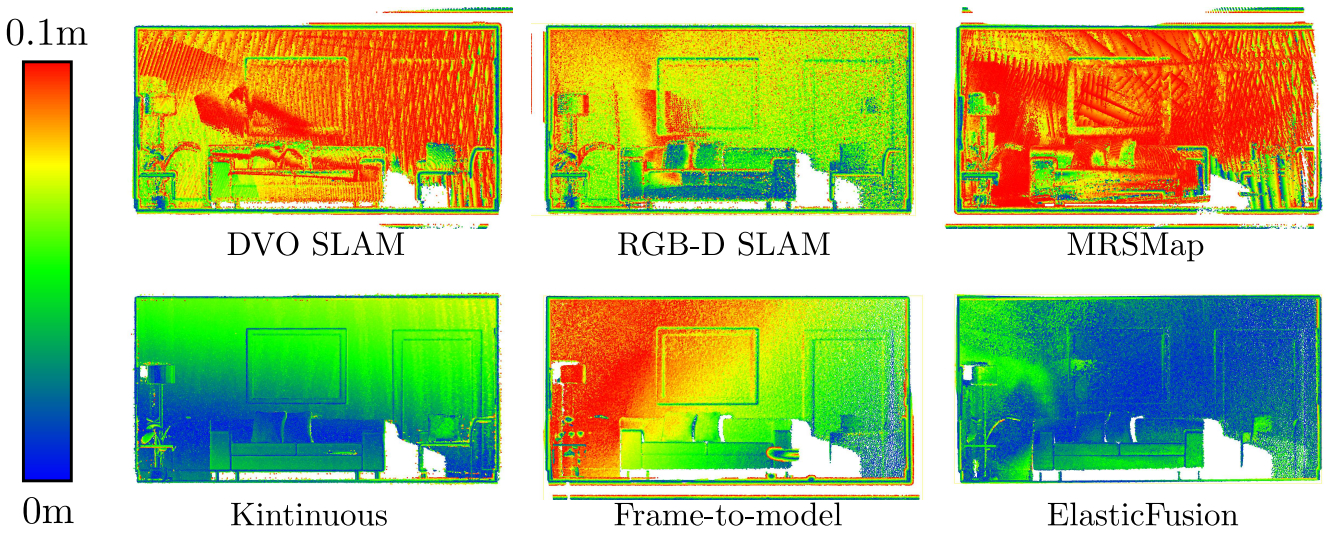


Figure 10: Orthogonal frontal view heat maps showing reconstruction error on the kt0 dataset. Points more than 0.1m from ground truth have been removed for visualisation purposes.

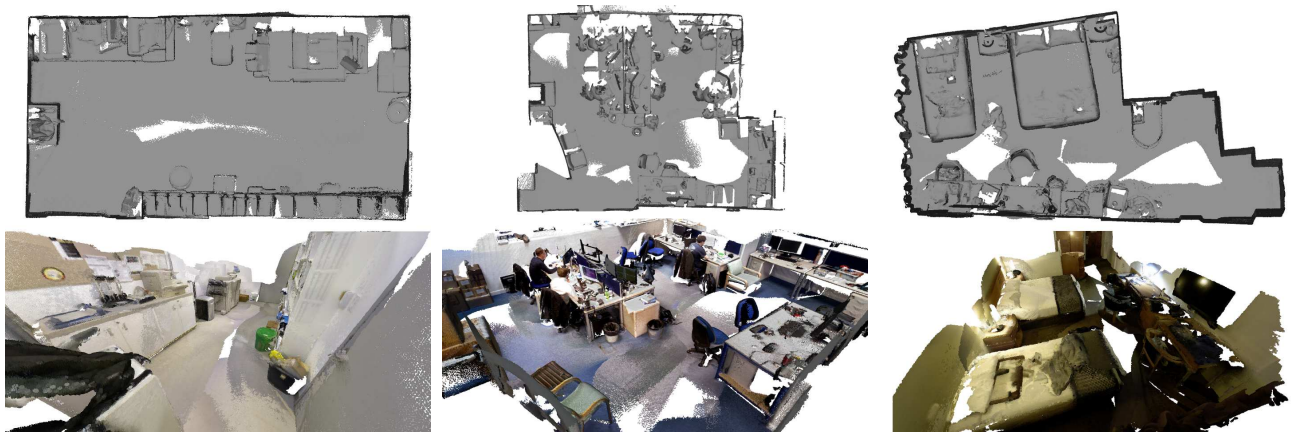


Figure 11: Qualitative datasets; (i) a comprehensive scan of a copy room; (ii) a loopy large scan of a computer lab; (iii) a comprehensive scan of a twin bed hotel room (note that the actual room is not rectilinear). To view small details we recommend using the digital zoom function in a PDF reader.

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 4: Comparison of surface reconstruction accuracy results on the evaluated synthetic datasets of Handa et al. (2014). Quantities shown are the mean distances from each point to the nearest surface in the ground truth 3D model.

Name (Fig.)	Copy (11i)	Lab (11ii)	Hotel (11iii)	Office (1)
Frames	5490	6533	7725	5000
Surfels	4.4×10^6	3.5×10^6	4.1×10^6	4.8×10^6
Map size (m)	5×3×2	8×6×2	8×4×2	3×3×2
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 5: Statistics on qualitative datasets.

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 an extensive 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 (`https://youtu.be/XySrhZpODYs`, `https://youtu.be/-dz_VauPjEU`).

8.3 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 12 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 (Whelan et al. (2012); Salas-Moreno et al. (2014); Chen et al. (2013); Keller et al. (2013)), and as shown in our qualitative results more than adequate for real-time operation. The current limitation in terms of computational performance is the cost of rendering predicted colour and depth images for tracking, where at around 8 million surfels the full pipeline begins to take more than 66ms to execute (15Hz), which manifests itself in requiring camera motion to slow down in order for tracking to succeed.

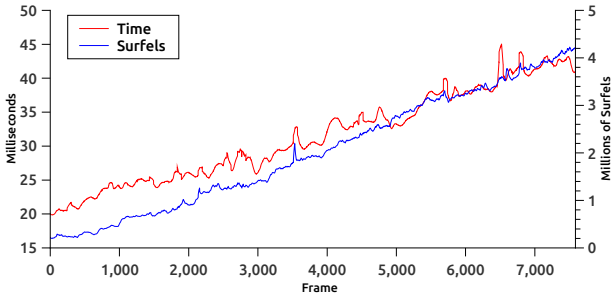


Figure 12: Per-frame processing time and number of surfels in the Hotel dataset.

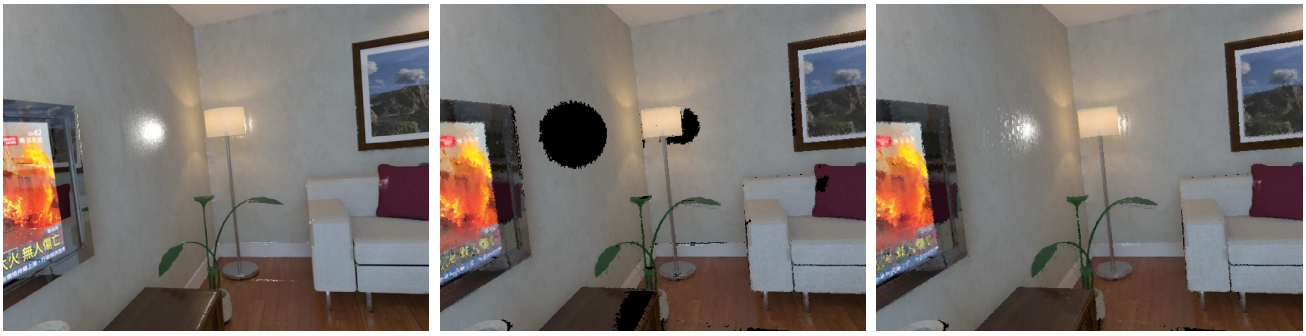
9 Light Detection Evaluation

In this section we present both quantitative and qualitative results on our light source estimation pipeline, demonstrating its usefulness in both camera pose estimation and realistic scene rendering for AR.

9.1 Specular Predictive Photometric Tracking

Given the discrete light source detection process outlined in Section 7.2 it is possible to augment the rendered colour image prediction of the model used for photometric tracking (previously referred to as $\hat{\mathcal{C}}^a_{t-1}$ in Equation 8) with specular highlights according to any detected light sources in the scene. However, given the enormous complexity in modeling light sources accurately enough to perform strong predictions, we found in this work it was most beneficial to simply mask out pixels believed to be under the effect of a specular light source reflection rather than use a possibly incorrect photometric prediction for tracking. In addition to this, as in previous work we assume tracking operates over a narrow baseline (Newcombe et al. (2011b)). However, like image areas on occlusion boundaries, for a correct prediction re-rendering the predicted specular reflection would be required during each iteration of the Gauss-Newton optimisation used to solve Equation 9. This can be quite costly and we find that conservatively masking out potentially specular pixels in the optimisation achieves promising results in the direction of utilising the light source information directly in the tracking pipeline.

In these experiments, we simulate a simple point light source specular reflection for each detected light source in the scene and mask out all pixels in the predicted colour image which may have their appearance altered under reflection of those light sources. We modified and augmented a number of sequences from the noiseless ICL-NUIM dataset of Handa et al. (2014) with specular light sources in order to generate ground truth data. We were unable to evaluate the light source estimation algorithm on the RGB-D benchmark of Sturm et al. (2012) as the automatic exposure and white balance camera functions were enabled during capture, which violates the as-



(i) (ii) (iii)

Figure 13: Example of predicting the effect of specular lighting on model views for tracking; (i) the real image of the scene from the current camera pose; (ii) a predicted view of the scene taking into account the specular reflections of any light sources that have been detected and masked out; (iii) a predicted view of the scene without any predicted specular light source reflection masking. Not only is this prediction worse for tracking, but it runs the risk of smearing the moving specular reflection captured in the raw data into the diffuse estimated colour of the surface over time.

Prediction	kt0	kt1	kt2
Diffuse	0.02974m	0.04791m	0.08085m
Masked	**0.00385m**	**0.03760m**	**0.03453m**

Table 6: Comparison of ATE RMSE on our augmented ICL-NUIM dataset for diffuse predictive rendering and specular predictive masked rendering taking detected scene light sources into account. Note that this experiment only evaluates pure tracking and does not include loop closures in order to perform a fair comparison.

sumption highlighted in Section 7.1.

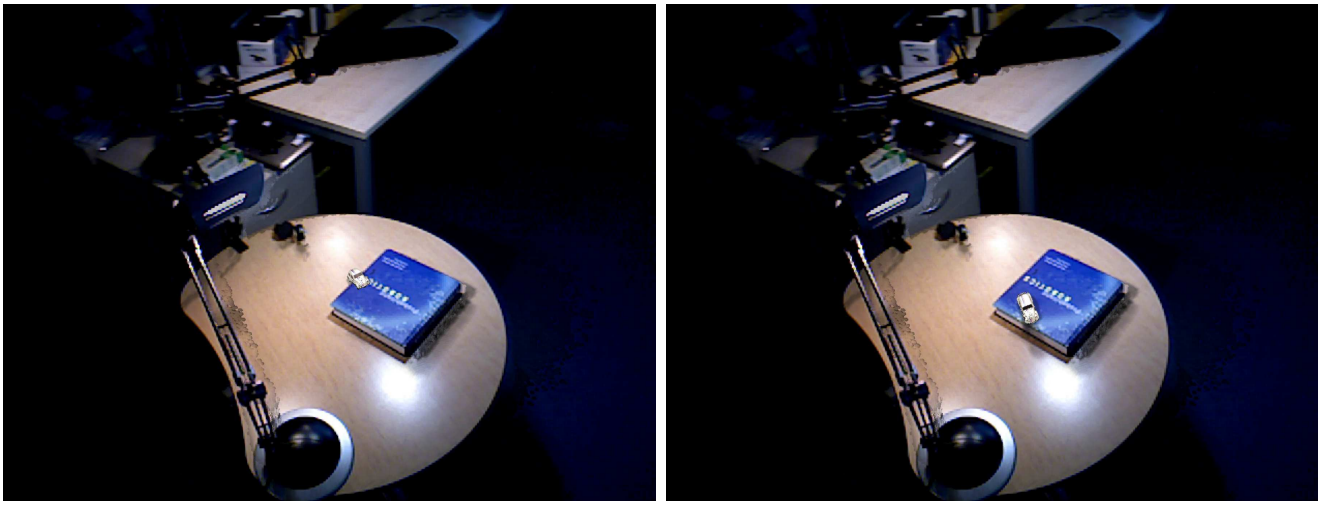
Figure 13 shows a comparison between an actual real image including specular highlights and two model renders for predictive tracking; one with masked out specular highlights and one without. We quantitatively evaluate the performance of our photometric camera tracking method on three simulated sequences both with our light source estimation pipeline running (masking predicted specular highlights) and without. The results are listed in Table 6. It is evident that including information hinted at by these specular highlight predictions to the real-time tracking pipeline has a very positive effect on pose estimation performance. There is clearly a benefit to introducing high level reasoning and modeling of the environment being captured beyond simple geometric and diffuse appearance properties. This kind of scene understanding is also clearly useful for motion planning and active vision systems in a robotics context. Information on bright light source positions and directions can be taken into account when deciding where a visual sensor should be oriented, potentially avoiding saturation or dynamic range issues during real-time perception.

9.2 Spotlight Synthesis for Augmented Reality

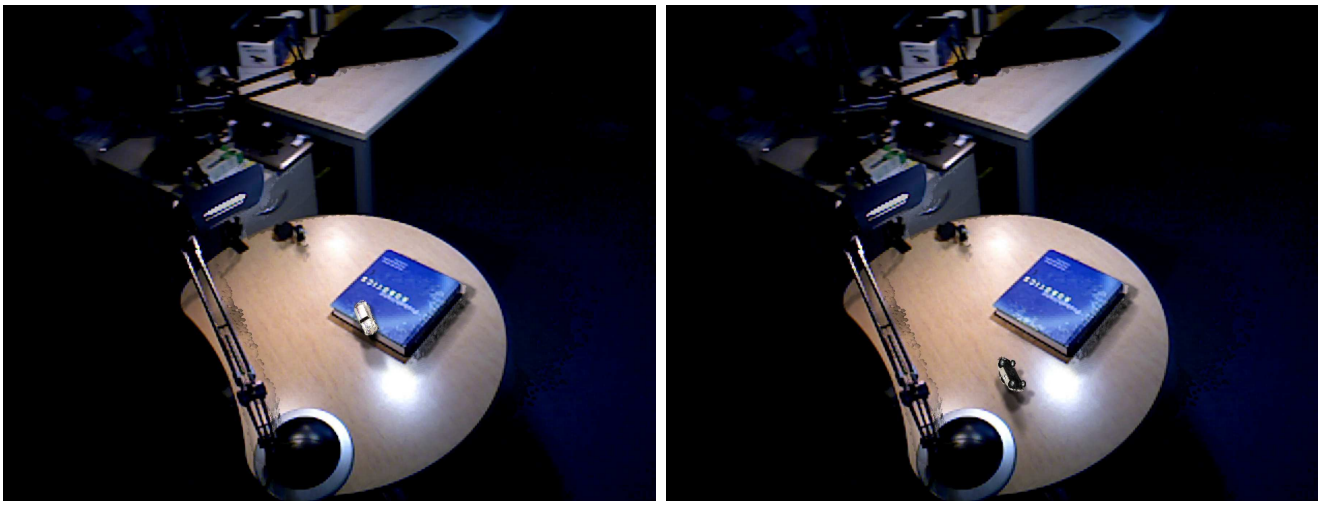
Beyond robotics light source information also has huge benefits in augmented reality applications. Since the inception of real-time dense scene perception methods physically realistic virtual object-scene interaction has been possible. There has been previous work along the lines of approximating a full hemispherical environment lighting map for a single plane, which produces convincing re-lighting, shadowing and reflection effects, although quite constrained in operation (Jachnik et al. (2012)). In Figures 14 and 15 along with our accompanying video (`https://youtu.be/QFDnFjV9YdM`), we show real-time physically interactive augmented reality compositing which respects the geometry, occlusion properties, point-source lighting effects and shadowing of the scenes in which the sequences take place. Our video also shows a qualitative comparison of stability when utilising specular predictive photometric tracking, as outlined above. We believe that this initial step towards approximating the scene lightfield with a discrete set of simply modeled light sources is already quite compelling and gives a clear indicator of the future direction in which general real-time 3D scene perception must go beyond dense surface diffuse appearance and geometry.

10 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 addition to this we have presented a novel method for detecting multiple discrete point light sources in a scene in



(i) **(ii)**



(iii) **(iv)**

Figure 14: Example AR sequence involving three specular light source detections showing geometry interaction, realistic virtual object specular reflections and shadowing; (i) here the virtual car can been seen driving up on top of the book, respecting the scene geometry; (ii) specular and shadowing effects which respect the detected light sources in the scene are evidient; (iii)-(iv) further movement again highlights the geometry and lighting effects that can be rendered in real-time to augment the scene.

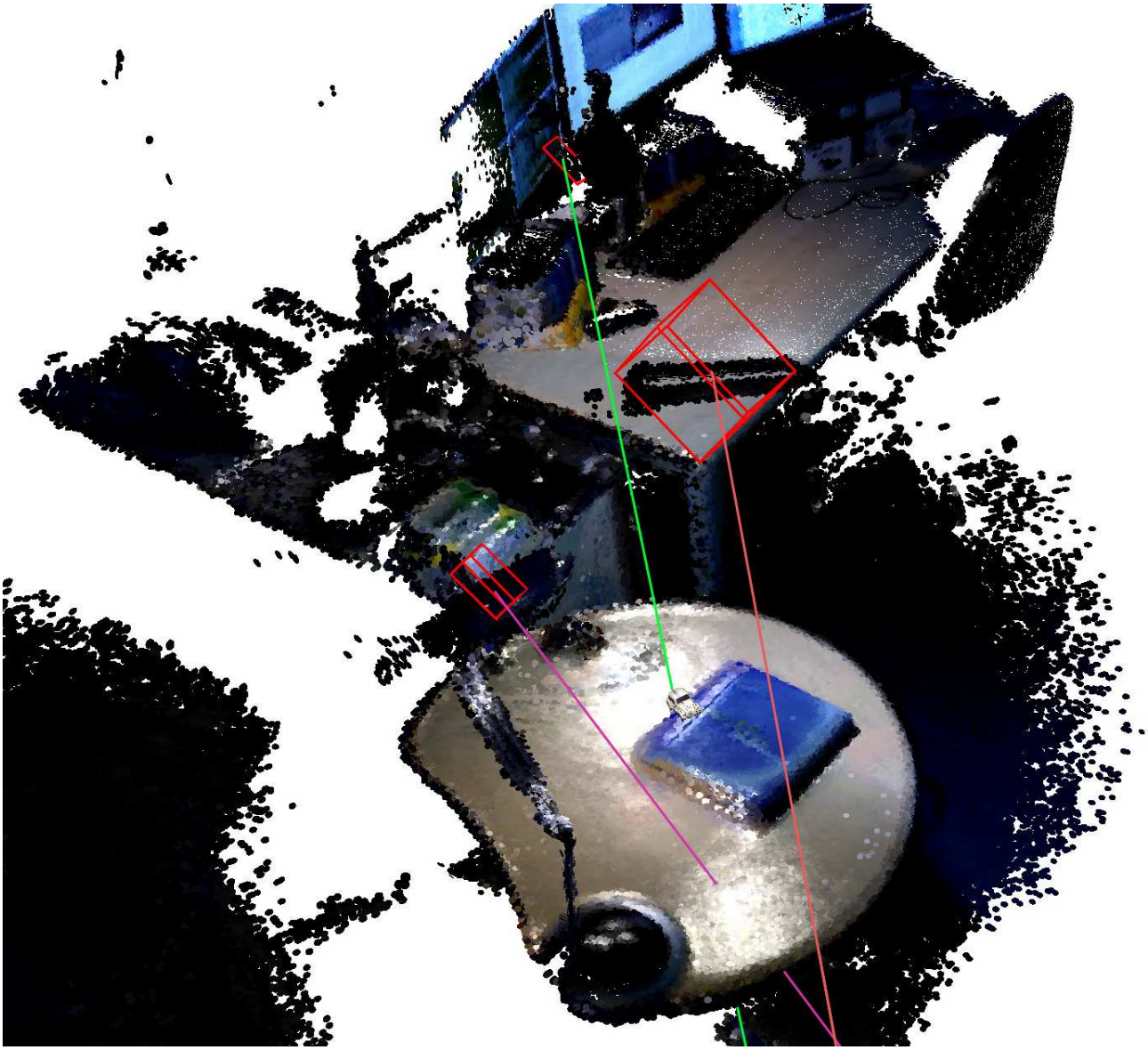


Figure 15: The reconstructed surfel map showing scene geometry and detected light sources (including estimated directionality and extent) for the AR sequence shown in Figure 14.

real-time. This secondary contribution demonstrates its usefulness both in camera tracking performance improvements and realistic AR rendering effects. In future work we wish to address the problem of map scalability beyond whole rooms, investigate the problem of dense globally consistent SLAM as $t \rightarrow \infty$ and a more complete method for modeling scene lighting configurations.

11 Acknowledgements

Research presented in this paper has been supported by Dyson Technology Ltd. The authors would like to thank Richard Newcombe for his input and suggestions.

References

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.

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

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

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.

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.

N. Fioraio, J. Taylor, A. Fitzgibbon, L. D. Stefano, and S. Izadi. Large-Scale and Drift-Free Surface Reconstruction Using Online Subvolume Registration. In *CVPR*, 2015.

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.

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`.

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.

J. Jachnik, R. A. Newcombe, and A. J. Davison. Real-Time Surface Light-field Capture for Augmentation of Planar Specular Surfaces. In *Proceedings of the International Symposium on Mixed and Augmented Reality (ISMAR)*, 2012.

O. Kahler, V. A. Prisacariu, C. Y. Ren, X. Sun, P. H. S. Torr, and D. W. Murray. Very High Frame Rate Volumetric Integration of Depth Images on Mobile Device. In *Proceedings of the International Symposium on Mixed and Augmented Reality (ISMAR)*, 2015.

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.

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.

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.

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

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.

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.

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)*, 2011a.

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)*, 2011b.

R. A. Newcombe, D. Fox, and S. M. Seitz. Dynamicfusion: Reconstruction and tracking of non-rigid scenes in real-time. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

M. Nießner, M. Zollhöfer, S. Izadi, and M. Stamminger. Real-time 3D Reconstruction at Scale using Voxel Hashing. In *Proceedings of SIGGRAPH*, 2013.

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.

F. Steinbrücker, 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.

F. Steinbrücker, 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.

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.

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.

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.

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

T. Weise, T. Wismer, B. Leibe, and L. V. Gool. In-hand scanning with online loop closure. In *Proceedings of the International Conference on Computer Vision Workshops (ICCV)*, 2009.

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.

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, 2015a.

T. Whelan, S. Leutenegger, R. F. Salas-Moreno, B. Glocker, and A. J. Davison. ElasticFusion: Dense SLAM without a pose graph. In *Proceedings of Robotics: Science and Systems (RSS)*, 2015b.

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

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