Dense Planar SLAM

Renato F. Salas-Moreno* Ben Glocker† Paul H. J. Kelly‡ Andrew J. Davison§

Imperial College London

Figure 1: Dense Planar SLAM in action. **(left)** The stairs of a house have been mapped with both planar and non-planar region surfels. **(mid-left)** Normal map shows high-quality reconstruction. Observe the lower quality normals on the highlighted red area lacking planar measurements. **(mid-right)** Planar-only regions. **(right)** Some planar regions detected on a kitchen are used to display user's content.

Abstract

Using higher-level entities during mapping has the potential to improve camera localisation performance and give substantial perception capabilities to real-time 3D SLAM systems. We present an efficient new real-time approach which densely maps an environment using bounded planes and surfels extracted from depth images (like those produced by RGB-D sensors or dense multi-view stereo reconstruction). Our method offers the every-pixel descriptive power of the latest dense SLAM approaches, but takes advantage directly of the planarity of many parts of real-world scenes via a data-driven process to directly regularize planar regions and represent their accurate extent efficiently using an occupancy approach with on-line compression. Large areas can be mapped efficiently and with useful semantic planar structure which enables intuitive and useful AR applications such as using any wall or other planar surface in a scene to display a user's content.

Index Terms: Computing methodologies [Scene understanding]; Computing methodologies [Reconstruction]. Computing methodologies [Image Processing and Computer Vision]: Segmentation. Information Systems [Information Interfaces and Presentation]: Artificial, augmented, and virtual realities.

*e-mail: r.salas-moreno10@imperial.ac.uk
†e-mail: b.glocker@imperial.ac.uk
‡e-mail: p.kelly@imperial.ac.uk
§e-mail: a.davison@imperial.ac.uk

1 Introduction

In augmented reality applications, the goals of SLAM systems which can operate in unknown environments with hand-held or wearable cameras have now clearly progressed beyond pure localisation towards capturing significant information about the scene to enable meaningful automatic annotation. We define a 'dense SLAM' system as one which produces not a point cloud but a closed surface geometry scene model, enabling every-pixel depth prediction and occlusion reasoning. A breakthrough in real-time dense SLAM was provided by the KinectFusion algorithm [17], and it was shown that in AR a dense geometric reconstruction can be annotated in any number of interactive or automatic ways [12]. However, KinectFusion and related methods [27, 19, 13] use non-parametric representations which are both heavyweight in terms of computing resources and lack semantic description.

Salas-Moreno *et al.* [21] attacked both of these weaknesses in their SLAM++ algorithm, which operates by detecting instances of known 3D objects in the live image stream from a hand-held depth camera and creating a highly efficient object-level map consisting solely of the objects' configuration. The estimated object configuration is used to generate a dense surface prediction for accurate and robust camera pose tracking using ICP as in KinectFusion; and the known objects can easily be used as the basis for content-aware AR effects. However, SLAM++ relies on a database of specific known objects only, and apart from the ability to define and use a ground plane under objects such as chairs and tables, cannot cope with non-object *regions* such as walls, which often have a large extent well beyond the field of view of a camera.

We believe that the crucial measure of the performance of a dense SLAM system is what fraction of the pixels in each new image it is able to explain with its scene model, and that this must be driven up closer to the near 100% that KinectFusion achieves for

the efficient and semantic object-based SLAM paradigm to be fully competitive. This requires both the ability to model and use the non-object-like regions of a scene; and to find and add new types of objects which are not present in the prior database. In man-made scenes, both of these requirements strongly motivate the capability to discover and model significant planar scene structures. Planes are extremely common and often occupy large fractions of the field of view of typical images from indoor scenes. Mapping them explains these pixels, potentially with great efficiency; and crucially if all planes in a scene can be mapped then the regions which are *not* planar are relatively few and are often clearly segmented against the planar surfaces which surround them.

In this work we focus on the detection and modelling of accurate, bounded planar regions with arbitrary boundary shape. These are extracted and refined over time to form a real-time dense planar SLAM system using depth images such as those produced by RGB-D sensors or via dense multi-view stereo reconstruction methods [18]. While there are many planes in most man-made scenes, mapping them is not as simple as identifying these and instantiating infinite planes in a map. Rather, accurately representing the shape and extent of planes is crucial, and for us this is what defines 'Dense Planar SLAM'. Planar regions will often have irregular boundaries, or holes (when an object is on a table top, or hangs on a wall, for instance). Starting from a surfel map generated in real-time using the point fusion method of [13], we show how planar regions at arbitrary orientations can be segmented and incrementally grown over time. We introduce an efficient representation of the accurate extent of 3D planar regions using a 2D occupancy map approach. This representation is incrementally extensible so that large planar regions can be grown, refined and joined over long observation periods. Our representation also allows on the fly compression and efficient use of memory and processing resources, and we particularly show how it is amenable to parallel GPGPU implementation.

While this work was first motivated by high level goals in object aware SLAM, we show that dense planar SLAM alone is very interesting and practical for a number of novel AR applications; in particular the use of walls or other real-world surfaces for the display of information, which will be particularly attractive when used with see-through AR headsets in the near future.

2 Related Work

Our work relates to previous real-time and off-line SLAM methods which have attempted to efficiently map scenes using planar assumptions. In the broader context, it relates to the literature on augmenting 3D reconstructions with semantic meaning. Our approach, benefits from the simplification, efficiency and predictive power of semantic model-fitting *in the loop* of real-time operation, a principle that also underpins Salas-Moreno *et al.*'s SLAM++ [21], which recognises instances of objects from a pre-scanned library and directly builds a map at the object level. This is in contrast to numerous approaches which consider reconstruction and semantic labelling as processes to be applied one after the other (e.g. the sophisticated work of Kim *et al.* [14]).Other work which does jointly perform reconstruction and object fitting, such as that by Bao *et al.* [3], is far from being feasible in real-time operation, unlike [21].

Earlier approaches for real-time SLAM using planes include work from Gee *et al.* [9] and Chekhlov *et al.* [6]. Their systems used planes to replace point features and reduce the state space of estimation, because having large number of points becomes unbearable in Kalman filter-based systems. This was improved by Carranza and Calway [16] who directly mapped using planes and points without initialisation delay.

Dou *et al.* [8] improved indoor 3D reconstruction quality via bundle adjustment method that incorporated planar surface alignment errors in addition to 3D point reprojection errors. Their system however was not aimed at real-time scenarios, taking 3 seconds for plane extraction and a few minutes for global optimisation.

Trevor *et al.* [26] combined a Kinect sensor with a 2D laser range scanner to map both close and distant line and plane features. Taguchi *et al.* [23] performed camera tracking by detecting point and plane features and matching them in the complete global map. However this approach resulted in perceptual aliasing and slow tracking. Most recently Ataer-Cansizoglu *et al.* [2] improves on Taguchi's method by predicting the camera motion via a constant velocity model using optical flow, a condition which is difficult to satisfy with handheld camera scenarios. Our method imposes no such assumption and instead directly tracks via ICP alignment from a dense model, giving a fine pose update that eases the data-association task.

3 System Overview

A schematic overview of our system is shown in Figure 2. Our starting point is the Point-based Fusion method of Keller *et al.* [13] to densely map the environment with surfels: small disk-shaped entities to describe locally planar regions without connectivity information. Mapping from noisy depth sensors using surfels provides easier management of data-association, insertion, averaging and removal of map entities compared to structured meshes like triangles as well as memory savings compared to voxel-based methods like KinectFusion [17].

In our approach, we aim to label each surfel in the 3D map either with one of a number of discrete plane labels, or to leave it with no label if it is not part of any major plane in the scene. *Planar region surfels* describe large areas with little or no curvature and therefore share common properties (normals and closest distance to the common plane) and are managed together to enforce this. *Non-planar region surfels* on the other hand are located in areas of high curvature and are managed as in the original method of Keller *et al.* [13].

The dense and incremental mapping nature of our method enables easy data-association of modelled and measured planes across consecutive frames: After the current pose of the camera is estimated, plane association is simply done by counting pixel-level matches of projected modelled planes into the currently measured depth image and handling mismatches in a logical manner.

Data-associated planes are then converted into the same world reference frame and refined with a running average. All modelled surfels belonging to the same plane are enforced to share the same refined normals and closest distance, unlike non-planar region surfels which average in isolation.

Finally, two or more overlapping modelled planes with similar properties are merged together to incrementally extend areas that initially fail to connect due to noise or occlusion.

While a particular planar region is still densely represented by many surfels, it is worth mentioning that surfels' position and orientation are controlled by the same set of plane parameters. This may seem redundant and memory inefficient at first but allows us to densely represent complex planes with holes in the middle [1] that would otherwise be challenging to model and render incrementally using closed polygons with hulls (e.g. as done in [26, 23, 2]). Section 5 demonstrates how we compress planes with a 9-to-1 ratio to achieve lightweight maps particularly of indoor environments composed of several planes.

3.1 Preliminaries

As in [13], we represent the SLAM map with a set of k unstructured surfels $\bar{\mathbf{P}}_k$ with properties such as position $\bar{\mathbf{v}}_k \in \mathbb{R}^3$, normal $\bar{\mathbf{n}}_k \in \mathbb{R}^3$, radius $\bar{r}_k \in \mathbb{R}$, confidence $\bar{c}_k \in \mathbb{R}$, and timestamp[2] $\bar{t}_k \in \mathbb{N}$.

[1] see for example the plane hole on the washing machine door in Figure 8.

[2] timestamp is the moment when a measurement is taken such as a simple frame counter.

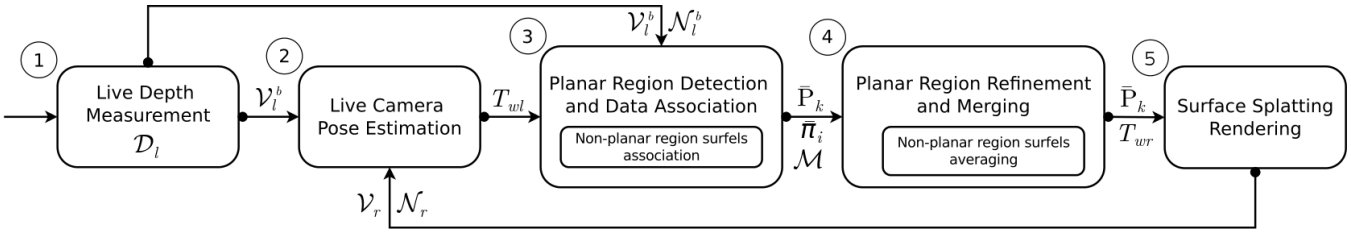

Figure 2: Outline of the Dense Planar SLAM pipeline. **(1)** A bilateral filtered depth measurement $\mathcal{D}_l$ is transformed into a metric vertex $\mathcal{V}_l^b$ and normal $\mathcal{N}_l^b$ map and used for both camera pose estimation and plane detection. **(2)** We update the live camera pose T_{wl} by densely aligning with ICP the measured vertex map $\mathcal{V}_l^b$ against the predicted vertex $\mathcal{V}_r$ and normal map $\mathcal{N}_r$. **(3)** Planes are detected via connected component labelling and incrementally extended with projective data-association. **(4)** Modelled planes $\bar{\pi}_i$ are merged and refined with a running average. **(5)** View prediction is generated by rendering surfels $\bar{\mathbf{P}}_k$ via surface-splatting using the reference pose T_{wr}.

Additionally we include a plane ID $\bar{o}_k = i; i = 1, ..., p \in \mathbb{N}$ with $\bar{o}_k = 0$ signalling non-planar region surfels.

A surfel radius is representative of the local surface area with its value chosen to minimise holes between neighbouring surfels; we compute them as: $\bar{r}_k = \sqrt{2}\bar{\mathbf{v}}_{k(z)}/f$, with f the camera focal length.

A live depth measurement $\mathcal{D}_l$ is transformed into a metric vertex map $\mathcal{V}_l(\mathbf{u}) = \mathbf{K}^{-1}\dot{\mathbf{u}}\mathcal{D}_l(\mathbf{u})$, with $\mathbf{K}$ the camera intrinsic matrix, $\mathbf{u} = (x, y)^\top$ a pixel position in the image domain $\mathbf{u} \in \Omega \subset \mathbb{R}^2$ and $\dot{\mathbf{u}}$ its homogeneous representation. The live normal map $\mathcal{N}_l$ is simply generated from the vertex map by central differences.

Additionally, we apply a bilateral filter [24] to $\mathcal{D}_l$ generating discontinuity preserved vertex $\mathcal{V}_l^b$ and normal $\mathcal{N}_l^b$ map with reduced noise.

To update the live camera to world transform $T_{wl} = [\mathbf{R}, \mathbf{t}] \in \mathbb{SE}(3)$, $\mathbf{R} \in \mathbb{SO}(3)$, $\mathbf{t} \in \mathbb{R}^3$; a pair of vertex map $\mathcal{V}_r$ and normal map $\mathcal{N}_r$ is rendered via surface-splatting using the previously reference pose $T_{wr} \in \mathbb{SE}(3)$ that is incrementally aligned to $\mathcal{V}_l^b$ using dense ICP [17] with a point-plane error metric [20]. This produces a series of incremental updates $\{\tilde{T}_{rl}^n\}_{n=1}^m$ composed together to generate $T_{wl} \leftarrow T_{wr}\tilde{T}_{rl}^m$.

3.2 Relocalisation

In line with the dense nature of our system, we avoid extracting features to match (such as SIFT or SURF) to retrieve known places and continue tracking when this is lost. Instead we perform whole image encoding of keyframes using Ferns, following the method of Glocker *et al.* [10]. This enables the fast retrieval of near poses when tracking is lost that are later refined with ICP on the dense model.

4 Mapping with Planes

Assuming an updated live camera pose T_{wl}, mapping a scene consists of integrating new measurements into the global model. To do so, a $4x$ super-resolution index map $\mathcal{I}^s$ is created by recording the point index k projected into the live frame at pixel $\mathbf{u}^s = \pi(\mathbf{K}^s T_{wl}^{-1}\bar{\mathbf{v}}_k)$ via standard pin-hole projection function π. Modelled surfels can now be associated with measurements according to sensor uncertainty, normals agreement, confidence value and distance to viewing ray [13], producing data-associated pairs $a_{surfels} = \{(i, j)\}; i = 1, ..., k; j = 1, ..., w \times h$.

4.1 Planar Region Detection

Similar to Trevor *et al.* [25] we detect planes using connected component labelling [7]. This produces a label map $\mathcal{L}(\mathbf{u}) = i; i = 1, ..., q \in \mathbb{N}$ identifying to which of the q measured planes each pixel belongs to. $\mathcal{L}(\mathbf{u}_i) = 0$ is reserved for regions with few connecting components or high curvature (see Figure 3 **left**).

Planes are parametrised by $\pi = (n_x, n_y, n_z, d)^\top$ with $\mathbf{n}_\pi = (n_x, n_y, n_z)^\top$ the plane normal and d the closest distance to the common plane.

Detection proceeds by first computing a similarity map efficiently using the GPU. This generates for every pixel a bitmask indicating if the pixel above and to the left of the current one have similar plane distance and normal:

$$Mask(\mathbf{u}) = (S(\mathbf{u}, \mathbf{up}) \ll 1) \mid S(\mathbf{u}, \mathbf{left}) \tag{1}$$

$$S(\mathbf{x}, \mathbf{y}) = \begin{cases} 1 & \text{if } \|\mathcal{V}_l^b(\mathbf{x}) \cdot \mathcal{N}_l^b(\mathbf{x}) - \mathcal{V}_l^b(\mathbf{y}) \cdot \mathcal{N}_l^b(\mathbf{y})\| \\ & < \delta_1(\mathcal{V}_l^b(\mathbf{x})_{(z)})^2 \text{ and } \mathcal{N}_l^b(\mathbf{x}) \cdot \mathcal{N}_l^b(\mathbf{y}) > cos(\Theta_1) \\ 0 & \text{otherwise} \end{cases}$$

$$\mathbf{up} = \mathbf{u} - (0, 1)^\top, \ \mathbf{left} = \mathbf{u} - (1, 0)^\top$$

The process continues on the CPU by assigning unique labels to similar and contiguous pixels followed by a union-find algorithm to merge equivalent labels. To prevent merging regions with low-quality normals, the process is avoided at depth-discontinuity boundaries (see Figure 3 **right**), i.e. zeros in the map:

$$Disc(\mathbf{u}) = \prod_{\mathbf{u}_i \in \omega} S(\mathbf{u}, \mathbf{u}_i) \tag{2}$$

with ω a window with radius 3 centered in $\mathbf{u}$.

Following [25], we discard regions with few connected pixels ($<$ min-inliers). From the rest we fit planes by performing Principal Component Analysis (PCA): first the vertices are normalised by subtracting its mean $\hat{\mathbf{v}}$, followed by the computation of a covariance matrix $\mathbf{\Sigma}$ and its corresponding eigendecomposition. The eigenvector with minimum eigenvalue λ_{min} becomes the plane normal $\mathbf{n}_\pi$. The plane distance is computed as: $d = -\mathbf{n}_\pi \cdot \hat{\mathbf{v}}$ while the plane curvature is: $\kappa = \frac{\lambda_{min}}{\sum_{i=1}^3 \lambda_i}$. We discard planes having curvature $\kappa >$ max-curvature.

4.2 Data-Association with Planes

A SLAM system should be able to incrementally expand its map during exploration. To enable this in our system we need to expand modelled planes as the camera browses a new scene.

Once the current sensor pose is estimated, modelled planar and non-planar region surfels are projected into the current live frame. For each pixel $\mathbf{u}$, one of several data-association cases may occur (see Figure 4):

(A) A modelled plane and a measured plane intersect. This situation indicates a data-association between the modelled and measured planes, producing pairs $a_{planes} = \{(i, j)\}; i = 1, ..., p; j = 1, ..., q$.

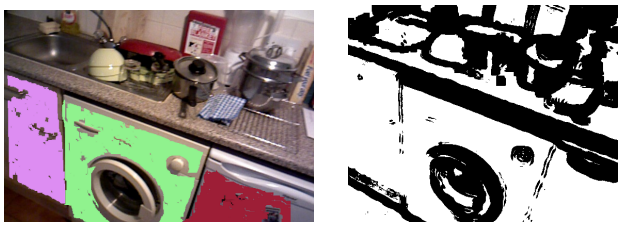

Figure 3: Planar region detection. **(left)** Three planes have been detected with connected component labelling using the measured vertex and normal maps. **(right)** A depth discontinuity map is used to avoid labelling regions with low quality normals (black pixels).

(B) Modelled surfels lack a planar measurement. This indicates a non-planar region due to high-curvature or noisy depth measurements preventing planes to be fitted.

(C, D) Modelled planar region surfels lack planar measurement. Due to noise, not every plane is expected to be detected on measured depth maps.

(E, F) Unmodelled (measured) planar regions. This happens when a new plane is detected on the live depth map and is yet to be incorporated into the SLAM map.

(G) Invalid data. Occurs at pixels where the live depth data from the sensor is invalid (has holes).

Cases C and D as well as E and F have to be disambiguated. To do this we first identify the intersecting pixels (Case A) giving them a unique ID that is flooded into the regions C and E, thus expanding the modelled plane of regions C and A towards region E. At this point we are left with only cases D and F which are purely modelled or unmodelled (measured) respectively.

This disambiguation is efficiently performed in practice via parallel operations evaluated on the GPU [1, 5]. First the set of pixels associations a_{planes} corresponding to Case A are transformed into a hash value $h = j\rho + i$ (with ρ a constant to ensure uniqueness) and sorted in parallel to group pixels belonging to the same modelled plane first and same measured plane second. This is followed by a parallel reduction operation using the hash values as keys and a constant 1 (one) as value, giving a list of possible associations and number of pixels supporting this: $\{(j, i, count)\}$.

A measured plane $\bar{\pi}_j$ could be associated to more than one modelled plane $\bar{\pi}_i$ due to noise or occlusion. To prevent wrong associations to the measured plane we traverse the previous list looking for a modelled plane with similar coefficients: $\|d'_j - d_i\| < \delta_2$, $\mathbf{n}'_j \cdot \mathbf{n}_i > cos(\Theta_2)$. Similar modelled planes IDs are added into a merge map $\mathcal{M}$ and the one with maximum count $count^\star >=$ min-assoc-count is chosen as the final associated modelled plane and used as key for $\mathcal{M}$.

4.3 Planar Region Refinement and Merging

A modelled plane $\bar{\pi}_i$ is refined with the associated measured plane π_j using a simple running average. First the measured plane coefficients are transformed into the world reference frame:

$$\mathbf{n}'_j = \mathbf{R}\mathbf{n}_j \tag{3}$$

$$d'_j = -\mathbf{n}'_j \cdot \mathbf{t} + d_j \tag{4}$$

The modelled plane coefficients are then refined with:

$$\mathbf{n}_i \leftarrow \frac{w\mathbf{n}_i + \mathbf{n}'_j}{w+1},\ d_i \leftarrow \frac{wd_i + d'_j}{w+1},\ w \leftarrow w+1 \tag{5}$$

We traverse the merge map $\mathcal{M}$ and for each entry we rename the contained plane IDs with that of the corresponding key.

Finally all surfels belonging to the measured plane π_j are projected onto the refined modelled plane $\bar{\pi}_i$.

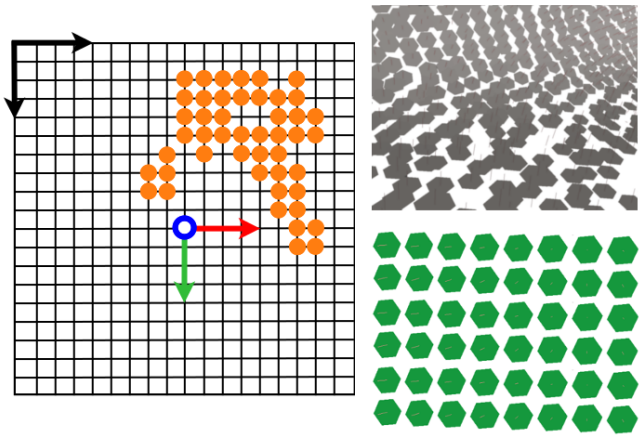

Figure 5: (**left**)Virtual Image with on-pixels surfels (orange) representing planar region coverage. The coloured coordinate frame are the eigenvectors centred at the plane centroid, while the black coordinate frame is the virtual image origin. (**top-right**) Non-planar region surfels around a region of high curvature need to explicitly represent individual position and orientation information. (**bottom-right**) Planar region surfels are evenly organised and share common properties. ***Note***: Surfel radius size reduced for easier visualisation

4.4 Non-Planar region surfels mapping

Surfels not having planar region measurements have their associated properties updated with a running average α weighted for radial noise as in [13]. This is reproduced here for easier reading:

$$\bar{\mathbf{v}}_k \leftarrow \frac{\bar{c}_k\bar{\mathbf{v}}_k + \alpha T_{wl}\mathbf{v}_l}{\bar{c}_k + \alpha},\ \bar{\mathbf{n}}_k \leftarrow \frac{\bar{c}_k\bar{\mathbf{n}}_k + \alpha\mathbf{R}\mathbf{n}_l}{\bar{c}_k + \alpha} \tag{6}$$

$$\bar{r}_k \leftarrow \frac{\bar{c}_k\bar{r}_k + \alpha r_l}{\bar{c}_k + \alpha},\ \bar{c}_k \leftarrow \bar{c}_k + \alpha,\ \bar{t}_k \leftarrow t \tag{7}$$

5 Map Compression

Planar region surfels need to densely populate their area of coverage, however many of their properties are shared between them (normals, radius size, confidence, timestamp and plane ID). Furthermore their planar position requires a two-dimensional representation only.

We will compress planar regions whenever they become non-visible (i.e. outside the view frustum). First we execute frustum culling by intersecting the plane bounding box with each of the 6 planes enclosing the frustum. Only if the planar region does not intersect all of them we can safely compress the plane, move its data down the memory hierarchy (i.e. from GPU memory to RAM or disk) and reclaim the working GPU memory. This form of occlusion culling also helps to maintain an almost consistent frame rate independent of map size, while reducing the need for additional space partitioning techniques like octrees.

Compression begins by performing an additional PCA step in order to estimate the major x-y axis of the extended plane. As in the plane fitting procedure, we first normalise the vertices by subtracting its mean $\hat{\mathbf{v}}$, followed by calculation of the covariance matrix $\mathbf{\Sigma}$ and corresponding eigendecomposition to obtain the x-y axis.

We can think of the plane compression mechanism as a way of representing the plane as a binary image: with on-pixels signalling the areas of coverage and off-pixels for holes (see Figure 5). To do so we will convert vertex coordinates into pixel locations in a *Virtual Image* of dimensions: $w_{vi} \times h_{vi}$. In practice both dimensions are set to 65536, allowing us to represent planes with dimensions in the range $(-32.768m, -32.768m)$ to $(32.767m, 32.767m)$ at

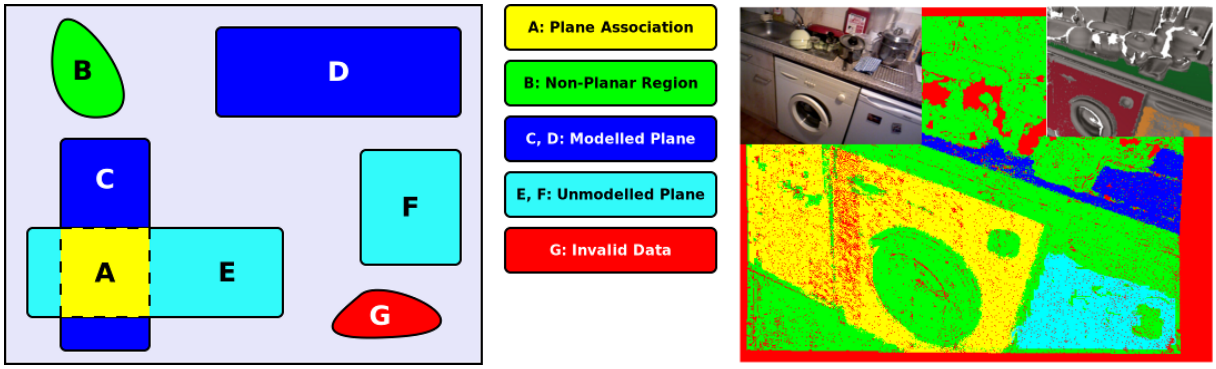

Figure 4: Data-association cases. **(left)** The diagram shows all the possible pixel-wise association cases when projecting the SLAM map into the measured live frame. **(right)** Colour-coded visualisation of pixel-wise association cases. The top-left inset shows the rgb data while the top-right shows the densely reconstructed planar and non-planar region surfels.

millimetre accuracy. The fixed virtual image dimensions allow us to further linearise the coordinates in 1D. Computing the compressed index of a surfel is detailed below:

$$\mathbf{v}_c = \mathbf{v} - \hat{\mathbf{v}} \tag{8}$$

$$\mathbf{v}_p = (\mathbf{x}_{axis} \cdot \mathbf{v}_c, \mathbf{y}_{axis} \cdot \mathbf{v}_c)^\top \tag{9}$$

$$\mathbf{v}_{vi} = round(\mathbf{v}_p \times 1000) \tag{10}$$

$$\mathbf{v}_o = \mathbf{v}_{vi} + (w_{vi}, h_{vi})^\top / 2 \tag{11}$$

$$index = \mathbf{v}_{o(y)} \times w_{vi} + \mathbf{v}_{o(x)} \tag{12}$$

$\mathbf{v}_c$ is the normalised (centred) vertex, $\mathbf{v}_p$ is the vertex on the plane after projecting the $\mathbf{v}_c$ coordinates with the plane axis, $\mathbf{v}_{vi}$ is the vertex on the virtual image with integer units pre-scaled to preserve millimetre accuracy, $\mathbf{v}_o$ is the vertex on the top-left reference frame, finally $index$ is the linearised $\mathbf{v}_o$ coordinates in 1D.

In this way planar region surfels are compressed with an approximate ratio of 9-to-1 as we only need to store their indices with 4 bytes per surfel[3]. In contrast, non-planar region surfels require 36 bytes each: vertices (3 floats), normals (2 floats), radius (1 float), confidence (1 float), timestamp (1 uint), plane ID (1 uint).

Scenes composed of planar and non-planar region surfels produces combined compression ratios of about 2.27. Some compression results on real and synthetic data are shown on the chart in Figure 6.

Further compression ratios could be achieved for example by performing run-length encoding of on-pixels but this is not explored yet.

Decompression is trivially achieved by performing the inverse compression steps (12 - 8) and in practice both tasks take less than 2ms, allowing online operation.

6 Results

Quantitative experiments using synthetic scenes were performed to measure the quality of tracking with and without planar mapping. Qualitative results of the enhanced planar representation are shown for real scenes obtained with the Asus Xtion Pro RGB-D sensor.

The plane detection parameters were set to min-inliers $= 1000$, $\delta_1 = 0.01m$, $\delta_2 = 0.2m$, $\Theta_1 = \Theta_2 = 20°$, max-curvature $= 0.00015$, and min-assoc-count $= 100$.

The noise characteristics of the depth sensor make plane detection only usable in the close range (< 4m). Our thresholds discourage false-positives by discarding planes with large-curvature

[3] planar regions surfels share the same normal, radius, confidence, timestamp and plane ID.

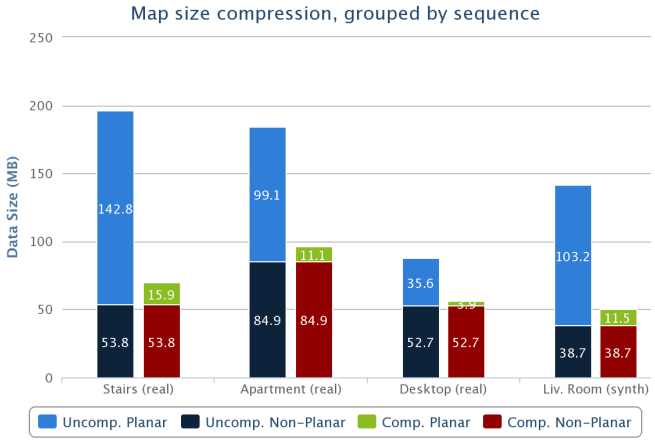

Figure 6: Chart showing the map data size of planar and non-planar region surfels with and without compression.

(as described in Section 4.1). Nevertheless, the fact that mapping and tracking are still possible in the absence of plane detection reinforces the integrated planar/non-planar approach presented.

6.1 Synthetic scenes

We evaluate our system on synthetic scenes with groundtruth camera poses for two trajectories produced by Handa *et al.* [11]. Depth maps are further corrupted by the noise model proposed by Barron and Malik [4] to create data closer to the Kinect sensor. Reconstructions results of the 'living room' sequence are shown on Figure 7.

Table 1 summarises the Absolute Trajectory Error (ATE) as proposed by Sturm *et al.* [22]. ATE computes the absolute difference between the groundtruth and estimated poses after alignment.

Although tracking from a globally consistent dense model is shown to be already of high quality [17] we can see from the RMSE values that using planar regions surfels decreases the trajectory error slightly, this is because the scene contains a large number of planar regions affected by noise and alleviated earlier by our method (compared to KinectFusion [17] or Point-based Fusion [13] that require a few frames to denoise).

While the RMSE of *trajectory-0* is large compared to *trajectory-1* it is worth highlighting that *trajectory-0* is challenging when tracking with ICP since the camera cannot be locked based on 2

planes only (wall and ceiling)[4] and therefore large drift occurs. This artefact was also reported in [11] section VI-D-1.

Table 1: Absolute Trajectory Error (ATE) in synthetic scene

Error	trajectory-0		trajectory-1	
	non-planar	planar	non-planar	planar
RMSE	0.254134	**0.246437**	0.018997	**0.016940**
Mean	0.222794	0.218559	0.016906	0.015043
Median	0.179679	0.182547	0.014714	0.016449
Std	0.122258	0.113857	0.008666	0.007789
Min	0.055287	0.070420	0.003252	0.002228
Max	0.728229	0.645284	0.032742	0.028430

6.2 Real-world scenes

Examples of real-world scenes are shown in Figure 1 and 8. Here the stairs of a house and an apartment have been reconstructed and major planes parametrised incrementally and in real-time. The first case could be particularly useful for stair-climbing robots navigating new environments.

6.3 Augmented Reality with dense planar maps

We can take advantage of the dense and real-time nature of our system to perform novel Augmented Reality (AR) interactions with fine occlusion handling, requiring very little user input [5].

As a first example, we let the user choose a set of planes to augment the original input with an application display using the Oculus Rift paired with an Xtion sensor (see Figure 9 and the rightmost image on Figure 1), essentially converting the real-world into a window manager. To enable this, we first extract the bounding-box of the selected plane(s) and convert it into a quad polygon for efficient texture mapping. This feature could also be very useful in see-through AR headsets as it can replace small floating widgets with large projections on planes without interfering with the wearer's field of view (e.g. a limiting factor in the current Google Glass).

Another useful example is virtually replacing the floor style of a house many times until the user is satisfied with the result (see Figure 10).

The idea of overlaying information on real planes instead of floating windows allows the user to safely navigate environments without fear of collisions and make tasks like zooming in/out as natural as walking closer/further from surfaces.

6.4 System Statistics

Table 2 summarises the results when mapping the stairs sequence shown in Figure 1. This was executed on a high-performance laptop equipped with an Intel i7 Quad Core CPU at 2.50GHz and NVidia GTX 580M GPU with 2GB of memory.

While the overhead of plane detection, data association and compression are not present in systems like [13], we note that further optimisation can be easily engineered to increase frame rate such as moving mapping to a lower priority thread as in Klein *et al.* [15].

7 Conclusions and Future Work

We have presented a Dense Planar SLAM algorithm which identifies, merges and compresses the arbitrary planar regions which are present in many man-made scenes, and leads to an efficient, robust and real-time plane-aware SLAM system.

In addition, we have shown the highly practical AR applications this permits, in particular the use of planar regions for the display

[4] as can be seen on minute 0:32 at: http://youtu.be/4O-OaV0h4AQ

[5] The supplementary video contains demos of the described AR use cases.

Figure 9: Facebook Wall on a real wall using the Oculus Rift. The user chooses a wall from which to read his Facebook Wall.

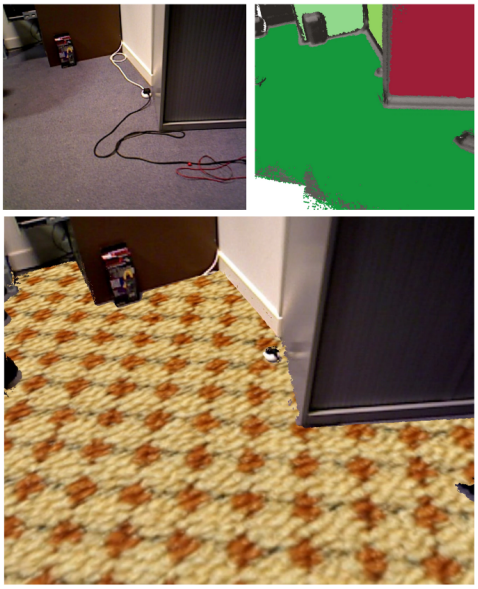

Figure 10: Floor carpet change. The ground plane is selected and overlaid with a new carpet.

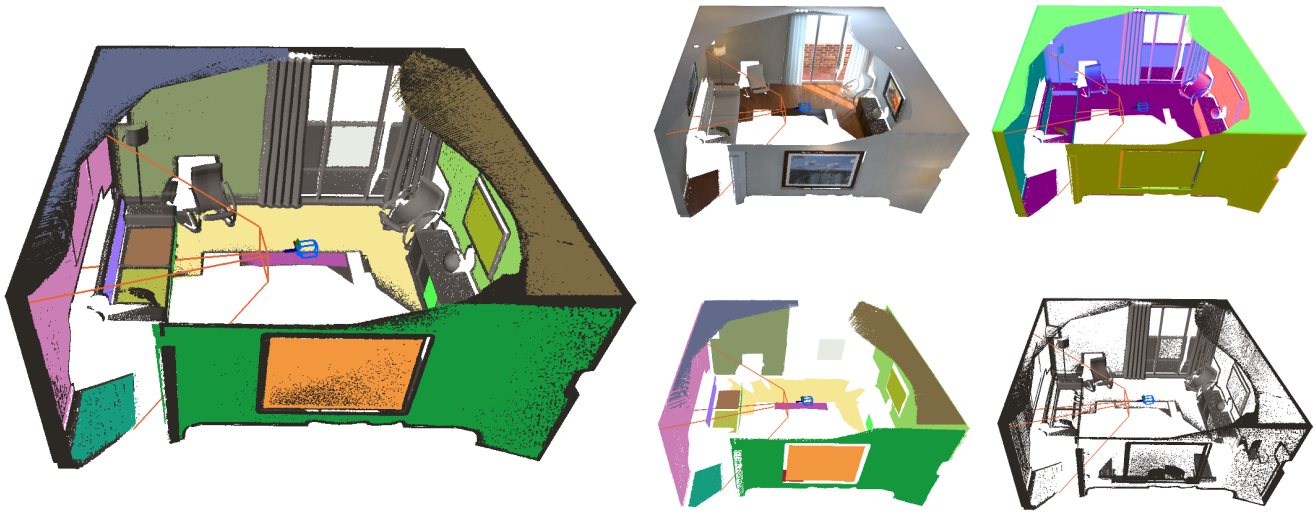

Figure 7: Synthetic scene reconstruction of a living room. **(left)** Displaying both planar and non-planar regions surfels. **(right)** In clockwise order: Colour output, normal map, non-planar region surfels only, planar region surfels only.

Table 2: System statistics for the stairs sequence

Memory usage	
Non-Planar region surfels count	1'566,063
Planar region surfels count	4'159,902
Plane Count	30
Point-Based fusion Memory [13]	196.58 MB
Dense Planar SLAM Memory (this work)	69.64 MB
Compression Ratio	2.82
Timings	
Frame Prediction	11.8 ms
ICP	6.08ms
Plane Detection	10.1 ms
Data Association	24.02 ms
Averaging	2.58 ms
Surfel Addition/Removal	9.4 ms
Compression and Decompression	1.8 ms
Total time	65.98 ms
Frame Rate	15.16 fps

of information in a very natural manner which will fit well with see-through head mounted displays.

We remain interested in our long-term goals of a fully object-based SLAM system within which this planar mapping will form a crucial component.

Adding a graph-based loop closure optimisation for consistency over long loopy trajectories is a clear short-term goal, and we hope that standard methods as used in [21] will be suitable here. Though careful thought has to be given to how non-planar regions of surfels should be treated when the locations of planes is optimised due to loop closure constraints; presumably each small blob of non-planar surfels should be attached and adjusted rigidly with the same transformation of one or more of its neighbouring planar regions rather than being broken up or sheared.

Also, we will continue to work on using these non-planar regions to extend a library of object types; the main challenge here is rapid learning of efficient detectors for these new object types during real-time operation when the resources for extravagant training are not available.

References

[1] Advanced Micro Devices Inc. Bolt C++ Template Library. 2012.

[2] E. Ataer-Cansizoglu, Y. Taguchi, S. Ramalingam, and T. Garaas. Tracking an RGB-D Camera Using Points and Planes. In *ICCV CDC4CV Workshop*, 2013.

[3] S. Y. Bao, M. Bagra, Y.-W. Chao, and S. Savarese. Semantic Structure From Motion with Points, Regions, and Objects. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[4] J. Barron and J. Malik. Intrinsic Scene Properties from a Single RGB-D Image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013.

[5] N. Bell and J. Hoberock. Thrust: C++ Template Library for CUDA. 2009.

[6] D. Chekhlov, A. Gee, A. Calway, and W. Mayol-Cuevas. Ninja on a Plane: Automatic Discovery of Physical Planes for Augmented Reality Using Visual SLAM. In *Proceedings of the International Symposium on Mixed and Augmented Reality (ISMAR)*, 2007.

[7] M. B. Dillencourt, H. Samet, and M. Tamminen. A General Approach to Connected-component Labeling for Arbitrary Image Representations. *Journal of the ACM*, 39(2):253–280, Apr. 1992.

[8] M. Dou, L. Guan, J. Frahm, and H. Fuchs. Exploring High-Level Plane Primitives for Indoor 3D Reconstruction with a Hand-held RGB-D Camera. In *ACCV Workshop on Color Depth Fusion, in conjunction with 11th Asian Conference on Computer Vision (ACCV)*, 2012.

[9] A. Gee, D. Chekhlov, W. Mayol, and A. Calway. Discovering planes and collapsing the state space in visual slam. In *Proceedings of the British Machine Vision Conference (BMVC)*, 2007.

[10] B. Glocker, S. Izadi, J. Shotton, and A. Criminisi. Real-Time RGB-D Camera Relocalization. In *International Symposium on Mixed and Augmented Reality (ISMAR)*, 2013.

[11] A. Handa, T. Whelan, J. 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.

[12] S. Izadi, D. Kim, O. Hilliges, D. Molyneaux, R. A. Newcombe, P. Kohli, J. Shotton, S. Hodges, D. Freeman, A. J. Davison, and A. Fitzgibbon. KinectFusion: Real-Time 3D Reconstruction and Interaction Using a Moving Depth Camera. In *Proceedings of ACM Symposium on User Interface Software and Technolog (UIST)*, 2011.

[13] M. Keller, D. Lefloch, M. Lambers, S. Izadi, T. Weyrich, and A. Kolb. Real-time 3D Reconstruction in Dynamic Scenes using Point-based

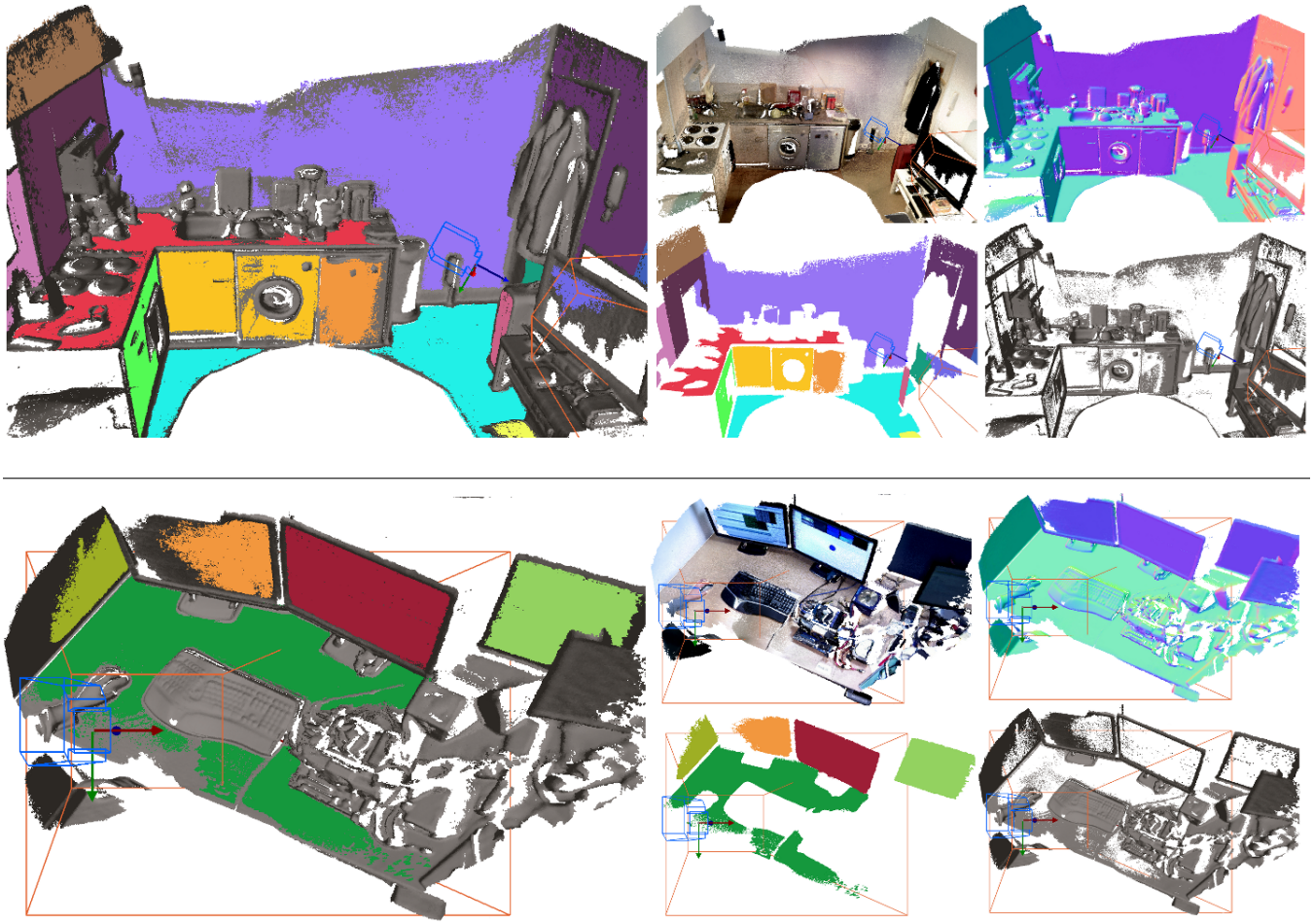

Figure 8: Real scene reconstruction of an apartment **(top)** and desktop **(bottom)**. **(left)** Displaying both planar and non-planar regions surfels. **(right)** In clockwise order: Colour output, Normal Map, Non-Planar region surfels only, Planar region surfels only.

Fusion. In *Proc. of Joint 3DIM/3DPVT Conference (3DV)*, June 2013.

[14] Y. M. Kim, N. J. Mitra, D.-M. Yan, and L. Guibas. Acquiring 3D Indoor Environments with Variability and Repetition. In *SIGGRAPH Asia*, 2012.

[15] 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.

[16] J. Martinez-Carranza and A. Calway. Unifying planar and point mapping in monocular slam. In *BMVC*, 2010.

[17] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohli, J. Shotton, S. Hodges, and A. Fitzgibbon. KinectFusion: Real-Time Dense Surface Mapping and Tracking. In *Proceedings of the International Symposium on Mixed and Augmented Reality (ISMAR)*, 2011.

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

[19] H. Roth and M. Vona. Moving Volume KinectFusion. In *Proceedings of the British Machine Vision Conference (BMVC)*, 2012.

[20] S. Rusinkiewicz and M. Levoy. Efficient Variants of the ICP Algorithm. In *Proceedings of the IEEE International Workshop on 3D Digital Imaging and Modeling (3DIM)*, 2001.

[21] R. F. Salas-Moreno, R. A. Newcombe, H. Strasdat, P. H. J. Kelly, and A. J. Davison. SLAM++: Simultaneous Localisation and Mapping at the Level of Objects. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013.

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

[23] Y. Taguchi, Y. Jian, S. Ramalingam, and C. Feng. Point-Plane SLAM for Hand-Held 3D Sensors. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2013.

[24] C. Tomasi and R. Manduchi. Bilateral Filtering for Gray and Color Images. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 1998.

[25] A. Trevor, S. Gedikli, R. Rusu, and H. Christensen. Efficient Organized Point Cloud Segmentation with Connected Components. In *3rd Workshop on Semantic Perception Mapping and Exploration (SPME)*, 2013.

[26] A. Trevor, J. Rogers III, and H. Christensen. Planar Surface SLAM with 3D and 2D Sensors. 2012.

[27] T. Whelan, J. 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.