Kintinuous: Spatially Extended KinectFusion

Kintinuous: Spatially Extended KinectFusion

Thomas Whelan, John McDonald
Department of Computer Science, National of Ireland Maynooth, Co. Kildare, Ireland.
Email: Thomas.J.Whelan@nuim.ie

Michael Kaess, Maurice Fallon, Hordur Johannsson, John J. Leonard
Computer Science and Artificial Intelligence Laboratory (CSAIL),
Massachusetts Institute of Technology (MIT), Cambridge, MA 02139, USA.

***Abstract*—In this paper we present an extension to the KinectFusion algorithm that permits dense mesh-based mapping of extended scale environments in real-time. This is achieved through (i) altering the original algorithm such that the region of space being mapped by the KinectFusion algorithm can vary dynamically, (ii) extracting a dense point cloud from the regions that leave the KinectFusion volume due to this variation, and, (iii) incrementally adding the resulting points to a triangular mesh representation of the environment. The system is implemented as a set of hierarchical multi-threaded components which are capable of operating in real-time. The architecture facilitates the creation and integration of new modules with minimal impact on the performance on the dense volume tracking and surface reconstruction modules. We provide experimental results demonstrating the system's ability to map areas considerably beyond the scale of the original KinectFusion algorithm including a two story apartment and an extended sequence taken from a car at night. In order to overcome failure of the iterative closest point (ICP) based odometry in areas of low geometric features we have evaluated the Fast Odometry from Vision (FOVIS) system as an alternative. We provide a comparison between the two approaches where we show a trade off between the reduced drift of the visual odometry approach and the higher local mesh quality of the ICP-based approach. Finally we present ongoing work on incorporating full simultaneous localisation and mapping (SLAM) pose-graph optimisation.**

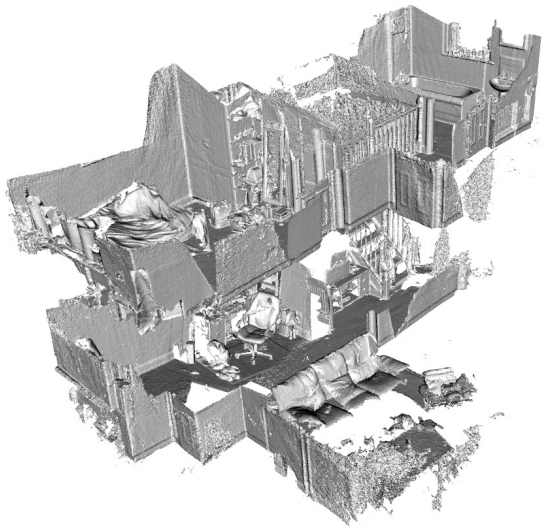

Fig. 1. Real-time 6DOF extended scale map reconstruction of a dataset captured using a handheld Kinect traversing multiple rooms over two floors of an apartment. (see Section V-B)

I. Introduction

In recent years visual SLAM has reached a significant level of maturity with a number of robust real-time solutions being reported in the literature [9]. Although these techniques permit the construction of an accurate map of an environment, the fact that they are feature-based means that they result in sparse point cloud maps that cannot be used directly or have limited utility in many robotic tasks (*e.g.* obstacle avoidance, path planning, manipulation, *etc.*). This issue has motivated the development of dense mapping approaches that aim to use information from every pixel from the input video frames to create 3D surface models of the environment [12, 15]. The emergence of RGB-D cameras, and in particular the Microsoft Kinect®, has seen this work being taken a step further. Newcombe *et al.* introduced the KinectFusion algorithm [11] which uses a volumetric representation of the scene, known as the truncated signed distance function (TSDF), in conjunction with fast iterative closest point (ICP) pose estimation to provide a real-time fused dense model of the scene at an unprecedented level of accuracy.

However this algorithm does suffer from a number of limitations in part derived from the chosen underlying TSDF voxel model. These limitations include an inflexible surface model that cannot properly model deformations, the inability to use the system in an unbounded extended area and tracking failures in environments with poor 3D geometry.

In this paper we present ongoing research to extend the work of Newcombe *et al.* to permit KinectFusion style mapping in an unbounded environment in real-time. At any point in time our system maintains a TSDF of the region of space that is currently being mapped. The region of space represented by this TSDF varies dynamically during processing. As new regions of space enter the TSDF, previously mapped regions are extracted into a more parsimonious triangular mesh representation.

We present results that demonstrate the technique's ability to create highly detailed maps of extended scale environments. We also present some early stage work which allows the KinectFusion tracking and surface reconstruction algorithm to function correctly in areas with few 3D features.

II. Related Work

Other researchers have addressed the problem of spatially extended mapping with RGB-D cameras however they have taken more traditional approaches to localisation and scene estimation and therefore have required post-processing of the map to generate a representation suitable, for example, for navigation [7]. Henry *et al.* [6] combine visual feature matching with ICP-based pose estimation to build a pose-graph which they optimise to create a globally consistent map. The resulting point cloud map is post-processed to generate a *surfel* model that significantly reduces the map storage requirements whilst providing a visually smoother representation of the environment. Endres *et al.* [3] developed a similar system for computing globally consistent pose-graph maps from which they derive an efficient volumetric model of the environment that explicitly represents free space and unmapped areas.

Audras and Comport [2] present an approach by defining a warping function that incorporates both depth and graylevel information to map the appearance of the curent image to that of a reference image. Pose estimation is then achieved through a non-linear least squares cost between the warped image and the original image. Real-time performance is achieved with the resulting map being constructed by stitching individual depth maps from the keyframes based on their computed pose. However given that the map does not fuse information in a manner similiar to the KinectFusion algorithm it does not achieve the same level of fidelity and does not make any attempt to derive a surface representation of the environment.

Recently Pirker *et al.* [13] presented the GPSlam algorithm that combines sparse appearance-based simultaneous localisation and mapping with dense volumetric modelling of large scale environments. Here they separate the processing into either tracking or exploration based on a simple test of the volume of previously mapped space in the current view. An interesting aspect of the work is a framework for the detection and correction of inconsistencies during updates and loop closures. This approach may be complimentary to ours and is something we are currently investigating.

During the writing of this paper we became aware of a similar effort to ours that is currently underway by Francisco Heredia and Raphael Favier [1]. The main differences between their solution to date and the work reported here is that our implementation is capable of full 6-DOF camera motion, extraction of point clouds as they leave the KinectFusion volume, incremental mesh construction of the extracted points, and hence spatially extended mapping. Furthermore, we present first results of an extended system that integrates place recognition for loop closure detection, pose-graph optimisation through iSAM, and depth map synthesis for reintegration of the mesh into the KinectFusion volume.

III. Approach

In this section we describe the improvements we made to the KinectFusion algorithm. We based our system on the open source C++ implementation of the KinectFusion algorithm released by the PCL project [14].

At the core of the KinectFusion algorithm is a truncated signed distance function (TSDF), a volumetric representation of the scene, where each location stores the distance to the closest surface. A weight that is proportional to the uncertainty of the surface measurement is also stored for each value in the TSDF. To integrate the raw data from each new frame into the TSDF, KinectFusion first computes a vertex map and normal map pyramid. This pyramid is then used to compute the pose of the camera using ICP in conjunction with a predicted surface model derived from the current TSDF. Extraction of a predicted surface from the TSDF is achieved by detecting the zero crossings through a GPU based raycasting operation. Given the output of the ICP procedure, new measurements are integrated by first transforming them into the frame of reference of the global TSDF. The TSDF is then updated via a weighted running average using the weights mentioned above. The weights themselves either accumulate over time resulting is an averaging process over all measurements or are truncated at some maximum value resulting in a moving average, allowing reconstruction of scenes with dynamic object motion.

A. Continuous Representation

As described by Newcombe *et al.* [11], the surface currently being reconstructed and tracked by the system is represented by a TSDF. This voxel data structure is implemented as a 3D array stored in GPU memory with each cell containing a distance value and a weight. All computation involving the TSDF including tracking and integration is carried out on the GPU. There are two main parameters used to configure the TSDF, namely the size of the TSDF in voxels v_s and the dimension in meters d (assuming a cubic TSDF volume as we do henceforth). These two values affect the size of the area which can be reconstructed and the resolution of the resulting reconstruction, which in turn affect tracking performance and accuracy. The metric size of one voxel is simply calculated as:

$$v_m = \frac{d}{v_s} \tag{1}$$

The 6DOF pose of the camera within the TSDF at time i, denoted by C_i, is represented by a 3×3 rotation matrix R_i and a 3×1 translation vector $\mathbf{t}_i$. We set the origin of the TSDF coordinate system to be positioned at the center of the TSDF volume with the basis vectors to be aligned with the TSDF axes. The initial pose of the camera is set to $R_0 = I, \mathbf{t}_0 = (0, 0, 0)^\top$.

Unlike the work of Newcombe *et al.* we do not restrict the tracking and surface reconstruction to the region around the point of initialisation of the TSDF, but rather permit the area mapped by the TSDF to move over time. This is implemented with a cyclical buffer type data structure which allows us to continuously augment the reconstructed surface in an incremental fashion as the camera translates and rotates in the real world. The basic process of our system is:

1) Check how far the camera is from the origin.

2) If above a specified threshold, virtually translate the TSDF so that the camera is once again centred.
    a) Extract surface from region no longer in the TSDF and add to pose-graph
    b) Initialise new region entering the TSDF as unmapped

The main configurable component of this system is a movement threshold b which denotes the distance in meters in all directions from the current origin that the camera may move before the TSDF recentres.

The system functions identically to the original KinectFusion algorithm while the camera remains within the region encompassed by the movement threshold b. Upon crossing this boundary in any one of the three translation dimensions x, y and z, the TSDF volume is virtually translated about the camera pose (in discrete voxel units) to shift the camera to within a distance of less than v_m from the TSDF origin. The new pose of the camera C_{i+1} after a boundary crossing is given as:

$$C_{i+1} = (R_{i+1}, \mathbf{t'}_{i+1}) \tag{2}$$

Where R_{i+1} is obtained from the standard KinectFusion algorithm and $\mathbf{t'}_{i+1}$ is calculated by firstly obtaining the number of whole voxel units crossed $\mathbf{u}$ since the last boundary crossing:

$$\mathbf{u} = \left\lfloor \frac{\mathbf{t}_{i+1}}{v_m} \right\rfloor \tag{3}$$

Subtracting $\mathbf{u}$ from the unshifted $\mathbf{t}_{i+1}$ gives,

$$\mathbf{t'}_{i+1} = \mathbf{t}_{i+1} - v_m \mathbf{u} \tag{4}$$

Finally, we compute the global position of the new TSDF, $\mathbf{g}_{i+1}$ as:

$$\mathbf{g}_{i+1} = \mathbf{g}_i + \mathbf{u} \tag{5}$$

where the position of the intial TSDF is given by $\mathbf{g}_0 = (0, 0, 0)^\top$. Note in the case where the camera does not cross any boundary between two frames the vector $\mathbf{u}$ is zero and as such $\mathbf{g}_{i+1} = \mathbf{g}_i$.

Following a boundary crossing, $\mathbf{g}_{i+1}$ can be used in conjunction with $\mathbf{t}'_{i+1}$ to produce a new pose in our pose-graph (detailed in Section III-C) while the $\mathbf{u}$ quantity is used to extract out a "slice" of the TSDF surface which moves out of one side of the cubic 3D voxel array before we calculate a cyclical offset for future interactions with the array (detailed next in Section III-B).

B. Implementation

In order to efficiently allow the TSDF voxel array to move relative to the tracked surface and recentre itself around the camera, as opposed to moving considerable regions of voxels within the array, we simply treat the array as cyclical and hence need only to update the base index offsets for each dimension of the array.

There are two main parts of the KinectFusion algorithm which require indexed access to the TSDF volume; 1) Volume Integration and 2) Raycasting, both of which access individual elements in the TSDF volume through direct array indexing. Functionality with a continuously changing base index has been achieved by modifying how this indexing is performed. When looking up an element at index (x, y, z) in a cubic 3D array stored in row major order with dimension v_s the 1D position a can be calculated as:

$$a = (x + y v_s + z v_s^2) \tag{6}$$

At the latest time i the $\mathbf{g}_i$ vector described in Section III-A contains the number of voxels we have travelled in each dimension. By use of the modulo operation the indices passed into Equation 6 can be modified to function seamlessly with a cycling base index provided by the value in $\mathbf{g}_i$:

$$x' = (x + \mathbf{g}_{ix}) \mod v_s \tag{7}$$
$$y' = (y + \mathbf{g}_{iy}) \mod v_s \tag{8}$$
$$z' = (z + \mathbf{g}_{iz}) \mod v_s \tag{9}$$
$$a = (x' + y' v_s + z' v_s^2) \tag{10}$$

Also mentioned in Section III-A is the extraction of the surface that falls outside of the boundary of the TSDF when it recentres. The $\mathbf{u}$ quantity calculated in Equation 3 is used in conjunction with $\mathbf{g}_i$ to index a 3D slice of the TSDF to extract surface points from. Surface points are extracted by casting rays orthogonally along each dimension of the TSDF slice where zero crossings in the distance values are extracted as vertices of the reconstructed surface. The 3D slice which was just ray cast is then zeroed to allow new surface measurement integration. The extracted vertices are downloaded from GPU memory and added to the point cloud set M_i for the camera pose at time i.

This orthogonal ray casting can result in duplicate points if the TSDF voxel array is obliquely aligned to the reconstructed surface. In order to remove these duplicate points we apply a voxel grid filter. This filter overlays a voxel grid (in our implementation with a leaf size of v_m) on the extracted point cloud and returns a new point cloud with a point for each voxel that represents the centroid of all points that fell inside that voxel.

C. Pose-Graph Representation

To represent the external mesh we employ a pose-graph representation, Q, where each pose stores an associated *surface slice*. Each time the camera pose moves across the movement boundary and triggers a TSDF cycle, a new element is added into Q. In reality we only add to the pose-graph on boundary crossings that result in a minimum number of points extracted from the TSDF surface as a pose without any associated surface is useless. Figure 2 shows a simplified 2D example of the movement of the camera and the TSDF virtual frame over the course of five consecutive frames while Figure 3 shows the final step with greater detail highlighting the $\mathbf{t}'$ vector. At the end of the example shown Q, would contain three elements

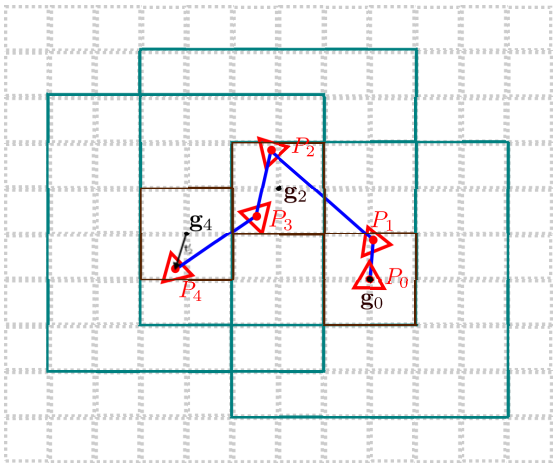

Fig. 2. In this figure the global position of the TSDF (denoted by $\mathbf{g}_i$) is updated twice over the course of five frames. The continuous pose of the camera P_i is shown for each of the five frames. The current TSDF virtual window in global space is shown in teal, the current movement boundary is shown in dark brown and the underlying voxel grid quantization is shown in light dashed lines. This simplified example has $b = 1$ and $v_s = 6$.

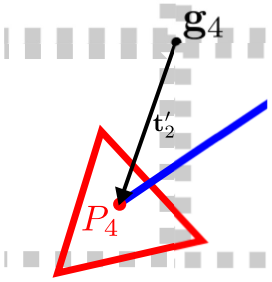

Fig. 3. The figure gives a more detailed look at the value represented by the $\mathbf{t}'$ quantity at the end of the example shown in Figure 2.

from times $i = 0, 2$ and 4. Each element n in Q is composed of four components computed at the corresponding pose time i such that:

$$Q_n = (\mathbf{g}_i, \mathbf{t}'_i, R_i, M_i) \tag{11}$$

D. Mesh Generation

The process described in the three previous sections results in a system which continuously streams a dense point cloud to main system memory as the camera moves through an environment. Given the quality, density and incremental nature of this point cloud, real-time streaming mesh computation is feasible. By extracting with a slight overlap in each TSDF volume slice (two voxels in our implemention) we can produce a seamless and continuous mesh.

We note that there are other approaches to surface extraction such as marching cubes, however in the current work we have exploited the existing raycasting operation to minimise the time to extract the data, delegating the mesh construction procedure to the CPU.

As new poses and associated TSDF volume slices are streamed from the GPU memory, they are immediately fed into a greedy mesh triangulation algorithm described by Marton *et al.* [10]. The result is a high polygon count surface representation of the mapped environment. Additionally as this is stored in system memory the size of the computed mesh surface is bounded only by available memory, with sample values detailed in Table I.

E. Visual Odometry

A critical shortcoming of the original KinectFusion system is its inability to function in environments with a low number of 3D geometric features. An obvious approach to ameliorating this issue is to incorporate visual odometry (VO) into the processing pipeline. As part of one of the more experimental aspects of our system we have substituted the KinectFusion ICP odometry estimate with the output of the feature-based FOVIS visual odometry system [7]. Results from this development are presented independently in Section VI which demonstrate the potential advantage of combining both approaches.

IV. System Architecture

In order to provide truly constant time mapping and localisation, bounded only by memory capacity, we adopted a multi-level threaded system architecture based on parallel processing and thread wait and signal mechanisms in between each level. This design allows the "front-end" TSDF tracking and surface integration to run without having to wait for operations on the outputted point cloud slices to complete.

Upon encountering a boundary crossing and subsequently outputing a new pose-graph element, the top level TSDF thread pushes back this new element to a vector of such elements and signals the CloudSliceProcessor thread to begin processing the new data. The CloudSliceProcessor determines the transformation for the extracted point cloud of each slice required for map rendering and stitching and also carries out the voxel grid downsampling discussed in Section III-B. Once this component is finished processing, it signals all other ComponentThreads so they may begin processing of their own.

A scalable and easily extendable modular system of threaded independent components is achieved by inheritance and polymorphism. By means of inheritence and polymorphism in C++ we derive a base class for all bottom level ComponentThreads, allowing easy creation and destruction of such threads and sharing of commonly used data (e.g. the pose-graph and mapped surface so far). As an example the mesh generation functionality of our system discussed in Section III-D is implemented as such a thread, along with many of the work in progress extensions we discuss later in Section VII. This scalable and easily extendable modular system of threaded independent components provides a very cohesive and maintainable interface for processing the output of the main TSDF tracking and surface construction thread. The only

limitations are those imposed by processing power, such as number of CPU cores and amount of main system memory.

Using this incremental system architecture in combination with the techniques described in Section III we have been able to densely map extended areas in real-time with no performance impact as the size of the surface grows.

V. Experimental Evaluation

As part of the ongoing development of the system we have carried out a number of experiments evaluating both the qualitative performance of the system and the computational performance of each of the components. This section describes the experimental setups we used.

A. Hardware

For all tests we ran our system on a standard desktop PC running 32-bit Ubuntu 10.10 Linux with an Intel Core i7-2600 3.4GHz CPU, 8GB DDR 1333MHz RAM and an nVidia GeForce GTX 560 Ti 2GB GPU. The RGB-D camera used was a standard unmodified Microsoft Kinect.

B. Datasets

All data was recorded on a laptop computer with a human controlling the Kinect. The capture rate was 15FPS due to the hardware limitations of the capture platform. We evaluated three datasets in the context of continuous dense visual SLAM and mesh generation as well as a fourth dataset which evaluates the performance of FOVIS odometry in place of ICP odometry. A value of $v_s = 512$ and $b = 14$ was used for all tests. The four datasets were as follows :

1) Walking within a research lab, (LAB).
2) Walking within and between two floors of an apartment, (APT).
3) Driving within a suburban housing estate at night, with the camera pointing out of a passenger window, (CAR).
4) Walking the length of a straight corridor (VO *vs.* ICP evaluation dataset), (CORR).

VI. Results

In this section we present both qualitative and computational performance results for all four datasets.

A. Qualitative Performance

Qualitative results are provided in the associated video contribution where we show the output of the system over the four datasets mentioned in the previous section. It should be noted that in the associated video the visualisation thread was running at a lower priority than the main TSDF update thread. For this reason, at points where the surface has grown quite large the system will appear to slow down. However, this slow down is restricted to the visualisation thread only while tracking and mapping is in fact running in real-time and not dropping any frames. Our video contribution and point clouds are available at http://www.cs.nuim.ie/research/vision/data/rgbd2012.

We have also provided figures of the final outputs of the system on the APT, CAR, and LAB datasets in Figures 1, 5, and 4, respectively. As can be seen from the results in the video along with the data presented in Table I the scale of the environments mapped is considerably larger than what was previously possible with the standard KinectFusion algorithm. Furthermore, the detail of the resulting maps and their ability to model occlusions within the environment is maintained (e.g. see inset on Figure 4). A reduction in model smoothness is apparent in the FOVIS dataset, this is due to the fact that the ICP odometry matches depth frames against the dense volume accumulated from numerous past frames whereas FOVIS odometry is purely based on frame to frame correspondences.

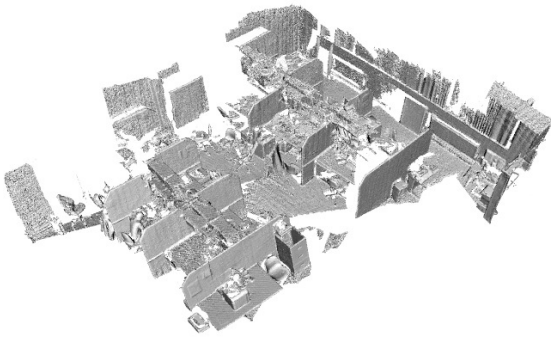

Fig. 4. Map reconstruction of indoor handheld research lab dataset.

B. Computational Performance

Given the asynchronous nature of the different components of the system we evaluate the computational performance of each level of the thread hierarchy discussed in Section IV separately. In particular we are interested in the framerate of the top level TSDF front-end tracking and surface mapping thread, the speed of the intermediate CloudSliceProcessor (CSP) thread and finally the performance of the bottom level ComponentThreads, namely the Mesh Generation module. We also evaluate the computational performance of FOVIS visual odometry estimation versus the original ICP odometry estimation.

In Table II we present execution time results for each system component on all datasets. These measurements were recorded with all visualisation modules disabled. The values shown for the TSDF and CloudSliceProcessor components are the maximum average execution times in milliseconds where the average is computed over a thirty frame sliding window throughout the experiments. The values listed in the Mesh Generator row indicate the average and maximum number of cloud slices in the mesh generation thread queue when the data is played in real-time at capture rate (15FPS in all cases), formatted in the order average/maximum/total amount of cloud slices. The execution time of the TSDF update for each dataset shows that although the data was only captured at 15FPS the system is capable of executing at the full

Dataset	LAB	APT	CAR	CORR
TSDF Volume Size (m)	6	6	20	10
Pose-to-pose Odometry (m)	31.07	42.31	136.18	56.08
Bounding Cuboid (m) [x, y, z]	[14.05, 4.99, 10.54]	[11.22, 6.93, 6.23]	[81.56, 22.77, 81.47]	[5.25, 8.3, 58.67]
Bounding Cuboid Volume (m^3)	739.7	483.8	151332	2558.77
Vertices (without overlap) [size (MB)]	1.2×10^6 [18.34]	1.2×10^6 [19.1]	8.8×10^5 [13.57]	1.6×10^6 [24.59]
Mesh Triangles (with overlap) [size (MB)]	2.3×10^6 [88.21]	2.5×10^6 [93.24]	1.5×10^6 [62.97]	3.1×10^6 [117.7]

TABLE I
SUMMARY OF PROCESSING AND MAP STATISTICS FOR EACH OF THE DATASETS DESCRIBED IN SECTION V-B.

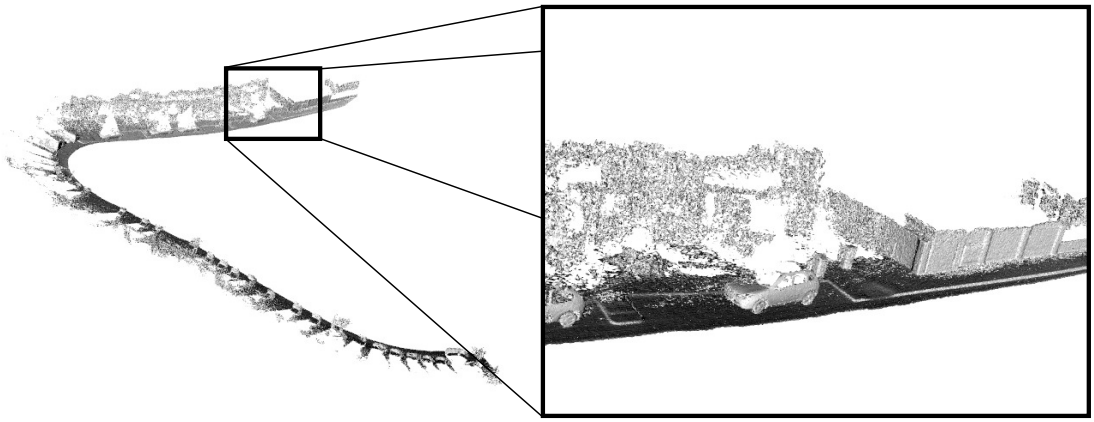

Fig. 5. Map reconstruction of an outdoor dataset captured from a car. Inset: zoomed region showing detail level of the final map.

Dataset	LAB	APT	CAR	CORR
TSDF (ms)	33.94±3.54	33.83±4.85	36.79±4.49	41.25±7.03
CSP (ms)	3.53±3.4	3.52±3.38	2.62±2.39	4.7±3.87
Mesh Gen.	1.27/5/135	1.36/7/178	2.57/25/254	2.28/6/220
Odometry	ICP	ICP	ICP	FOVIS

TABLE II
COMPUTATIONAL PERFORMANCE RESULTS FOR LAB, APT, CAR AND CORR DATASETS, SHOWING THE MAXIMUM AVERAGE COMPUTATION TIMES FOR THE KINECTFUSION (TSDF), CLOUDSLICEPROCESSOR (CSP), AND THE AVERAGE/MAX/TOTAL QUEUE SIZES FOR MESH GENERATION.

30FPS frame rate of the Kinect sensor. Additionally it can be seen that our extension to the original algorithm does not affect the real-time performance of the system. The processing done by the CloudSliceProcessor intermediate thread, while asynchronous to the TSDF thread, does not make the pool of ComponentThreads wait more than 2 - 4 ms for an outputted slice from the TSDF tracking module. The performance of this particular thread is affected by the number of points extracted from the surface in a given slice of the TSDF volume. With regards to mesh generation a queue length of 1 implies that the mesh generator is keeping up with the output of the CloudSliceProcessor and real-time mesh construction is succeeding. On average our system keeps up in producing the mesh surface and catches up when computationally feasible. The large maximum queue length value is expected in the CAR dataset due to the fast motion of the camera; many large slices are outputted by the TSDF and CloudSliceProcessor in such runs.

Analysing the performance of the FOVIS odometry replacement we note that it has poorer computational performance than the original CUDA implemented KinectFusion ICP odometry estimator at 14.71±4.39ms versus 10.54±0.21ms respectively. This is to be expected given that the FOVIS implementation is CPU based. However it is still sufficient to execute in real-time given the frame rate of the captured data. This increase in execution time is reflected in the TSDF value for the CORR dataset in Table II.

VII. CURRENT INVESTIGATIONS

In the long term the aim of this work is to incorporate a full SLAM approach including loop closure detection, mesh reintegration (*i.e.* into the TSDF), and global pose and mesh optimisation. In this section we discuss ongoing work on extensions to the system with regard to these and other issues.

A. Loop Closure Detection and Handling

In order to provide visual loop closure detection we have integrated the DBoW place recognition system [5] in conjunction with SURF feature descriptors as a separate ComponentThread. Based on the well established bag-of-words model, when the system identifies a loop closure a pose constraint is computed between the matched frames. As this pose constraint is between two RGB images it is then propagated back to the centre of the TSDF virtual frame in order to properly adjust the associated poses. This is made possible by transforming back by the $\mathbf{t}'$ vector associated with the matched frames. We

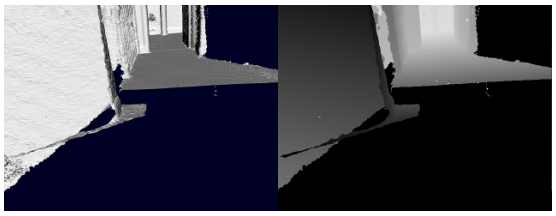

Fig. 6. Shown is the cumulative generated mesh as viewed from the camera estimate and a depth map synthesized from this render.

have also experimented with integrating our pose-graph with iSAM [8] for smoothing and global pose-graph optimisation. Given the dense structure that is produced by the TSDF front-end we have observed that a mesh deformation of some kind is required in conjunction with the iSAM optimisation which adjusts poses.

B. Map Reintegration

Another aspect of the system we are currently investigating is the reintegration of previously mapped surfaces to the front-end TSDF as they are encountered, aiding in loop closure and drift reduction. One of the approaches we are experimenting with involves rendering the computed mesh surface at the estimated position of the camera in the global frame from the camera's perspective. Subsequently we synthesize a depth map from this mesh render (similar to the work of Fallon et al. [4]) and either merge this with the current Kinect depth map or interleave it with the live Kinect data and pass it in independently. Figure 6 shows some preliminary results of this work.

C. Mapping Improvements

One of the advantages of capturing such a dense and rich model in real-time is the possibility of higher level processing and reasoning such as object recognition and other semantic processing. One of the first steps forward in this direction we are currently exploring is the integration of texture in the surface building process. In addition to this we are looking at fitting surface models to the data, such as planes and curves using sample consensus methods. The extraction of such primitives from the data provides higher level representations more suitable for semantic understanding and also reduces the overall complexity of the data.

VIII. Conclusion

We have developed an algorithm which extends the KinectFusion framework to overcome one of the principal limitations of the original system by Newcombe *et al.*; the inability to work over extended areas. In addition to this, we have also implemented a real-time triangular mesh generation module for representing the extended scale map thereby maintaining the ability to characterise topological and occlusion relationships within the environment. We have also integrated the FOVIS visual odometry library into the processing pipeline and evaluated its potential in increasing robustness and reducing overall drift.

The system is organised as a set of hierarchical multi-threaded components which are capable of operating in real-time. The software framework we have developed facilitates the easy creation and integration of new modules which then also have a minimal performance impact on the main front-end dense TSDF tracking and surface reconstruction module. In the future we will extend the system to implement a full SLAM approach including loop closure detection, mesh reintegration, and global pose and mesh optimisation.

Acknowledgments

Research presented in this paper was funded by a Strategic Research Cluster grant (07/SRC/I1168) by Science Foundation Ireland under the Irish National Development Plan and the Embark Initiative of the Irish Research Council for Science, Engineering and Technology.

References

[1] KinectFusion extensions to large scale environments. http://www.pointclouds.org/blog/srcs/fheredia/index.php, May 11th 2012.

[2] C. Audras, A. I. Comport, M. Meilland, and P. Rives. Real-time dense RGB-D localisation and mapping. In *Australian Conference on Robotics and Automation*, Monash University, Australia, December 2011.

[3] 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 Int. Conf. on Robotics and Automation (ICRA)*, St. Paul, MA, USA, May 2012.

[4] M. F. Fallon, H. Johannsson, and J. J. Leonard. Efficient scene simulation for robust Monte Carlo localization using an RGB-D camera. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, St. Paul, MN, May 2012.

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

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

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

[8] M. Kaess, A. Ranganathan, and F. Dellaert. iSAM: Incremental smoothing and mapping. *IEEE Transactions on Robotics (TRO)*, 24(6):1365–1378, December 2008.

[9] G. Klein and D. Murray. Parallel tracking and mapping for small AR workspaces. In *Proceedings Sixth IEEE and*

ACM International Symposium on Mixed and Augmented Reality (ISMAR'07), Nara, Japan, November 2007.

[10] Z. C. Marton, R. B. Rusu, and M. Beetz. On fast surface reconstruction methods for large and noisy datasets. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Kobe, Japan, May 2009.

[11] 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 2011 10th IEEE International Symposium on Mixed and Augmented Reality*, ISMAR '11, pages 127–136, Washington, DC, USA, 2011. IEEE Computer Society.

[12] R. A. Newcombe, S. J. Lovegrove, and A. J. Davison. DTAM: Dense tracking and mapping in real-time. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2320 –2327, November 2011.

[13] K. Pirker, M. Rüther, G. Schweighofer, and H. Bischof. GPSlam: Marrying sparse geometric and dense probabilistic visual mapping. In *Proceedings of the British Machine Vision Conference*, pages 115.1–115.12. BMVA Press, 2011.

[14] R. B. Rusu and S. Cousins. 3D is here: Point cloud library (PCL). In *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, May 2011.

[15] J. Stuehmer, S. Gumhold, and D. Cremers. Real-time dense geometry from a handheld camera. In *Pattern Recognition (Proceedings DAGM)*, pages 11–20, Darmstadt, Germany, September 2010.