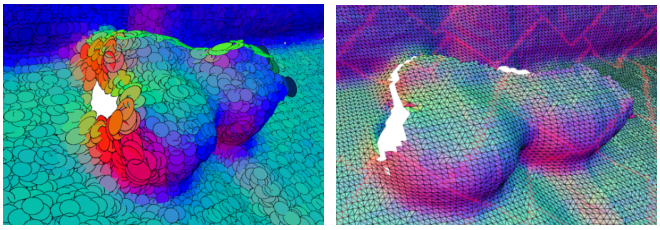
2019 International Conference on Robotics and Automation (ICRA)
Palais des congres de Montreal, Montreal, Canada, May 20-24, 2019

ScalableFusion: High-resolution Mesh-based Real-time 3D Reconstruction

Simon Schreiberhuber, Johann Prankl, Timothy Patten and Markus Vincze

***Abstract*— Dense 3D reconstructions generate globally consisent data of the environment suitable for many robot applications. Current RGB-D based reconstructions, however, only maintain the color resolution equal to the depth resolution of the used sensor. This firmly limits the precision and realism of the generated reconstructions. In this paper we present a real-time approach for creating and maintaining a surface reconstruction in as high as possible geometrical fidelity with full sensor resolution for its colorization (or surface texture). A multi-scale memory management process and a Level of Detail scheme enable equally detailed reconstructions to be generated at small scales, such as objects, as well as large scales, such as rooms or buildings. We showcase the benefit of this novel pipeline with a PrimeSense RGB-D camera as well as combining the depth channel of this camera with a high resolution global shutter camera. Further experiments show that our memory management approach allows us to scale up to larger domains that are not achievable with current state-of-the-art methods.**



(a) ElasticFusion. Surfels are color coded by normal vectors with additional outlines in black for the borders.

(b) ScaleableFusion (direct mesh based reconstruction) with outlined triangles.

Fig. 1: A meshed set of three apples (zoomed in from Fig. 2). Due to not relying on excessive overlapping of surfels to achieve a tight surface model, mesh reconstructions allow surfaces to be described more precisely and efficiently.

I. Introduction

The introduction of affordable RGB-D sensors, initially for game consoles, has also lead to their adoption in other applied fields. The ability to fuse multiple frames into one consistent 3D reconstruction of a scene has fueled expectations for a large variety of applications. Robotics as well as Augmented Reality, to name two examples, benefit greatly from a complete representation of environments.

Generating consistent 3D reconstructions based on these sensors is challenging due to two main reasons. Firstly, geometrical consistency is not provided by the depth sensor due to distortions and incorrect depth values, noise and other effects caused by rolling shutter sensors. Secondly, photometric consistency can not be acheived by the RGB sensor due to limited geometrical resolution of the reconstruction and the dependence of perceived intensity on the camera and object position as well as lighting conditions.

While future sensors may mitigate these problems, present sensors all suffer same problems [14], [5]. Therefore, it is the task of reconstruction algorithms and the supporting hardware to cope with the challenges. The most common approaches are based on KinectFusion ([7], [10], [15]) and ElasticFusion ([16], [17], [8]). The first class of methods utilize Truncated Signed Distance Functions (TSDFs) to carve out the free space of a regular voxel grid of uniform resolution. The second class of methods stores the geometrical information in Surfels, which are small unconnected discs of different size and color. Essentially both methods attempt to achieve geometrical and colometric consistency by averaging measurements from multiple perspectives. The averaging schemes achieve pleasing results for geometry but they are unsatisfactory for the color. This is firstly due to the widespread practice of ignoring exposure and gain information of the sensors [2]. But more importantly, the limitation is due to the underlying data structures that enforce a one to one relationship between color and geometric resolution, which leads to a loss of color information. It is also common for these algorithms to steadily aggregate new data on the GPU until it runs out of memory. This disqualifies their use for serious applications in robotics.

Our solution to the described shortcomings is a third paradigm for 3D reconstruction to work directly on meshes. Our contribution is the introduction of *ScaleableFusion*: A triangulated mesh, as shown in Fig. 1, represents surfaces more accurately by breaking up the direct coupling of geometry and color resolution. This requires extra effort to maintain in comparison to surfels or TSDFs but it comes with several benefits. (1) The vertices making up the geometry do not need to be evenly spaced, like the voxels in a TSDF, and the triangles spanning the vertices still sufficiently describe tight surfaces unlike unconnected surfels. (2) The ability to handle textures allows the progression from averaging colors of unconnected surface elements to selecting fitting keyframes for entire surface regions. These keyframes are selected with awareness of exposure time [2] and corrected for vignetting [1] to further improve photometric consistency.

All authors are with the Vision4Robotics group, Automation and Control Institute (ACIN), TU Wien, Austria {schreiberhuber, prankl, patten, vincze}@acin.tuwien.ac.at

This work is partially supported by the European Comission through the Horizon 2020 Programme (H2020-ICT-2014-1, Grant agreement no. 645376), FLOBOT and Austrian funding from WWTF ICT15-045, RALLI.

Real-time performance is achieved through our Level of Detail (LOD) system (example in Fig. 2) that is novel to RGB-D reconstruction. This scheme offloads the geometry from GPU to CPU memory to enable large scale reconstructions.

The remainder of this paper is as follows. In Section III we present the ScaleableFusion approach. In Secion IV we then describe the implementation of the memory management system that is necessary for live continous reconstruction. Section V gives a detailed comparison between KinectFusion, ElasticFusion and our approach to highlight where and why ScalableFusion generates higher quality reconstructions. We additionally collect data with a higher resolution RGB sensor to demonstrate the improvements over the classic VGA-resolution RGB-D sensor as well as the benefits of a higher resolution RGB sensor in the high resolution pipeline. Our experiments also establish the limits up to which current methods can reconstruct scenes before running out of GPU memory while demonstrating that our system efficiently offloads unused data to the system memory. The paper concludes in Section VI with a discussion of future work.

II. Related Work

One of the first reconstruction algorithms introduced by Izadi et al. is KinectFusion [7]. This approach maintains a volume in the form of a 3D grid that is populated with values of a TSDF to indicate where the closest surface resides. The initial implementation is only able to map small volumes of fixed position, size and resolution. By dynamically changing the position of the active reconstruction volume, Kintinuous [15] extends the basis algorithm and enables the reconstruction of larger scenes. The use of voxel hashing [11] allows higher resolution reconstructions by reducing the memory footprint required for the reconstruction volume. KinectFusion spawned further notable expansions such as DynamicFusion [10], which introduces a warp-able volume to reconstruct non-rigid objects. Another thoroughly researched approach is made popular by ElasticFusion [16]. The captured points are stored as surfels, which are small discs with diameter, orientation, position and color. This allows surfaces with varying spatial resolution to be stored depending on which distance was perceived by the sensor.

These approaches directly couple color and geometry to one resolution. This means that one resolution cannot be increased without increasing the other. A feasible approach to decouple the geometrical and photometric resolution is to span a triangle mesh with textures as done in other work (e.g. [4], [13], [18]) but these are generally not usable for real-time applications.

The approach by Fu et al. [4] enhances a TSDF based reconstruction with textures. Following a TSDF based reconstruction phase a mesh is exported from the reconstruction volume. This mesh is partitioned into regions that are textured by their own RGB frame taken during the reconstruction phase. This operation alone would introduce stitching artifacts into the region borders. Therefore, the work assumes that all of these textured region borders contain overlapping color information for their bordering regions.

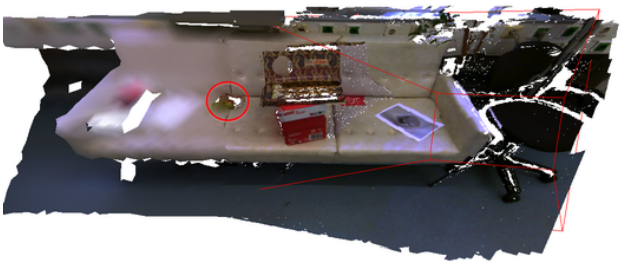


Fig. 2: Demonstration of the Level of Detail system. The younger part of the reconstruction (right) is in memory being processed and displayed at full detail, while the older part (left) is downloaded to system memory and displayed with a low detail replacement. The apples shown in Fig. 1 are taken from this reconstruction (red circle) and meant to demonstrate the dynamic range of detail.

As such, a global optimization step is performed to align these overlaps by adjusting the intrinsics and extrinsics of keyframes. Since photoconsistency can not be fully achieved due to the limited knowledge of the scene's geometry and other factors, a local optimization step is utilized to optimize texture coordinates in these regions. This approach generates high fidelity results, but comes at the cost of a long and extensive offline optimization phase. Our approach, on the other hand, strikes a tradeoff of texture quality and computation time to achieve real-time capabilities.

III. ScalableFusion

Our approch to 3D reconstruction directly utilizes triangle meshes, which by their nature maintain point neighborhoods and have several textures attached. Incoming points are first partitioned into smaller patches. These are then meshed and finally stitched together at their borders. The patches are then enhanced with textures that store geometrical standard deviation and colors. To update the reconstruction on the fly, keyframes are selected to donate textures to surface segments. The possible misalignment between segments prior to stitching (as a result of no costly global optimization step) is minimized by processing the captures images with exposure time awareness and vignetting correction.

In the following, we describe the key steps of our framework. We outline the geometry refinement udpdates, map expansion, meshing and segment stitching.

A. Geometry Refinement Update

The noise characteristics of depth sensors make it impossible to attain precise geometry at large distances when accumulating multiple measurements. After a certain amount of measurements at a large distance, high quality results can only be achieved by positioning the sensor closer to the surface. This is made possible by current methods by employing a scheme that allows close range measurements to outrule long range measurements, no matter how many samples are taken at long range.

ElasticFusion [16] implicitly handles this problem by introducing a "weight" property for surfels. The weight of

a surfel is increased with each observation while position, color and orientation of the surfel are updated. An increasing weight for a surfel implies that it becomes progressively and eventually static while the influence of additional measurements vanishes. Static surfels are not moved but replaced with a denser set of lighter (in terms of weight) surfels when the sensor makes observations from a closer range.

Our approach takes inspiration from these weights but instead of spawning new geometry when the sensor approaches a surface we only do this when it benefits the reconstruction. For example, when approaching a planar surface, ElasticFusion would spawn new surfels just to accommodate the additional color information whereas our approach directly updates the texture and existing mesh.

This behavior is achieved by storing additional textures containing values for every sampled surface point p. The values contained for each texture pixel (texel) are:

μ_k The average deviation of the k measurements from the actual surface. This is used to indicate where the meshed surface deviates from the sensor's perception.

σ_k An estimate of the noise level that decreases with every additional measurement. The smaller the value, the less influence new measurements have on the geometry. Additionally, $\sigma_{s,k}$ defines the estimated noise level of the sensor projected onto the surface point p.

$\sigma_{m,k}$ The estimated minimal noise level achievable with all measurements until step k. This assumes quantization is the only limiting factor. Similar to beforehand, $\sigma_{m,s,k}$ refers to the projected value of the sensor.

The estimated noise level and minimal noise level are updated according to

$$\sigma_{m,k+1} = \min(\sigma_{m,k}, \sigma_{m,s,k}), \quad (1)$$

$$\sigma_{k+1} = \sigma'_{k+1} + \sigma_{m,k+1}, \quad (2)$$

$$(3)$$

where

$$\sigma'_{k+1} = \frac{\sigma'_k \sigma'_{s,k}}{\sigma'_k + \sigma'_{s,k}}, \quad (4)$$

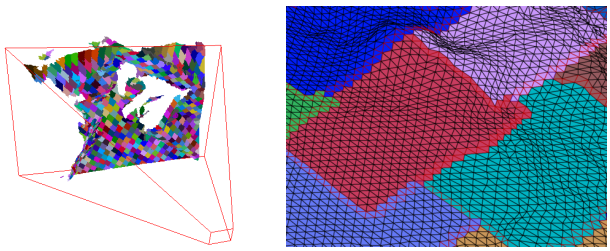
$$\sigma'_{s,k} = \sigma_{s,k} - \sigma_{m,k+1}, \quad (5)$$

$$\sigma'_k = \sigma_k - \sigma_{m,k+1}. \quad (6)$$

Note that σ'_{k+1} is smaller than σ'_k and $\sigma'_{s,k}$, which implies the assumption that every further measurement improves the result. This system guarantees that σ_k approaches $\sigma_{m,k}$ with increasing iteration count k but never falls below it. The resulting values $\sigma'_{s,k}$ and σ'_{k+1} are used to update μ by

$$\mu_{k+1} = \left(\frac{\mu_k}{\sigma'_k} + \frac{d_{s,k} - d_k}{\sigma'_{s,k}} \right) \sigma'_{k+1}, \quad (7)$$

where $d_{s,k}$ is the distance of the surface point perceived by the sensor and d_k is the distance of the reconstructed point/texel to the sensor. Transcribing the texture updates to the vertices is done by shifting the vertex positions along the view rays such that μ_{k+1} is equal to 0 wherever possible.



(a) Coarse segmentation of the point cloud into smaller patches. (b) Patches meshed into a regular pattern of triangles.

Fig. 3: Triangle creation process applied to a single frame.

These updates do not necessarily need to occur every time new sensor values are available for a certain surface pixel. When the estimated noise level of the sensor data σ_k is higher than $\sigma_{s,k}$ on the surface, no update needs to be made. The same applies when the perceived depth values significantly differ from the mapped values. This would indicate either unmapped geometry of an already reconstructed surface or an invalid surface.

B. Expand Update

As soon as the sensor generates a new frame from a new position, the formerly mapped surface elements are used to render a depth map for the current sensor position. The artificial depth map is compared to the depth values currently perceived by the sensor. Depth values that are in close proximity to the mapped values result in updates to the already existing surfaces. If the captured surface leaves this proximity towards the camera, it is added (meshed) to the current reconstruction. The thresholds for these operations as well as $\sigma_{s,k}$ depend on the depth, pixel position and the sensor itself.

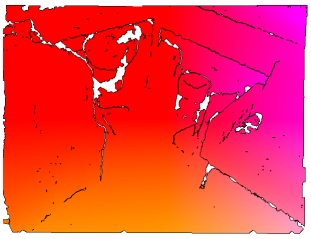
C. Meshing

After identifying the novel parts of the captured depth map, 3D points are created by applying the pinhole model to project the depth pixel. These points are segmented into smaller blocks depending on their distance to each other and the estimated normal vector. The neighborhood information derived from the organized point cloud is directly used for this operation and also for spanning triangles between each neighboring set of 3 points. During this process, care is taken to ensure that no triangles are created where neighboring depth values are within the thresholds mentioned in Sec. III-B. An example of the output from this segmentation and meshing process is shown in Fig. 3.

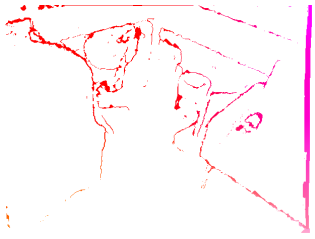
D. Stitching

Generating a mesh on a single organized depth map is computationally undemanding due to the presence of neighborhood information in the 2D image plane. The situation changes, however, as soon as new sensor data is to be integrated into an existing reconstruction.

To adress this problem, all visible triangles captured prior to the current frame are searched for open edges. This refers to every edge where a triangle does not border to another.

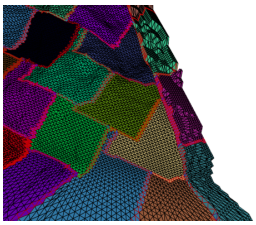


(a) The already existing geometry rendered with open edges outlined in black.



(b) The novel geometry that is not overlapping with the existing geometry.

(c) The coarse segmentation of the novel geometry with smaller regions not being mapped.



(d) The novel and still rough geometry (right) is stitched to the existing geometry.

Fig. 4: Connecting novel data to existing geometry: Open edges (a) outlined in white are connected to the coarse segmentation (c). The result (d) shows an obvious line between old and new geometry in the segmentation pattern.

The open edges are projected to the pixel space of the current frame's depth map. After this process, finding a potential neighbor for a reconstructed triangle within the set of novel triangles is a simple lookup in the current image plane. This process of expanding the reconstruction is shown in Fig. 4.

IV. Implementation

Modern desktop hardware distinguishes between memory bound to the GPU and system memory that is bound to the CPU. Access can not be done across memory spaces without performing expensive data transfer over the PCI-E bus. Therefore, our data structures are designed to mirror the information between CPU and GPU, and are only synchronized when necessary.

Modifications of the geometry occur on either the GPU or the CPU depending on which processor is more fit for the performed task. This has implications on the data structure. While data stored on the CPU is vastly interconnected, the structures on the GPU only store very few references between elements.

We introduce a threading system to simultaneously process tasks that are not fully interdependent. For example, while the geometry of one frame is updating the mesh, data that is not needed can be simultaneously transferred from GPU to system memory (download). The odometry is likewise not explicitly reliant on the most current version of geometry. Pose estimation of a new frame can therefore be performed while the last frame is still being integrated. An exemplary timeline of this system is shown in Fig. 5.

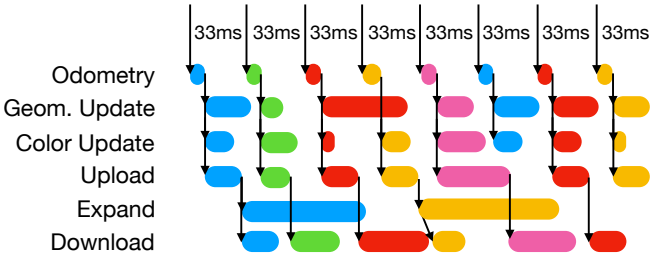


Fig. 5: The employed threading system. Black arrows depict the flow of data. Colored stripes present actual processing.

The depth noise characteristics used in Sections III-A, III-B and III-C are derived by Halmetschlaeger-Funek et al. [5] and approximated as a polynomial. We additionally altered these characteristics to mimic the assumption of [16] whereas data in the center of a depth frame is more reliable than in the peripheral areas.

V. Evaluation

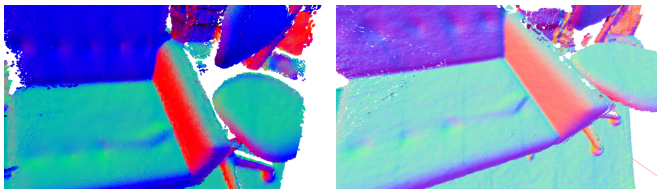
Comparison of 3D reconstruction pipelines is traditionally done by either comparing the reconstructed point cloud with a ground-truth mesh or by comparing the odometry with the ground-truth trajectory. A ground-truth mesh is inherent to rendered datasets [6], while a trajectory can either be input for the rendering process or captured with external cameras and markers [12].

These benchmarks mainly reflect the quality of odometry instead of the perceived quality. To showcase photometric quality instead of tracking quality we capture selected scenes with a combination of a Xtion Live Pro and a rigidly attached IDS global shutter camera that delivers a resolution of 1600x1200 pixels. While recording depth and color from the Xtion sensor at 30Hz we also capture high resolution data from the IDS sensor at 60Hz. We evaluate the quality of our texturing algorithm against the output of ElasticFusion in a similar manner prevalent in the literature focused on high fidelity texturing [4].

A. Geometry Comparison

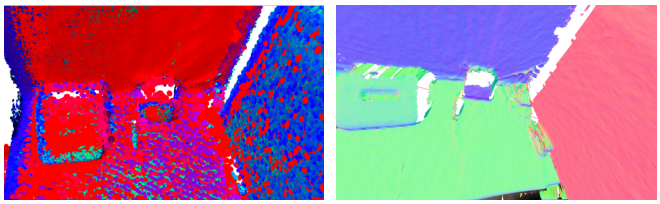
The weighting scheme used by ElasticFusion and our standard deviation based approach are similar. The results do not differ significantly as underlined by the images in Fig. 6. It is, however, visible that the surfels of ElasticFusion are of lower density than the triangles in our reconstruction. Furthermore, these surfels are intended to be overlapping, resulting in a smooth surface. A disadvantage is that the overlapping surfels regularly create redundant (duplicated) versions of surfaces, see Fig. 7.

Methods that directly operate on surfaces are sensitive to tracking errors in a different way. If it is not possible to attribute a captured surface to an reconstructed surface a duplicate will be mapped. Volumetric approaches, such as KinectFusion, are the best to handle these situations because they update a reconstruction volume allowing them to derive a correct surface model. We mitigate this problem



(a) ElasticFusion. (b) ScaleableFusion.

Fig. 6: Normals acquired from ElasticFusion and ScaleableFusion. While both methods deliver similar results, it is notable that our method retains sharper features.



(a) ElasticFusion surfels colorized by count of observations (blue, few vs. red, many). (b) Our reconstruction colorized by normals.

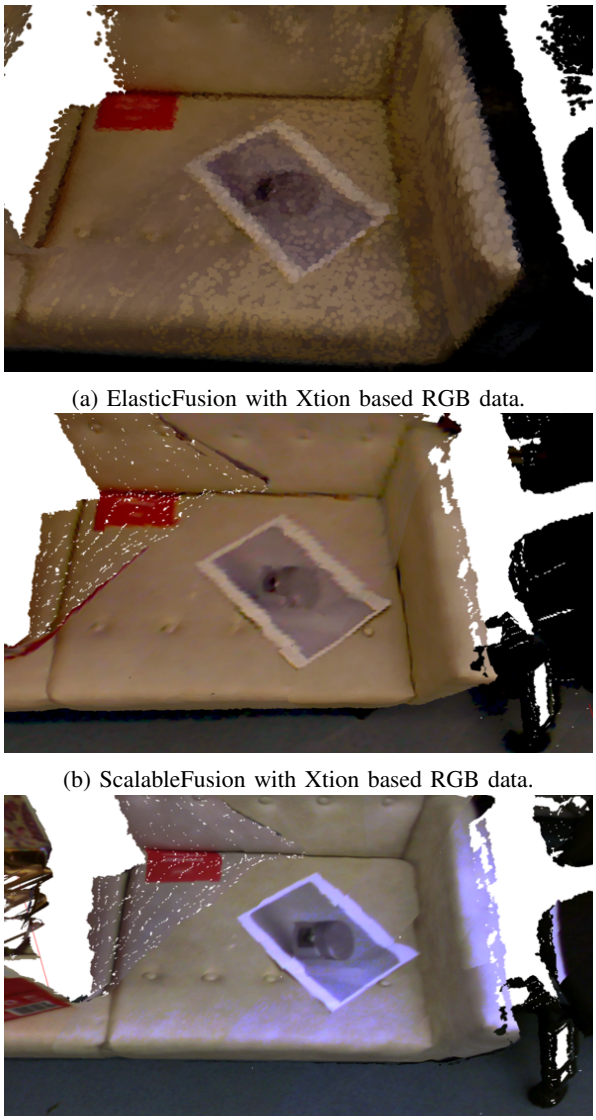
Fig. 7: Reconstruction conducted with ElasticFusion and ScalableFusion. (a) Shows how younger surfels (blue) are partially layed on top of older (red) surfaces. Our reconstruction (b) on the other hand maintains smooth nonredundant surfaces.

through our noise dependent thresholding scheme to achieve more consistent maps than traditional surface based methods. Despite this effort, these artifacts can sometimes still arise when depth and tracking errors exceed the given thresholds.

B. Texture Comparison

Fig. 8 shows that our method makes better use of the RGB sensors than ElasticFusion, which like other methods only features one color per geometrical primitive. Given that there is only space for 9 million surfels as seen in Fig. 10 the entire reconstruction can only store as much color detail as a 9 Megapixel image. Furthermore, the averaging schemes for color, together with tracking inaccuracies and no consideration of photometric sensor characteristics leads to washed out and speckled reconstructions. Our texturing approach overcomes many of these insufficiencies: The texel density is no longer coupled to the vertex density anymore, but instead taken from optimal camera frames to batchwise advance the reconstruction. This enables the usage of high resolution cameras as seen in Fig. 8c. We also take care of exposure times and vignetting according to [2] and [1], to reduce seams along segment borders.

The images in Fig. 8 for our approach still show misalignments of texture along segment borders, which is due to tracking inaccuracies. All methods are of course improved with better odometry. Offline optimization schemes such as presented in [4] yield more consistent results, but unfortunately suffer from heavy computation and are unusable for real-time reconstruction.



(a) ElasticFusion with Xtion based RGB data.

(b) ScalableFusion with Xtion based RGB data.

(c) ScalableFusion with high resolution RGB camera.

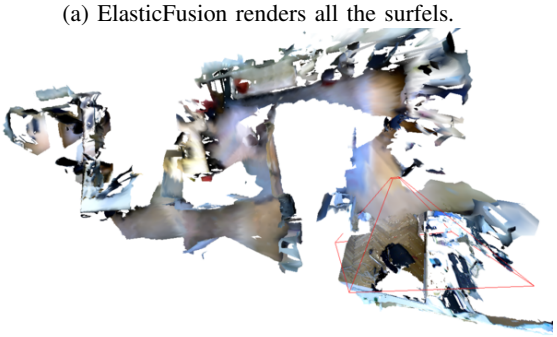
Fig. 8: Most methods average the color observations, leading to washed out and speckled results (a). Our texturing based approach ((b), (c)) samples the color patchwise, therefore, retaining sharper detail. This enables the use of higher resolution cameras (c).

C. Memory Consumption and Computational Performance

The implementations of TSDF and surfel based approaches do not offload irrelevant geometry from the GPU to CPU memory. This becomes a limiting factor when reconstructing large scenes. We therefore benchmark our system against ElasticFusion in two large scale scenes.

The first experiment is a live and continuous reconstruction of a large office space consisting of multiple rooms (Fig. 9). Data is offloaded to the CPU memory and only the currently observed region is maintained on the GPU for immediate processing. This is shown by the different LOD of the bottom right of the reconstruction in Fig. 9b. Our scheme vastly reduces the memory consumption for large scale

(a) ElasticFusion renders all the surfels.



(b) Whereas our system renders the same scene more efficiently.

Fig. 9: Partial reconstruction of a large office space.

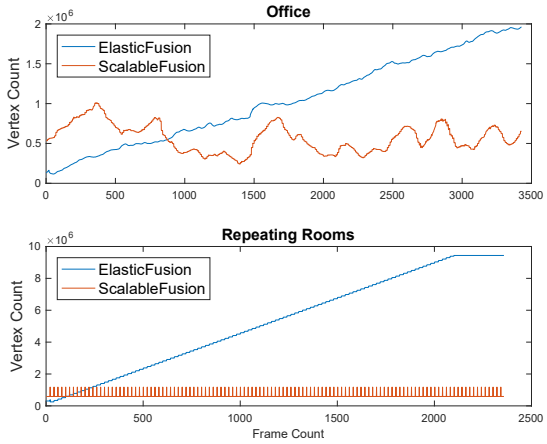


Fig. 10: Count of vertices currently residing on GPU. Top: Reconstruction of the office in Fig. 9. Bottom: Dataset repeating the same room with pose offsets. ScalableFusion stays below 1 million active vertices, while ElasticFusion accumulates data in correlation to the size of the environment.

reconstructions because it is regulated by the current demand. As such, the vertex count residing on the GPU is constant on average, only fluctuating depending on the complexity of the reconstructed frames (Fig. 10). In comparison to ElasticFusion, the surfel count growth is correlated with the overall size of the captured scene.

The second experiment is an extreme stress test for both systems. We repeatedly reconstruct a room dataset [6] while offsetting the pose by 5 m to artificially create a large environment. In this test, ElasticFusion halts at 2200 frames after accumulating 9 million surfels because the capacity of the GPU memory is reached. Our system, on the other hand, seamlessly continues to operate with a constant demand on the same memory.

On our system (Intel Core i7-7700K, Geforce GTX 1070), we achieve a constant framerate of 30 Hz for our texturing and geometry refinement tasks while still allowing the geometry to be expanded, meshed and stitched at 5 Hz. This is independant of the overall reconstruction size.

VI. Conclusion

In this paper we introduced ScalableFusion, a novel reconstruction approach for directly texturing triangles and vertices that significantly improves the quality of 3D reconstruction. While the use of a triangulated mesh imposes an overhead in terms of data management, the benefits vastly outweigh the downsides of increased computational effort. This particularly applies to real-world environments that feature very small objects. A meshed-based approach describes these elements with an outstanding level of detail. ScaleableFusion performs the entire meshing process in the pixel space of the camera frame to eliminate the expensive neighborhood search. This method proves to be superior to colorized geometrical primitives because the camera resolution is utilized to its full potential. We also provide a sophisticated memory management scheme to enable real-time operation using a Level of Detail approach. A detailed comparison showed that ScaleableFusion creates better reconstruction for robotic applications. It creates sharper features (Fig. 8), overcomes the issue of generating double planes (Fig. 7), maintains the full resolution of color cameras (Fig. 2) and overcomes memory limits of present methods (Fig. 10).

While results show significant improvements of state-of-the-art methods, hands on experience reveals that the current odometry method, based on ElasticFusion, is insufficient. The increased resolution exposes the problem of tracking inaccuracies, which manifest themselves as sudden changes in intensity at the borders between textured patches. We are currently considering more elaborate methods such as ORB SLAM [9] and Direct Sparse Odometry [3] to improve this issue. Other future work will entail the task of simplifying and tesselating the geometry, which is necessary to apply the system in changing environments typical in robot applications.

References

[1] S. V. Alexandrov, J. Prankl, M. Zillich, and M. Vincze, "Calibration and correction of vignetting effects with an application to 3D mapping," in *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2016, pp. 4217–4223.

[2] ——, "Towards dense SLAM with high dynamic range colors," in *Proceedings of Compute Vision Winter Workshop*, 2017.

[3] J. Engel, V. Koltun, and D. Cremers, "Direct sparse odometry," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 3, 2018.

[4] Y. Fu, Q. Yan, L. Yang, J. Liao, and C. Xiao, "Texture mapping for 3D reconstruction with RGB-D sensor," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4645–4653.

[5] G. Halmetschlager-Funek, M. Suchi, M. Kampel, and M. Vincze, "Xtion's gone! What's next? An evaluation of ten different depth sensors for robotic systems," *IEEE Robotics Automation Magazine*, 2018, (to appear).

[6] A. Handa, T. Whelan, J. McDonald, and A. Davison, "A benchmark for RGB-D visual odometry, 3D reconstruction and SLAM," in *Proceedings of IEEE International Conference on Robotics and Automation*, 2014, pp. 1524–1531.

[7] S. Izadi, D. Kim, O. Hilliges, D. Molyneaux, R. Newcombe, P. Kohli, J. Shotton, S. Hodges, D. Freeman, A. 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 Technology*, 2011, pp. 559–568.

[8] J. McCormac, A. Handa, A. J. Davison, and S. Leutenegger, "SemanticFusion: Dense 3D semantic mapping with convolutional neural networks," in *Proceedings of IEEE International Conference on Robotics and Automation*, 2017, pp. 4628–4635.

[9] R. Mur-Artal and J. D. Tardós, "ORB-SLAM2: An open-source SLAM system for monocular, stereo, and RGB-D cameras," *IEEE Transactions on Robotics*, vol. 33, no. 5, pp. 1255–1262, 2017.

[10] R. A. Newcombe, D. Fox, and S. M. Seitz, "DynamicFusion: Reconstruction and tracking of non-rigid scenes in real-time," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 343–352.

[11] M. Nießner, M. Zollhöfer, S. Izadi, and M. Stamminger, "Real-time 3D reconstruction at scale using voxel hashing," *ACM Transactions on Graphics*, vol. 32, no. 6, pp. 169:1–169:11, 2013.

[12] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of RGB-D SLAM systems," in *Proceedings of IEEE/RSJ International Conference on Intelligent Robot Systems*, 2012, pp. 573–580.

[13] M. Waechter, N. Moehrle, and M. Goesele, "Let there be color! Large-scale texturing of 3D reconstructions," in *Proceedings of European Conference on Computer Vision*, 2014, pp. 836–850.

[14] O. Wasenmüller and D. Stricker, "Comparison of kinect v1 and v2 depth images in terms of accuracy and precision," in *Computer Vision – ACCV 2016 Workshops*, C.-S. Chen, J. Lu, and K.-K. Ma, Eds. Cham: Springer International Publishing, 2017, pp. 34–45.

[15] T. Whelan, M. Kaess, M. Fallon, H. Johannsson, J. Leonard, and J. McDonald, "Kintinuous: Spatially extended kinectfusion," in *Proceedings of Robotics: Science and Systems (Workshop on RGB-D: Advanced Reasoning with Depth Cameras)*, 2012.

[16] T. Whelan, S. Leutenegger, R. S. Moreno, B. Glocker, and A. Davison, "ElasticFusion: Dense SLAM without a pose graph," in *Proceedings of Robotics: Science and Systems*, 2015.

[17] T. Whelan, R. F. Salas-Moreno, B. Glocker, A. J. Davison, and S. Leutenegger, "ElasticFusion: Real-time dense SLAM and light source estimation," *International Journal of Robotics Research*, vol. 35, no. 14, pp. 1697–1716, 2016.

[18] Q.-Y. Zhou and V. Koltun, "Color map optimization for 3D reconstruction with consumer depth cameras," *ACM Transactions on Graphics*, vol. 33, no. 4, pp. 155:1–155:10, 2014.