RGB-D Mapping: Using Depth Cameras for Dense 3D Modeling of Indoor Environments

Peter Henry^1, Michael Krainin^1, Evan Herbst^1, Xiaofeng Ren^2, Dieter Fox^{1,2}

Abstract RGB-D cameras are novel sensing systems that capture RGB images along with per-pixel depth information. In this paper we investigate how such cameras can be used in the context of robotics, specifically for building dense 3D maps of indoor environments. Such maps have applications in robot navigation, manipulation, semantic mapping, and telepresence. We present RGB-D Mapping, a full 3D mapping system that utilizes a novel joint optimization algorithm combining visual features and shape-based alignment. Visual and depth information are also combined for view-based loop closure detection, followed by pose optimization to achieve globally consistent maps. We evaluate RGB-D Mapping on two large indoor environments, and show that it effectively combines the visual and shape information available from RGB-D cameras.

1 Introduction

Building rich 3D maps of environments is an important task for mobile robotics, with applications in navigation, manipulation, semantic mapping, and telepresence. Most 3D mapping systems contain three main components: first, the spatial alignment of consecutive data frames; second, the detection of loop closures; third, the globally consistent alignment of the complete data sequence. While 3D point clouds are extremely well suited for frame-to-frame alignment and for dense 3D reconstruction, they ignore valuable information contained in images. Color cameras, on the other hand, capture rich visual information and are becoming more and more the sensor of choice for loop closure detection [21, 16, 30]. However, it is extremely hard to extract dense depth from camera data alone, especially in indoor environments with very dark or sparsely textured areas.

^{1}University of Washington, Department of Computer Science & Engineering, Seattle, WA
^{2}Intel Labs Seattle, Seattle, WA

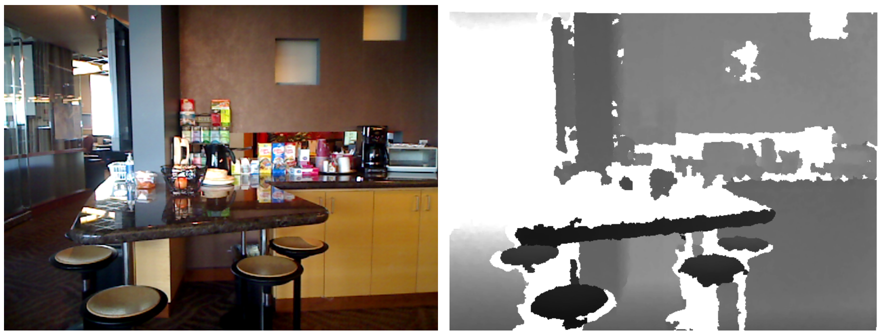

Fig. 1 (left) RGB image and (right) depth information captured by an RGB-D camera. Recent systems can capture images at a resolution of up to 640x480 pixels at 30 frames per second. White pixels in the right image have no depth value, mostly due to occlusion, max distance, relative surface angle, or surface material.

RGB-D cameras are sensing systems that capture RGB images along with per-pixel depth information. RGB-D cameras rely on either active stereo [15, 25] or time-of-flight sensing [3, 12] to generate depth estimates at a large number of pixels. While sensor systems with these capabilities have been custom-built for years, only now are they being packaged in form factors that make them attractive for research outside specialized computer vision groups. In fact, the key drivers for the most recent RGB-D camera systems are computer gaming and home entertainment applications [25].

RGB-D cameras allow the capture of reasonably accurate mid-resolution depth and appearance information at high data rates. In our work we use a camera developed by PrimeSense [25], which captures 640x480 registered image and depth points at 30 frames per second. This camera is equivalent to the visual sensors in the recently available Microsoft Kinect [20]. Fig. 1 shows an example frame observed with this RGB-D camera. As can be seen, the sensor provides dense depth estimates. However, RGB-D cameras have some important drawbacks with respect to 3D mapping: they provide depth only up to a limited distance (typically less than 5m), their depth estimates are very noisy and their field of view ($\sim 60^\circ$) is far more constrained than that of the specialized cameras and laser scanners commonly used for 3D mapping ($\sim 180^\circ$).

In this paper we introduce RGB-D Mapping, a framework for using RGB-D cameras to generate dense 3D models of indoor environments. RGB-D Mapping exploits the integration of shape and appearance information provided by these systems. Alignment between frames is computed by jointly optimizing over both appearance and shape matching. Our approach detects loop closures by matching data frames against a subset of previously collected frames. To generate globally consistent alignments we use TORO, an optimization tool developed for SLAM [9]. The overall system can accurately align and map large indoor environments in near-real time and is capable of handling situations such as featureless corridors and completely dark rooms.

RGB-D Mapping maintains and updates a global model using small planar colored surface patches called *surfels* [23]. This representation enables the approach to efficiently reason about occlusions, to estimate the appropriate color extracted for each part of the environment, and to provide good visualizations of the resulting model. Furthermore, surfels automatically adapt the resolution of the representation to the resolution of data available for each patch.

After discussing related work, we introduce RGB-D Mapping in Section 3. Experiments are presented in Section 4, followed by a discussion.

2 Related Work

The robotics and computer vision communities have developed many techniques for 3D mapping using range scans [31, 32, 19, 21], stereo cameras [1, 16], monocular cameras [5], and even unsorted collections of photos [30, 7]. Most mapping systems require the spatial alignment of consecutive data frames, the detection of loop closures, and the globally consistent alignment of all data frames.

The solution to the frame alignment problem strongly depends on the data being used. For 3D laser data, the iterated closest point (ICP) algorithm and variants thereof are popular techniques [2, 31, 19, 27]. The ICP algorithm iterates between associating each point in one time frame to the closest point in the other frame and computing the rigid transformation that minimizes distance between the point pairs. The robustness of ICP in 3D has been improved by, e.g., incorporating point-to-plane associations or point reflectance values [4, 28, 19].

Passive stereo systems can extract depth information for only a subset of feature points in each stereo pair. These feature points can then be aligned over consecutive frames using an optimization similar to a single iteration of ICP, with the additional advantage that appearance information can be used to solve the data association problem more robustly, typically via RANSAC [16, 1]. Monocular SLAM and mapping based on unsorted image sets are similar to stereo SLAM in that sparse features are extracted from images to solve the correspondence problem. Projective geometry is used to define the spatial relationship between features [22, 5, 30], a much harder problem to solve than correspondence in ICP.

For the loop closure problem, most recent approaches to 3D mapping rely on fast image matching techniques [30, 5, 16, 21]. Once a loop closure is detected, the new correspondence between data frames can be used as an additional constraint in the graph describing the spatial relationship between frames. Optimization of this *pose graph* results in a globally aligned set of frames [9].

While RGB-D Mapping follows the overall structure of recent 3D mapping techniques, it differs from existing approaches in the way it performs frame-to-frame matching. While pure laser-based ICP is extremely robust for the 3D point clouds collected by 3D laser scanning systems such as panning SICK scanners or 3D Velodyne scanners [21, 28], RGB-D cameras provide depth and color information for a small field of view (60° in contrast to 180°) and with less depth precision ($\approx$3cm at

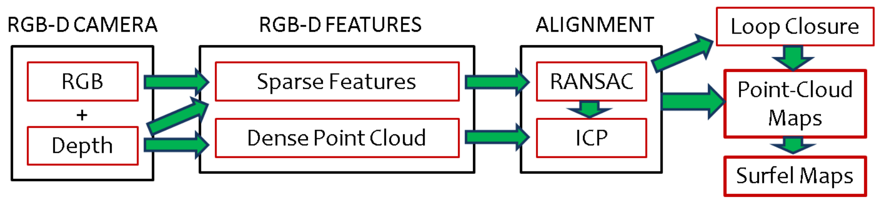

Fig. 2 Overview of RGB-D Mapping. The algorithm uses both sparse visual features and dense point clouds for frame-to-frame alignment and loop closure detection. The surfel representation is updated incrementally.

3m depth) [25]. The limited field of view can cause problems due to a lack of spatial structure needed to constrain ICP alignments.

There has been relatively little attention devoted to the problem of combining shape and visual information for scan alignment. Ramos and colleagues introduce CRF-matching [26], which uses conditional random fields and adds visual information into the matching of 2D laser scans. While this approach provides excellent results, it is computationally expensive and does not scale to large 3D clouds. May and colleagues [19] use laser reflectance values to improve ICP but do not take full advantage of the improved data association provided by visual features. In contrast, we use the visual channels to locate point features, constrain their correspondences, and incorporate them into scan matching.

A common addition to ICP is to augment each point in the two point clouds with additional attributes. The correspondence selection step acts in this higher-dimensional space. This approach has been applied to point color [13], geometric descriptors [29], and point-wise reflectance values [19]. In comparison, our algorithm uses rich visual features along with RANSAC verification to add fixed data associations into the ICP optimization. Additionally, the RANSAC associations act as an initialization for ICP, which is a local optimizer.

Our objective is not only alignment and registration, but also building 3D models with both shape and appearance information. In one recent work, Kim et al [14] used a set of time-of-flight cameras in a fixed calibrated configuration and with no temporal alignment of sensor streams. In contrast, we use a single freely moving camera to build dense models for large indoor environments. In the vision community, there has been a large amount of work on dense reconstruction from videos (e.g. [24]) and photos (e.g. [6, 8]), mostly on objects or outdoor scenes. One interesting line of work [7] attacks the harder problem of indoor reconstruction, using a Manhattan-world assumption to fit simple geometric models for visualization purposes. Such approaches are computationally demanding and not very robust in feature-sparse environments.

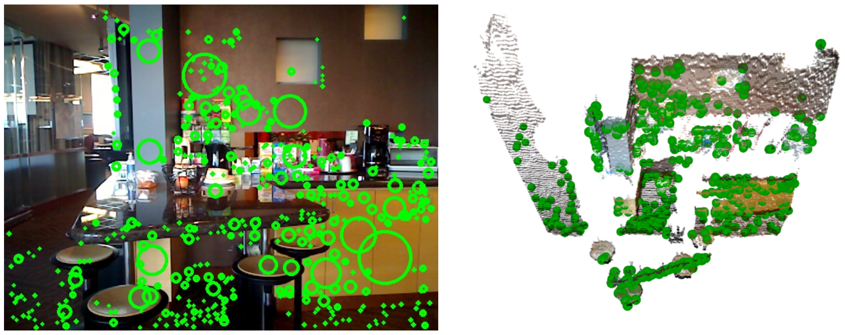

Fig. 3 Example frame for RGB-D frame alignment. Left, the locations of SIFT features in the image. Right, the same SIFT features shown in their position in the point cloud.

3 RGB-D Mapping

This section describes the different components of RGB-D Mapping. A flow chart of the overall system is shown in Fig. 2.

To align the current frame to the previous frame, the alignment step uses RGBD-ICP, our enhanced ICP algorithm that takes advantage of the combination of RGB and depth information. After this alignment step, the new frame is added to the dense 3D model. This step also updates the surfels used for visualization and occlusion reasoning. A parallel loop closure detection thread uses the sparse feature points to match the current frame against previous observations, taking spatial constraints into account. If a loop closure is detected, a constraint is added to the pose graph and a global alignment process is triggered.

3.1 RGBD-ICP

In the *Iterative Closest Point* (ICP) algorithm [2], points in a source cloud $\mathbf{P}_s$ are matched with their nearest neighboring points in a target cloud $\mathbf{P}_t$ and a rigid transformation is found by minimizing the n-D error between associated points. This transformation may change the nearest neighbors for points in $\mathbf{P}_s$, so the two steps of association and optimization are alternated until convergence. ICP has been shown to be effective when the two clouds are already nearly aligned. Otherwise, the unknown data association between $\mathbf{P}_s$ and $\mathbf{P}_t$ can lead to convergence at an incorrect local minimum.

Alignment of images, by contrast, is typically done using sparse feature-point matching. A key advantage of visual features is that they can provide alignments without requiring initialization. One widely used feature detector and descriptor is the Scale Invariant Feature Transform (SIFT)[18]. Though feature descriptors are very distinctive, they must be matched heuristically and there can be false matches selected. The RANSAC algorithm is often used to determine a subset of feature pairs

```
RGBD-ICP ($\mathbf{P}_s, \mathbf{P}_t$):
1: $F$ = Extract_RGB_point_features($\mathbf{P}_s$)
2: $F_{\text{target}}$ = Extract_RGB_point_features($\mathbf{P}_t$)
3: ($\mathbf{t}^*$, $A_f$) = Perform_RANSAC_Alignment( $F$, $F_{\text{target}}$)
4: repeat
5:     $A_d$ = Compute_Closest_Points( $\mathbf{t}^*$, $\mathbf{P}_s$, $\mathbf{P}_t$)
6:     $\mathbf{t}^* = \text{argmin}_{\mathbf{t}}\, \alpha \left( \frac{1}{|A_f|} \sum_{i \in A_f} w_i \left| \mathbf{t}(f_s^i) - f_t^i \right|^2 \right) + (1-\alpha) \left( \frac{1}{|A_d|} \sum_{j \in A_d} w_j \left| (\mathbf{t}(p_s^j) - p_t^j) \cdot n_t^j \right|^2 \right)$
7: until (Error_Change($\mathbf{t}^*$) $\leq \theta$) or (maxIter reached)
8: return $\mathbf{t}^*$
```

Table 1 RGBD-ICP algorithm for matching two RGB-D frames.

corresponding to a consistent rigid transformation. However, in 2D this problem is not fully constrained due to the scale indeterminacy.

Since we have RGB-D frames, we can fuse these two approaches to exploit the advantages of each. The RGBD-ICP algorithm is described in Table 1. It takes as input a source RGB-D frame, $\mathbf{P}_s$, and a target frame, $\mathbf{P}_t$. Steps 1 and 2 extract sparse visual features from the two frames and associate them with their corresponding depth values to generate feature points in 3D. These steps can be implemented with arbitrary visual features (in our implementation, we compute SIFT features using SIFTGPU [33]). Step 3 uses RANSAC to find the best rigid transformation, $\mathbf{t}^*$, between these feature sets. *Perform_RANSAC_Alignment* does this by first finding matching features between the two frames. It then repeatedly samples three pairs of feature points, determines the optimal transformation for this sample, and counts the number of inliers among the remaining 3D feature points. For the sample resulting in the maximum inlier count, the function then efficiently computes a more accurate transformation including all inlier points using the technique of [11]. The function also returns a set of associations A_f containing the feature pairs that generated the best transformation.

Steps 4 through 7 perform the main ICP loop. Step 5 determines the associations A_d between the points in the dense point cloud. This is done by transforming the 3D points in the source cloud, $\mathbf{P}_s$, using the current transformation $\mathbf{t}$. In the first iteration, $\mathbf{t}$ is initialized by the visual RANSAC transformation, which allows RGBD-ICP to match frames without any knowledge of their relative pose (if enough visual features are present). For each point in $\mathbf{P}_s$, Step 5 then determines the nearest point in the target cloud $\mathbf{P}_t$. While it is possible to compute associations between points based on a combination of Euclidean distance, color difference, and shape difference, we found Euclidean distance along with a fast kd-tree search to be sufficient in most cases. Step 6 minimizes the alignment error of both the visual feature associations and the dense point associations. The first part of the error function measures average distances for the visually associated feature points, and the second part computes a similar error term for the dense point associations. For the dense points we employ a *point-to-plane* error term that minimizes the distance error along each target point's normal. These normals, $\{n_t^j\}$, are computed efficiently by principal component analysis over a small neighborhood of each target point. Point-to-plane ICP has been shown to generate more accurate alignments than point-to-point ICP due to an improved interpolation between points [28]. Finally, the two components are weighted using a factor α. Since the point-to-plane error metric has no known closed-form solution, and thus requires the use of a nonlinear optimizer, RGBD-ICP performs the minimization using Levenberg-Marquardt.

The loop exits after the error no longer decreases significantly or a maximum number of iterations is reached. Otherwise, the dense data associations are recomputed using the most recent transformation. Note that feature point data associations are *not* recomputed after the RANSAC procedure. This avoids that the dense ICP components might cause the point clouds to drift apart, which can happen in underconstrained cases such as large flat walls.

We find that downsampling the source and target clouds given to ICP by a factor of 4 to 10 gives the best compromise between matching speed and accuracy.

If RANSAC fails to find a large number of inliers, we initialize the ICP transformation using a constant-velocity motion model: assume the motion between frames n and $n+1$ is similar to that between frames $n-1$ and n.

3.2 Loop Closure Detection and Global Optimization

Alignment between successive frames is a good method for tracking the camera position over moderate distances. However, errors in alignment between a particular pair of frames, and noise and quantization in depth values, cause the estimation of camera position to drift over time, leading to inaccuracies in the map. This is most noticeable when the camera follows a long path, eventually returning to a location previously visited. The cumulative error in frame alignment results in a map that has two representations of the same region in different locations. This is known as the *loop closure problem*, and our solution to it has two parts. First, loop closure *detection* is needed to recognize when the camera has returned to a previously visited location. Second, the map must be *corrected* to merge duplicate regions. Our overall strategy is to represent constraints between frames with a graph structure, with edges between frames corresponding to geometric constraints. The relative transformations from the alignment of sequential frames give us some constraints, so without any loop closure, the graph consists of a linear chain. Loop closures are represented as constraints between frames that are not temporally adjacent.

To keep the graph relatively sparse we define *keyframes*, which are a subset of the aligned frames. We determine keyframes based on visual overlap, adapting the density of keyframes to camera motion and local appearance. After we align a frame F, we reuse the SIFT features to find a rigid transformation with the most recent keyframe, using the same RANSAC procedure defined for frame-to-frame alignment. As long as the number of RANSAC inliers is above a threshold, we do not need to add F as a keyframe. As the camera continues to move, its view contains progressively fewer 3D feature point matches with the previous keyframe. The first frame that fails to match against the previous keyframe becomes the next keyframe. Because we never remove pose-graph edges, contiguous keyframes are always constrained in the graph.

Each time we create a new keyframe we attempt to detect a loop closure with each previous keyframe. A closure is detected if enough geometrically consistent 3D feature point matches are recovered by RANSAC, and if so, we add an edge to the graph representing this newly discovered constraint. For this stage we modify RGBD-ICP slightly to return FAIL if no RANSAC match is found with sufficient inliers, so that the same algorithm that performs frame-to-frame matching also performs loop closure detection and initializes pose-graph edges. We only perform the RANSAC check with keyframes that are within a small distance of our current position estimate.

In order to minimize the conflict between sequential constraints and loop closure constraints we employ TORO [9, 10], a system for efficiently minimizing the error in such graphs where vertices are parameterized by translation and rotation components and edges represent constraints between the parameters with associated covariance matrices. TORO uses stochastic gradient descent to maximize the likelihood of the vertex parameters subject to the constraints. We run TORO to convergence each time loop closure is detected, initializing it with the output of the previous TORO run and the contiguous-frame constraints added since then.

3.3 Surfel Representation

Considering that each frame from the RGB-D camera gives us roughly 250,000 points, it is necessary to create a more concise representation of the map. One option is to downsample the clouds. However, it is more appealing to incorporate all the information from each frame into a concise representation for visualization. One method for doing this is *surfels* [23, 17]. A surfel consists of a location, a surface orientation, a patch size and a color. As more point clouds are added to the surfel representation, we follow rules similar to those of [23] and [17] for updating, adding, and removing surfels. Surfels store a measure of confidence, which is increased through being seen from multiple angles over time. Surfels with low confidence are removed from the representation. Because surfels have a notion of size (obtained initially from the depth of the original point in the RGB-D frame), we can reason about occlusion, so if an existing surfel is seen through too often, it can be removed.

Based on the estimated normals within each RGB-D frame, the surfel normal directions can be updated as well. We can also wait to add surfels until their normal is pointed (within some angle) towards the camera position, which leads to a more accurate recovery of the surfel size. The color of a surfel is determined from the RGB-D frame most aligned with the normal direction.

4 Experiments

We performed several experiments to evaluate different aspects of RGB-D Mapping. Specifically, we demonstrate the ability of our system to build consistent maps of large scale indoor environments, we show that our joint ICP algorithm improves accuracy of frame to frame alignment, and we illustrate the advantageous properties of the surfel representation.

4.1 Large-Scale Environments

We tested RGB-D Mapping in two indoor environments: the Intel Labs Seattle offices and the University of Washington computer science building. During mapping, the camera was carried by a person, and generally pointed in the direction of travel. The left panels in Fig. 4 show 3D maps built for large loops in these environments. The loops in the upper and lower panel are 71 and 114 meters long, respectively. To assess the consistency of these maps we overlaid our 3D maps onto 2D layouts generated by different means. The right panels in Fig. 4 show overlays of our maps onto an Intel lab map built with a SICK laser scanner using a standard SLAM approach and an architectural floor plan of the Allen Center. For clarity, most floor and ceiling points were removed from our 3D maps; the remaining 3D points are shown in red. RGB-D Mapping produces very consistent maps.

4.2 Benefits of RGBD-ICP

To more thoroughly evaluate the benefits of RGBD-ICP, we determined ground-truth poses in the Intel lab loop and also used a Barrett WAM manipulator to provide ground-truth poses in a small-scale experiment.

In the first experiment, we placed 16 markers around the Intel loop and measured the true distance between consecutive markers. A challenging dataset was then collected at night with the lights turned off in a hallway section of the building. The camera was carried sequentially between the marker locations and placed carefully on a tripod at each marker location, returning finally to the starting marker. In this way we obtained 16 measurements of sequential frame alignment error over small sections of the map by measuring the difference between real-world distance and between-marker distance determined by successive frame-to-frame alignments. Table 2 summarizes the results for different α values, which weigh the RGB and dense point contributions in RGBD-ICP, resulting in pure SIFT alignment or ICP in the extreme cases.

α weight	0.0 (ICP)	0.2	0.5	0.8	1.0 (SIFT)
Mean error [m]	0.22	0.19	0.16	0.18	1.29

Table 2 Sequential alignment comparison on large scale marker sequence.

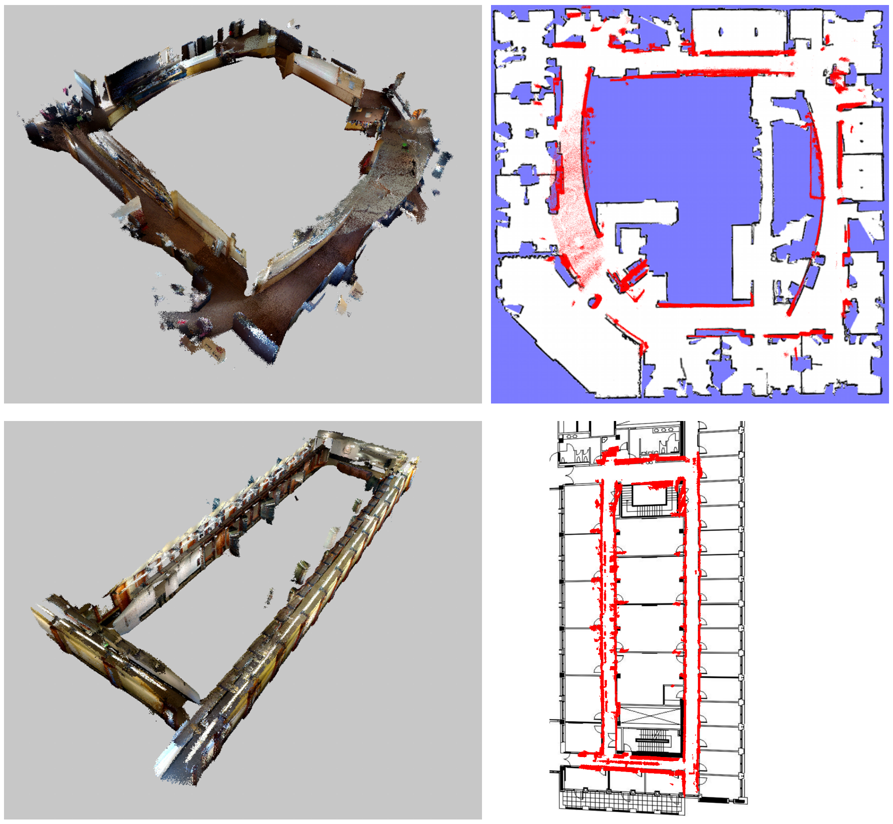

Fig. 4 (left) 3D maps generated by RGB-D Mapping for large loops in the Intel lab (upper) and the Allen Center (lower). (right) Accuracy: Maps (red) overlaid on a 2D laser scan map of the Intel Lab and on a floor plan of the Allen Center. For clarity, most floor and ceiling points were removed from the 3D maps.

As can be seen, by combining RGB and depth into a joint ICP algorithm, RGBD-ICP achieves accuracy that is superior to using either component individually. The results are robust to different α weightings between the SIFT and dense point components. The larger error of SIFT alone is due to failed alignments in the dark section of the environment. However, when incorporated into RGBD-ICP, it provides additional information that improves the alignment accuracy. Though on this sequence ICP alone performs in nearly as well as RGBD-ICP it should be pointed out that ICP requires more iterations when not initialized with RANSAC, and that frames containing a single textured flat wall provide no constraints to ICP.

In a second experiment, we collected a ground-truth dataset with the camera mounted on a moving Barrett WAM arm, which has accurate motion estimation. Errors are the average distance between the camera motion determined by the arm's odometry and the poses estimated by the different algorithms. Table 3 summarizes

Method	Avg. Translational Error (m)	Avg. Angular Error (deg)
SIFT	.168	4.5
ICP	.144	3.9
RGBD-ICP ($\alpha = 0.5$)	**.123**	**3.3**
RGBD-ICP + TORO	.069	2.8

Table 3 Sequential alignment comparison on WAM-arm sequence.

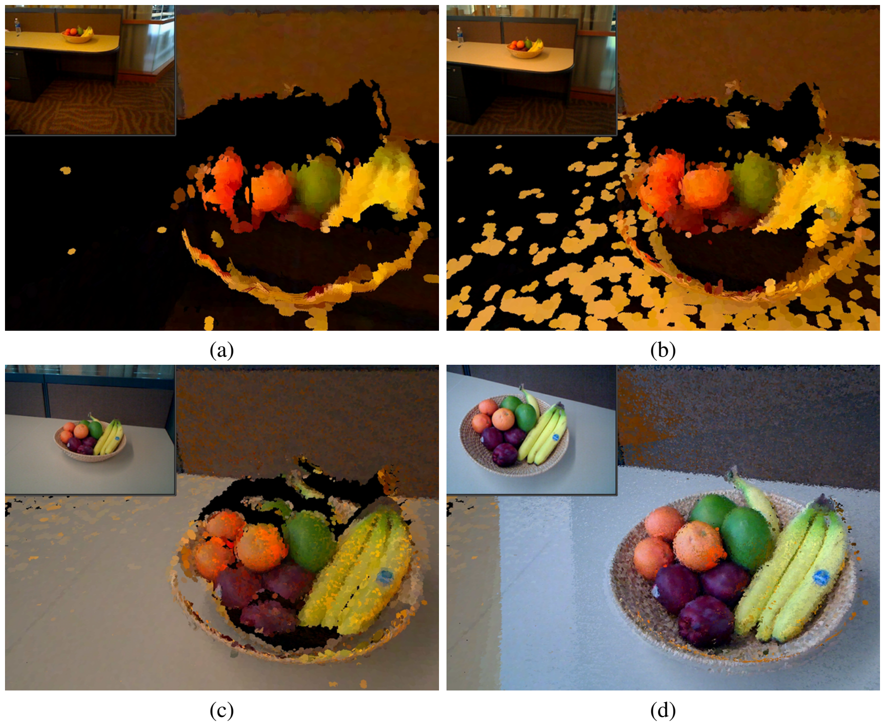

Fig. 5 Surfel updates: a) The initial surfels from the first frame. b,c) As the camera moves closer, more surfels are added and existing ones are refined. d) The completed surfel model from 95 aligned RGB-D images.

the results achieved by the different alignment algorithms and the additional optimization after loop closure.

This experiment confirms our findings on the large-scale marker sequence, where RGBD-ICP outperforms the individual components. Furthermore, it shows that loop closure followed by global optimization via TORO further improves results.

4.3 Surfel Representation

To demonstrate the value of the surfel representation of aligned point clouds, we first point out that the surfel representation shown in Fig. 5 is formed from a sequence of

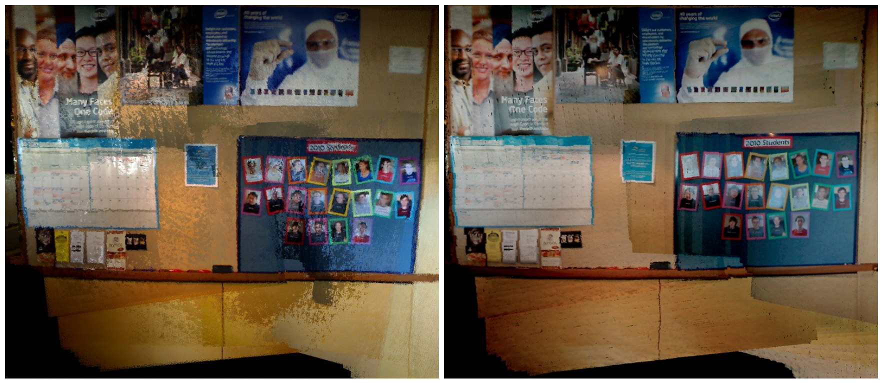

Fig. 6 Surfel representation: (left) raw point clouds (56.6 million points); (right) corresponding surfel representation (2.2 million surfels).

95 aligned frames, each containing roughly 250,000 RGB-D points. Simply merging the point clouds would result in a representation containing roughly 23,750,000 points. Furthermore, these points duplicate much information, and are not consistent with respect to color and pose. In contrast, the surfel representation consists of only 730,000 surfels. This amounts to a reduction in size by a factor of 32. The color for each is selected from the frame most in line with the surfel normal. The final surfel representation shown in Fig. 5(d) has concisely and faithfully combined the information from all input frames.

In a second demonstration of surfels from an indoor map, in Fig. 6, the representation on the left shows all points from each cloud, totaling 56.6 million points. Our surfel representation on the right consists of only 2.2 million surfels, a factor of 26 reduction. In addition to being a more efficient representation, the surfel model exhibits better color and geometric consistency.

We have created software that allows surfel maps to be navigated in real time, including a stereoscopic 3D mode which create an immersive experience we believe is well suited for telepresence and augmented reality applications.

Videos of our results can be found at
`http://www.cs.washington.edu/robotics/projects/rgbd-mapping/`.

4.4 Timing and Lessons Learned

Per frame, our current implementation extracts features in 150 ms, runs RANSAC in 80 ms, and runs dense ICP in an average of 500 ms. Loop closure checks use this same alignment procedure, making its running time largely dependent on the other number of keyframes in the nearby vicinity of the map. If ICP is *not* initialized with the RANSAC transform, it takes an average of about 1 second per frame, showing that the RANSAC initialization significantly helps the overall speed of the algorithm in addition to providing robustness. Surfel generation takes roughly 6 seconds per frame, but this is currently the least optimized component of our system.

RANSAC is the faster and more reliable alignment component when considered individually. However, there are situations where it fails and the joint optimization is required. For some frames, many detected visual features are out of range of the depth sensor, so those features have no associated 3D points and do not participate in the RANSAC procedure. Also, when the majority of the features lie in a small region of the image, they do not provide very strong constraints on the motion. For example, in badly lit halls it is common to see features only on one side wall. It is in situations such as these that RGBD-ICP provides notably better alignment.

5 Conclusion

Building accurate, dense models of indoor environments has many applications in robotics, telepresence, gaming, and augmented reality. Limited lighting, lack of distinctive features, repetitive structures, and the demand for rich detail are inherent to indoor environments, and handling these issues has proved a challenging task for both robotics and computer vision communities. Laser scanning approaches are typically expensive and slow and need additional registration to add appearance information (visual details). Vision-only approaches to dense 3D reconstruction often require a prohibitive amount of computation, suffer from lack of robustness, and cannot yet provide dense, accurate 3D models.

We investigate how potentially inexpensive depth cameras developed mainly for gaming and entertainment applications can be used for building dense 3D maps of indoor environments. The key insights of this investigation are, first, that existing frame matching techniques are not sufficient to provide robust visual odometry with these cameras; second, that a tight integration of depth and color information can yield robust frame matching and loop closure detection; third, that building on best practices in SLAM and computer graphics makes it possible to build and visualize accurate and extremely rich 3D maps with such cameras; and, fourth, that it will be feasible to build complete robot navigation and interaction systems solely based on inexpensive depth cameras.

We introduce RGB-D Mapping, a framework that can generate dense 3D maps of indoor environments despite the limited depth precision and field of view provided by RGB-D cameras. At the core of this framework is RGBD-ICP, a novel ICP variant that takes advantage of the richness of information contained in RGB-D data. RGB-D Mapping also incorporates a surfel representation to enable occlusion reasoning and visualization.

Given that RGB-D cameras will soon be available to the public at a low price (we believe less than 100 dollars), an RGB-D-based modeling system will potentially have a huge impact on everyday life, allowing people to build 3D models of arbitrary indoor environments. Furthermore, our results indicate that RGB-D cameras could be used to build robust robotic mapping, navigation, and interaction systems. Along with the potential decrease in cost of the resulting navigation platform, the application of RGB-D cameras might be an important step to enable the development of useful, affordable robot platforms.

Despite these encouraging results, our system has several shortcomings that deserve future effort. Our current implementation of RGB-D Mapping is not real-time, but we believe that an efficient implementation taking advantage of modern GPU hardware can certainly achieve the speedup needed to operate online. The global alignment process of RGB-D Mapping is still limited. Instead of optimizing over camera poses only, a joint optimization over camera poses and 3D points could result in even more consistent reconstructions. The computer graphics community has developed extremely sophisticated visualization techniques, and incorporating these into RGB-D mapping could improve the visual quality of the 3D maps. Another interesting avenue for research is the extraction of object representations from the rich information contained in dense 3D maps. Other areas for future research include the development of exploration techniques for building complete 3D maps and the extension to dynamic environments.

We would like to thank Louis LeGrand and Brian Mayton for their support with the PrimeSense camera, and Marvin Cheng for his work on visualization. This work was funded in part by an Intel grant, by ONR MURI grants number N00014-07-1-0749 and N00014-09-1-1052, and by the NSF under contract number IIS-0812671. Part of this work was also conducted through collaborative participation in the Robotics Consortium sponsored by the U.S Army Research Laboratory under the Collaborative Technology Alliance Program, Cooperative Agreement W911NF-10-2-0016.

References

1. A. Akbarzadeh, J. M. Frahm, P. Mordohai, B. Clipp, C. Engels, D. Gallup, P. Merrell, M. Phelps, S. Sinha, B. Talton, L. Wang, Q. Yang, H. Stewenius, R. Yang, G. Welch, H. Towles, D. Nistér, and M. Pollefeys. Towards urban 3D reconstruction from video. In *Proc. of the Third International Symposium on 3D Data Processing, Visualization and Transmission (3DPVT)*, 2006.
2. P. J. Besl and N. D. McKay. A method for registration of 3-d shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 14(2), 1992.
3. Canesta. `http://www.canesta.com/`.
4. Yang Chen and Gérard Medioni. Object modeling by registration of multiple range images. *Image Vision Comput.*, 10(3):145–155, 1992.
5. L. Clemente, A. Davison, I. Reid, J. Neira, and J. Tardós. Mapping large loops with a single hand-held camera. In *Proc. of Robotics: Science and Systems (RSS)*, 2007.
6. P. Debevec, C. J. Taylor, and Jitendra Malik. Modeling and rendering architecture from photographs: A hybrid geometryand image-based approach. In *SIGGRAPH*, 1996.
7. Y. Furukawa, B. Curless, S. Seitz, and R. Szeliski. Reconstructing building interiors from images. In *Proc. of the International Conference on Computer Vision (ICCV)*, 2009.
8. Y. Furukawa and J. Ponce. Patch-based multi-view stereo software (PMVS): `http://grail.cs.washington.edu/software/pmvs/`.
9. G. Grisetti, S. Grzonka, C. Stachniss, P. Pfaff, and W. Burgard. Estimation of accurate maximum likelihood maps in 3D. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2007.

10. G. Grisetti, C. Stachniss, S. Grzonka, and W. Burgard. A tree parameterization for efficiently computing maximum likelihood maps using gradient descent. In *Proc. of Robotics: Science and Systems (RSS)*, 2007.
11. B. K. P. Horn. Closed-form solution of absolute orientation using unit quaternions. *J. Opt. Soc. Am. A*, 4(4):629–642, 1987.
12. Mesa Imaging. `http://www.mesa-imaging.ch/`.
13. A. Johnson and S. B. Kang. Registration and integration of textured 3-d data. In *International Conference on Recent Advances in 3-D Digital Imaging and Modeling (3DIM '97)*, pages 234 – 241, May 1997.
14. Y. M. Kim, C. Theobalt, J. Diebel, J. Kosecka, B. Micusik, and S. Thrun. Multi-view image and ToF sensor fusion for dense 3D reconstruction. In *Workshop on 3-D Digital Imaging and Modeling (3DIM)*, 2009.
15. K. Konolige. Projected texture stereo. In *Proc. of the IEEE International Conference on Robotics & Automation (ICRA)*, 2010.
16. K. Konolige and M. Agrawal. FrameSLAM: From bundle adjustment to real-time visual mapping. *IEEE Transactions on Robotics*, 25(5), 2008.
17. M. Krainin, P. Henry, X. Ren, and D. Fox. Manipulator and object tracking for in hand 3D object modeling. Technical Report UW-CSE-10-09-01, University of Washington, 2010. `http://www.cs.washington.edu/ai/Mobile_Robotics/projects/hand_tracking/`.
18. D. Lowe. Discriminative image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2), 2004.
19. S. May, D. Dröschel, D. Holz, E. Fuchs, S. Malis, A. Nüchter, and J. Hertzberg. Three-dimensional mapping with time-of-flight cameras. *Journal of Field Robotics (JFR)*, 26(11-12), 2009.
20. Microsoft. `http://www.xbox.com/en-US/kinect`, 2010.
21. P. Newman, G. Sibley, M. Smith, M. Cummins, A. Harrison, C. Mei, I. Posner, R. Shade, D. Schröter, L. Murphy, W. Churchill, D. Cole, and I. Reid. Navigating, recognising and describing urban spaces with vision and laser. *International Journal of Robotics Research (IJRR)*, 28(11-12), 2009.
22. D. Nister. An efficient solution to the five-point relative pose problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 26(6):756–77, 2004.
23. H. Pfister, M. Zwicker, J. van Baar, and M. Gross. Surfels: Surface elements as rendering primitives. In *ACM Transactions on Graphics (Proc. of SIGGRAPH)*, 2000.
24. M. Pollefeys, D. Nister, J.-M. Frahm, A. Akbarzadeh, P. Mordohai, B. Clipp, C. Engels, D. Gallup, S.-J. Kim, P. Merrell, C. Salmi, S. Sinha, B. Talton, L. Wang, Q. Yang, H. Stewenius, R. Yang, G. Welch, and H. Towles. Detailed real-time urban 3D reconstruction from video. *International Journal of Computer Vision*, 72(2):143–67, 2008.
25. PrimeSense. `http://www.primesense.com/`.
26. F. Ramos, D. Fox, and H. Durrant-Whyte. CRF-matching: Conditional random fields for feature-based scan matching. In *Proc. of Robotics: Science and Systems (RSS)*, 2007.
27. S. Rusinkiewicz and M. Levoy. Efficient variants of the ICP algorithm. In *Third International Conference on 3D Digital Imaging and Modeling*, 2001.
28. A. Segal, D. Haehnel, and S. Thrun. Generalized-ICP. In *Proc. of Robotics: Science and Systems (RSS)*, 2009.
29. G. C. Sharp, S. W. Lee, and D. K. Wehe. ICP registration using invariant features. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 24(1):90–102, 2002.
30. N. Snavely, S. Seitz, and R. Szeliski. Photo tourism: Exploring photo collections in 3D. In *ACM Transactions on Graphics (Proc. of SIGGRAPH)*, 2006.
31. S. Thrun, W. Burgard, and D. Fox. A real-time algorithm for mobile robot mapping with applications to multi-robot and 3D mapping. In *Proc. of the IEEE International Conference on Robotics & Automation (ICRA)*, 2000.
32. R. Triebel and W. Burgard. Improving simultaneous mapping and localization in 3D using global constraints. In *Proc. of the National Conference on Artificial Intelligence (AAAI)*, 2005.
33. C. Wu. SiftGPU: A GPU implementation of scale invariant feature transform (SIFT). `http://cs.unc.edu/~ccwu/siftgpu`, 2007.