The International Journal of Robotics Research
31(5) 647–663

DOI: 10.1177/0278364911434148
ijr.sagepub.com

RGB-D mapping: Using Kinect-style depth cameras for dense 3D modeling of indoor environments

Peter Henry[1], Michael Krainin[1], Evan Herbst[1], Xiaofeng Ren[2] and Dieter Fox[1]

Abstract

RGB-D cameras (such as the Microsoft Kinect) 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 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.

Keywords

SLAM, localization, mapping, range sensing, vision, RGB-D, kinect

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; and third, the globally consistent alignment of the complete data sequence. While 3D point clouds are 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 (Snavely et al. 2006; Konolige and Agrawal 2008; Newman et al. 2009). However, it is extremely hard to extract dense depth from camera data alone, especially in indoor environments with very dark or sparsely textured areas.

RGB-D cameras are sensing systems that capture RGB (visual) images along with per-pixel depth information. RGB-D cameras rely on either active stereo[1] (Konolige 2010a) or time-of-flight sensing[2] 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.[1]

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,[3] which captures 640 × 480 registered image and depth points at 30 frames per second. This camera is equivalent to the sensor underlying the commercially available Microsoft Kinect gaming system (Shotton et al. 2011).[4] Figure 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 5 m), their depth estimates are noisy (∼3 cm at

[1]Department of Computer Science and Engineering, University of Washington, Seattle, WA, USA
[2]ISTC-Pervasive Computing, Intel Labs, Seattle, WA, USA

Corresponding author:
Peter Henry, Department of Computer Science and Engineering, University of Washington, AC101 Paul G. Allen Center, Box 352350, 185 Stevens Way, Seattle, WA 98195-2350, USA
Email: peter@cs.washington.edu

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 640 × 480 pixels at 30 frames per second. White pixels in the right image have no depth value, mostly due to occlusion, distance, relative surface angle, or surface material.

3 m depth),[3] and their field of view (∼60°) is far more constrained than that of the specialized cameras and laser scanners commonly used for 3D mapping (∼180°).

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 matches. Visual appearance is incorporated by extracting sparse feature points from the RGB images and matching them via a random sample consensus (RANSAC) (Fischler and Bolles 1981) procedure. The resulting feature matches are then combined with dense *Iterative Closest Point* (ICP) matching to determine the best alignment between the frames. Our approach detects loop closures by matching data frames against a subset of previously collected frames. We show that globally consistent alignments can be achieved either via highly efficient pose graph optimization such as TORO (Lu and Milios 1997; Konolige 2004; Thrun et al. 2005; Grisetti et al. 2007a) or via sparse bundle adjustment (Lourakis and Argyros 2009; Konolige 2010b). 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 builds a global model using small planar colored surface patches called *surfels* (Pfister et al. 2000). This representation enables the approach to estimate the appropriate location, surface orientation, and 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.

This paper is an extension of our previous work presented at the International Symposium on Experimental Robotics (Henry et al. 2010). The major additions are:

- we utilize re-projection error for frame-to-frame alignment RANSAC and demonstrate that it performs better than the Euclidean-space RANSAC used in Henry et al. (2010);
- we use FAST features and Calonder descriptors instead of scale-invariant feature transform (SIFT);
- we improve the efficiency of loop closure detection through place recognition;
- we improve global optimization by using *sparse bundle adjustment* (SBA) and compare it with TORO;
- we show how to incorporate ICP constraints into SBA, a crucial component in featureless areas.

2. Related work

The robotics and computer vision communities have developed many techniques for 3D mapping using range scans (Thrun et al. 2000; Triebel and Burgard 2005; May et al. 2009; Newman et al. 2009), stereo cameras (Nister et al. 2004; Akbarzadeh et al. 2006; Konolige and Agrawal 2008), monocular cameras (Clemente et al. 2007), and even unsorted collections of photos (Snavely et al. 2006; Furukawa et al. 2009). 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 ICP algorithm and variants thereof are popular techniques (Besl and McKay 1992; Thrun et al. 2000; May et al. 2009; Rusinkiewicz and Levoy 2001). 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 incorporating ideas such as point-to-plane associations or point reflectance values (Chen and Medioni 1992; May et al. 2009; Segal et al. 2009).

Passive stereo systems can extract depth information for only a subset of feature points in each stereo pair. The SIFT features of Lowe (2004) are commonly used as well as fast descriptors based on random trees from Calonder et al. (2008), which can be computed for any desired keypoints, such as the FAST keypoints described by Rosten and Drummond (2006). Sparse feature points can then be aligned over consecutive frames minimizing distance between matched points, with the additional advantage that appearance information can be used to solve the data association problem more robustly, typically via RANSAC (Fischler and Bolles 1981) as in Akbarzadeh et al. (2006) and Konolige and Agrawal (2008). Monocular simultaneous localization and mapping (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 (Nister 2004; Snavely et al. 2006; Clemente et al. 2007).

For the loop-closure problem, most recent approaches to 3D mapping rely on fast image matching techniques (Snavely et al. 2006; Clemente et al. 2007; Konolige and Agrawal 2008; Newman et al. 2009). 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 (Grisetti et al. 2007a). More generally, *bundle adjustment* (Triggs et al. 2000) simultaneously optimizes the poses and a map consisting of sparse features. Mature algorithms and libraries have been developed for bundle adjustment (Lourakis and Argyros 2009), with efficient versions exploiting sparse connections between poses (Konolige 2010b).

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 (Newman et al. 2009; Segal et al. 2009), RGB-D cameras provide depth and color information for a small field of view (60° in contrast to 180°) and with less depth precision (≈3 cm at 3 m depth).[3] 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 et al. (2007) use conditional random fields for matching 2D laser scans together with visual information, which is computationally expensive and does not scale to large 3D clouds. Prusak et al. (2008) combine a time-of-flight (ToF) camera with a CCD camera for robot self-localization. May et al. (2009) use laser reflectance values from ToF cameras to improve ICP but do not take full advantage of the improved data association provided by sparse visual features. The related work of Droeschel et al. (2009) uses sparse features (SIFT) only and focuses on 2D motions on a robot platform. Andreasson and Lilienthal (2010) also use SIFT features, with depth interpolated from a laser scanner, to estimate full 6D registration. In contrast to these existing works, we use the visual channels to locate point features, constrain their correspondences, and incorporate them into scan matching to build dense 3D maps.

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 (Johnson and Kang 1997), geometric descriptors (Sharp et al. 2002), and point-wise reflectance values (May et al. 2009). In comparison, our algorithm uses rich visual features along with RANSAC verification to add fixed data associations into the ICP optimization. In addition, the RANSAC associations act as an initialization for ICP, which is a local optimizer.

Our objective is not only registration, but also building 3D models with both shape and appearance information. Cui et al. (2010) combine super-resolution and expectation–maximization (EM)-based non-rigid registration to scan 3D objects with a ToF camera. Strobl et al. (2009) combine a ToF camera with a stereo camera to build 3D object models in real-time. Kim et al. (2009) used a set of time-of-flight cameras in a fixed calibrated configuration and with no temporal alignment of sensor streams. Se and Jasiobedzki (2008) use a stereo camera combined with SIFT features to create 3D models of environments, but make no provision for loop closure or global consistency. Newcombe and Davison (2010) developed an impressive system that does real-time dense 3D reconstruction with a monocular camera, although their system is still limited to small feature-rich scenes. In contrast, we use a single freely moving camera to build dense models at near-real-time for large indoor environments.

In the vision and graphics communities, there has been a large amount of work on dense reconstruction from videos (e.g. Pollefeys et al. 2008) and photos (e.g. Debevec et al. 1996; Furukawa and Ponce 2010),[5] mostly on objects or outdoor scenes. One interesting line of work (Furukawa et al. 2009) attacks the arguably 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.

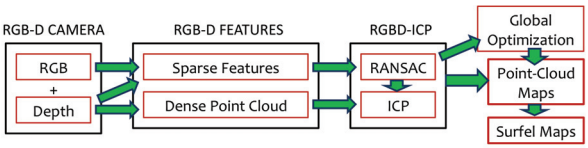


Fig. 2. Overview of RGB-D Mapping. The algorithm uses both sparse visual features and dense point clouds for frame-to-frame alignment and loop-closure detection, which run in parallel threads.

3. RGB-D Mapping

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

To align the current frame to the previous frame, the alignment step uses RGB-D 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. A parallel loop-closure detection thread uses the sparse feature points to match the current frame against previous observations, taking spatial considerations into account. If a loop closure is detected, a constraint is added to the pose graph and a global alignment process is triggered. A surfel map is used to faithfully incorporate input data in an efficient representation once camera poses are determined.

3.1. RGB-D ICP

In the ICP algorithm (Besl and McKay 1992), points in a source cloud P_s are matched with their nearest neighboring points in a target cloud P_t and a rigid transformation is found by minimizing the sum of squared spatial error between associated points. This transformation may change the nearest neighbors for points in 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 P_s and 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. For example, one widely used feature detector and descriptor is SIFT (Lowe 2004). Although feature descriptors are very distinctive, they must be matched heuristically and false matches may be selected. The RANSAC algorithm (Fischler and Bolles 1981) is often used to determine a subset of feature pairs corresponding to a consistent rigid transformation. However, in the monocular case this problem is not fully constrained due to scale indeterminacy.

Algorithm 1: RGB-D ICP algorithm for matching two RGB-D frames.

input : source RGB-D frame P_s, target RGB-D frame P_t, previous transformation $\mathbf{T}_\mathrm{p}$
output: Optimized relative transformation $\mathbf{T}^*$

1 $F_s \leftarrow$ `Extract_RGB_Point_Features` (P_s);
2 $F_t \leftarrow$ `Extract_RGB_Point_Features` (P_t);
3 $(\mathbf{T}^*, A_f) \leftarrow$ `Perform_RANSAC_Alignment` (F_s, F_t);
4 **if** $|A_f| < \gamma$ **then**
5 $\quad \mathbf{T}^* = \mathbf{T}_\mathrm{p}$;
6 $\quad A_f = \emptyset$;
7 **end**
8 **repeat**
9 $\quad A_d \leftarrow$ `Compute_Closest_Points` $(\mathbf{T}^*, P_s, P_t)$;
10 $\quad \mathbf{T}^* \leftarrow$ `Optimize_Alignment` $(\mathbf{T}^*, A_f, A_d)$;
11 **until** *(*`Change` $(\mathbf{T}^*) \leq \theta$*)* **or** *(*Iterations $>$ MaxIterations*)*;
12 **return** $\mathbf{T}^*$;

Since RGB-D cameras provide both color and depth, we can fuse these two approaches to exploit the advantages of each. Briefly, our RGB-D ICP algorithm uses visual features and their associated depth values to obtain an initial alignment, and then jointly optimizes over the sparse feature matches and dense point cloud alignment. The RGB-D ICP algorithm is described in Algorithm 1.

3.1.1. RANSAC RGB-D ICP takes as input a source RGB-D frame, P_s, and a target frame, P_t. It also has access to the previous relative transformation $\mathbf{T}_\mathrm{p}$, which is initialized to the identity. For readability, we use the notation $\mathbf{T}(p)$ to mean the application of rigid transformation $\mathbf{T} \in SE(3)$ to the point p. In other words, if $\mathbf{T}$ consists of rotation matrix $\mathbf{R}$ and translation component t, then $\mathbf{T}(p) = \mathbf{R}p + t$. The final output is the optimized relative transform $\mathbf{T}^*$.

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 the conference version of this paper (Henry et al. 2010), we used SIFT features computed with SIFTGPU (Wu 2007). We have since found that the FAST feature detector (Rosten and Drummond 2006) combined with the Calonder feature descriptor (Calonder et al. 2008) as implemented in OpenCV (Bradski 2000) provide faster and more reliable visual keypoints in our system. See Figure 3 for an example of these features.

The function `Perform_RANSAC_Alignment` in Step 3 uses RANSAC to find the best rigid transformation, $\mathbf{T}^*$, between these two feature sets. The error metric used to find

Fig. 3. Example frame for RGB-D frame alignment. Left: the locations of FAST features in the image. Right: the same features shown in their 3D positions in the point cloud.

the optimal alignment depends on how the RGB-D camera determines depth values. For cameras that use time-of-flight to measure depth,[2] the point-to-point squared distance error often used to align 2D laser scans and 3D point clouds is well suited:

$$\mathbf{T}^* = \underset{\mathbf{T}}{\operatorname{argmin}} \left(\frac{1}{|A_f|} \sum_{i \in A_f} w_i \left| \mathbf{T}(f_s^i) - f_t^i \right|^2 \right). \quad (1)$$

Here, A_f contains the associations between feature points in the two sets, and each term in the summation measures the squared distance between a feature point f_t^i in the target frame and the transformed pose of the associated feature point f_s^i in the source frame. The weight w_i can be used to control the contribution of the particular association to the overall error considering factors such as distance from the sensor, reflectance values in the case of laser scans, or color difference in the case of colored 3D point clouds.

Alternatively, for active stereo RGB-D cameras such as those used in Kinect, a standard stereo-vision-based re-projection error measure is more appropriate:

$$\mathbf{T}^* = \underset{\mathbf{T}}{\operatorname{argmin}} \left(\frac{1}{|A_f|} \sum_{i \in A_f} \left| Proj(\mathbf{T}(f_s^i)) - Proj(f_t^i) \right|^2 \right). \quad (2)$$

Instead of measuring the distance between the feature points in 3D space, this metric measures the error in pixel space when re-projecting associated features into the camera frame. The projection function $Proj : \mathbb{R}^3 \rightarrow \mathbb{R}^3$ provides the projection of feature point $f = (x, y, z) \in \mathbb{R}^3$ from its Euclidean 3D position relative to the camera into the image space of the camera, giving $(u, v, d) \in \mathbb{R}^3$, where u and v are RGB pixel coordinates and d is the disparity, which represents the depth value in pixel space. Just as with a standard passive stereo camera, the depth of a pixel is measured through triangulation. In the case of our RGB-D camera, this is done using the baseline B between the infrared (IR) emitter and IR camera sensor. It therefore makes sense to minimize the depth error in the measurement space of the IR sensor, which is in pixels. Note that this is the same measurement space as the RGB pixel coordinates.[6] Knowing the focal length F of the camera and the center of the image (C_u, C_v) in pixels, we compute $Proj((x, y, z))$ as

$$\begin{aligned} u &= \frac{F}{z} x + C_u \\ v &= \frac{F}{z} y + C_v \\ d &= \frac{F}{z} B \end{aligned} \quad (3)$$

An implicit assumption of (2) is that errors in u, v, and d are independent and equally weighted, which is the assumption made in the SBA implementation of Konolige (2010b) which we use.

To jointly optimize the data association A_f and the transformation $\mathbf{T}^*$ in (1) or (2), `Perform_RANSAC_Alignment` first finds matching features between the two frames. It then repeatedly samples three pairs of feature points and determines the optimal transformation for this sample using the method of Horn (1987). Proposed transformations are evaluated according to the number of inliers among the remaining 3D feature points. When optimizing the distance metric used in (1), inlier correspondences are determined by their 3D distance falling below a threshold. When optimizing the re-projection metric used in (2), inlier correspondences are determined by thresholding pixel distances. For the sample resulting in the maximum inlier count, the function then efficiently computes a more accurate transformation taking all inlier points into account. To minimize (1), we again use Horn's method on all inlier points. (2) is minimized through *two-frame SBA* as opposed to rigid 3D alignment (see Section 3.2.3). The function `Perform_RANSAC_Alignment` also returns the set of associations A_f containing the feature pairs that generated the best transformation. The two-frame SBA refinement may result in a small number of inliers now falling outside the inlier threshold, which are discarded.

In the relatively rare case that the final number of inliers $|A_f|$ is below a minimal number γ, we cannot be confident that any of the associations are truly correct correspondences; they may in fact be coincidentally consistent spurious matches. In this case, we initialize $\mathbf{T}^* = \mathbf{T}_\mathrm{p}$ and clear the untrustworthy associations. This provides an initial estimate to the ICP portion of the algorithm based on a constant-velocity motion model, which assumes that the motion between frames n and $n+1$ is similar to that between frames $n-1$ and n.

In the algorithm introduced by Henry et al. (2010), we used the 3D version of RANSAC for alignment. Later, we show that the re-projection metric yields improved tracking results when used on the PrimeSense active stereo cameras.[3] While we call this technique *re-projection error RANSAC* (RE-RANSAC), we refer to other version (1) as *Euclidean error RANSAC* (EE-RANSAC).

3.1.2. Joint optimization Steps 4 through 7 of RGB-D ICP 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, P_s, using the current transformation $\mathbf{T}$. In the first iteration, $\mathbf{T}$ is initialized by the visual RANSAC transformation (if enough visual features are present). For each point in P_s, Step 5 then determines the nearest point in the target cloud 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 k–d 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. When using EE-RANSAC, the joint error function is

$$\mathbf{T}^* = \underset{\mathbf{T}}{\operatorname{argmin}} \left[\alpha \left(\frac{1}{|A_f|} \sum_{i \in A_f} w_i \left| \mathbf{T}(f_s^i) - f_t^i \right|^2 \right) + (1-\alpha) \left(\frac{1}{|A_d|} \sum_{j \in A_d} w_j \left| (\mathbf{T}(p_s^j) - p_t^j) \cdot n_t^j \right|^2 \right) \right], \quad (4)$$

where the first part measures average squared distances for the visually associated feature points. For the dense points used in the second part, we employ a *point-to-plane* error term that minimizes the squared 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 (Segal et al. 2009), as well as being resistant to boundary point matches. The two components in (4) are weighted using a factor α. Since the point-to-plane error metric has no known closed-form solution, minimization requires the use of a nonlinear optimizer. RGB-D ICP performs the minimization using Levenberg–Marquardt.

For the re-projection error used in RE-RANSAC, we optimize a measure incorporating re-projection error for feature point associations:

$$\mathbf{T}^* = \underset{\mathbf{T}}{\operatorname{argmin}} \left[\left(\frac{1}{|A_f|} \sum_{i \in A_f} \left| Proj(\mathbf{T}(f_s^i)) - Proj(f_t^i) \right|^2 \right) + \beta \left(\frac{1}{|A_d|} \sum_{j \in A_d} w_j \left| (\mathbf{T}(p_s^j) - p_t^j) \cdot n_t^j \right|^2 \right) \right]. \quad (5)$$

The first part of (5) minimizes the re-projection error between the fixed feature point associations obtained from RANSAC, thereby using the same error metric by which they were originally obtained. Again, optimization is done using Levenberg–Marquardt.

The loop exits in Step 7 after the transformation no longer changes above a small threshold θ 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 situations in which the dense ICP components alone might cause the point clouds to drift apart, which can happen in under-constrained cases such as large flat walls.

3.1.3. Two-Stage RGB-D ICP In practice, we found that when using RE-RANSAC, the joint optimization provides only marginal improvement if there are sufficient RANSAC inliers. Based on this insight, Algorithm 2 presents a fast alternative algorithm Two-Stage RGB-D ICP which avoids performing the more expensive ICP components of the combined optimization by utilizing the RE-RANSAC transform when reliable, and resorts to dense point-to-plane ICP only in cases where RE-RANSAC has a small number of inliers ($\leq \phi$).

3.1.4. Implementation details We implement our system using the Robot Operating System (ROS)[7] framework. The different components are implemented as separate nodes (processes) which communicate via the ROS messaging protocol.

In (4) we use a value of $\alpha = 0.5$ as per our results in Henry et al. (2010). The value of β in (5) is heuristically set to 1,000, which brings the two different units (pixels and meters) into roughly equivalent contribution to the error. We consider inliers to be matches for (1) and (4) if they fall within 0.03 m. For (2) and (5), the inlier distance is 2.0 pixels. RANSAC is considered to have failed if it obtains fewer than $\gamma = 10$ inliers. In Two-Stage RGB-D ICP, ϕ is a parameter to trade off speed and accuracy. In the

Algorithm 2: Two-Stage RGB-D ICP algorithm for more quickly matching two RGB-D frames.

input : source RGB-D frame P_s and target RGB-D frame P_t
output: optimized relative transformation $\mathbf{T}^*$

$F_s \leftarrow$ `Extract_RGB_Point_Features` (P_s);
$F_t \leftarrow$ `Extract_RGB_Point_Features` (P_t);
$(\mathbf{T}^*, A_f) \leftarrow$ `Perform_RANSAC_Alignment` (F_s, F_t);
if $|A_f| < \gamma$ **then**
  $\mathbf{T}^* = \mathbf{T}_\mathrm{p}$;
  $A_f = \emptyset$;
end
if $|A_f| \geq \phi$ **then**
  return $\mathbf{T}^*$;
else
  repeat
    $A_d \leftarrow$ `Compute_Closest_Points` $(\mathbf{T}^*, P_s, P_t)$;
    $\mathbf{T}^* \leftarrow$ `Optimize_Alignment` $(\mathbf{T}^*, A_f, A_d)$;
  until *(*`Change` $(\mathbf{T}^*) \leq \theta$*)* **or** *(*Iterations > MaxIterations*)*;
  return $\mathbf{T}^*$;
end

extreme case, setting $\phi = \gamma$ means that ICP is run only when RANSAC fails, which we did for the experiments in Section 4.2.

We find that downsampling the source cloud given to ICP by a factor of 10 gives a good compromise between matching speed and accuracy. Our implementation allows for setting the value of w_i in the point-to-plane component based on factors such as distance from the camera, normal angle agreement, and color difference. However, for the experiments in Section 4.2 we leave $w_i = 1$.

There exist several variants of RANSAC with alternative inlier selection schemes, such as MLESAC (Torr and Zisserman 1996). An investigation of these is left for future work.

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 previously visited location. 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.

3.2.1. Loop-closure detection The simplest approach to loop closure detection would be to run RANSAC alignment between the current frame and each of the previous frames and detect a loop closure whenever the number of inliers is above a threshold. This is prohibitively expensive, and the computation required for each new frame grows quickly with the number of previous frames. We employ a combination of techniques to make loop detection more efficient.

First, we define *keyframes*, which are a subset of the aligned frames. These can be selected in various ways. The simplest is to choose every *n*th frame. Another option is to determine keyframes based on visual overlap. In our previous work (Henry et al. 2010), after we align a frame F, we reuse the visual 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 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, which is still localized relative to the previous keyframe through intervening non-keyframes. We have found that a more efficient estimate of visual overlap is to compose all relative poses since the previous keyframe and establish a new keyframe whenever the accumulated rotation or translation is above a threshold. This allows the density of keyframes to adjust to camera motion without requiring an additional RANSAC alignment for each new frame. Keyframes are the only frames considered as potential loop-closure frames.

Each time we create a new keyframe we attempt to detect loop closures with previous keyframes. Not all previous keyframes need be considered. We *prefilter* keyframes based on the currently estimated global poses, avoiding a relatively expensive RANSAC alignment against frames more than a few meters away from our current position estimate.

We also take advantage of work on *place recognition* to further prefilter loop-closure candidates. By using a vocabulary tree (Nister and Stewenius 2006) based on the Calonder feature descriptors, we are able to identify *n* previous frames with similar appearance. Briefly, this technique hierarchically quantizes feature descriptors, which allows for each keyframe to be represented as a 'bag of visual words' which can be rapidly compared with other

keyframes providing likely loop-closure candidates. We use an implementation available from within ROS.[7]

Only keyframes passing these prefilters are subjected to RANSAC alignment with the current keyframe. A closure is detected if enough geometrically consistent feature point matches are recovered by RANSAC.

3.2.2. Pose graph optimization One strategy for global optimization is to represent constraints between frames with a *pose 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.

In order to minimize the conflict between sequential constraints and loop closure constraints we employ TORO (Grisetti et al. 2007a,b). TORO efficiently minimizes 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 a loop closure is detected, initializing it with the output of the previous TORO run and the contiguous-frame constraints added since then.

3.2.3. Sparse bundle adjustment A second strategy is to globally minimize the re-projection error of feature points across all frames in which these points are matched using SBA (Triggs et al. 2000; Lourakis and Argyros 2009; Konolige 2010b). This has the advantage of taking into account the same visual features appearing in multiple frames, as well as adjusting the estimated 3D locations of feature points along with the camera poses making the optimization robust to uncertain depth estimates. Formally, SBA solves for a set of camera poses C (where $c_i \in SE(3)$) and 3D feature point locations P (where $p_j \in \mathbb{R}^3$) which minimize the global re-projection error

$$\sum_{c_i \in C} \sum_{p_j \in P} v_{ij} \left| Proj(c_i(p_j)) - (\bar{u}, \bar{v}, \bar{d}) \right|^2 . \quad (6)$$

The function *Proj* is described in (3) and $(\bar{u}, \bar{v}, \bar{d})$ is the observed projection of p_j onto camera c_i. The notation $c_i(p_j)$ indicates the application of the 6DOF transformation taking point p_j from world coordinates to 3D Euclidean coordinates relative to the camera center of c_i. The dummy variable v_{ij} indicates whether feature point p_j is observed in camera c_i. Although SBA can be used to solve for additional camera parameters, here we have a single camera with known parameters and are interested in recovering the optimized camera poses.

Bundle adjustment has an advantage over pose graph optimization because it can reason about the underlying mutually observed feature points which constrain the relative poses between frames. However, it is not immediately clear how pairs of frames which have fewer than the minimum RANSAC inliers γ can be incorporated into this optimization. Such pairs of frames have a relative pose constraint between them obtained from the ICP component of our system, and thus are no different from RANSAC constraints in the context of pose-graph optimization (Section 3.2.2), where this issue does not arise. In the context of bundle adjustment, we solve this problem by selecting pairs of points from the dense point clouds to constrain the relative pose between such frames. We sample points from one of the frames across a grid in image space, and find the closest point in the dense cloud of the other frame according to the relative pose obtained from ICP. Over these pairs of points, we keep those pairs whose distance is below a threshold. We further filter the pairs by only keeping those in which the corresponding normals deviate by less than an angular threshold. These remaining pairs are considered equivalent to feature point matches, and are included as additional points within the global SBA system. Without this modification, frame sequences containing any ICP-only constraints could result in a disconnected, unsolvable bundle adjustment system.

In addition to being a valuable global optimization tool, SBA is also our choice for refining the inliers of RE-RANSAC used in RGB-D ICP (see Section 3.1.1). To do this, we simply set up a small SBA system consisting of the two frames with all inliers as feature points.

We use an implementation of SBA available in ROS[7] which uses the techniques described by Konolige (2010b).

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* (Pfister et al. 2000; Krainin et al. 2011). 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 Pfister et al. (2000) and Krainin et al. (2011) for updating, adding, and removing surfels. Surfels store a measure of confidence, which is increased through being seen from multiple angles over time. This is represented as a two-dimensional histogram of orientations relative to the surfel normal from which the surfel has been observed. Surfels with low confidence are removed from the representation. Because surfels have a notion of size (obtained initially from the depth of the

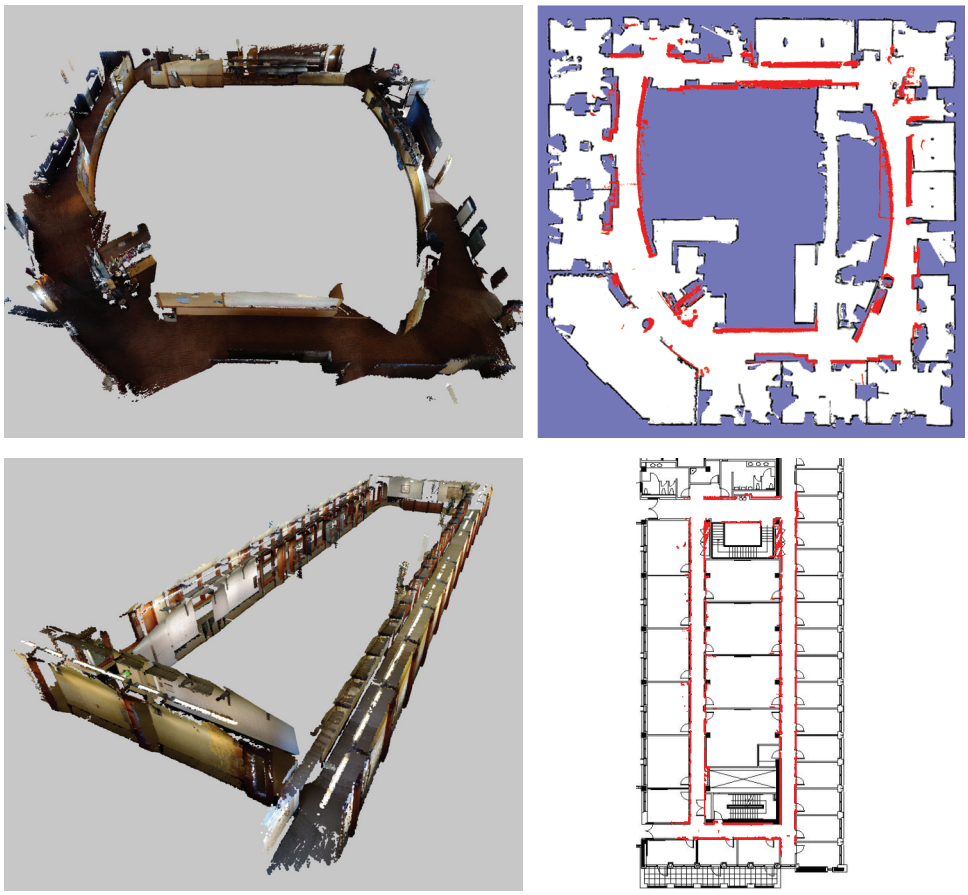


Fig. 4. (Left) 3D maps generated by RGB-D Mapping for large loops in the Intel lab (upper) and the Allen Center (lower). (Right) Accuracy: Maps (red) overlaid on a 2D laser scan map of the Intel Lab and on a floor plan of the Allen Center. For clarity, most floor and ceiling points were removed from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

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. We direct the reader to Krainin et al. (2011) for more details of our surfel implementation.

Currently surfel maps are created from point cloud maps as a post-process, as the incremental nature of surfel map construction requires rebuilding the surfel map when camera poses are updated through global optimization.

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 RGB-D ICP algorithm improves accuracy of frame-to-frame alignment, and we illustrate the advantageous properties of the surfel representation.

4.1. Global consistency

We tested RGB-D Mapping in two indoor environments: the Intel Labs Seattle offices and the Paul G. Allen Center for Computer Science and Engineering at the University of Washington. During mapping, the camera was carried by a person, and generally pointed in the direction of travel. The left panels in Figure 4 show 3D maps built for large loops in these environments. The loop in the upper panel consists of 906 frames over a length of 71 m, while the lower panel is from 716 frames over a length of 114 m (the difference in frames per meter is due to variations in the carrier's rate of motion). To assess the consistency of these maps we overlaid our 3D maps onto 2D layouts generated by different means. The right panels in Figure 4 shows these

Table 1. Comparison of Euclidean error RANSAC and re-projection error RANSAC on the *Intel-Day* sequence.

	EE-RANSAC	RE-RANSAC
Mean inliers per frame	60.3	116.7

overlays. For the Intel lab, we compare against a map built with a SICK laser scanner using a standard two-dimensional SLAM approach. An architectural floor plan is used in the case 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. Properties of RGB-D ICP

To more thoroughly evaluate the benefits of RGB-D ICP, we determined ground-truth poses in the Intel lab loop. We placed 16 markers around the Intel loop and measured the true distance between consecutive markers, which varied between 3 and 5.5 m. We collected two datasets: *Intel-Day* and *Intel-Night*, where the second is a challenging dataset 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.

We first evaluate the benefit of using RE-RANSAC instead of EE-RANSAC on the *Intel-Day* sequence. As can be seen in Table 1, RE-RANSAC averages nearly twice as many inliers per frame. This is because visual features with more uncertain depth estimates can be utilized using a re-projection inlier threshold. Additionally, it can be seen in Table 2 that RE-RANSAC achieves lower error than EE-RANSAC on the *Intel-Day* sequence due to optimizing a more appropriate error metric.

We now investigate the contributions of various components of RGB-D ICP. As can be seen in Table 2, RGB-D ICP provides lower error than either component alone. The large error of RE-RANSAC and EE-RANSAC on *Intel-Night* is due to failed alignments in the dark section of the environment, where they resort to a constant motion assumption. Two-Stage RGB-D ICP is equivalent to RE-RANSAC on the *Intel-Day* sequence because RE-RANSAC never fails on this sequence, and thus the ICP portion of Two-Stage RGB-D ICP is never executed. Two-Stage RGB-D ICP also exhibits only marginally higher error than RGB-D ICP across both sequences, suggesting that it is a valuable alternative for more rapid alignment of frames.

Corresponding to the error results in Table 2, we present timing results for each of the alignment methods in Table 3. The two RANSAC methods are consistently fastest. One thing to note is that RGB-D ICP is faster than ICP alone because RGB-D ICP initializes the combined optimization procedure from the RE-RANSAC solution and therefore requires fewer iterations to converge. Observe that Two-Stage RGB-D ICP is equivalent to RE-RANSAC on the *Intel-Day* sequence because ICP is never required, and Two-Stage RGB-D ICP is more efficient than RGB-D ICP on the *Intel-Night* sequence since the combined optimization is required only on frames where RE-RANSAC fails.

4.3. Properties of global optimization

We show results from aligning a more complex data sequence to investigate the optimization properties of bundle adjustment with SBA. In this challenging sequence, several frames in the kitchen area have insufficient RANSAC inliers ($\leq \gamma$), and thus ICP is required. Figure 5 shows that SBA optimization is still able to operate on a sequence which contains pose constraints generated from ICP alignments. The resulting surfel map, even in the kitchen where the ICP component of RGB-D ICP was required, is visually consistent.

We now compare results between using pose-graph optimization (via TORO) or bundle adjustment (via SBA) as a global optimizer. In Figure 6 we aligned roughly 100 frames using Two-Stage RGB-D ICP and globally optimized the camera poses using TORO in one case and SBA in the second. SBA results in more consistent alignment due to jointly optimizing re-projection error over feature matches as opposed to simply optimizing relative pose constraints directly as TORO does.

A second comparison of bundle adjustment (SBA) and pose-graph optimization (TORO) is shown in Figure 7. This sequence is a portion of the single large Intel Labs office building loop following optimization of the entire loop sequence. Although TORO does 'close' the loop (not shown), the resulting point cloud map has inconsistencies due to TORO failing to consistently combined the sequential pose constraints with medium range constraints generated by loop-closure detection. The same sequence is globally optimized by SBA in a more consistent fashion, because SBA is a joint optimization over camera poses and feature locations, allowing it to smoothly combine sequential and loop-closure constraints. It should be noted that incorporating this type of medium range loop-closure constraint helps reduce camera drift.

Next we examine the running time requirements for TORO and SBA on the large loop shown in Figure 5. A graph of the time required for global optimization of all frames is shown in Figure 8. TORO is considered to have converged when the error change between iterations falls

Table 2. Sequential alignment comparison on large-scale marker sequences. Values are mean error in meters measured against ground truth distances between markers with 95% confidence intervals.

	RE-RANSAC	EE-RANSAC	ICP	RGB-D ICP	Two-Stage RGB-D ICP
Intel-Day	0.11 (±0.05)	0.16 (±0.07)	0.15 (±0.05)	**0.10 (±0.04)**	0.11 (±0.05)
Intel-Night	1.09 (±0.88)	1.15 (±0.89)	0.17 (±0.06)	**0.15 (±0.08)**	0.15 (±0.09)

Table 3. Timing results for sequential alignment techniques. Values are mean seconds per frame with 95% confidence intervals.

	RE-RANSAC	EE-RANSAC	ICP	RGB-D ICP	Two-Stage RGB-D ICP
Intel-Day	0.21 (±0.03)	0.20 (±0.05)	0.72 (±0.73)	0.48 (±0.10)	0.21 (±0.03)
Intel-Night	0.20 (±0.05)	0.20 (±0.05)	0.43 (±0.64)	0.57 (±0.47)	0.37 (±0.63)

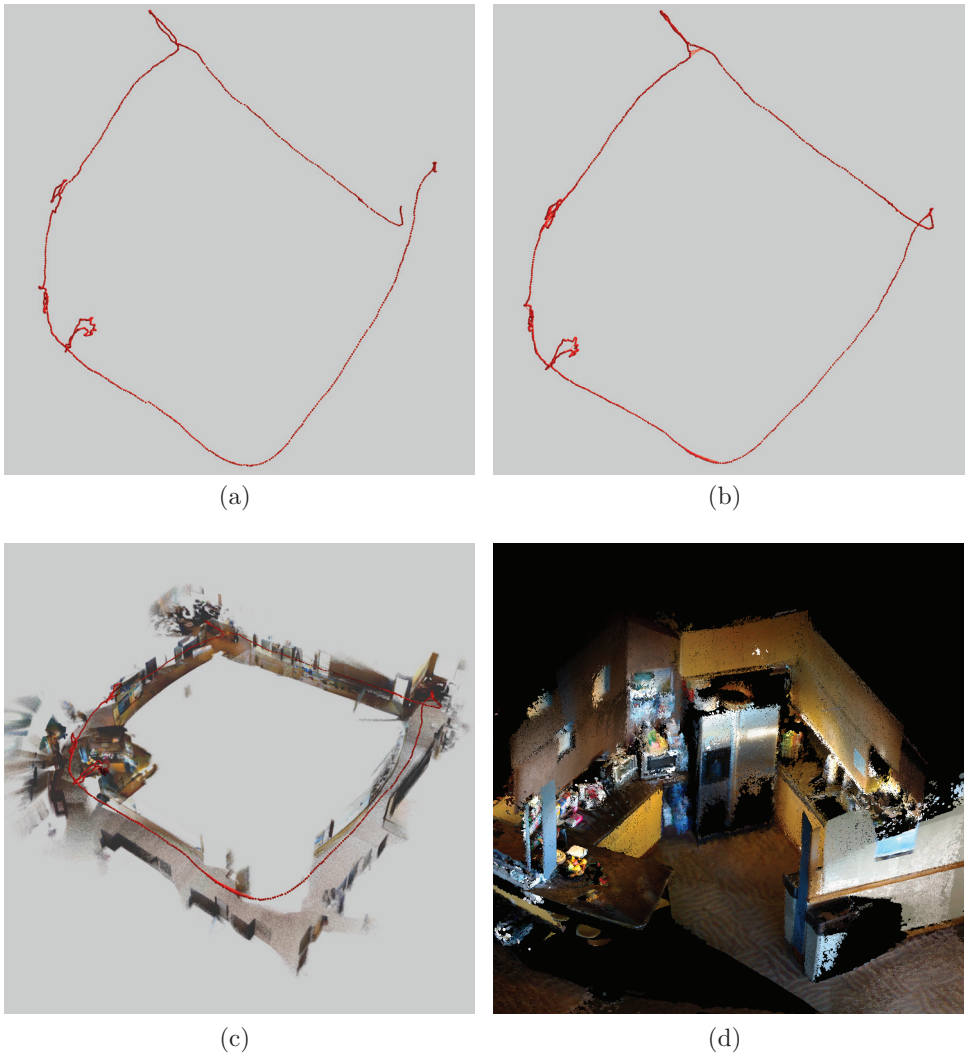
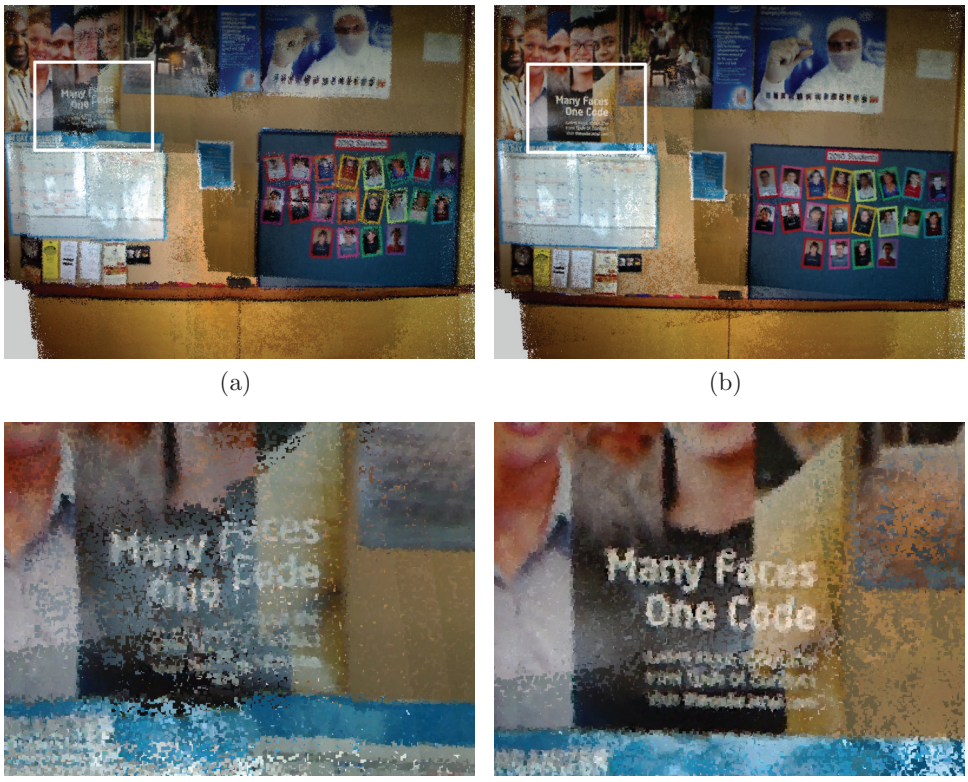


Fig. 5. Results on a challenging data sequence in the Intel Labs Seattle office. (a) RGB-D ICP poses before optimization. (b) SBA optimized pose graph. (c) SBA pose graph and point cloud. (d) Detailed view of a kitchen area in the surfel map, including counters and a refrigerator.

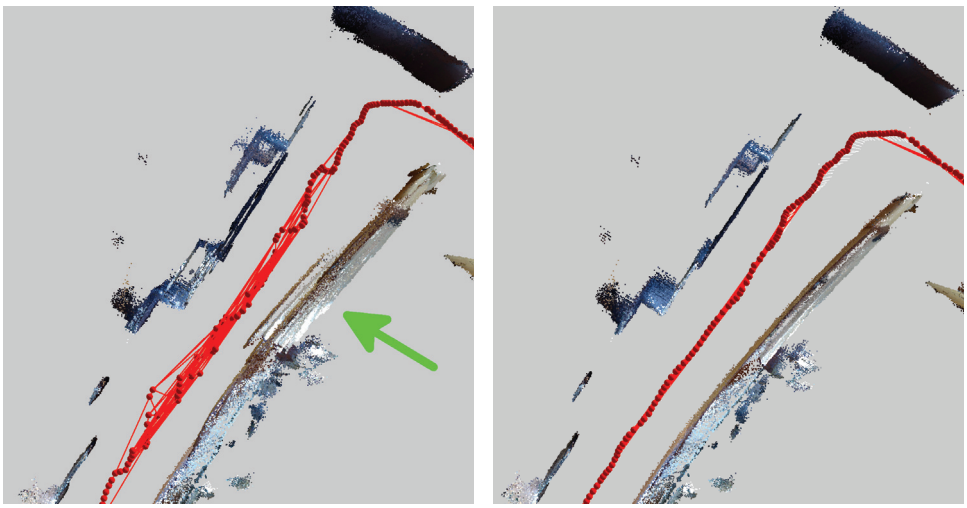
below a small threshold. SBA is run for five iterations, then outlier projections with error greater than 2.0 pixels are removed, followed by another five iterations. We see that SBA takes notably longer than TORO, and that the time taken by SBA grows roughly linearly with the size of the map, which agrees with the observations about



(a) (b)

(c) (d)

Fig. 6. Comparison of detailed visual consistency between TORO and SBA. In the upper images a white box shows the region detailed in the lower images. Note the more consistent alignment from SBA. Color variation is due to camera auto-exposure and white balance. (a) Point cloud using TORO. (b) Point cloud using SBA. (c) Close-up of TORO cloud. (d) Close-up of SBA cloud.



(a) (b)

Fig. 7. Comparison of structural consistency between TORO and SBA. Both are optimizing the same set of loop-closure constraints. Note the more jagged path produced by optimizing over non-sequential pose constraints with TORO. Ceiling and floor points are removed for clarity. (a) Point cloud using TORO (b) Point cloud using SBA.

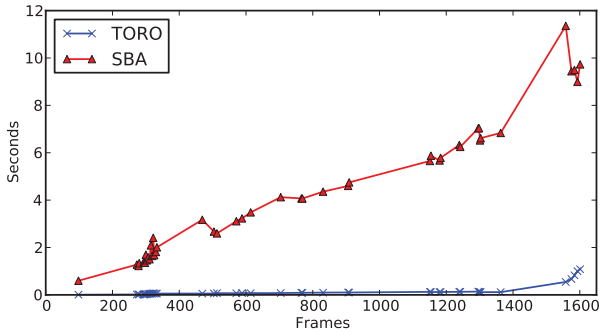


Fig. 8. Time taken by global optimization for the map shown in Figure 5. The final optimization with TORO and SBA takes 1.08 and 9.73 seconds, respectively. The increase in time taken by TORO towards the end of the sequence occurs after the large-scale loop is closed, requiring more iterations for TORO to converge.

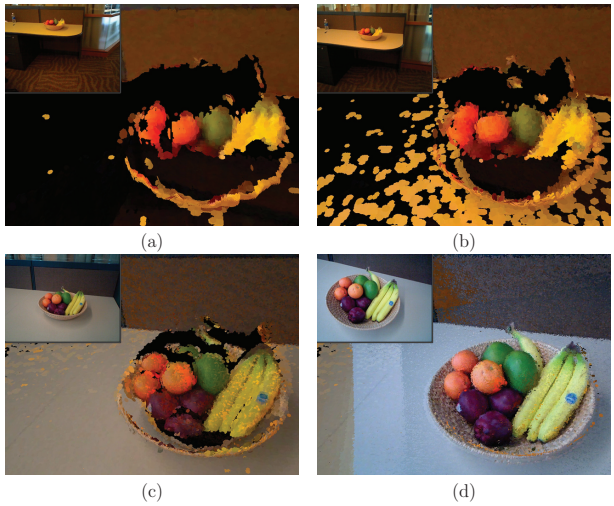


Fig. 9. Surfel updates. (a) The initial surfels from the first frame. (b,c) As the camera moves closer, more surfels are added and existing ones are refined. (d) The completed surfel model from 95 aligned RGB-D images.

Fig. 10. Optimized point cloud map showing SBA feature point locations.

sparse maps made in Konolige (2010b). Even with the relatively large amount of time taken by SBA towards the end of the sequence, when amortized over the total number of frames, SBA optimization takes only 0.12 seconds per frame, which is less than the time required for frame-to-frame RGB-D ICP. Clearly this will not continue to hold as the map grows arbitrarily large, and more sophisticated strategies will be required, such as running SBA over only a window of recent frames or applying global SBA less frequently.

4.4. Surfel representation

To demonstrate the value of the surfel representation of aligned point clouds, we first point out that the surfel representation shown in Figure 9 is formed from a sequence of 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 Figure 9(d) has concisely and faithfully combined the information from all input frames.

We now examine the relationship between point cloud maps, surfel maps, and SBA feature points. In Figure 10, the feature points used for SBA optimization are shown superimposed over the point cloud. In Figure 11, it can be seen that the surfel map is smoother and more consistent than the point cloud representation. The surfel representation is also more efficient, as it combines information from roughly 28 million points into 1.4 million surfels. Generating a surfel map takes roughly 3 seconds per frame when updating and adding surfels in a map of the size of our large loops. However, this is currently the least optimized component of our system.

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.uw.edu/robotics/rgbd-mapping/.

4.5. Lessons learned

RANSAC is the faster and more reliable alignment component when considered individually. Using re-projection

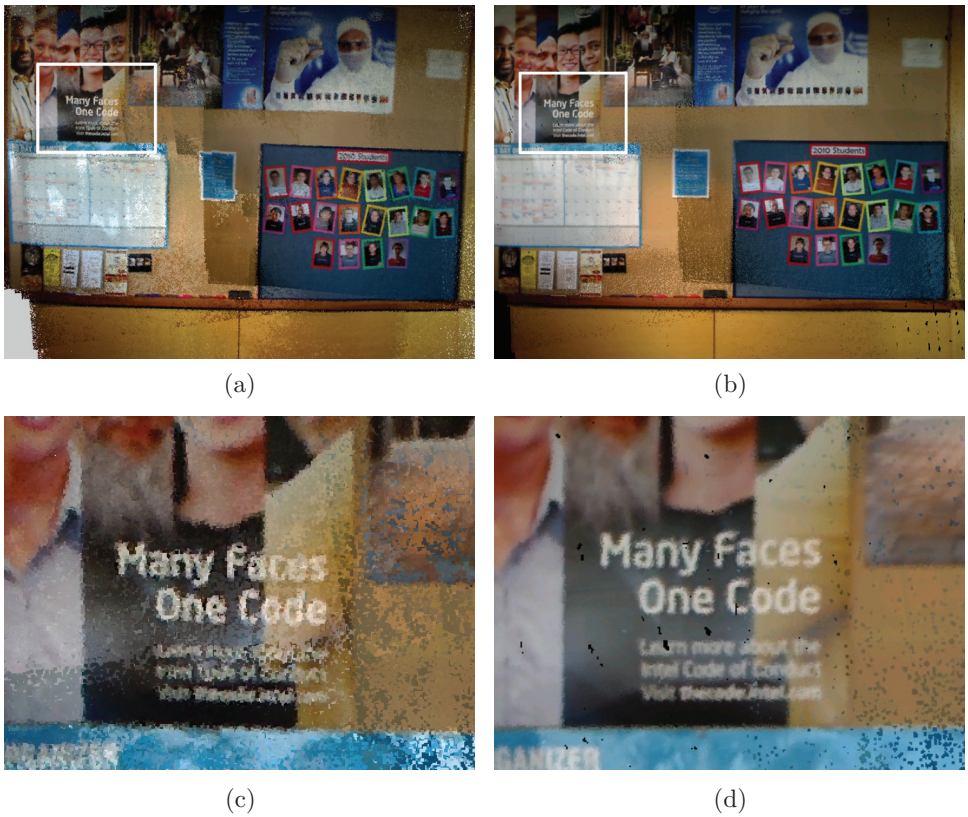


Fig. 11. The surfel map created after SBA optimization is compared with the point cloud map (duplicated from Figure 6 for comparison purposes). The surfel map is smoother and more consistent. (a) Point cloud using SBA. (b) Surfels using SBA. (c) Close-up of SBA cloud. (d) Close-up of SBA surfels.

error as we do in RE-RANSAC improves alignment accuracy considerably. However, there are situations where it is unreliable 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 RGB-D ICP provides notably better alignment than RANSAC alone.

Our experiments also show that for active stereo depth cameras such as PrimeSense or Kinect, a stereo-based reprojection error metric for RANSAC yields better alignments than a Euclidean error metric typically used for point cloud alignment. To generate globally aligned maps, we investigate both pose-based optimization (TORO) and SBA. While TORO can readily incorporate pose constraints generated by both ICP and RANSAC, SBA requires that all consecutive frames are connected via point-based correspondences. To enable the use of SBA in environments with featureless areas, we incorporate point associations found via RGB-D ICP into the SBA optimization. Visual comparisons of the resulting maps indicate that SBA yields more accurate fine-grained alignments than TORO. However, TORO has the advantage of being far more efficient and more recent versions of this approach should generate yet more accurate results than those presented here (Ruhnke et al. 2011).

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 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 that:

1. purely vision-based or purely depth-based frame matching techniques are not sufficient to provide robust and accurate visual odometry with these cameras;
2. a tight integration of depth and color information can yield robust frame matching and loop-closure detection;
3. 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;
4. it will be feasible to build complete robot navigation and interaction systems solely based on inexpensive RGB-D camera.

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. RGB-D Mapping incorporates a surfel representation to enable compact representations and visualizations of 3D maps. At the core of RGB-D Mapping is RGB-D ICP, a novel ICP variant that takes advantage of the rich information contained in RGB-D data. The main idea behind RGB-D ICP is to use visual information to identify sparse point features in each camera frame. These point features can be used in a RANSAC optimization to generate alignments between consecutive frames for visual odometry, and between temporally far apart frames to detect loop closures. While sparse visual features typically provide better alignment accuracies than dense point cloud alignment via ICP, they might fail in areas that lack visual information, such as very dark rooms. RGB-D ICP overcomes this limitation by combining visual feature matching (RANSAC) with dense point cloud alignment (point-to-plane ICP) to get the best of both techniques: maximally accurate alignment when visual features are available, and robust alignment when there are no visual features. We introduce two variants of RGB-D ICP, one that combines RANSAC and ICP into a joint optimization, and another, slightly more efficient approach, that performs either RANSAC or the full joint optimization depending on the situation.

Given that RGB-D cameras are available to the public at an extremely low price, 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 is an important step to towards enabling the development of useful, affordable robot platforms.

Despite these encouraging results, every module of RGB-D Mapping has several shortcomings that deserve future effort. RGB-D Mapping only uses two consecutive frames to estimate the motion of the camera. Based on our findings, we believe that a sequential window-based bundle adjustment technique that reasons about multiple frames at a time could certainly improve the accuracy of alignments, without substantial increase in complexity. Currently, loop closures are detected by frame-to-frame visual feature matching. While this approach is able to detect obvious loop closures, it is not sufficient to add all of the constraints needed to generate fully consistent 3D maps. More work on loop closure detection taking the full depth and color information into account is needed. We hope to perform more extensive and detailed ground-truth experiments, for example using motion capture. In recent work, we have begun to investigate the benefits and opportunities afforded through interactive mapping, in which a human operator receives real-time feedback which can be used to correct mapping failures and generate complete maps (Du et al. 2011).

The computer graphics community has developed extremely sophisticated visualization techniques, and incorporating these into RGB-D Mapping could further improve the alignment and visual quality of the 3D maps. For example, patch-based multi-view stereo (PMVS; Furukawa and Ponce, 2010)[5] can generate quite accurate reconstructions using a visual consistency measure, and it would be exciting to apply these techniques to our 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.

Notes

1. See http://www.xbox.com/en-US/kinect and http://www.primesense.com/
2. See http://www.canesta.com/ and http://www.mesa-imaging.ch/
3. See http://www.primesense.com/.
4. See http://www.xbox.com/en-US/kinect
5. See also http://grail.cs.washington.edu/software/pmvs/.
6. A small transformation is applied within the camera driver which shifts the points from the frame of the IR camera to the physically adjacent frame of the RGB camera, so the depth (z) values in Euclidean space (as obtained from the camera driver) are relative to the RGB camera.
7. See http://www.ros.org/.

Acknowledgements

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.

Funding

This work was funded in part by an Intel grant, by the ONR (MURI grant number N00014-07-1-0749), and by the NSF (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

Akbarzadeh A, Frahm JM, Mordohai P, Clipp B, Engels C, Gallup D, et al.
(2006) Towards urban 3D reconstruction from video. In *Proceedings of the Third International Symposium on 3D Data Processing, Visualization and Transmission (3DPVT)*.

Andreasson H and Lilienthal A (2010) 6D scan registration using depth-interpolated local image features. *Robotics and Autonomous Systems* 58(2): 157–165.

Besl PJ and McKay ND (1992) A method for registration of 3-D shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 14(2): 239–256.

Bradski G (2000) The OpenCV Library. *Dr. Dobb's Journal of Software Tools*. Available from: http://drdobbs.com/open-source/184404319

Calonder M, Lepetit V and Fua P (2008) Keypoint signatures for fast learning and recognition. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 58–71.

Chen Y and Medioni G (1992) Object modeling by registration of multiple range images. *Image and Vision Computing* 10: 145–155.

Clemente L, Davison A, Reid I, Neira J and Tardós J (2007) Mapping large loops with a single hand-held camera. In *Proceedings of Robotics: Science and Systems (RSS)*.

Cui Y, Schuon S, Chan D, Thrun S and Theobalt C (2010) 3D shape scanning with a time-of-flight camera. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*.

Debevec P, Taylor CJ and Malik J (1996) Modeling and rendering architecture from photographs: A hybrid geometry and image-based approach. In: SIGGRAPH '96, Proceedings of the 23rd Annual Conference on Computer Graphics, pp. 11–20.

Droeschel D, May S, Holz D, Ploeger P and Behnke S (2009) Robust ego-motion estimation with ToF cameras. In *European Conference on Mobile Robots (ECMR)*.

Du H, Henry P, Ren X, Cheng M, Goldman DB, Seitz SM, et al. (2011) Interactive 3D modeling of indoor environments with a consumer depth camera. In *International Conference on Ubiquitous Computing (UbiComp)*.

Fischler MA and Bolles RC (1981) Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM* 24: 381–395.

Furukawa Y and Ponce J (2010). Accurate, Dense, and Robust Multiview Stereopsis. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32(8): 1362–1376.

Furukawa Y, Curless B, Seitz S and Szeliski R (2009) Reconstructing building interiors from images. In *Proceedings of the International Conference on Computer Vision (ICCV)*.

Grisetti G, Grzonka S, Stachniss C, Pfaff P and Burgard W (2007a) Estimation of accurate maximum likelihood maps in 3D. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

Grisetti G, Stachniss C, Grzonka S and Burgard W (2007b). A tree parameterization for efficiently computing maximum likelihood maps using gradient descent. In *Proceedings of Robotics: Science and Systems (RSS)*.

Henry P, Krainin M, Herbst E, Ren X and Fox D (2010) RGB-D mapping: Using depth cameras for dense 3D modeling of indoor environments. In *Proceedings of the International Symposium on Experimental Robotics (ISER)*.

Horn BKP (1987) Closed-form solution of absolute orientation using unit quaternions. *Journal of Optical Society of America A* 4: 629–642.

Johnson A and Kang SB (1997) Registration and integration of textured 3-D data. In *International Conference on 3-D Digital Imaging and Modeling (3DIM)*, pp. 234–241.

Kim YM, Theobalt C, Diebel J, Kosecka J, Micusik B and Thrun S (2009) Multi-view image and ToF sensor fusion for dense 3D reconstruction. In *International Conference on 3-D Digital Imaging and Modeling (3DIM)*.

Konolige K (2004) Large-scale map making. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*.

Konolige K (2010a) Projected texture stereo. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*.

Konolige K (2010b) Sparse sparse bundle adjustment. In *Proceedings of the British Machine Vision Conference (BMVC)*.

Konolige K and Agrawal M (2008) FrameSLAM: From bundle adjustment to real-time visual mapping. *IEEE Transactions on Robotics* 25(5): 1066–1077.

Krainin M, Henry P, Ren X and Fox D (2011) Manipulator and object tracking for in-hand 3D object modeling. *The International Journal of Robotics Research* 30(11): 1311–1327.

Lourakis M and Argyros A (2009) SBA: A software package for generic sparse bundle adjustment. *ACM Transactions on Mathematical Software* 36: 1–30.

Lowe D (2004) Discriminative image features from scale-invariant keypoints. *International Journal of Computer Vision* 60(2): 91–110.

Lu F and Milios E (1997) Globally consistent range scan alignment for environment mapping. *Autonomous Robots* 4: 333–349.

May S, Dröschel D, Holz D, Fuchs E, Malis S, Nüchter A, et al.(2009) Three-dimensional mapping with time-of-flight cameras. *Journal of Field Robotics* 26(11-12): 934–965.

Newcombe R and Davison A (2010) Live dense reconstruction with a single moving camera. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1498–1505.

Newman P, Sibley G, Smith M, Cummins M, Harrison A, Mei C, et al. (2009) Navigating, recognising and describing urban spaces with vision and laser. *The International Journal of Robotics Research* 28(11–12): 1406–1433.

Nister D (2004) An efficient solution to the five-point relative pose problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 26: 756–777.

Nister D, Naroditsky O and Bergen J (2004) Visual odometry. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*.

Nister D and Stewenius H (2006) Scalable recognition with a vocabulary tree. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*.

Pfister H, Zwicker M, van Baar J and Gross M (2000) Surfels: Surface elements as rendering primitives. In: SIGGRAPH 2000, Proceedings of the 27th Annual Conference on Computer Graphics, pp. 335–342.

Pollefeys M, Nister D, Frahm J-M, Akbarzadeh A, Mordohai P, Clipp B, et al. (2008) Detailed real-time urban 3D reconstruction from video. *International Journal of Computer Vision* 72: 143–167.

Prusak A, Melnychuk O, Roth H, Schiller I and Koch R (2008) Pose estimation and map building with a time-of-flight-camera for robot navigation. *International Journal of Intelligent Systems Technologies and Applications* 5: 355–364.

Ramos F, Fox D and Durrant-Whyte H (2007) CRF-matching: Conditional random fields for feature-based scan matching. In *Proceedings of Robotics: Science and Systems (RSS)*.

Rosten E and Drummond T (2006) Machine learning for high-speed corner detection. In *Proceedings of the European Conference on Computer Vision (ECCV)*.

Ruhnke M, Kuemmerle R, Grisetti G and Burgard W (2011) Highly accurate maximum likelihood laser mapping by jointly optimizing laser points and robot poses. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, to appear.

Rusinkiewicz S and Levoy M (2001) Efficient variants of the ICP algorithm. In *International Conference on 3-D Digital Imaging and Modeling (3DIM)*.

Se S and Jasiobedzki P (2008) Stereo-vision based 3D modeling and localization for unmanned vehicles. *International Journal of Intelligent Control and Systems* 13: 47–58.

Segal A, Haehnel D and Thrun S (2009) Generalized-ICP. In *Proceedings of Robotics: Science and Systems (RSS)*.

Sharp GC, Lee SW and Wehe DK (2002) ICP registration using invariant features. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 24: 90–102.

Shotton J, Fitzgibbon A, Cook M, Sharp T, Finocchio M, Moore R, et al. (2011) Real-time human pose recognition in parts from single depth images. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*.

Snavely N, Seitz S and Szeliski R (2006) Photo tourism: Exploring photo collections in 3D. *ACM Transactions on Graphics* 25(3): 835–846.

Strobl K, Mair E, Bodenmuller T, Kielhofer S, Sepp W, Suppa M, et al. (2009) The self-referenced DLR 3D-modeler. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 21–28.

Thrun S, Burgard W and Fox D (2000) A real-time algorithm for mobile robot mapping with applications to multi-robot and 3D mapping. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*.

Thrun S, Burgard W and Fox D (2005) *Probabilistic Robotics*. Cambridge, MA: MIT Press.

Torr PHS and Zisserman A (1996) MLESAC: A new robust estimator with application to estimating image geometry. *Computer Vision and Image Understanding* 78(1): 138–156.

Triebel R and Burgard W (2005) Improving simultaneous mapping and localization in 3D using global constraints. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*.

Triggs B, McLauchlan P, Hartley R and Fitzgibbon A (2000) Bundle adjustment—a modern synthesis. *Vision Algorithms: Theory and Practice* 1883: 153–177.

Wu, C. (2007). SiftGPU: A GPU implementation of scale invariant feature transform (SIFT). Available at: http://cs.unc.edu/∼ccwu/siftgpu.