Chapter 2
Real-Time RGB-D Mapping and 3-D Modeling on the GPU Using the Random Ball Cover

Sebastian Bauer, Jakob Wasza, Felix Lugauer, Dominik Neumann, and Joachim Hornegger

Abstract In this chapter, we present a system for real-time point cloud mapping and scene reconstruction based on an efficient implementation of the iterative closest point (ICP) algorithm on the graphics processing unit (GPU). Compared to state-of-the-art approaches that achieve real-time performance using projective data association schemes which operate on the 3-D scene geometry solely, our method allows to incorporate additional complementary information to guide the registration process. In this work, the ICP's nearest neighbor search evaluates both geometric and photometric information in a direct manner, achieving robust mappings in real-time. In order to overcome the performance bottleneck in nearest neighbor search space traversal, we exploit the inherent computation parallelism of GPUs. In particular, we have adapted the random ball cover (RBC) data structure and search algorithm, originally proposed for high-dimensional problems, to low-dimensional RGB-D data. The system is validated on scene and object reconstruction scenarios. Our implementation achieves frame-to-frame registration runtimes of less than 20 ms on an off-the-shelf consumer GPU.

S. Bauer (✉) · J. Wasza · F. Lugauer · D. Neumann
Pattern Recognition Lab, Department of Computer Science, Friedrich-Alexander-Universität Erlangen-Nürnberg, Martensstr. 3, 91058 Erlangen, Germany
e-mail: sebastian.bauer@cs.fau.de

J. Wasza
e-mail: jakob.wasza@cs.fau.de

F. Lugauer
e-mail: felix.lugauer@gmail.com

D. Neumann
e-mail: dominik.neumann@gmail.com

J. Hornegger
Erlangen Graduate School in Advanced Optical Technologies (SAOT) & Pattern Recognition Lab, Department of Computer Science, Friedrich-Alexander-Universität Erlangen-Nürnberg, Martensstr. 3, 91058 Erlangen, Germany
e-mail: joachim.hornegger@cs.fau.de

A. Fossati et al. (eds.), *Consumer Depth Cameras for Computer Vision*, Advances in Computer Vision and Pattern Recognition, DOI 10.1007/978-1-4471-4640-7_2,

2.1 Introduction

In the past, the acquisition of dense 3-D range data was both tedious, time consuming and expensive. Lately, advances in RGB-D sensor design have rendered metric 3-D surface acquisition at convenient resolutions (up to 300k points) and framerates (up to 40 Hz) possible, holding potential for a variety of applications where real-time demands form a key aspect. The advent of Microsoft's Kinect [14], with more than 10 million sales within a few months, has caused a furor in the field of consumer electronics. In fact, the device has attracted the attention of various research communities.

This chapter addresses the field of 3-D scene and model reconstruction that provides the basis for many practical applications. Among others, 3-D modeling is a key component for the acquisition of virtual 3-D models from real objects, the digitalization of archaeological buildings or sculptures for restoration planning or archival storage [11], and the construction of environment maps in robot or vehicle navigation [19, 28]. In particular, in the field of robotics, there is an increasing interest in both 3-D environment reconstruction and simultaneous localization and mapping (SLAM) solutions [2, 6, 32].

We present a framework that is capable of mapping RGB-D point cloud data streams on-the-fly, enabling real-time 3-D scene modeling. We have implemented a hybrid 6-D ICP variant that performs the alignment by considering both photometric appearance and geometric shape [24]. Photometric (color) data may be an essential source of information to guide the registration process in cases when geometric surface information is not discriminative enough to achieve a correct alignment, see Fig. 2.1 for an example. Without loss of generality, we have designed the framework in a manner that allows to incorporate further complementary information into an n-dimensional point signature. In order to enable on-the-fly processing, the corpus of the framework is implemented on the GPU. For the nearest neighbor search, being the performance bottleneck in the majority of previous ICP implementations, we use a data structure that is specifically designed to benefit from the parallel architecture of modern GPUs. In this work, we investigated the fitness of the random ball cover (RBC) data structure and search algorithm [7, 8] for low-dimensional 6-D data. Trading accuracy against runtime, we propose a modified approximate RBC variant that is optimized in terms of performance. Please note that this chapter is a substantial extension of previous work by the authors [30]. In particular, we further enhanced the GPU implementation and achieved significant speedups.

The remainder of this chapter is organized as follows. In Sect. 2.2, we review relevant literature. We present our method for RGB-D mapping and 3-D modeling in Sect. 2.3. Implementation details are given in Sect. 2.4. In Sect. 2.5, we evaluate the proposed framework and discuss experimental results. Eventually, we draw a conclusion in Sect. 2.6.

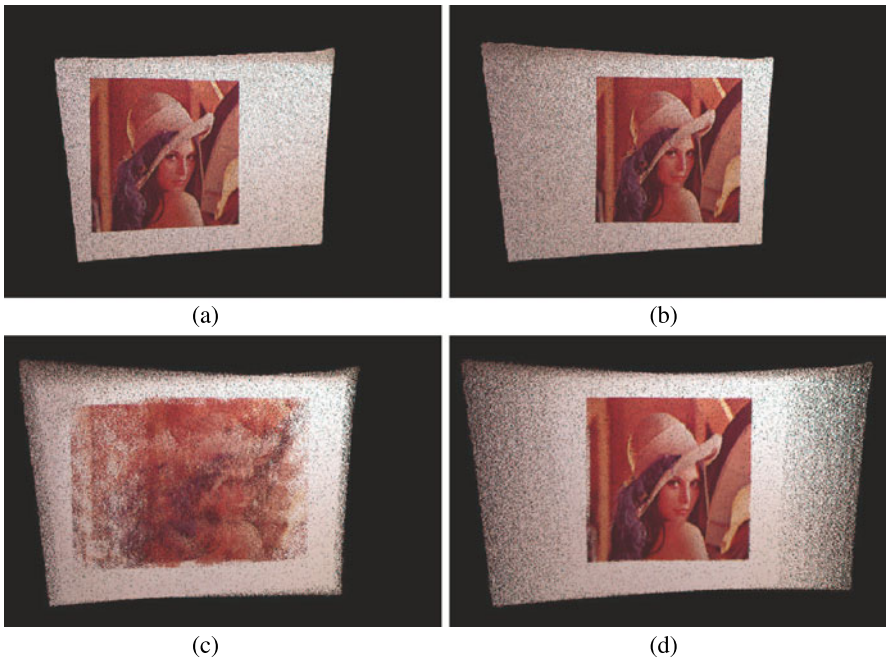


Fig. 2.1 Illustration of the benefit of incorporating photometric information into the point cloud alignment process in situations of non-salient surface geometry. The *top row* (**a**, **b**) depicts the first and last frame of an RGB-D sequence capturing a colored poster stuck to a plane wall from changing perspectives. Using scene geometry as the only source of information for the registration algorithm results in an erroneous alignment (**c**). Instead, by considering both geometric and photometric information, the correct alignment is found using the proposed framework (**d**)

2.2 Related Work

The iterative closest point (ICP) algorithm is state-of-the-art for the rigid alignment of 3-D point clouds [4, 9, 36], and the vast majority of related work builds upon this established scheme. However, in the field of 3-D environment and model reconstruction, only few existing approaches have achieved interactive frame-rates so far [12, 13, 19, 22]. Huhle et al. proposed a system for on-the-fly 3-D scene modeling using a low resolution Time-of-Flight camera (160 $\times$ 120 px), typically achieving per-frame runtimes of $>$2 s [22]. Engelhard et al. presented similar runtimes on Microsoft Kinect data (640 $\times$ 480 px) for an ICP-based RGB-D SLAM framework [12]. The RGB-D mapping framework of Henry et al. performs ICP registration in an average of 500 ms [19].

Only recently, real-time frame-rates were reported for geometric ICP variants [13, 23, 31]. In particular, the *KinectFusion* framework [23, 31] has gained popularity in the field of 3-D reconstruction. The fundamental core of this framework is based on the work of Rusinkiewicz et al. [35], combining projective data association [5] and a point-to-plane metric [9] for rigid ICP surface registration

and sensor pose estimation, respectively. While the original work was limited to a frame-to-frame alignment [35], KinectFusion tracks the depth frame against a globally fused implicit surface model of the observed scene [10]. This limits the drift behavior and results in an increased robustness and reconstruction accuracy, respectively. Real-time capability is achieved using a parallelized implementation on the GPU.

Compared to related methods based on projective data association [5] that primarily consider the surface geometry for finding corresponding points, our approach allows to incorporate multiple complementary sources of information (in our case geometry and photometry) into the nearest neighbor search. Furthermore, explicitly performing a nearest neighbor search according to a point signature potentially allows one to extend the framework to handle large misalignments by a feature-based initial pre-alignment [3].

More than a decade ago, Johnson and Kang presented the first approach to incorporate photometric information into the ICP framework (*Color-ICP*) in order to improve its robustness [24]. The basic idea is that photometric information can compensate for regions with non-salient topologies, whereas geometric information can guide the pose estimation for faintly textured regions. In experiments, Johnson and Kang observed that the additional use of color information decreased the registration error by one order of magnitude. Recently, modifications have been proposed that try to accelerate the color ICP's nearest neighbor search by pruning the search space w.r.t. photometrically dissimilar points [11, 25]. However, this reduction typically comes with a loss in robustness.

Since modern RGB-D devices produce and propagate an immense data stream, efficient implementations are inevitable in order to fulfill real-time constraints. For the ICP algorithm in general, a comprehensive survey of efficient implementation variants was given by Rusinkiewicz and Levoy [36]. However, their survey did not include hardware acceleration techniques.

For the nearest neighbor search, being a major bottleneck in terms of runtime, CPU architectures have shown to benefit from space-partitioning data structures like k-d trees [1]. In contrast to algorithmic improvements, hardware acceleration techniques are increasingly attracting the attention of the community. Garcia et al. have shown that a GPU-based brute-force implementation outperforms a CPU-based k-d tree [15]. The reason for this lies in the fact that the brute-force primitive can be implemented efficiently using techniques known from the well understood problem of GPU-based matrix–matrix multiplication. Implementations of traditional nearest neighbor search acceleration strategies on the GPU are challenging due to the non-parallel and recursive nature of construction and/or traversal of the underlying data structures. For instance, Qiu et al. [33] achieved excellent frame-rates for GPU-based k-d tree queries. However, the construction of the tree is performed on the CPU, thus limiting performance when the tree must be constructed on a per-frame basis as in the application scenarios considered in this chapter. Recently, space-partitioning strategies that are specifically designed for GPU architectures have been addressed. A promising approach is the random ball cover (RBC) proposed by Cayton [7, 8]. The basic principle behind the RBC is a two-tier nearest neighbor search,

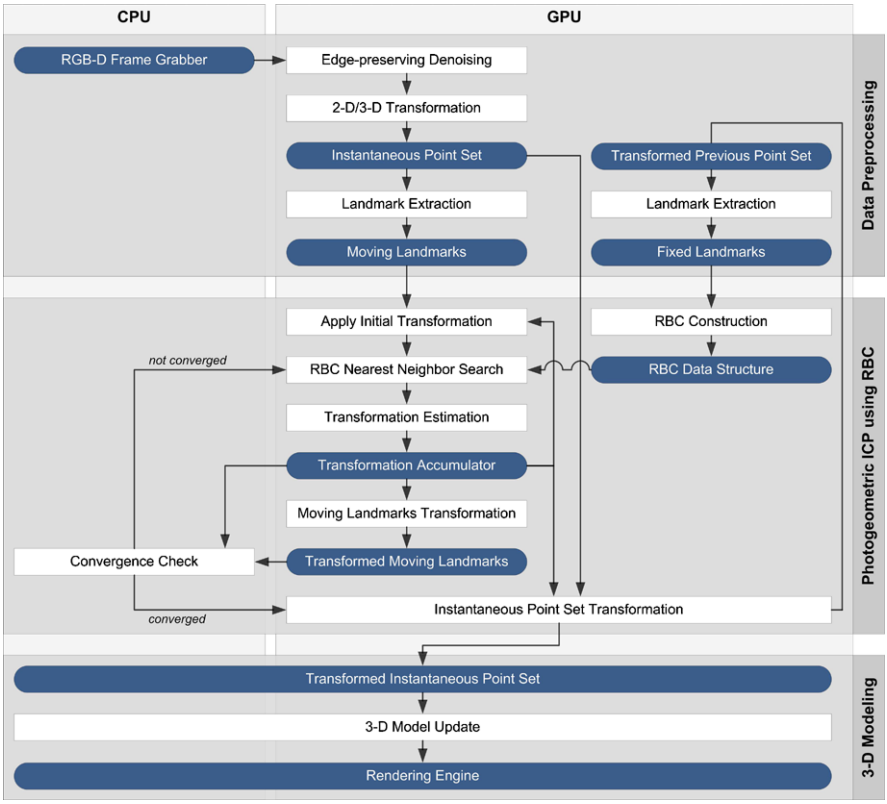


Fig. 2.2 Flowchart of the proposed 3-D scene reconstruction framework. Apart from the camera hardware interface and the ICP control flow management, the corpus of the computational load of both data preprocessing and photogeometric ICP alignment using RBC is outsourced to the GPU

building on the brute-force primitive, to prune the search space. In this work, we adapted the random ball cover data structure and search algorithm, originally proposed for high-dimensional problems, to low-dimensional RGB-D data for accelerating the ICP alignment.

2.3 Methods

The proposed RGB-D mapping and modeling framework is composed of three stages, as depicted in Fig. 2.2. In an initial stage, the sensor data consisting of orthogonal distance measurements and photometric color information are transferred to the GPU where the corpus of the pipeline is executed. On the GPU, first, data preprocessing and the transformation from orthogonal range measurements in the 2-D sensor domain to 3-D world coordinates are performed (Sect. 2.3.1). Second, based on a set of extracted landmarks, the proposed color ICP variant is applied

(Sect. 2.3.2). Our method exploits the arithmetic power of modern GPUs for efficient nearest neighbor search with an inherently parallel data structure and query framework (RBC, Sect. 2.3.3). Third and last, the instantaneous point cloud is attached to the global reconstructed model based on the estimated transformation. We point out that the rigid body transformation is estimated in a frame-to-frame manner, i.e. the pose of the instantaneous frame is estimated by registration against the previous frame. In the remainder of this section, we outline the essential steps of the proposed ICP framework. GPU implementation details are discussed in Sect. 2.4.

2.3.1 Data Preprocessing on the GPU

The Microsoft Kinect device acquires RGB-D data with VGA resolution (640 × 480 px) at 30 Hz. With respect to real-time constraints and regardless of the specific application, this spatial and temporal data density poses a challenge to data processing solutions. Hence, in addition to the actual point cloud alignment, we perform RGB-D data preprocessing on-the-fly on the GPU. First, we apply edge-preserving denoising (e.g. guided image filtering [18, 37]) on the raw depth and RGB data, respectively, as acquired by the Microsoft Kinect sensors. Next, the enhanced depth measurements are transformed to the 3-D world coordinate system. Indeed, for each point $\mathbf{x}_c \in \mathbb{R}^2$ in the camera plane, its depth value $z(\mathbf{x}_c)$ describes a world coordinate position vector $\mathbf{x}_w \in \mathbb{R}^3$. The transformation can be computed independently for each pixel, thus fitting perfectly for parallel processing on the GPU (see Sect. 2.5.2).

Nomenclature Let us introduce the notation for this chapter. Let $\tilde{\mathcal{M}}$ denote a *moving* set of template points $\tilde{\mathcal{M}} = \{\mathbf{m}\}$, where $\mathbf{m} \in \mathbb{R}^6$ concatenates a point's geometric and photometric information $\mathbf{m}_g \in \mathbb{R}^3$ and $\mathbf{m}_p \in \mathbb{R}^3$:

$$\mathbf{m} = \begin{pmatrix} \mathbf{m}_g \\ \mathbf{m}_p \end{pmatrix}. \tag{2.1}$$

The indices g and p denote that only the geometric and photometric part is considered, respectively. In order to compensate for inconsistencies due to changes in illumination and viewpoint direction, the photometric information is transformed to the normalized RGB space [16]:

$$\mathbf{m}_p = (i_\mathrm{r} + i_\mathrm{g} + i_\mathrm{b})^{-1} \begin{pmatrix} i_\mathrm{r} \\ i_\mathrm{g} \\ i_\mathrm{b} \end{pmatrix}, \tag{2.2}$$

where $i_\mathrm{r}, i_\mathrm{g}, i_\mathrm{b}$ denote the intensities of the red, green and blue photometric channel.

In analogy to the moving set of template points $\tilde{\mathcal{M}}$, let $\tilde{\mathcal{F}} = \{\mathbf{f}\}$ denote a *fixed* set of $|\tilde{\mathcal{F}}|$ reference points $\mathbf{f} \in \mathbb{R}^6$, where $\mathbf{f}^\top = (\mathbf{f}_g^\top, \mathbf{f}_p^\top)$.

Landmark Extraction Considering the application of 3-D scene or object modeling using a real-time, hand-held and steadily moved RGB-D device implies that a portion of the scene that was captured in the previous frame $\tilde{\mathcal{F}}$ is no longer visible in the instantaneous data $\tilde{\mathcal{M}}$ and vice versa. Facing these issues, we heuristically discard the set of points that correspond to range measurements at the edge of the 2-D sensor domain in order to improve the robustness of ICP alignment. This clipping is performed in conjunction with the extraction of the sparse sets of ICP landmarks, denoted by $\mathcal{M} \subset \tilde{\mathcal{M}}$ and $\mathcal{F} \subset \tilde{\mathcal{F}}$. In practice, the landmark extraction is performed by sub-sampling the clipped point set.

For the case of 3-D object reconstruction, we apply a dedicated scheme for landmark extraction. Instead of considering the entire scene, we segment the foreground using a depth threshold. From the set of foreground pixels, we then select a set of landmarks.

2.3.2 Photogeometric ICP Framework

Being the state-of-the-art in rigid point cloud alignment [4, 9, 36], the ICP estimates the optimal rigid transformation $(\mathbf{R}, \mathbf{t})$ that brings $\mathcal{M}$ in congruence with $\mathcal{F}$, where $\mathbf{R} \in \mathbb{R}^{3\times3}$ denotes a rotation matrix with $\mathbf{R}^\top = \mathbf{R}^{-1}$, $\det(\mathbf{R}) = 1$ and $\mathbf{t} \in \mathbb{R}^3$ denotes a translation vector. Based on an initial guess $(\mathbf{R}^0, \mathbf{t}^0)$, the ICP scheme iteratively estimates this transformation by minimizing an error metric assigned to repeatedly generated pairs of corresponding landmarks $(\mathbf{m}, \mathbf{y})$ where $\mathbf{m} \in \mathcal{M}$ and $\mathbf{y} \in \mathcal{F}$. In terms of correspondence search, our *photogeometric* ICP variant incorporates both geometric and photometric information. Let us note that competing strategies, including projective data association, typically rely on the pure geometry and cannot incorporate additional information in a straightforward manner. We now outline the essential steps of our photogeometric ICP variant.

In the geometric case, the distance d between an individual moving landmark $\mathbf{m}_g$ and the set of reference landmarks $\mathcal{F}_g = \{\mathbf{f}_g\}$ is defined as

$$d(\mathbf{m}_g, \mathcal{F}_g) = \min_{\mathbf{f}_g \in \mathcal{F}_g} \|\mathbf{f}_g - \mathbf{m}_g\|_2^2, \tag{2.3}$$

where $\|\cdot\|_2$ denotes the Euclidean norm. In order to incorporate the additional photometric information available with modern RGB-D sensors, let us modify the distance metric d:

$$d(\mathbf{m}, \mathcal{F}) = \min_{\mathbf{f} \in \mathcal{F}} \big((1-\alpha)\|\mathbf{f}_g - \mathbf{m}_g\|_2^2 + \alpha\|\mathbf{f}_p - \mathbf{m}_p\|_2^2\big), \tag{2.4}$$

where $\alpha \in [0, 1]$ is a non-negative constant weighting the influence of the photometric information. The benefit of this hybrid approach is that photometric information compensates for regions with non-salient surface topology, and geometric information compensates for faintly textured regions or photometric inconsistencies due to

changes in illumination and viewpoint direction. The landmark $\mathbf{y} \in \mathcal{F}$ yielding the minimum distance to $\mathbf{m}$ is then given by

$$\mathbf{y} = \arg\min_{\mathbf{f} \in \mathcal{F}} \big((1-\alpha) \|\mathbf{f}_g - \mathbf{m}_g\|_2^2 + \alpha \|\mathbf{f}_p - \mathbf{m}_p\|_2^2 \big). \tag{2.5}$$

By assigning a nearest neighbor $\mathbf{y}$ to all $\mathbf{m} \in \mathcal{M}$, a set of nearest neighbors $\mathcal{Y}$ is given as $\mathcal{Y} = \{\mathbf{y}\}$, $\mathbf{y} \in \mathcal{F}$, $|\mathcal{Y}| = |\mathcal{M}|$, and the landmark correspondences can be denoted by $(\mathcal{M}, \mathcal{Y})$. The GPU-based nearest neighbor search framework that we use to establish these landmark correspondences is described in Sect. 2.3.3. Next, based on the landmark correspondences $(\mathcal{M}^k, \mathcal{Y}^k)$ found in the kth ICP iteration, the transformation $(\hat{\mathbf{R}}^k, \hat{\mathbf{t}}^k)$ is estimated by either minimizing a point-to-point error metric in a least-squares sense using a unit quaternion optimizer [21],

$$\big(\hat{\mathbf{R}}^k, \hat{\mathbf{t}}^k\big) = \arg\min_{\mathbf{R}^k, \mathbf{t}^k} \frac{1}{|\mathcal{M}_g^k|} \sum_{\mathcal{M}_g^k, \mathcal{Y}_g^k} \big\| \big(\mathbf{R}^k \mathbf{m}_g^k + \mathbf{t}^k\big) - \mathbf{y}_g^k \big\|_2^2, \tag{2.6}$$

or by minimizing a point-to-plane distance metric [9] using a nonlinear solver,

$$\big(\hat{\mathbf{R}}^k, \hat{\mathbf{t}}^k\big) = \arg\min_{\mathbf{R}^k, \mathbf{t}^k} \frac{1}{|\mathcal{M}_g^k|} \sum_{\mathcal{M}_g^k, \mathcal{Y}_g^k} \Big(\big(\big(\mathbf{R}^k \mathbf{m}_g^k + \mathbf{t}^k\big) - \mathbf{y}_g^k \big)^\top \mathbf{n}_{\mathbf{y}_g^k} \Big)^2. \tag{2.7}$$

Here, $\mathbf{n}_{\mathbf{y}_g^k}$ denotes the surface normal associated with the point $\mathbf{y}_g^k \in \mathcal{F}$. After each iteration, the global solution $(\mathbf{R}, \mathbf{t})$ is accumulated:

$$\mathbf{R} = \hat{\mathbf{R}}^k \mathbf{R}, \qquad \mathbf{t} = \hat{\mathbf{R}}^k \mathbf{t} + \hat{\mathbf{t}}^k, \tag{2.8}$$

and $\mathcal{M}_g^k$ is updated according to $\mathbf{m}_g^k = \mathbf{R}\mathbf{m}_g + \mathbf{t}$. The two stages of first finding the set of nearest neighbors $\mathcal{Y}^k$ and then estimating the optimal transformation for the correspondences $(\mathcal{M}^k, \mathcal{Y}^k)$ are repeated iteratively until a convergence criterion is fulfilled, see Fig. 2.2 and Sect. 2.4.1.

2.3.3 6-D Nearest Neighbor Search Using RBC

The Random Ball Cover (RBC) is a novel data structure for efficient nearest neighbor (NN) search on the GPU proposed by Cayton [7, 8]. By design, it exploits the parallel architecture of modern graphics cards hardware. In particular, both the construction of the RBC and dataset queries are performed using brute-force (BF) primitives. Using techniques known from matrix–matrix multiplication, the BF search can be performed in a highly efficient manner on the GPU. The RBC data structure relies on randomly selected points $\mathbf{r} \in \mathcal{F}$, called *representatives*. Each of them manages a local subset of $\mathcal{F}$. This indirection creates a hierarchy in the database such that a nearest neighbor query is processed by (i) searching the nearest neighbor $\mathbf{r}$ among the set of representatives and (ii) performing another search for the subset of

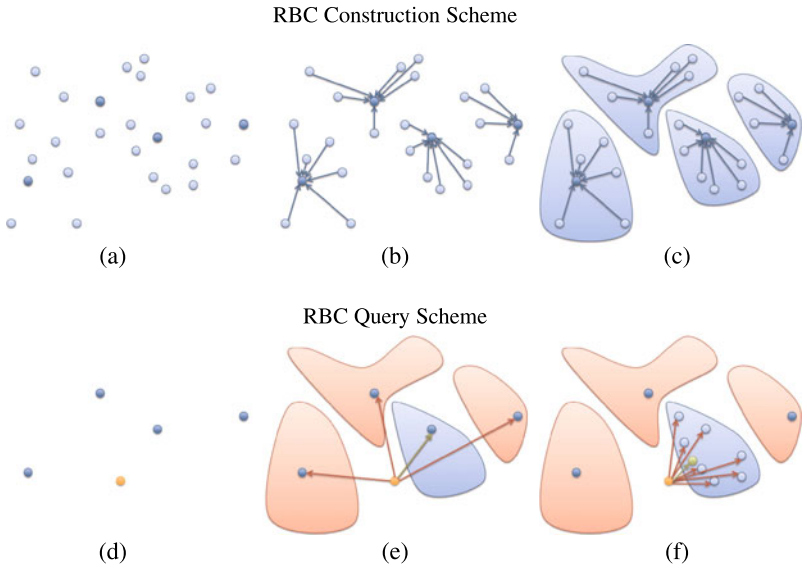


Fig. 2.3 Illustration of the RBC construction (**a**–**c**) and the two-tier nearest neighbor query scheme (**d**–**f**) for the simplified case of 2-D data. (**a**) Selection of a set of representatives $\mathcal{R}$ (*labeled in dark blue*) out of the set of database entries $\mathcal{F}$ (*light blue*). (**b**) Nearest representative search over the set of database entries, to establish a landmark-to-representative mapping. (**c**) Nearest neighbor set of each representative (*shaded in blue*). (**d**) Query data (*orange*) and set of representatives $\mathcal{R}$ (*dark blue*). (**e**) Identification of the closest representative **r**, in a first brute-force (BF) run. (**f**) Identification of the nearest neighbor (*green*) in the subset of entries managed by **r** (*shaded in blue*), in a second BF run

entries managed by **r**. This two-tier approach outperforms a global BF search due to the fact that each of the two successive stages explore a heavily pruned search space.

In this work, we have investigated the fitness of the RBC for acceleration of the 6-D nearest neighbor search of our photogeometric ICP. Optimizing this particular ICP stage is motivated by the fact that it is a major performance bottleneck—see Sect. 2.5.2 and [30].

Cayton proposed two alternative RBC search strategies [8]. The *exact* search is the appropriate choice when the exact nearest neighbor is required. Otherwise, if a small error may be tolerated, the approximate *one-shot* search is typically faster. Originally, in order to set up the *one-shot* data structure, the representatives are chosen at random, and each **r** manages its s closest database elements. Depending on s, points typically belong to more than one representative. However, this implies a sorting of all database entries for each representative—hindering a high degree of parallelization for implementation on the GPU—or the need for multiple BF runs [7]. Hence, we introduce a modified version of the *one-shot* approach that is even further optimized in terms of performance. In particular, we simplified the RBC construction, trading off accuracy against runtime, see Fig. 2.3 (a–c). First, we select a random set of representatives $\mathcal{R} = \{\mathbf{r}\}$ out of the set of fixed points $\mathcal{F}$. Second,

each representative $\mathbf{r}$ is assigned a local subset of $\mathcal{F}$. This is done in an inverse manner by simply computing the nearest representative $\mathbf{r}$ for each point $\mathbf{f} \in \mathcal{F}$. The query scheme of our modified *one-shot* RBC variant is basically consistent with the original approach and can be performed efficiently using two subsequent BF runs [8], see Fig. 2.3 (d–f). First, the closest representative is identified among $\mathcal{R}$. Second, based on the associated subset of entries managed by $\mathbf{r}$, the nearest neighbor is located.

Please note that this modified RBC construction scheme results in an approximate nearest neighbor search being error-prone from a theoretical point of view. In practice, facing the trade-off between accuracy and runtime, we tolerate this approximation, cf. Sect. 2.5.2. Let us further remark that the scheme is not limited to 6-D data but can be applied to data of any dimension. For application in 3-D reconstruction, this potentially allows us to extend the point signature from 6-D to higher dimensions, e.g. appending additional complementary information or local feature descriptors to the raw geometric and photometric measurements acquired by the sensor, cf. [19].

2.4 Implementation Details

In this section, we discuss implementation details and comment on practical issues. In particular, we address the RBC implementation on the GPU.

2.4.1 Details Regarding the ICP Framework

Regarding the quality and robustness of point cloud alignment, we observed a strong impact of outliers that occur in RGB-D data particularly due to sensor noise, quantization, occlusion, and changes in viewpoint direction. Sensor noise and quantization issues are reduced using edge-preserving denoising filters in the preprocessing stage of the framework, recall Fig. 2.2. We typically apply the concept of guided image filtering [18] or median filtering that both can be parallelized in an efficient manner on the GPU [29, 37].

The remaining set of outliers arise from a change in viewpoint direction or occlusion and cannot be eliminated by denoising. To take them into account, we optionally reject low-grade correspondences in the transformation estimation stage. The term *low-grade* is quantified by comparing the distance of a corresponding pair of landmarks (Eq. 2.4) w.r.t. an empirically set threshold δ. The set of low-grade correspondences is re-computed for each ICP iteration and discarded in the subsequent transformation estimation step.

As initialization for the ICP alignment, we incorporate the estimated global transformation $(\mathbf{R}^0, \mathbf{t}^0)$ from the previously aligned frame, see Fig. 2.2, assuming a smooth trajectory of the hand-guided acquisition device. In practice, this speeds up convergence and reconstruction, respectively.

In our implementation, the ICP transformation is estimated by minimizing the point-to-point distance metric (Eq. 2.6). The estimation of the transformation matrix according to Horn [21] is performed on the GPU. Both the computation of the centroids of $\mathcal{F}$ and $\mathcal{M}$ and the summation of the intermediate M-matrix are implemented using the established parallel reduction technique [17]. For details on Horn's scheme we refer to [21]. Note that low-grade correspondences may have been removed from $\mathcal{F}$ and $\mathcal{M}$ at this stage. The resulting eigenvalue problem is solved using the iterative Jacobi scheme on the GPU. This is motivated by practical experience: on the one hand, using a CPU-based implementation of Jacobi's scheme would result in notable host-device and device-host transfer times, depending on the number of ICP iterations. On the other hand, solving the eigenvalue problem on the GPU using Ferrari's closed form solution [26] as proposed by Loop and Blinn [27] would imply a non-negligible number of branches and root calculations that are also performed iteratively in hardware [34].

As ICP convergence criterion we analyze the variation of the estimated transformation over the iterations. In particular, we evaluate the change in translation magnitude and rotation angle w.r.t. heuristically set thresholds of 0.01 mm and 0.001°, respectively.

2.4.2 RBC Construction and Queries on the GPU

Originally designed for offline and high-dimensional data queries, utilizing the RBC for real-time low-dimensional RGB-D mapping requires certain adaptations. We found that the originally proposed RBC construction routine does not satisfy runtime constraints imposed by the frame-rate of modern RGB-D imaging devices. We therefore employ a different RBC construction routine as introduced in Sect. 2.3.3. As a consequence, this implies a query approach that slightly differs from the original proposal. Below, we describe the details and hardware related considerations of our RBC implementation. An illustration of the workflow for RBC construction and query, as well as data interaction, is depicted in Fig. 2.4.

RBC Construction As a first step in the RBC construction, we extract the set of representatives $\mathcal{R} = \{\mathbf{r}\}$ from the given fixed landmarks $\mathcal{F}$. For each landmark $\mathbf{f} \in \mathcal{F}$, we then compute the nearest representative $\mathbf{r}$ by a brute-force search strategy. This can be done efficiently in parallel over the landmarks using block-decomposition techniques known from matrix–matrix multiplication on the GPU. These landmark-to-representative (LR) mappings are subsequently used to (i) set up the RBC *meta information* and (ii) to generate a compact and cache friendly *permuted database* of the original landmarks $\mathcal{F}$ for RBC queries. An illustration is given in Fig. 2.5. For meta information generation, let us note that the number of managed landmarks for each representative can be derived in the LR mapping computation directly by using synchronized counters employing atomic operations. We found this approach more performant compared to a separate approach. Next,

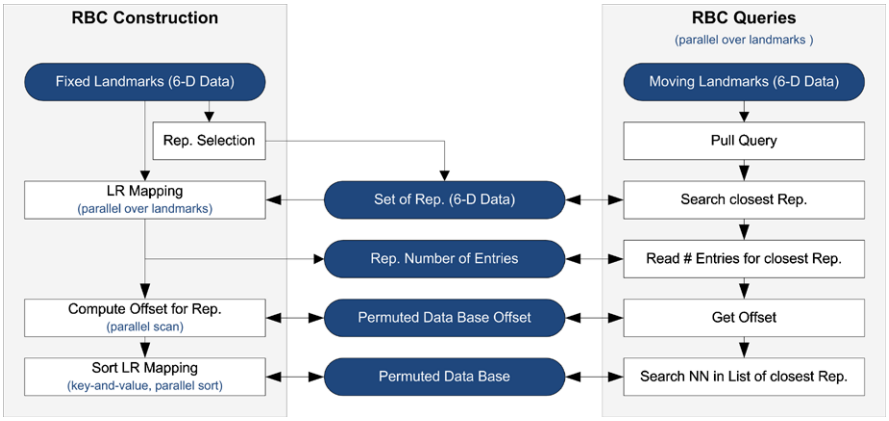


Fig. 2.4 Flowchart describing the GPU workflow and data interaction for RBC construction (*left*) and queries (*right*). Note the high degree of parallelism for both construction and queries. For details on the landmark-to-representative (LR) mapping see Fig. 2.5

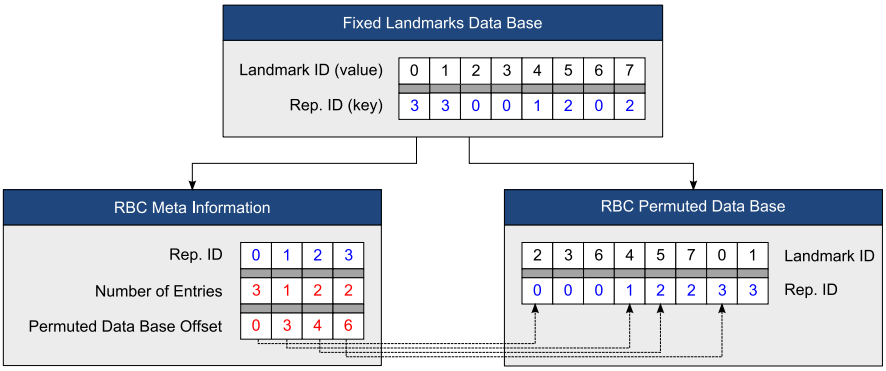


Fig. 2.5 Data structures for RBC construction and queries. Note the differentiation between meta information (*left*) and the permuted database (*right*) to improve cache hit ratio for queries

we compute an offset table by performing a parallel scan [17] on the number of managed entries. This offset table ultimately defines the unique position for each representative's first managed entry in the permuted database. To re-arrange the original data into a cache friendly layout for RBC queries, we perform a key-and-value sort [20] on the LR mappings. Here, a landmark ID denotes the value and the associated representative ID defines the key. By using such a database layout, a representative's managed entries are located in contiguous memory regions, improving cache hit ratio for RBC queries. We note that our approach still requires sorting, however, sorting breaks down to $|\mathcal{F}|$ elements in contrast to $|\mathcal{F}| \cdot |\mathcal{R}|$ entries as originally described [8].

RBC Nearest Neighbor Queries As described in Sect. 2.3.3, RBC queries rely on a two-tier approach—each employing a brute-force search—to prune the search space. The first tier consists of finding the nearest representative $\mathbf{r}$ for each query element by a BF search. This is basically the same procedure as for deriving the LR mappings during RBC construction and can be performed efficiently in parallel over the query elements by using a block-decomposition scheme. The second tier consists of finding the nearest entry managed by the representative $\mathbf{r}$ identified in the first tier. Again, this is done by utilizing a BF search, however, an efficient block-decomposition scheme is not a performant option here. In the first tier this scheme is efficient and possible due to the prior knowledge that all query elements have to visit exactly the same representatives. However, in the second tier, each query element must examine (i) different entries and/or (ii) a different number of entries. Both are given by the entry's nearest representative which in general is not consistent across different query elements. Though sophisticated techniques to implement a block-decomposition-like scheme can be used, in most cases they are counterproductive. We found that due to the computational overhead a potential performance gain is lost. Instead, we employ a simple BF search over a representative's contiguous memory region in the *permuted database* which allows to increase the cache hit ratio and results in lower runtimes.

2.5 Experiments and Results

We have evaluated the proposed framework for on-the-fly 3-D reconstruction and modeling of real data (640 × 480 px, 30 Hz) from a hand-held Microsoft Kinect sensor. Below, first, we present qualitative results for both indoor scene mapping and object reconstruction scenarios, and investigate the influence of the parameter settings (Sect. 2.5.1). Second, being a major focus of this system, we demonstrate its real-time capability in a comprehensive performance study (Sect. 2.5.2). Third, we compare our approximate RBC variant to an exact nearest neighbor search in terms of accuracy (Sect. 2.5.3). For all experiments, the number of representatives was set to $|\mathcal{R}| = \sqrt{|\mathcal{F}|}$ according to Cayton's rule of thumb [8], if not stated otherwise. The ICP transformation was estimated by minimizing the point-to-point distance metric, see Eq. 2.6. The performance study was conducted on an off-the-shelf consumer desktop computer equipped with an NVIDIA GeForce GTX 460 GPU and a 2.8 GHz Intel Core 2 Quad Q9550 CPU. The GPU framework is implemented using CUDA.

2.5.1 Qualitative Results

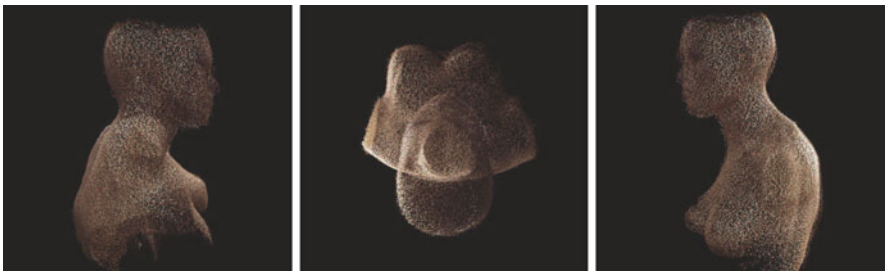
Qualitative results for a scene reconstruction scenario in indoor environments are depicted in Fig. 2.6. The three point cloud sequences were acquired from a static

Fig. 2.6 On-the-fly 3-D scene reconstruction for different types of room. *First row*: bedroom (295 frames). *Second row*: lounge (526 frames). *Third row*: family room (380 frames). For each sequence, the *left column* depicts a bird-eye view of the respective room layout. The *remaining columns* provide a zoom-in for selected regions. All reconstructions were performed using our default parameter settings as stated in Sect. 2.5.1. Note that for visualization of the reconstructed scenes, we rendered a subset of the global model point cloud

observer location by rotating the hand-held sensor around the observer's body axis. RGB-D data were aligned on-the-fly. The different rooms were reconstructed using identical preprocessing pipeline and ICP/RBC parameter settings (default configuration): Edge-preserving denoising (geometric median, geometric and photometric guided image filter), $|\mathcal{F}| = |\mathcal{M}| = 16{,}384$ ICP landmarks, 10 % edge clipping, photogeometric weight $\alpha = 0.8$, no elimination of low-grade correspondences ($\delta \to \infty$).

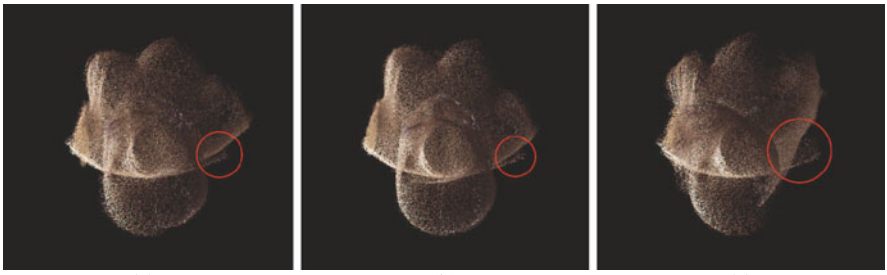
In order to demonstrate the effectiveness of our system for reconstruction of scenes with non-salient 3-D geometry, we refer to Fig. 2.1. Facing a colored poster stuck to a plane wall, the reconstruction could benefit significantly from incorporating the photometric domain as a complementary source of information.

In addition to scene reconstruction, the proposed framework can also be employed for 3-D model digitalization scenarios. Here, the hand-held acquisition device is moved around an object to acquire RGB-D data from different perspectives while continuously merging the data into a global model using the proposed framework. As stated in Sect. 2.3.1, for the case of 3-D object reconstruction, we select the set of landmarks from a defined foreground region only. Background data points



(a) (b) (c)

Fig. 2.7 3-D reconstruction of a female torso model, where the hand-held acquisition device was moved around the model in a 360°-fashion in order to cover the entire object. RGB-D data from different perspectives (525 frames) were merged into a global model on-the-fly. For visualization of the reconstructed model, we rendered a subset of the global model point cloud



(a) (b) (c)

Fig. 2.8 Influence of parameter settings, again for the reconstruction of the female torso model, cf. Fig. 2.7(**b**). Subfigure (**a**) depicts the reconstruction result when edge-preserving denoising was disabled. In subfigures (**b**, **c**), we increased the low-grade correspondence threshold to $\delta = 10$ mm (**b**) and $\delta \to \infty$ (**c**), leading to decreasing reconstruction quality. For instance, please note the labeled issues regarding loop closure

that are located beyond a certain depth level are ignored within the ICP alignment procedure. For object reconstruction, our default settings are: Edge-preserving denoising (geometric guided image filter), $|\mathcal{F}| = |\mathcal{M}| = 16{,}384$ ICP landmarks, $\alpha = 0$ (invariance to illumination issues), $\delta = 3$ mm.

Qualitative results for model reconstruction are depicted in Fig. 2.7. Note that by setting a rather rigorous threshold for discarding low-grade correspondences ($\delta = 3$ mm), our framework is able to achieve a sufficient degree of loop closure although it relies on a frame-to-frame alignment.

The influence of different parameter settings is investigated in Fig. 2.8. As a baseline, we refer to the reconstruction results in Fig. 2.7(b) using our default settings (guided image filter denoising, $\delta = 3$ mm). Disabling edge-preserving denoising increases issues regarding loop closure, see Fig. 2.8(a). Relaxing the low-grade correspondence threshold δ results in similar effects (Fig. 2.8(b), $\delta = 10$ mm) and can eventually lead to model reconstruction failures (Fig. 2.8(c), $\delta \to \infty$).

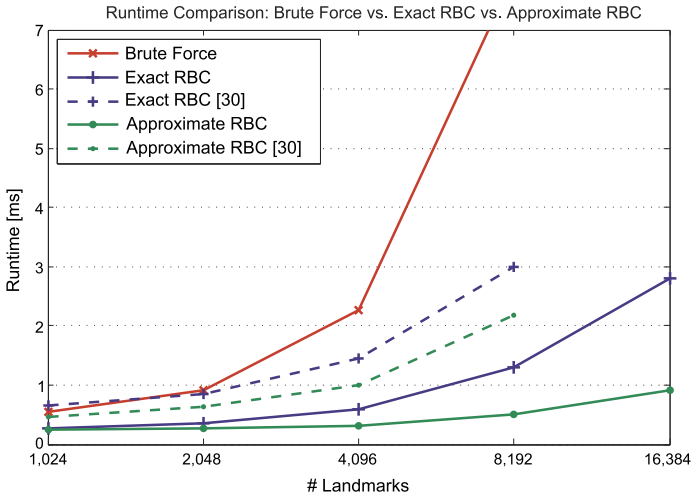


Fig. 2.9 Comparison of the average runtime for a single ICP iteration based on a GPU brute-force primitive, the exact RBC and our optimized approximate RBC variant as described in Sect. 2.3.3, for increasing number of landmarks. The number of representatives is chosen according to Cayton's rule of thumb, $|\mathcal{R}| = \sqrt{|\mathcal{F}|}$. Note that our modified approximate RBC approach outperforms the exact RBC up to a factor of 3. The BF primitive scales quadratically w.r.t. the number of landmarks

2.5.2 *Performance Study*

The corpus of the proposed framework including both preprocessing and RGB-D mapping is executed on the GPU, recall Fig. 2.2. This section presents quantitative results for individual modules of the framework.

Preprocessing Pipeline Edge-preserving image filtering is parallelized in an efficient manner on the GPU [29, 37]. The computation of 3-D world coordinates from the measured depth values requires less than 1 ms for Microsoft Kinect data of VGA resolution, including CPU-GPU memory transfer of the RGB-D data. The subsequent edge clipping and landmark extraction for $\mathcal{M}$ and $\mathcal{F}$ in scene reconstruction scenarios depends on $|\mathcal{M}| = |\mathcal{F}|$, denoting the number of landmarks (LMs), with typical runtimes of less than 0.3 ms. Let us conclude that runtimes for data preprocessing assume a minor role. As we target scene reconstruction in the first place, landmark extraction for object reconstruction scenarios including foreground segmentation and random landmark selection was implemented on the CPU with a runtime of about 5 ms, as proof-of-concept.

ICP Using RBC Being the cornerstone of our framework, we have investigated the performance of our GPU-based ICP/RBC implementation in detail. A single ICP iteration consists of three steps: (i) nearest neighbor search using RBC, (ii) transformation estimation and (iii) application of the transformation. With an increasing

Table 2.1 Runtimes [ms] for the construction of the RBC data structure ($t_{\mathrm{RBC,C}}$) and ICP execution for reconstructing a typical indoor scene, for varying number of landmarks. In the *first rows*, average runtimes for our default setting $|\mathcal{R}| = \sqrt{|\mathcal{F}|}$ are given. In the *second rows*, we state performance numbers for $|\mathcal{R}|$ being optimized in terms of runtime. Note that optimizing runtime comes with a loss in accuracy, cf. Fig. 2.10. We state both the runtime for a single ICP iteration (t_{ICP}) and typical total ICP runtimes t_{tot} (including RBC construction) for 10 and 20 iterations, respectively

# Landmarks	$\lvert\mathcal{R}\rvert$	$t_{\mathrm{RBC,C}}$ [ms]	t_{ICP} [ms]	t_{tot} (10 its) [ms]	t_{tot} (20 its) [ms]
1,024	$\sqrt{\lvert\mathcal{F}\rvert} = 32$	0.58	0.25	3.13	5.68
1,024	128	0.59	0.12	1.79	3.00
2,048	$\sqrt{\lvert\mathcal{F}\rvert} = 45$	0.60	0.27	3.31	6.03
2,048	128	0.60	0.14	2.02	3.44
4,096	$\sqrt{\lvert\mathcal{F}\rvert} = 64$	0.63	0.32	3.80	6.97
4,096	128	0.67	0.21	2.76	4.86
8,192	$\sqrt{\lvert\mathcal{F}\rvert} = 91$	0.76	0.50	5.80	10.82
8,192	256	1.22	0.40	5.22	9.22
16,384	$\sqrt{\lvert\mathcal{F}\rvert} = 128$	0.90	0.91	9.96	19.07
16,384	256	1.49	0.78	9.25	17.04

number of landmarks, the nearest neighbor search dominates the runtime considerably [30]. Hence, we have put emphasis on optimizing the RBC construction and query performance. Note that for all subsequent performance evaluations, runtimes where averaged over several successive runs.

A comparison of absolute runtimes for a single ICP iteration is presented in Fig. 2.9. Our modified approximate RBC outperforms both a BF search and our reference implementation of Cayton's exact RBC. Note that the BF search scales quadratically with the number of landmarks. Our approximate RBC variant outperforms the exact RBC implementation up to a factor of 3. Compared to previous work by the authors [30], significant runtime speedups were achieved using the permuted database and its cache friendly layout as detailed in Sect. 2.4.2.

Typical scene reconstruction runtimes of the method are given in Table 2.1. From our experiments in indoor scene mapping, we observed the ICP to converge after 10–20 iterations using the stopping criterion described in Sect. 2.4.1. Hence, as an overall performance indicator, let us refer to the runtime of 19.1 ms for 16,384 landmarks, $|\mathcal{R}| = \sqrt{|\mathcal{F}|}$, for 20 iterations.

2.5.3 Approximate RBC

As motivated in Sect. 2.3.3, our approximate RBC construction and nearest neighbor search trades exactness for runtime speedup. We quantitatively investigated the error

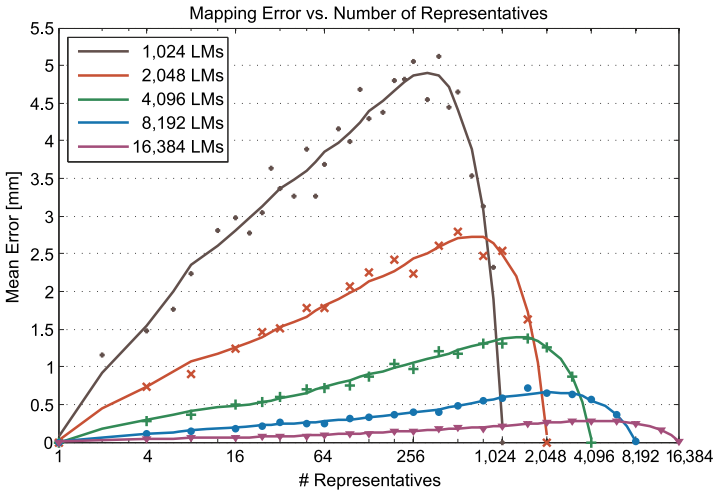


Fig. 2.10 Evaluation of the influence of $|\mathcal{R}|$ on mapping accuracy, compared to an exact BF search, for varying number of landmarks. Given is the mean Euclidean distance [mm] between the mapped points $\hat{\mathbf{m}}_{\mathrm{RBC}}$ and $\hat{\mathbf{m}}_{\mathrm{BF}}$. Increasing the number of landmarks decreases the error. The graph shows both discretized measurements and a trendline for each setting. Note the semi-log scale

that results from our approximate nearest neighbor search compared to an exact BF scheme, considering the aligned point clouds $\hat{\mathcal{M}}_{\mathrm{RBC}}$ and $\hat{\mathcal{M}}_{\mathrm{BF}}$, see Fig. 2.10. The error measures the mean pointwise Euclidean distance [mm] between the points $\hat{\mathbf{m}}_{\mathrm{RBC}}$ and $\hat{\mathbf{m}}_{\mathrm{BF}}$, being transformed w.r.t. different estimations for $(\mathbf{R}, \mathbf{t})$. With an increasing number of representatives $|\mathcal{R}|$, the mapping error rises increasingly until dropping sharply when approaching $|\mathcal{R}| = |\mathcal{F}|$. In general, increasing the number of landmarks decreases the error. Please note that both situations of $|\mathcal{R}| = 1$ and $|\mathcal{R}| = |\mathcal{F}|$ correspond to a BF search, hence yielding an identical transformation/mapping estimate and a mean error of zero.

In order to further illustrate the impact of the relation between the number of landmarks and representatives on reconstruction accuracy, we refer to Fig. 2.11. For $|\mathcal{R}| \ll |\mathcal{F}|$, decreasing $|\mathcal{R}|$ with a fixed number of landmarks reduces the error. This results from our approximate RBC construction scheme, where the probability of erroneous nearest neighbor assignments increases with the number of representatives. Again, increasing the number of landmarks decreases the error. We remark that by using our default configuration (16,384 LMs, $|\mathcal{R}| = \sqrt{|\mathcal{F}|}$), the mapping error is less than 0.25 mm. This is an acceptable scale for the applications considered in this work.

Furthermore, we have related the runtime per ICP iteration to $|\mathcal{R}|$, see Fig. 2.12. Apart from the runtime minimum that is located around $|\mathcal{R}| = 2\sqrt{|\mathcal{F}|}$, the computational load rises when increasing or decreasing the number of representatives. Simultaneously, the error decreases, recall Fig. 2.10. Hence, the application-related requirements in terms of runtime and accuracy motivates the choice of $|\mathcal{R}|$. Together, Figs. 2.10–2.12 illustrate the trade-off between error and runtime.

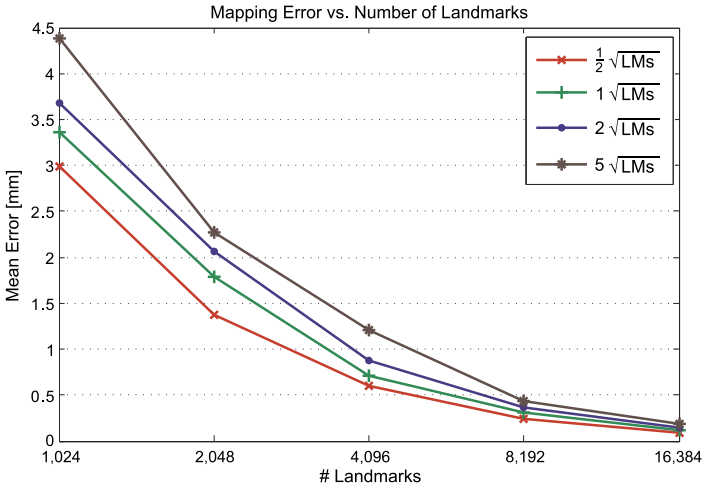


Fig. 2.11 Investigation of the mean mapping error vs. number of landmarks, for varying $|\mathcal{R}|$. Here, the analysis is restricted to $|\mathcal{R}| \ll |\mathcal{F}|$. Note that decreasing $|\mathcal{R}|$ with a fixed number of landmarks reduces the error

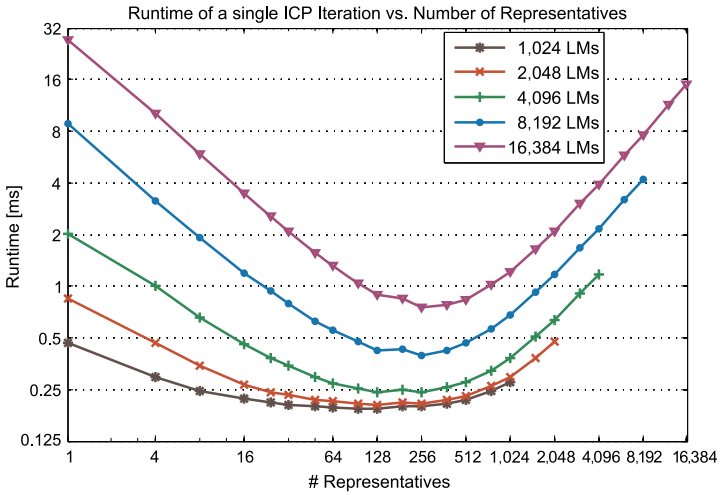


Fig. 2.12 Runtimes of a single ICP iteration, for varying number of landmarks and representatives. The runtime minimum is located around $|\mathcal{R}| = 2\sqrt{|\mathcal{F}|}$. Note the logarithmic scale

2.6 Discussion and Conclusions

In this chapter, we have proposed a GPU framework for real-time mapping and modeling of textured point cloud streams enabling on-the-fly 3-D reconstruction with modern RGB-D imaging devices. Our quantitative RBC experiments demonstrate that using a data structure which is specifically designed to exploit the parallel

computing power of GPUs is beneficial even for low-dimensional (6-D) data. Using our optimized approximate RBC for the photogeometric nearest neighbor search, our system achieves reconstruction runtimes of less than 20 ms on an off-the-shelf consumer GPU in a frame-to-frame scenario.

The proposed framework was evaluated using a point-to-point metric for estimating the transformation within ICP. In general, minimizing a point-to-plane distance metric holds advantages over the point-to-point approach as it allows the surfaces described by $\mathcal{M}$ and $\mathcal{F}$ to slide over each other [9], avoiding snap-to-grid effects. However, solving the corresponding optimization problem as denoted in Eq. 2.7 would require an iterative scheme. We did not observe a negative impact on the reconstruction results using the point-to-point approach in our experiments.

Compared to a conventional ICP that relies on the pure 3-D geometry [4, 9], incorporating photometric appearance as a complementary source of information is advantageous in cases of non-salient surface topology, recall Fig. 2.1 and the experimental results in related work [24]. Approaches that combine dense geometric point associations with a sparse set of correspondences derived from local photometric features are limited to interactive frame-rates, as feature extraction is computationally expensive even if performed on the GPU [19]. In contrast, our approach evaluates both geometric and photometric information in a direct and dense manner, cf. [11, 24, 25]. We found that incorporating photometric appearance in such an elementary manner gives the best compromise between reconstruction robustness and runtime performance. Nonetheless, the proposed scheme using the RBC for efficient nearest neighbor queries on the GPU can be potentially extended to higher-dimensional point signatures.

Ongoing work includes the implementation of a multi-resolution ICP alignment scheme in order to improve the convergence behavior, and the transition from frame-to-frame to frame-to-model registration using an implicit surface model [10]. Furthermore, an automatic scene-dependent weighting of the photogeometric weight α by low-level analysis of the depth image as part of the preprocessing stage will be subject of our upcoming research.

Acknowledgements S. Bauer and J. Wasza gratefully acknowledge the support by the European Regional Development Fund (ERDF) and the Bayerisches Staatsministerium für Wirtschaft, Infrastruktur, Verkehr und Technologie (StMWIVT), in the context of the R&D program IuK Bayern under Grant No. IUK338. Furthermore, this research was supported by the Graduate School of Information Science in Health (GSISH) and the TUM Graduate School.

References

1. Akenine-Möller, T., Haines, E., Hoffman, N.: Real-Time Rendering, 3rd edn. AK Peters, Natick, MA (2008)
2. Bailey, T., Durrant-Whyte, H.: Simultaneous localization and mapping (SLAM): part II. IEEE Robot. Autom. Mag. **13**(3), 108–117 (2006)
3. Bauer, S., Wasza, J., Haase, S., Marosi, N., Hornegger, J.: Multi-modal surface registration for markerless initial patient setup in radiation therapy using Microsoft's Kinect sensor. In:

International Conference on Computer Vision—Workshop on Consumer Depth Cameras for Computer Vision, pp. 1175–1181 (2011)
4. Besl, P., McKay, N.: A method for registration of 3-D shapes. IEEE Trans. Pattern Anal. Mach. Intell. **14**(2), 239–256 (1992)
5. Blais, G., Levine, D.M.: Registering multiview range data to create 3-D computer objects. IEEE Trans. Pattern Anal. Mach. Intell. **17**(8), 820–824 (1995)
6. Castaneda, V., Mateus, D., Navab, N.: SLAM combining ToF and high-resolution cameras. In: IEEE Workshop on Applications of Computer Vision, pp. 672–678 (2011)
7. Cayton, L.: A nearest neighbor data structure for graphics hardware. In: International Workshop on Accelerating Data Management Systems Using Modern Processor and Storage Architectures (2010)
8. Cayton, L.: Accelerating nearest neighbor search on manycore systems. CoRR arXiv:1103.2635 (2011)
9. Chen, Y., Medioni, G.: Object modelling by registration of multiple range images. Image Vis. Comput. **10**(3), 145–155 (1992)
10. Curless, B., Levoy, M.: A volumetric method for building complex models from range images. In: Conference on Computer Graphics and Interactive Techniques, SIGGRAPH, pp. 303–312. ACM, New York (1996)
11. Druon, S., Aldon, M., Crosnier, A.: Color constrained ICP for registration of large unstructured 3D color data sets. In: IEEE International Conference on Information Acquisition, pp. 249–255 (2006)
12. Engelhard, N., Endres, F., Hess, J., Sturm, J., Burgard, W.: Real-time 3D visual SLAM with a hand-held RGB-D camera. In: RGB-D Workshop on 3D Perception in Robotics, European Robotics Forum (2011)
13. Fioraio, N., Konolige, K.: Realtime visual and point cloud SLAM. In: RGB-D Workshop: Advanced Reasoning with Depth Cameras, Robotics Science and Systems Conference (2011)
14. Garcia, J., Zalevsky, Z.: Range mapping using speckle decorrelation. US Patent No. 7433024 (2008)
15. Garcia, V., Debreuve, E., Barlaud, M.: Fast k nearest neighbor search using GPU. In: IEEE Conference on Computer Vision and Pattern Recognition—Workshop on Computer Vision on GPU (2008)
16. Gevers, T., Smeulders, A.W.: Color-based object recognition. Pattern Recognit. **32**(3), 453–464 (1999)
17. Harris, M., Sengupta, S., Owens, J.D.: Parallel prefix sum (scan) with CUDA. In: GPU Gems 3, pp. 851–876. Addison-Wesley, Reading (2007)
18. He, K., Sun, J., Tang, X.: Guided image filtering. In: European Conference on Computer Vision, pp. 1–14 (2010)
19. Henry, P., Krainin, M., Herbst, E., Ren, X., Fox, D.: RGB-D mapping: using depth cameras for dense 3D modeling of indoor environments. In: International Symposium on Experimental Robotics (2010)
20. Hoberock, J., Bell, N.: Thrust: a parallel template library (2010). URL http://code.google.com/p/thrust/. Version 1.3.0
21. Horn, B.: Closed-form solution of absolute orientation using unit quaternions. J. Opt. Soc. Am. **4**(4), 629–642 (1987)
22. Huhle, B., Jenke, P., Strasser, W.: On-the-fly scene acquisition with a handy multi-sensor system. Int. J. Intell. Syst. Technol. Appl. **5**, 255–263 (2008)
23. Izadi, S., Newcombe, R.A., Kim, D., Hilliges, O., Molyneaux, D., Hodges, S., Kohli, P., Shotton, J., Davison, A.J., Fitzgibbon, A.W.: KinectFusion: real-time dynamic 3D surface reconstruction and interaction. In: ACM Symposium on User Interface Software and Technology, p. 23 (2011)
24. Johnson, A., Kang, S.B.: Registration and integration of textured 3-D data. In: International Conference on Recent Advances in 3-D Digital Imaging and Modeling, pp. 234–241 (1997)

25. Joung, J.H., An, K.H., Kang, J.W., Chung, M.J., Yu, W.: 3D environment reconstruction using modified color ICP algorithm by fusion of a camera and a 3D laser range finder. In: IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 3082–3088 (2009)
26. Korn, G.A., Korn, T.M.: Mathematical Handbook for Scientists and Engineers: Definitions, Theorems, and Formulas for Reference and Review. Dover, New York (2000)
27. Loop, C., Blinn, J.: Real-time GPU rendering of piecewise algebraic surfaces. ACM Trans. Graph. **25**(3), 664–670 (2006)
28. May, S., Droeschel, D., Holz, D., Fuchs, S., Malis, E., Nüchter, A., Hertzberg, J.: Three-dimensional mapping with time-of-flight cameras. J. Field Robot. **26**, 934–965 (2009)
29. McGuire, M.: A fast, small-radius GPU median filter. In: ShaderX6, pp. 165–173. Charles River Media (2008)
30. Neumann, D., Lugauer, F., Bauer, S., Wasza, J., Hornegger, J.: Real-time RGB-D mapping and 3-D modeling on the GPU using the random ball cover data structure. In: International Conference on Computer Vision—Workshop on Consumer Depth Cameras for Computer Vision, pp. 1161–1167 (2011)
31. Newcombe, R.A., Izadi, S., Hilliges, O., Molyneaux, D., Kim, D., Davison, A.J., Kohli, P., Shotton, J., Hodges, S., Fitzgibbon, A.W.: KinectFusion: real-time dense surface mapping and tracking. In: IEEE International Symposium on Mixed and Augmented Reality, pp. 127–136 (2011)
32. Nüchter, A., Surmann, H., Lingemann, K., Hertzberg, J., Thrun, S.: 6D SLAM with an application in autonomous mine mapping. In: IEEE International Conference on Robotics and Automation, vol. 2, pp. 1998–2003 (2004)
33. Qiu, D., May, S., Nüchter, A.: GPU-accelerated nearest neighbor search for 3D registration. In: International Conference on Computer Vision Systems, pp. 194–203. Springer, Berlin (2009)
34. Reis, G., Zeilfelder, F., Hering-Bertram, M., Farin, G.E., Hagen, H.: High-quality rendering of quartic spline surfaces on the GPU. IEEE Trans. Vis. Comput. Graph. **14**(5), 1126–1139 (2008)
35. Rusinkiewicz, S., Hall-Holt, O., Levoy, M.: Real-time 3D model acquisition. ACM Trans. Graph. **21**(3), 438–446 (2002)
36. Rusinkiewicz, S., Levoy, M.: Efficient variants of the ICP algorithm. In: International Conference on 3-D Digital Imaging and Modeling, pp. 145–152 (2001)
37. Wasza, J., Bauer, S., Haase, S., Hornegger, J.: Real-time preprocessing for dense 3-D range imaging on the GPU: defect interpolation, bilateral temporal averaging and guided filtering. In: International Conference on Computer Vision—Workshop on Consumer Depth Cameras for Computer Vision, pp. 1221–1227 (2011)

http://www.springer.com/978-1-4471-4639-1

Consumer Depth Cameras for Computer Vision
Research Topics and Applications
Fossati, A.; Gall, J.; Grabner, H.; Ren, X.; Konolige, K.
(Eds.)
2013, XVI, 210 p., Hardcover
ISBN: 978-1-4471-4639-1