Computer Vision and Image Understanding 115 (2011) 1598–1609

Contents lists available at ScienceDirect

Computer Vision and Image Understanding

journal homepage: www.elsevier.com/locate/cviu

Stochastic global optimization for robust point set registration

Chavdar Papazov *, Darius Burschka

Department of Computer Science, Technische Universität München, Boltzmannstr. 3, 85748 Garching, Germany

ARTICLE INFO

Article history:
Available online 23 July 2011

Keywords:
Rigid registration
Robust cost function
Stochastic global optimization
Generalized BSP tree
Hierarchical decomposition of $SO(3)$
Uniform sampling from spherical boxes

ABSTRACT

In this paper, we propose a new algorithm for pairwise rigid point set registration with unknown point correspondences. The main properties of our method are noise robustness, outlier resistance and global optimal alignment. The problem of registering two point clouds is converted to a minimization of a nonlinear cost function. We propose a new cost function based on an inverse distance kernel that significantly reduces the impact of noise and outliers. In order to achieve a global optimal registration without the need of any initial alignment, we develop a new stochastic approach for global minimization. It is an adaptive sampling method which uses a generalized BSP tree and allows for minimizing nonlinear scalar fields over complex shaped search spaces like, e.g., the space of rotations. We introduce a new technique for a hierarchical decomposition of the rotation space in disjoint equally sized parts called spherical boxes. Furthermore, a procedure for uniform point sampling from spherical boxes is presented. Tests on a variety of point sets show that the proposed registration method performs very well on noisy, outlier corrupted and incomplete data. For comparison, we report how two state-of-the-art registration algorithms perform on the same data sets.

1. Introduction and related work

Point set registration is a fundamental problem in computational geometry with applications in the fields of computer vision, computer graphics, image processing and many others. The problem can be formulated as follows. Given two finite point sets $\mathbf{M} = \{\mathbf{x}_1, \ldots, \mathbf{x}_m\} \subset \mathbb{R}^3$ and $\mathbf{D} = \{\mathbf{y}_1, \ldots, \mathbf{y}_n\} \subset \mathbb{R}^3$ find a mapping $T : \mathbb{R}^3 \to \mathbb{R}^3$ such that the point set $T(\mathbf{D}) = \{T(\mathbf{y}_1), \ldots, T(\mathbf{y}_n)\}$ is optimally aligned in some sense to $\mathbf{M}$. $\mathbf{M}$ is referred to as the model point set (or just the model) and $\mathbf{D}$ is termed the data point set. Points from $\mathbf{M}$ and $\mathbf{D}$ are called model points and data points, respectively.

If T is a rigid transform, i.e., $T(\mathbf{x}) = R\mathbf{x} + \mathbf{t}$ for a rotation matrix R and a translation vector $\mathbf{t}$, we have to solve a rigid point set registration problem. This special case is of major importance for the tasks of object recognition, tracking, localization and mapping, and object modeling, just to name a few. The problem is especially hard when no initial pose estimation is available, the point sets are noisy, corrupted by outliers and incomplete and no correspondences between the points of the input sets are known. In Fig. 1, a model and a data set are shown before and after rigid registration.

1.1. Rigid point set registration

One class of rigid point set registration approaches consists of methods designed to solve the initial pose estimation problem.[1] These methods compute a (more or less) coarse alignment between the point sets without making any assumptions about their initial position and orientation in space. Classic initial pose estimators are the generalized Hough transform [2], geometric hashing [3] and pose clustering [4]. These algorithms are guaranteed to find the optimal alignment between the input point sets. However, because of their high computational cost and/or high memory requirements, these methods are only applicable to small data sets.

Johnson et al. introduced in their work [5] local geometric descriptors, called spin images, and used them for pose estimation and object recognition. The presented results are impressive, but no tests with noisy or outlier corrupted data were performed. Gelfand et al. [6] developed a local descriptor which performs well under artificially created noisy conditions, but still, defining robust local descriptors in the presence of significant noise and a large amount of outliers remains a difficult task.

A more recent approach to the initial pose estimation problem is the robust 4PCS algorithm introduced by Aiger et al. [7]. It is an efficient randomized generate-and-test approach. It selects an appropriate quadruple $\mathbf{B}$ (called a basis) of nearly coplanar points from the model set $\mathbf{M}$ and computes the optimal rigid transform

* Corresponding author.
E-mail addresses: papazov@in.tum.de (C. Papazov), burschka@in.tum.de (D. Burschka).

[1] Pose = position (translation) + orientation (rotation).

1077-3142/$ - see front matter © 2011 Elsevier Inc. All rights reserved.
doi:10.1016/j.cviu.2011.05.008

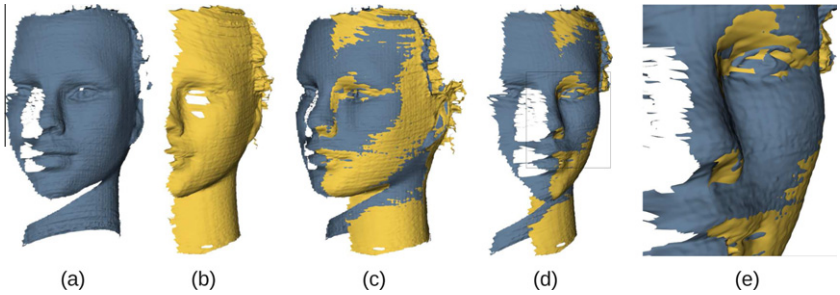

Fig. 1. Pairwise rigid point set registration obtained with our method. The input point sets, model and data, are shown in (a) and (b), respectively. Although rendered as meshes no surface information (like, e.g., normals) is used for the registration. Note that the scans are noisy and only partially overlapping. (c) and (d) Our registration result (shown from two different viewpoints) obtained without noise filtering, local ICP refinement [1] or any assumptions about the initial pose of the input scans. (e) A closer view of the part marked by the rectangle in (d). Observe the high quality of the alignment.

between **B** and each of the potential bases in the data set **D** and chooses the best one. In order to achieve high probability for success, the procedure is repeated several times for different bases $\mathbf{B} \subset \mathbf{M}$. Note, however, that the rigid transform, found by the algorithm, is optimal only for the two bases (i.e., for eight points). In contrast to this, the rigid transform we compute is optimal for all points of the input sets and thus we expect to achieve higher accuracy than the 4PCS algorithm. This is further validated in the experimental results in Section 5 of this paper.

Since the accuracy of the pose computed by the above mentioned methods is insufficient for many applications, an additional pose refinement step needs to be performed. The pose refining algorithms represent another class of registration approaches. The most popular one is the Iterative Closest Point (ICP) algorithm. Since its introduction by Chen and Medioni [8], and Besl and McKay [1], a variety of improvements has been proposed in the literature. A good summary as well as results in acceleration of ICP algorithms have been given by Rusinkiewicz and Levoy [9]. A major drawback of ICP and all its variants is that they assume a good initial guess for the pose of the data point set (with respect to the model). This pose is improved in an iterative fashion until an optimal rigid transform is found. The quality of the solution heavily depends on the initial guess. Furthermore, the methods compared by Rusinkiewicz and Levoy [9] use local surface features like surface normals which cannot be computed very reliably in the presence of noise.

Recently, a variety of registration algorithms based on robust statistics has been proposed. Granger and Pennec [10] formulated the rigid point set registration as a general maximum likelihood estimation problem which they solved using expectation maximization principles. Tsin and Kanade [11] introduced the kernel correlation approach as an extension of the well-known 2D image correlation technique to point sets. The model and data sets are represented by a collection of kernel functions each one centered at a model/data point. If each point in the model set has a close counterpart in the data set the kernel correlation value is large. Thus the registration problem is converted to the maximization of the kernel correlation of the input point sets. An extension of this approach through a Gaussian mixture model was proposed by Jian and Vemuri [12]. Instead of using one-to-one correspondences between the points of the input sets, the above cited methods work with multiple, weighted correspondences. Although this significantly widens the basin of convergence the resulting computational cost limits the applicability of the algorithms to small point sets only [13].

A further class of rigid registration methods is based on particle filtering. Ma and Ellis [14] pioneered the use of the unscented particle filter for registration of surfaces in the context of computer-assisted surgery. A major limitation of the method is its running time: it takes 1.5 s for a data set consisting of 15 points. Moreover, outlier robustness was not addressed by the authors. Further interesting approaches from this class are the algorithm of Moghari and Abolmaesumi [15] which is based on the unscented Kalman filter and the point set registration method via particle filtering and stochastic dynamics introduced by Sandhu et al. [16]. Although these algorithms have a band of convergence significantly wider than the one of local optimizers, they still depend on the initial alignment of the point sets.

1.2. Optimization-based point set registration

Solving the registration problem by minimizing a cost function with a general-purpose optimizer has already been introduced in the literature. Depending on the choice of either a global or a local optimization procedure the corresponding registration approach belongs to the class of initial pose estimators or pose refining methods, respectively.

Breuel [17] used a deterministic branch-and-bound method to globally maximize a quality measure which counts the number of data points a given rigid transform brings within an ϵ-neighborhood of some model point. Although this method always finds the global optimal solution its computational cost seems to be very high since only planar rigid transforms (with three degrees of freedom) were considered.

Olsson et al. [18] also used a deterministic branch-and-bound algorithm to globally minimize the sum of squared distances between corresponding entities (points, lines or planes) in **M** and **D**. This method is guaranteed to find the global optimal solution, however, at a high computational cost: a problem consisting of 10 point-to-plane, 4 point-to-line and 4 point-to-point correspondences is solved in about 10 s. Furthermore, when applied in the case of point set registration, the correspondences between the points have to be known in advance which is seldom the case in a real world setting.

Another deterministic solver based on Lipschitz global optimization theory was introduced by Li and Hartley [19]. On the positive side, the method does not assume any known correspondences across the point sets and it always solves the problem in a globally optimal way. Unfortunately, the algorithm is very costly (about 18 minutes for input sets consisting of 200 points each) and it is based on some unrealistic assumptions: (i) the model and data sets have exactly the same number of points, (ii) there are no outliers and (iii) there is no missing data, i.e., there is a 100% overlap between model and data.

Mitra et al. [20], Pottmann et al. [21] and Fitzgibbon [22] also formulated the registration problem as a minimization of a geometric cost function. For its minimization, however, a local

optimization method is used. This results in the already mentioned strong dependence on a good initial transform estimation.

1.3. Stochastic optimization

Stochastic optimization has received considerable attention in the literature over the last three decades. Much of the work has been devoted to the theory and applications of simulated annealing (SA in the following) as a minimization technique [23–25]. A comprehensive overview of this field is given in [26]. A major property of SA algorithms is their "willingness" to explore regions around points in the search space at which the objective function takes values greater than the current minimum [27]. This is what makes SA algorithms able to escape from local minima and makes them suitable for global minimization. A known drawback of SA algorithms is the fact that they waste a lot of iterations in generating candidate points, evaluating the objective function at these points, and finally rejecting them [26]. In order to reduce the number of rejections, Bilbro and Snyder [28] select candidate points from "promising" regions of the search space, i.e., from regions in which the objective function is likely to have low values. They achieve this by adapting a k-d tree to the objective function each time a new candidate point is accepted. If, however, the current point is rejected, the tree remains unchanged. This is a considerable waste of computation time since the information gained by the (expensive) evaluation of the objective function is not used. In contrast to this, our algorithm adapts a generalized BSP tree at every iteration and thus uses all the information collected during the minimization. Furthermore, the use of a generalized BSP tree allows for a minimization over complex shaped spaces and not only over rectangular regions as in the case of [28].

1.4. Contributions and overview

Our registration algorithm aims to robustly solve the initial pose estimation problem in the case of noisy, outlier corrupted and incomplete point sets with unknown correspondences between the points. Our main contributions are (i) a noise and outlier resistant cost function, (ii) a stochastic approach for its global minimization, (iii) a technique for a hierarchical rotation space decomposition in disjoint parts of equal volume and (iv) a procedure for uniform sampling from spherical boxes. The work presented here is a significant extension of the concept introduced in the conference paper [29].

The rest of the paper is organized as follows. In Section 2, we define the task of aligning two point sets as a nonlinear minimization problem and define our cost function. In Section 3, a stochastic approach for global minimization is presented. In Section 4, we motivate the choice of the rotation space parametrization we use in combination with our minimization approach and introduce a technique for a hierarchical rotation space decomposition. Furthermore, a procedure for uniform sampling from spherical boxes is described. Section 5 presents experimental results obtained with our registration algorithm as well as comparisons with two state-of-the-art registration methods. The paper ends with some conclusions in Section 6.

2. Registration as a minimization problem

Consider, we are given a model point set $\mathbf{M} = \{\mathbf{x}_1, \ldots, \mathbf{x}_m\} \subset \mathbb{R}^3$ and a data point set $\mathbf{D} = \{\mathbf{y}_1, \ldots, \mathbf{y}_n\} \subset \mathbb{R}^3$. Suppose, we have a continuous function $S : \mathbb{R}^3 \to \mathbb{R}$, called the model scalar field, which attains small values at the model points $\mathbf{x}_j$, $j \in \{1, \ldots, m\}$ and increases with increasing distance between the evaluation point and the closest model point. Our aim is to find a rigid transform $T : \mathbb{R}^3 \to \mathbb{R}^3$ of the form $T(\mathbf{x}) = R\mathbf{x} + \mathbf{t}$ for a rotation matrix R and a translation vector $\mathbf{t} \in \mathbb{R}^3$ such that the functional

$$\mathcal{F}(T) = \sum_{i=1}^{n} S(T(\mathbf{y}_i)), \quad \mathbf{y}_i \in \mathbf{D} \tag{1}$$

is minimized. This definition of $\mathcal{F}$ is based on the following idea common for most registration algorithms: we seek a rigid transform that brings the data points as close as possible to the model points.

2.1. Definition of the model scalar field

Given the model point set $\mathbf{M} = \{\mathbf{x}_1, \ldots, \mathbf{x}_m\}$, we want our model scalar field $S : \mathbb{R}^3 \to \mathbb{R}$ to attain its minimal value at the model points, i.e.,

$$S(\mathbf{x}_j) = s_{\min} \in \mathbb{R}, \quad \forall \mathbf{x}_j \in \mathbf{M}, \tag{2}$$

and to attain greater values for all other points in $\mathbb{R}^3$, i.e.,

$$S(\mathbf{x}) > s_{\min}, \quad \forall \mathbf{x} \in \mathbb{R}^3 \setminus \mathbf{M}. \tag{3}$$

Define

$$d_{\mathbf{M}}(\mathbf{x}) = \min_{\mathbf{x}_j \in \mathbf{M}} \|\mathbf{x} - \mathbf{x}_j\| \tag{4}$$

to be the distance between a point $\mathbf{x} \in \mathbb{R}^3$ and the set $\mathbf{M}$, where $\|\cdot\|$ is the Euclidean norm in $\mathbb{R}^n$. If we set

$$S(\mathbf{x}) = d_{\mathbf{M}}(\mathbf{x}), \tag{5}$$

we get an unsigned distance field which is implicitly used by ICP [1]. It is obvious that this choice for S fulfills both criteria (2) and (3).

Mitra et al. [20] and Pottmann et al. [21] considered in their work more sophisticated scalar fields. They assumed that the model point set $\mathbf{M}$ consists of points sampled from an underlying surface Φ. The scalar field S at a point $\mathbf{x} \in \mathbb{R}^3$ is defined to be the squared distance from $\mathbf{x}$ to Φ. In this context, S is called the squared distance function to the surface Φ. We refer to [20] for details on computing the squared distance function and its approximation for point sets.

The version of S given in (5) and the ones used by Mitra et al. [20] and Pottmann et al. [21] are essentially distance fields. This means that $S(\mathbf{x})$ approaches infinity as the point $\mathbf{x}$ gets infinitely far from the point set. This has the practical consequence that a registration technique which minimizes a cost function based on an unbounded scalar field will be sensitive to outliers in the data set. This is because data points lying far away from the model point set will have great impact on the sum in (1) and thus will prevent the minimization algorithm from converging towards the right alignment. A similar problem arises in the case of model and data sets with low overlap. In this case, there will be a lot of data points which have no corresponding model points and vice versa. The distance between such a data point and the closest model point will be large and thus will deteriorate the sum in (1). A simple way to overcome this is just to exclude data points which are too far away from the model set. However, this strategy introduces discontinuities in the cost function which cause a problem for many optimization methods.

Fitzgibbon presented in his work [22] a more convenient way to alleviate these difficulties which does not lead to a discontinuous cost function. He proposed to use either of the following two robust kernels:

$$S(\mathbf{x}) = \log\left(1 + \frac{(d_{\mathbf{M}}(\mathbf{x}))^2}{\sigma}\right) \text{(Lorentzian kernel)} \quad \text{or} \tag{6}$$

$$S(\mathbf{x}) = \begin{cases} (d_{\mathbf{M}}(\mathbf{x}))^2 & \text{if } d_{\mathbf{M}}(\mathbf{x}) < \sigma \\ 2\sigma d_{\mathbf{M}}(\mathbf{x}) - \sigma^2 & \text{otherwise} \end{cases} \text{(Huber kernel)}. \tag{7}$$

However, we still have $\lim_{d_{\mathbf{M}}(\mathbf{x})\to\infty} S(\mathbf{x}) = \infty$ for both kernels as in the case of (5). Thus a cost function based on (6) or (7) will still be sensitive to outliers. We further validate this in the experimental results presented in Section 5 of the paper.

To avoid this sensitivity, we propose to use a bounded scalar field satisfying (2) and (3) and having the additional property

$$\lim_{d_{\mathbf{M}}(\mathbf{x})\to\infty} S(\mathbf{x}) = 0. \tag{8}$$

We set

$$S(\mathbf{x}) = -\varphi(d_{\mathbf{M}}(\mathbf{x})), \tag{9}$$

where $\varphi : \mathbb{R}^* \to \mathbb{R}^*$, for $\mathbb{R}^* = \{x \in \mathbb{R} : x \geqslant 0\}$, is a strictly monotonically decreasing continuous function with

$$\max_{x\in\mathbb{R}^*} \varphi(x) = \varphi(0) \quad \text{and} \tag{10}$$

$$\lim_{x\to\infty} \varphi(x) = 0. \tag{11}$$

In our implementation, we use an inverse distance kernel of the form

$$\varphi(x) = \frac{1}{1+\alpha x^2}, \quad \alpha > 0 \tag{12}$$

because it is computationally efficient to evaluate and can be controlled by a single parameter α (see Fig. 2a). This results in the following model scalar field:

$$S_\alpha^{\mathbf{M}}(\mathbf{x}) = -\frac{1}{1+\alpha(d_{\mathbf{M}}(\mathbf{x}))^2}, \quad \alpha > 0. \tag{13}$$

It is easy to see that (2), (3) and (8) hold. Different values for α in (13) lead to different scalar fields. The greater the value the faster $S_\alpha^{\mathbf{M}}(\mathbf{x})$ convergences to zero as $d_{\mathbf{M}}(\mathbf{x}) \to \infty$ (see Fig. 2b). In Section 2.2, we will discuss how to choose a suitable value for α and why this particular form of $S_\alpha^{\mathbf{M}}(\mathbf{x})$ leads to an outlier robust cost function.

2.2. Cost function definition

The group of all rigid transforms in $\mathbb{R}^3$ is called the special Euclidean group and is denoted by $SE(3)$. At the beginning of Section 2, we formulated the rigid point set registration problem as a functional minimization problem over $SE(3)$. Using a parametrization of $SE(3)$, the functional $\mathcal{F}$ in (1) can be converted to a real-valued scalar field $F : \mathbb{R}^6 \to \mathbb{R}$ of the form

$$F(\varphi,\psi,\theta,x,y,z) = \sum_{i=1}^{n} S_\alpha^{\mathbf{M}}(R_{\varphi,\psi,\theta}\mathbf{y}_i + (x,y,z)), \tag{14}$$

where $\mathbf{y}_1, \ldots, \mathbf{y}_n$ are the data points, $S_\alpha^{\mathbf{M}}$ is the model scalar field defined in (13), $R_{\varphi,\psi,\theta}$ is a rotation matrix parametrized by φ, ψ, θ and $(x,y,z) \in \mathbb{R}^3$ is a translation vector. In order to achieve good optimization performance, it is very important to choose the right parametrization of the rotation group. We employ an axis-angle based parametrization which is especially well suited for our branch and "stochastic bound" minimization method. Furthermore, we introduce a new technique for a hierarchical decomposition of the rotation space in spherical boxes and describe a procedure for uniform sampling from them. Since the advantages of these techniques are best seen in the context of our minimization algorithm we postpone the detailed discussion to Section 4 after the introduction of the minimization method in Section 3.

A global minimizer $\mathbf{x}^* \in \mathbb{R}^6$ of F defines a rigid transform that brings the data points as close as possible to the model points. What makes the proposed cost function robust to outliers is the fact that outlier data points have a marginal contribution to the sum in (14) depending on α. More precisely, given a positive real number d, we can compute a value for α such that $\left|S_\alpha^{\mathbf{M}}(\mathbf{x})\right|$ is less than an arbitrary $\delta > 0$, if $d_{\mathbf{M}}(\mathbf{x}) > d$ holds. In this way, the contribution of an outlier point to the sum in (14) can be made arbitrary close to zero and F will behave like an outlier rejector. However, too large values for α will lead to the rejection of data points which do not have exact counterparts in a sparsely sampled model set, but still are not outliers. In our implementation we set

$$d = \frac{1}{4}\min\{bbox_x(\mathbf{M}), bbox_y(\mathbf{M}), bbox_z(\mathbf{M})\}, \tag{15}$$

$$\delta = 0.1, \tag{16}$$

where $bbox(\mathbf{M})$ denotes the bounding box of the model point set and $bbox_s(\mathbf{M})$, $s \in \{x,y,z\}$ is the extent of the bounding box along the x, y or z axis. Using the absolute value of the right side of (13) and solving for α yields

$$\alpha = \frac{1-\delta}{\delta d^2}. \tag{17}$$

The cost function given in (14) is nonconvex and has multiple local minima over the search space (see [19] where this is experimentally verified for a similar cost function). Using a local optimization procedure—common for many registration methods—will lead in most cases to a local minimizer of F and thus will not give the best alignment between model and data. To avoid this, we employ a new stochastic approach for global minimization described in the next Section.

3. Stochastic adaptive search for global minimization

Our stochastic minimization approach is inspired by the simulated annealing (SA) method of Bilbro and Snyder [28]. The main difference between their work and a typical SA algorithm is the way how the minimizer candidates are generated. As we already mentioned in Section 1.3, SA algorithms are known to waste many iterations in sampling candidate points from the search space, evaluating the cost function at these points and finally rejecting them [26]. In order to reduce the number of rejections, Bilbro

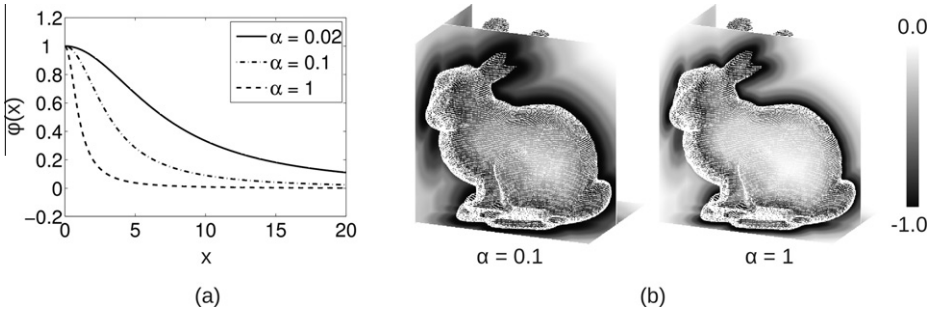

Fig. 2. (a) The inverse distance kernel (defined in (12)) for three different α values. (b) The model scalar field $S_\alpha^{\mathbf{M}}(\mathbf{x})$ (defined in (13)) based on the inverse distance kernel from (a) for $\alpha = 0.1$ and $\alpha = 1$. In this example, the Stanford bunny is used as the model set. $S_\alpha^{\mathbf{M}}(\mathbf{x})$ is visualized by evaluating it at a number of points lying on the three planes and mapping the scalar values to gray levels.

and Snyder [28] sampled the points from a distribution which is modified iteratively during the minimization such that its modes are built around minimizers of the cost function. They achieved this by building a k-d tree and sampling the candidates from those leaves of the tree which cover "promising" regions of the search space, i.e., regions in which the cost function is likely to attain low values. Although this leads to fewer candidate rejections and thus saves computation time the method in [28] still has two drawbacks. First, the candidate points are sampled directly from the tree leaves which are n-dimensional boxes of the form $[a_1,b_1]\times\cdots\times[a_n,b_n]$, where $[a_i,b_i]\subset\mathbb{R}$ is a closed interval. This strategy is based on the implicit assumption that the search space can be covered efficiently by such boxes. This, however, is not the case if we have a more complex shaped space, e.g., the space of rotations (see Section 4). Second, the k-d tree used in [28] is updated only if the generated candidate is accepted. In the case of a rejection, the tree remains unchanged. This is a waste of computation time since the information gained by the expensive cost function evaluation is not used.

We account for the first drawback by formulating our minimization algorithm using a more general spatial data structure, namely, a generalized binary space partitioning tree (we will call it a G-BSP tree in the following). As opposed to the classic BSP trees (see, e.g., [30]), we do not require that the subspaces represented by the tree nodes are convex sets. Thus we can minimize efficiently over more complex shaped search spaces like, e.g., the space of rotations (see Section 4). To avoid the second drawback, i.e., to use all the information gained by the cost function evaluation, we update the tree at every iteration—even in the cases of bad minimizer candidates. This apparently minor modification leads to a rather different algorithm (than [28]) and enables a faster rejection of the regions in which the cost function is likely to have high (i.e., poor) values and thus speeds up the convergence.

3.1. Generalized BSP trees

A binary space partitioning tree (BSP tree) is a spatial data structure which decomposes the real space $\mathbb{R}^n$ in a hierarchical manner. At each subdivision stage, the space is subdivided by a (hyper)plane in two disjoint parts of arbitrary size. Thus the resulting decomposition consists of arbitrarily shaped convex polygons [30]. Each node of the tree has exactly two or zero child nodes. A node with zero children is called a leaf. If we drop the assumption that the space subdivision is performed by planes we get a generalized BSP tree (G-BSP tree). This results in a decomposition made up of subspaces of arbitrary shape.

3.2. Problem definition

Given a set $\mathbf{X}$ (called the search space) and a function $f:\mathbf{X}\to\mathbb{R}$ our aim is to find a global minimizer of f, i.e., an $\mathbf{x}^*\in\mathbf{X}$ such that

$$f(\mathbf{x}^*)\leqslant f(\mathbf{x})\quad\forall\mathbf{x}\in\mathbf{X}. \tag{18}$$

The following assumptions about $\mathbf{X}$ should hold:

- $\mathbf{X}\subset\mathbb{R}^n$ is a bounded set of positive volume (Lebesgue measure in $\mathbb{R}^n$).
- There is an algorithm of acceptable complexity which can build a G-BSP tree for $\mathbf{X}$ such that each two subsets of $\mathbf{X}$ at the same level of the tree are of equal volume (have the same Lebesgue measure in $\mathbb{R}^n$).
- $\mathbf{X}$ is simple enough for sampling algorithms of acceptable complexity to be able to sample uniformly from the G-BSP tree nodes, i.e., from the subsets of $\mathbf{X}$ represented in the G-BSP tree.

Furthermore, the cost function f is required to be bounded and defined at each $\mathbf{x}\in\mathbf{X}$.

3.3. Overall algorithm description

We use a G-BSP tree to represent the n-dimensional search space $\mathbf{X}$. The root η_0^0 is at the 0th level of the tree and represents the whole space $\mathbf{X}_0=\mathbf{X}$. η_0^0 has two children, η_{00}^1 and η_{01}^1, which are at the next level. They represent the subsets $\mathbf{X}_{00}$ and $\mathbf{X}_{01}$, respectively, which are disjoint, have equal volume and their union equals $\mathbf{X}_0$. In general, a node η_s^k (where $k\geqslant 0$ and s is a binary string of length $k+1$) is at the kth level of the tree and has two children, η_{s0}^{k+1} and η_{s1}^{k+1}, which are at the next, $(k+1)$th, level. The volume of η_s^k is $1/2^k$ of the volume of $\mathbf{X}$. This concept is easily visualized in the case $n=2$ and $\mathbf{X}$ and its subsets being rectangles (see Fig. 3a).

During the minimization, the G-BSP tree is built in an iterative fashion beginning at the root. The algorithm adds more resolution to promising regions in the search space, i.e., the tree is built with greater detail in the vicinity of points in $\mathbf{X}$ at which the objective function attains low values. The overall procedure can be outlined as follows:

1. Initialize the tree (see Section 3.4) and set an iteration counter $j=0$.
2. Select a "promising" leaf according to a probabilistic selection scheme (see Section 3.5).
3. Expand the tree by bisecting the selected leaf. This results in the creation of two new child nodes. Evaluate the objective function at a point which is uniformly sampled from the subset of one of the two children (see Section 3.6).
4. If a stopping criterion is not met, increment the iteration counter j and go to step 2, otherwise terminate the algorithm (see Section 3.7).

3.4. Initializing the tree

For every tree node η_s^k the following items are stored: (i) a set $\mathbf{X}_s\subset\mathbf{X}$ and (ii) a pair $(\mathbf{x}_s,f(\mathbf{x}_s))$ consisting of a point $\mathbf{x}_s$, uniformly sampled from $\mathbf{X}_s$, and the corresponding function value $f(\mathbf{x}_s)$. The tree is initialized by storing the whole search space $\mathbf{X}$ and a pair $(\mathbf{x}_0,f(\mathbf{x}_0))$ in the root.

3.5. Selecting a leaf

At every iteration, the search for a global minimizer begins at the root and proceeds down the tree until a leaf is reached. In order to reach a leaf, we have to choose a concrete path from the root down to this leaf. At each node, we have to decide whether to take its left or right child as the next station. This decision is made probabilistically. For every node, two numbers p_0, $p_1\in(0,1)$ are computed such that $p_0+p_1=1$. Arriving at a node, we choose to descend via either its left or right child with probability p_0 or p_1, respectively. We make these left/right decisions until we reach a leaf.

Computing the probabilities p_0 and p_1. The idea is to compute the probabilities in a way such that the "better" child, i.e., the one with the lower function value, has greater chance to be selected. We compute p_0 and p_1 for each node η_s^k based on the function values associated with its children η_{s0}^{k+1} and η_{s1}^{k+1}. Let f_{s0} and f_{s1} be the function values associated with η_{s0}^{k+1} and η_{s1}^{k+1}, respectively. The following criterion should be fulfilled:

$$f_{s0}<f_{s1}\quad\Longleftrightarrow\quad p_0>p_1. \tag{19}$$

If $f_{s0}<f_{s1}$ we set

$$p_0=(t+1)/(1+2t),\quad p_1=t/(1+2t), \tag{20}$$

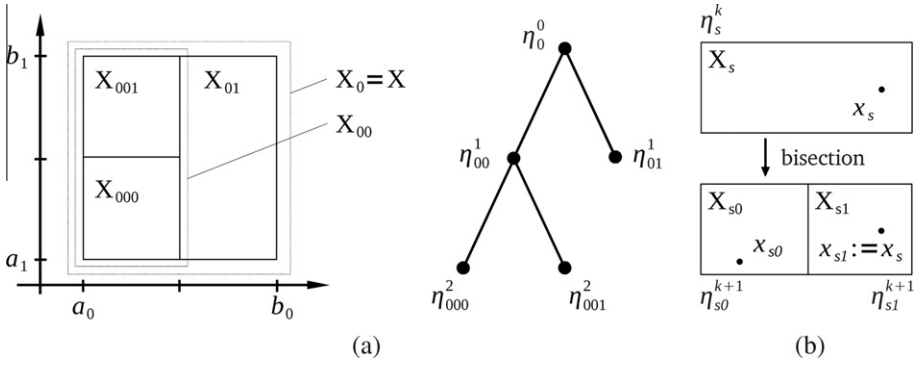

Fig. 3. (a) An example of a two-dimensional G-BSP tree and a rectangular search space $\mathbf{X}$. In this case, the G-BSP tree is a two-dimensional k-d tree. (b) Expanding the leaf η_s^k. In this example, after the bisection of η_s^k, the point $\mathbf{x}_s$ lies in the box $\mathbf{X}_{s1}$, hence η_{s1}^{k+1} adopts the pair $(\mathbf{x}_s, f(\mathbf{x}_s))$ from η_s^k. For the other child, η_{s0}^{k+1}, a point $\mathbf{x}_{s0}$ is sampled uniformly from $\mathbf{X}_{s0}$ and the objective function is evaluated at that point.

for a parameter $t \geqslant 0$. For $t \to \infty$ we get $p_0 = p_1 = \frac{1}{2}$ and our minimization algorithm becomes a pure random search. Setting $t = 0$ results in $p_0 = 1$ and $p_1 = 0$ and makes the algorithm deterministically choosing the "better" child of every node which leads to the exclusion of a large portion of the search space and in most cases prevents the algorithm from finding a global minimizer. For $f_{s1} < f_{s0}$ we set

$$p_0 = t/(1+2t), \quad p_1 = (t+1)/(1+2t). \tag{21}$$

Updating the probabilities. From the discussion above it becomes evident that t should be chosen from the interval $(0, \infty)$. For our algorithm the parameter t plays a similar role as the temperature parameter for a simulated annealing algorithm [23] so we will refer to t as temperature as well. Like in simulated annealing, the search begins at a high temperature level (large t) such that the algorithm samples the search space quite uniformly. The temperature is decreased gradually during the minimization process so that promising regions of the search space are explored in greater detail. More precisely, we update t according to the following cooling schedule:

$$t = t_{\max} \exp(-vj), \tag{22}$$

where $j \in \mathbb{N}$ is the current iteration number, $t_{\max} > 0$ is the temperature at the beginning of the search (for $j = 0$) and $v > 0$ is the cooling speed which determines how fast the temperature decreases.

3.6. Expanding the tree

After reaching a leaf η_s^k, the set $\mathbf{X}_s$ associated with it gets bisected in two disjoint subsets $\mathbf{X}_{s0}$ and $\mathbf{X}_{s1}$ of equal volume. The corresponding child nodes are η_{s0}^{k+1} and η_{s1}^{k+1}, respectively. In this way, we add more resolution in this part of the search space. Next, we evaluate the new children, i.e., we assign to the left and right one a pair $(\mathbf{x}_{s0}, f(\mathbf{x}_{s0}))$ and $(\mathbf{x}_{s1}, f(\mathbf{x}_{s1}))$, respectively.

Note that the parent of η_{s0}^{k+1} and η_{s1}^{k+1}, namely, the node η_s^k, stores a pair $(\mathbf{x}_s, f(\mathbf{x}_s))$. Since $\mathbf{X}_s = \mathbf{X}_{s0} \cup \mathbf{X}_{s1}$ and $\mathbf{X}_{s0} \cap \mathbf{X}_{s1} = \emptyset$ it follows that $\mathbf{x}_s$ is contained either in $\mathbf{X}_{s0}$ or in $\mathbf{X}_{s1}$. Thus we set

$$(\mathbf{x}_{s0}, f(\mathbf{x}_{s0})) = (\mathbf{x}_s, f(\mathbf{x}_s)) \quad \text{if } \mathbf{x}_s \in \mathbf{X}_{s0} \quad \text{or} \tag{23}$$

$$(\mathbf{x}_{s1}, f(\mathbf{x}_{s1})) = (\mathbf{x}_s, f(\mathbf{x}_s)) \quad \text{if } \mathbf{x}_s \in \mathbf{X}_{s1}. \tag{24}$$

To compute the other pair, we sample a point uniformly from the appropriate remaining set ($\mathbf{X}_{s0}$ or $\mathbf{X}_{s1}$) and evaluate the function at this point (see Fig. 3b for the case $n = 2$ and $\mathbf{X}$ and its subsets being rectangles).

Updating the tree. During the search we want to compute the random paths from the root down to a certain leaf such that promising regions—leaves with low function values—are visited more often than non-promising ones. Thus, after evaluating a new created leaf, we propagate its (possibly very low) function value as close as possible to the root. This is done by the following updating procedure. Suppose that the parent point $\mathbf{x}_s$ is contained in the set $\mathbf{X}_{s1}$ belonging to the new created child η_{s1}^{k+1}. Therefore, we randomly generate $\mathbf{x}_{s0} \in \mathbf{X}_{s0}$, compute $f(\mathbf{x}_{s0})$ and assign the pair $(\mathbf{x}_{s0}, f(\mathbf{x}_{s0}))$ to the child η_{s0}^{k+1}. Updating the tree consists of ascending from η_{s0}^{k+1} (via its ancestors) to the root and comparing at every parent node η_u^j the function value $f(\mathbf{x}_{s0})$ with the function value of η_u^j, i.e., with $f(\mathbf{x}_u)$. If $f(\mathbf{x}_{s0}) < f(\mathbf{x}_u)$ we update the current node by setting $(\mathbf{x}_u, f(\mathbf{x}_u)) = (\mathbf{x}_{s0}, f(\mathbf{x}_{s0}))$ and proceed to the parent of η_u^j. The updating procedure terminates if we reach the root or no improvement for the current node is possible.

Note that if $f(\mathbf{x}_{s0})$ is the lowest function value found so far, it will be propagated to the root, otherwise it will be propagated only to a certain level $l \in \{1, \ldots, k+1\}$. This means, that every node contains the minimum function value (and the point at which f takes this value) found in the subset associated with this node. Since the root represents the whole search space, it contains the point we are interested in, namely, the point at which f takes the lowest value found up to the current iteration.

3.7. Stopping rule

We break the search if the following two criteria are fulfilled. (i) The leaf η_s^k selected in the current iteration has a volume which is smaller than a user predefined value $\delta_v > 0$. (ii) The absolute difference between the minimal function value found so far and the function value computed in the current iteration is less than a user specified $\delta_f > 0$.

The first condition accounts for the desired precision of the solution and the second one assures that the algorithm makes no significant progress any more.

3.8. Remark

We want to emphasize that it is very important that each two nodes at the same tree level are of equal volume. Note that the points are uniformly sampled within the tree nodes (see Section 3.2). In this case, if two differently sized nodes at the same tree level are selected equally often, the part of the search space represented by the smaller node will be sampled more densely than the other part. Thus, the algorithm will possibly prefer parts of the search space only because the G-BSP tree is constructed in a particular way and not because of the cost function.

4. Processing in the space of rigid transforms

As already mentioned in Section 2.2, the choice of a parametrization of $SE(3)$ (the group of rigid transforms) is an important issue since different parametrizations lead to different optimization

performance. We decompose $SE(3)$ into a translational and a rotational part. While parametrizing translations is straightforward special care is needed when dealing with rotations since the geometry of the rotation space is more complex than the geometry of $\mathbb{R}^3$. In the following, we concentrate on the rotation space.

In view of our branch and "stochastic bound" minimization method, three specific problems have to be solved. (i) We need to parametrize rotations. (ii) We have to hierarchically decompose the rotation space in disjoint parts of equal volume. In other words, a G-BSP tree has to be computed in which the nodes are representing equally sized parts of the rotation space. (iii) We need to sample points (i.e., rotations) uniformly from each leaf of the G-BSP tree. These issues are discussed separately in the next three subsections.

4.1. Parametrizing rotations

There are many ways how to parametrize 3D rotations. Discussing all of them is far beyond the scope of this paper. An excellent introduction to this topic is included in the books by Kanatani [31] and Watt and Watt [32] in the context of computer vision and computer graphics, respectively. The set of all 3×3 rotation matrices is a group (under matrix multiplication) which is referred to as $SO(3)$. A parametrization of $SO(3)$ is a mapping R: $\mathbf{U} \rightarrow SO(3)$, where $\mathbf{U}$ is a subset of $\mathbb{R}^3$ since every rotation has three degrees of freedom.

Parametrizing rotation matrices using Euler angles is probably the most widely used technique which is, however, inefficient in conjunction with our minimization method. This is due to the fact that Euler angles are a redundant representation of rotations. In order to represent all elements in $SO(3)$ the following range, $\mathbf{E}$, for the three Euler angles is needed: $\mathbf{E} = [0,2\pi) \times [0,2\pi) \times [0,\pi]$. However, the corresponding parametrization R: $\mathbf{E} \rightarrow SO(3)$, which is given in [31], is not one-to-one. There are infinitely many combinations of Euler angles (within the range $\mathbf{E}$) which lead to the same rotation matrix (see [32]). A minimization method like ours which considers the whole search space will waste computation time exploring regions in $\mathbf{E}$ which should be completely ignored because they do not lead to "new" rotation matrices. The same applies to deterministic branch-and-bound methods (see, e.g., [33]).

In order to avoid this difficulty, we employ a redundant-free rotation space parametrization based on the axis-angle representation of $SO(3)$. According to Euler's theorem (see [31]), each rotation in $\mathbb{R}^3$ can be represented by an axis specified by a unit vector $\mathbf{n}$ and an angle θ of rotation around it. $\mathbf{n}$ can itself be parametrized using spherical coordinates φ and ψ:

$$\mathbf{n} = (\sin(\psi)\cos(\varphi), \quad \sin(\psi)\sin(\varphi), \quad \cos(\psi)). \tag{25}$$

Fig. 4a visualizes this concept. In order to represent all rotation matrices, we need to consider the following range for the spherical coordinates (φ,ψ) and the rotation angle θ:

$$(\varphi,\psi,\theta) \in [0,2\pi) \times [0,\pi] \times [0,\pi) = \mathbf{A}. \tag{26}$$

The parametrization R: $\mathbf{A} \rightarrow SO(3)$, which can be found in [31], is a one-to-one mapping between $\mathbf{A}$ and $SO(3)$.

4.2. Hierarchical decomposition of the rotation space

According to the axis-angle representation and to (26), it is possible to express the set of rotations by the open ball in $\mathbb{R}^3$ with radius π which we will denote by $\mathbf{B}^3(\pi)$ (see Fig. 4b). Thus a straightforward way to decompose the rotation space is to enclose $\mathbf{B}^3(\pi)$ in the cube $\mathbf{C}^3(\pi) = [-\pi,\pi]^3$ and to divide $\mathbf{C}^3(\pi)$ into smaller cubes by simply bisecting the x, y or z axis. Hartley and Kahl [33] used this technique in conjunction with a deterministic branch-and-bound minimization method to estimate the essential matrix and to solve the relative camera pose problem. However, if combined with our minimization algorithm, this technique leads to two problems. First, the sub-cubes of $\mathbf{C}^3(\pi)$ which do not lie within $\mathbf{B}^3(\pi)$ have to be ignored since the rotations they represent are included in other cubes within $\mathbf{B}^3(\pi)$. This gives rise to nodes in the corresponding G-BSP tree which have only one "legal" child. Second, the sub-cubes of $\mathbf{C}^3(\pi)$ which are partially intersecting $\mathbf{B}^3(\pi)$ represent a smaller region of the rotation space than sub-cubes at the same tree level which are fully enclosed in $\mathbf{B}^3(\pi)$. Thus the minimization algorithm will prefer rotations which are close to the boundary of $\mathbf{B}^3(\pi)$.

We solve these two problems by changing the shape of the building blocks of the decomposition. Since we are dealing with a three-dimensional ball the most natural shape is the shape of a spherical box (see Fig. 4b). In ball coordinates, we define a spherical box $\mathbf{S}^3$ to be a point set of the form

$$\mathbf{S}^3 = \{(\varphi,\psi,\theta) : (\varphi,\psi,\theta) \in [\varphi_1,\varphi_2) \times [\psi_1,\psi_2) \times [\theta_1,\theta_2)\}, \tag{27}$$

where $[\varphi_1,\varphi_2) \times [\psi_1,\psi_2)$ is the range of the spherical coordinates and $[\theta_1,\theta_2)$ limits the distance of the points to the origin. Decomposing the rotation space means to hierarchically subdivide $\mathbf{B}^3(\pi)$ into disjoint spherical boxes of equal volume (see Fig. 5). Note that the volume of $\mathbf{S}^3$ is given by

$$vol_{\mathbf{S}^3}(\varphi_1,\varphi_2,\psi_1,\psi_2,\theta_1,\theta_2) = \int_{\varphi_1}^{\varphi_2}\int_{\psi_1}^{\psi_2}\int_{\theta_1}^{\theta_2} \theta^2 \sin\psi \, d\theta \, d\psi \, d\varphi \tag{28}$$

$$= (\varphi_2-\varphi_1)(\cos\psi_1 - \cos\psi_2)\frac{\theta_2^3-\theta_1^3}{3}. \tag{29}$$

Our aim is to consecutively cut $\mathbf{S}^3$ along the φ, ψ or θ axis such that the resulting pieces have the same volume. Since $vol_{\mathbf{S}^3}$ depends in a different way from each of the ball coordinates φ, ψ and θ we get a different rule for cutting along each axis. We are looking for

$$\varphi \in (\varphi_1,\varphi_2), \quad \psi \in (\psi_1,\psi_2), \quad \theta \in (\theta_1,\theta_2) \tag{30}$$

such that

$$vol_{\mathbf{S}^3}(\varphi_1,\varphi) = vol_{\mathbf{S}^3}(\varphi,\varphi_2), \tag{31}$$

$$vol_{\mathbf{S}^3}(\psi_1,\psi) = vol_{\mathbf{S}^3}(\psi,\psi_2), \tag{32}$$

$$vol_{\mathbf{S}^3}(\theta_1,\theta) = vol_{\mathbf{S}^3}(\theta,\theta_2), \tag{33}$$

where, for the sake of clarity, $vol_{\mathbf{S}^3}$ is expressed as a function of two variables only, namely, the ones defining the interval which is currently cut. Using (29) to solve the Eqs. (31)–(33) leads to

$$\varphi = \frac{\varphi_1+\varphi_2}{2}, \quad \psi = \arccos\left(\frac{\cos\psi_1+\cos\psi_2}{2}\right),$$

$$\theta = \sqrt[3]{\frac{\theta_1^3+\theta_2^3}{2}}. \tag{34}$$

Thus we fully specified how to hierarchically decompose the space of rotations in disjoint equally sized parts such that a G-BSP tree can be built. Furthermore, the shape of the parts is optimally tailored to our minimization algorithm.

4.3. Uniform sampling from spherical boxes

Our method for sampling points uniformly from a spherical box is grounded on the following basic result from Statistics called the inverse probability integral transform. Since it is proved in many textbooks (like, e.g., in [34]) we state it here without a proof.

Theorem 1. *Let F be a cumulative distribution function (c.d.f.) on $\mathbb{R}$ and let U be a random variable uniformly distributed in $[0,1]$. Then the random variable $X = F(U)^{-1}$ has c.d.f. F.*

Based on this result we perform the uniform sampling from a spherical box $\mathbf{S}^3 = [\varphi_1,\varphi_2) \times [\psi_1,\psi_2) \times [\theta_1,\theta_2)$ in three steps:

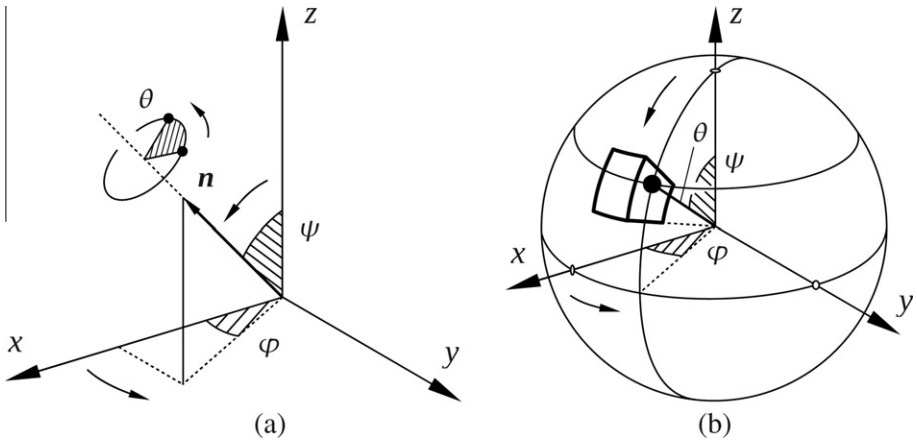

Fig. 4. (a) The axis-angle based parametrization of $SO(3)$. The two bold dots in the figure represent a point before and after rotation by the angle θ around the axis defined by the unit vector $\mathbf{n}$, which is itself parametrized using spherical coordinates (φ,ψ). (b) The rotation space represented as the open ball in $\mathbb{R}^3$ with radius π. The spherical coordinates (φ,ψ) of the point (shown as a bold dot) define the rotation axis and the distance to the origin gives the angle of rotation θ. The bold lines depict a spherical box.

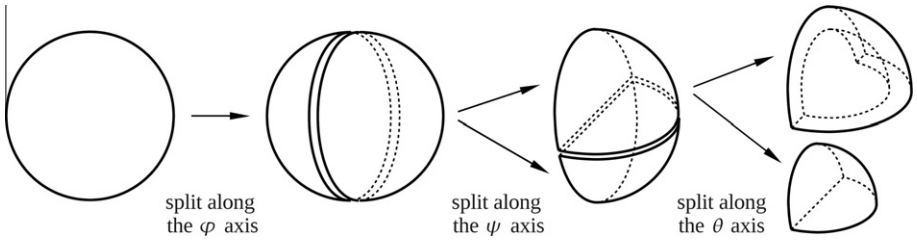

Fig. 5. Decomposing the rotation space (represented as $\mathbf{B}^3(\pi)$) into spherical boxes of equal volume. In this example, only one spherical box at each splitting step is further decomposed.

1. Sample a φ uniformly from $[\varphi_1,\varphi_2)$.
2. Sample a ψ from $[\psi_1,\psi_2)$ according to a c.d.f. F_2 such that the point in $\mathbb{R}^3$ with spherical coordinates (φ,ψ) is uniformly distributed on the spherical patch $\mathbf{S}^2 = [\varphi_1,\varphi_2)\times[\psi_1,\psi_2)$.
3. Sample a θ from $[\theta_1,\theta_2)$ according to a c.d.f. F_3 such that the point in $\mathbb{R}^3$ with ball coordinates (φ,ψ,θ) is uniformly distributed in the spherical box $\mathbf{S}^3$.

Step 1 is easy to perform. In step 2, we need to compute the area of a spherical patch (of the unit 2-sphere) as a function of an interval $[\varphi_1,\varphi_2)\times[\psi_1,\psi_2)$:

$$area_{\mathbf{S}^2}(\varphi_1,\varphi_2,\psi_1,\psi_2) = \int_{\varphi_1}^{\varphi_2}\int_{\psi_1}^{\psi_2} \sin\psi\, d\psi\, d\varphi \tag{35}$$

$$= (\varphi_2-\varphi_1)(\cos\psi_1-\cos\psi_2). \tag{36}$$

Thus the c.d.f. we need in step 2 is given by

$$F_2(\psi) = \frac{area_{\mathbf{S}^2}(\varphi_1,\varphi_2,\psi_1,\psi)}{area_{\mathbf{S}^2}(\varphi_1,\varphi_2,\psi_1,\psi_2)} \tag{37}$$

$$= \frac{\cos\psi_1-\cos\psi}{\cos\psi_1-\cos\psi_2}, \tag{38}$$

Analogously, we see that the c.d.f. in step 3 is given by

$$F_3(\theta) = \frac{vol_{\mathbf{S}^3}(\varphi_1,\varphi_2,\psi_1,\psi_2,\theta_1,\theta)}{vol_{\mathbf{S}^3}(\varphi_1,\varphi_2,\psi_1,\psi_2,\theta_1,\theta_2)} \tag{39}$$

$$= \frac{\theta^3-\theta_1^3}{\theta_2^3-\theta_1^3}, \tag{40}$$

where (40) follows from (29). Note that both F_2 and F_3 can easily be inverted and we can use Theorem 1 to sample according to F_2 and F_3 and hence uniformly from the spherical box $\mathbf{S}^3$.

4.4. Computing the search space and the G-BSP tree

Now since all details regarding the parametrization and decomposition of $SO(3)$ and the sampling from spherical boxes are given, we define the search space $\mathbf{X}$ and specify how to build the corresponding G-BSP tree. We set

$$\mathbf{X} = \mathbf{A}\times bbox(\mathbf{M}), \tag{41}$$

where $\mathbf{A}$ is, according to (26), the domain of the axis-angle based parametrization of $SO(3)$ and $bbox(\mathbf{M})$ (the bounding box of the model $\mathbf{M}$) represents the translational part of the search space. Since $bbox(\mathbf{M})$ is a rectangular box of the form $[x_1,x_2]\times[y_1,y_2]\times[z_1,z_2]\subset\mathbb{R}^3$ it can easily be broken up into smaller boxes of the same size by simply bisecting it along the x, y or z axis.

The root η_0^0 of the G-BSP tree represents the whole set $\mathbf{X}$. The child nodes of the root, namely, η_{00}^1 and η_{01}^1, represent the subsets $\mathbf{X}_0$ and $\mathbf{X}_1$, respectively, resulting from cutting the 0th interval of $\mathbf{X}$—which is $[0,2\pi)$ in (26)—using the rule $(34)_1$. In general, a node η_s^k (where $k\geqslant 0$ and s is a binary string of length $k+1$) is at the kth level of the tree, represents a subset $\mathbf{X}_s$ of the 6D search space and has two children, η_{s0}^k and η_{s1}^k. The child nodes represent the sets $\mathbf{X}_{s0}$ and $\mathbf{X}_{s1}$, respectively, which are computed by cutting the (k mod 6)th interval of $\mathbf{X}_s$ according to (34) if $0\leqslant k \bmod 6\leqslant 2$ (rotational part) or by dividing it in the middle if $3\leqslant k \bmod 6\leqslant 5$ (translational part).

5. Experimental results

In this Section, we test our registration method on a variety of point sets. All tests presented in the paper are performed on a laptop with a 3GHz CPU and 4GB RAM running a Linux operating

Table 1
The parameter values used in all experiments in this paper. The value of δ_v equals the volume of a spherical box with side one degree times the volume of a box with sides equal to one percent of the sides of the bounding box of the model point set.

	Parameter	Defined in	Value
Cost function	d	Eq. (15)	1/4 (min bbox side(**M**))
	δ	Eq. (16)	0.1
Cooling schedule	t_{max}	Eq. (22)	50.0
	v	Eq. (22)	0.00008
Stopping rule	δ_v	Section 3.7	1° rot. and 1% transl.
	δ_f	Section 3.7	0.1

system. The algorithm is implemented in C++. The parameter values used in all experiments presented here are given in Table 1.

Since our method is a probabilistic one, it computes each time a (slightly) different result. In order to make a statistical meaningful statement about its performance, we run 100 registration trials for each pair of inputs and report the mean performance values. We measure the success rate and the accuracy under varying amount of noise and outliers in the input sets. The success rate gives the percentage of registration trials in which a transform which is close to the global optimal one is found. The accuracy is measured using the RMS error (see [6]). The type of noise added to some of the model and data sets is Gaussian and the outliers are simulated by drawing points from a uniform distribution within the bounding box of the corresponding point set. We report the number of outliers as percentage of the original number of points and not as percentage of the points in the corrupted set. For example, 100% means that there are so many outliers in the corrupted point set as there are points in the outlier-free set. We did it so because the results in [7], which we use for comparison, are reported in this way.

We also measure the number of cost function evaluations and the computation time for varying cooling speed v (defined in (22)). We analyze the robustness of our method using two different kernels in the cost function. Furthermore, we report how two state-of-the-art registration approaches perform on the same point

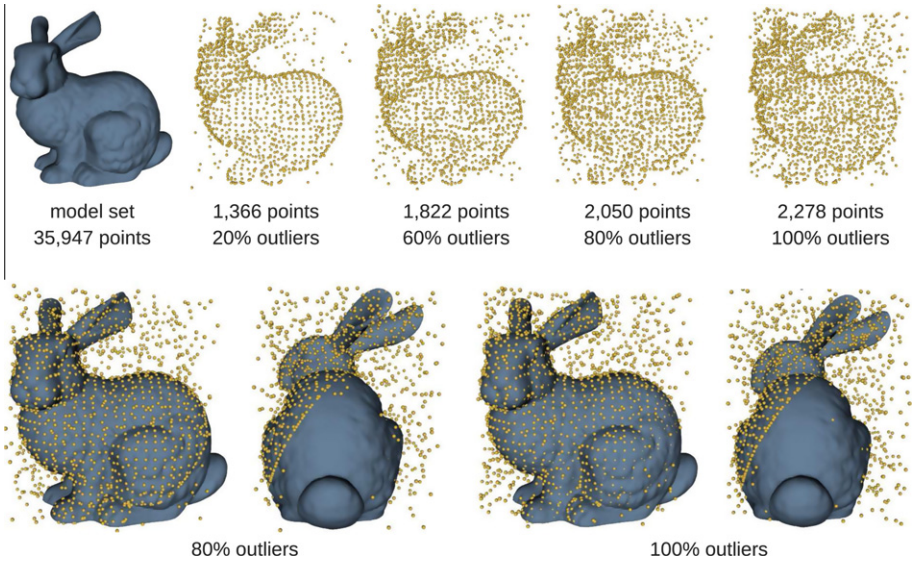

Fig. 6. (Top row) The model set is shown as a blue mesh (note that only the mesh vertices are used for the registration). The outlier corrupted data sets are rendered as yellow point clouds. The size of each point set and the number of outliers as percentage of the original number of input points are shown below each figure. Further note that the data sets are incomplete and sparsely sampled compared to the model. (Bottom row) Typical registration results obtained with our algorithm using the model scalar field (13) based on the inverse distance kernel (12). Observe the high quality of the alignment even in the presence of a significant amount of outliers. A registration trial took between 9 and 17 s (depending on the number of points). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

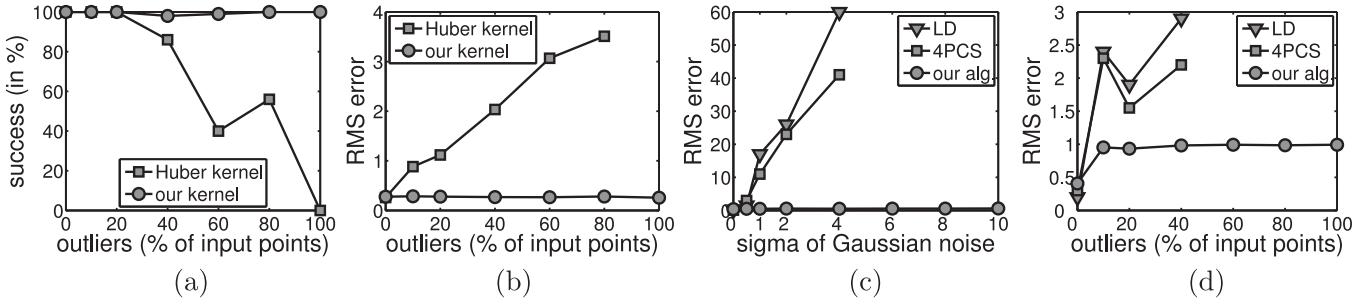

Fig. 7. (a) The success rate as a function of the percentage of outliers in the data sets shown in Fig. 6. The success rate of the registration is shown when using the inverse distance kernel (12) (our kernel) and the Huber kernel (7). Note that our kernel leads to an almost constant success rate of 100% even in the presence of a very large amount of outliers whereas at the level of 100% outliers the registration completely fails if the Huber kernel is used. (b) The RMS error between the ground truth pose for each data set and the estimated pose is shown as a function of the percentage of outliers. Only the successful trials are used for computing the RMS error. Note that our kernel leads to much more precise registration results which are almost independent of the amount of outliers. (c) and (d) We compare our method with the robust 4PCS algorithm [7] and a local descriptor based approach (LD). A combination of a spin-image based descriptor and integral invariants are used as local descriptors (see [7]). Note that the graphs corresponding to LD and 4PCS end by $\sigma = 4.0$ and 40% outliers. This is because the authors in [7] did not test their methods on point sets with more noise or outliers whereas we did. Observe that our algorithm is quite insensitive to noise and outliers and it outperforms both other methods. The alignment error is measured using the RMS error between the model and the data after registration. One unit corresponds to 1% of the bounding box diagonal length of the model set.

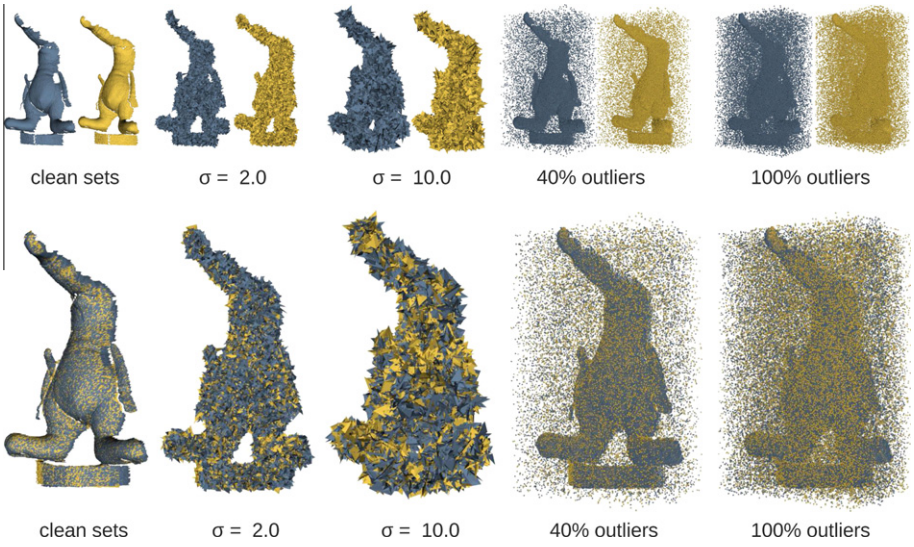

Fig. 8. Registration of partially overlapping noisy and outlier corrupted point sets. The models are shown in blue whereas the data sets in yellow. (Top row) Partial scans of the Coati model degraded by noise or outliers. The σ of the Gaussian noise or the amount of outliers as percentage of the original number of input points is indicated below each figure. One σ unit equals 1% of the bounding box diagonal length of the corresponding point set. (Bottom row) Typical registration results computed with our algorithm. 5,000 (randomly sampled) points from each point set are used for the registration. The results are obtained without any noise or outlier removal, ICP refinement [1] or assumptions about the initial pose of the point sets. Each registration trial took about 33 s. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

#points	40	60	80	100	150	200
time(sec) b &b[19]	10.2	48.6	107.5	156.2	722.1	1103.2
time(sec) our alg.	2.0	2.1	2.3	2.4	2.8	3.1

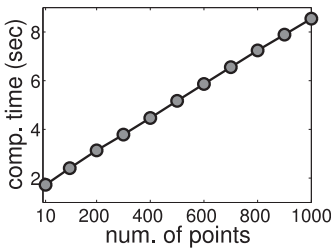

Fig. 9. (Left) Computation time comparison between our algorithm and the box-and-ball (b&b) registration algorithm of Li and Hartley [19] which is based on global deterministic Lipschitz optimization theory. The processing time is given in seconds. In the case of 200 input points, our algorithm outperforms [19] by three orders of magnitude. (Right) Runtime of our algorithm as a function of the number of input points. The figure clearly indicates a linear time complexity. Model and data used in this test case are downsampled copies of the outlier-free version of the data set shown in the top row of Fig. 6. In all tests, our method achieved a success rate of 100%.

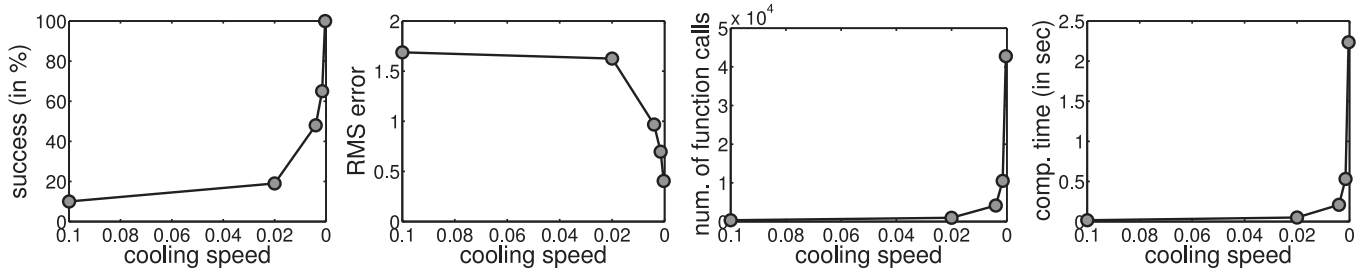

Fig. 10. From left to right: success rate, RMS error, number of cost function evaluations and computation time of our registration algorithm as a function of the cooling speed v (defined in (22)). Model and data used in this test case consist of 100 points randomly sampled from the outlier-free version of the data set shown in the top row of Fig. 6. One RMS error unit equals 1% of the bounding box diagonal length of the point set.

sets and compare the runtime of our algorithm with the one of a deterministic branch-and-bound method. In the following, we describe each test scenario in detail.

First, the success rate and the accuracy of our method are tested with two different kernels, namely, the inverse distance kernel (12) used in our cost function and the Huber kernel (7) used in [22]. The point sets used in this test together with some typical registration results are shown in Fig. 6. Note that outliers are added only to the data set and it is a subset of the model. This case occurs in real world scenarios in which one has a complete (relatively clean) model of an object and wants to align it to a low quality data set which only partially represents the object (due to visibility issues like, e.g., occlusion and scene clutter). As already mentioned in Section 2.1, we expect a registration method which minimizes a cost function based on the (unbounded) Huber kernel to have difficulties with outlier corrupted data sets. This is confirmed by the results of this test case which are summarized in the Fig. 7a and b.

In the second test case, we align two partially overlapping parts of the Coati model under varying conditions. This time, noise and outliers are added to both the model and the data set. This situation occurs in practice when building a complete object model out of multiple partially overlapping scans. We compare our

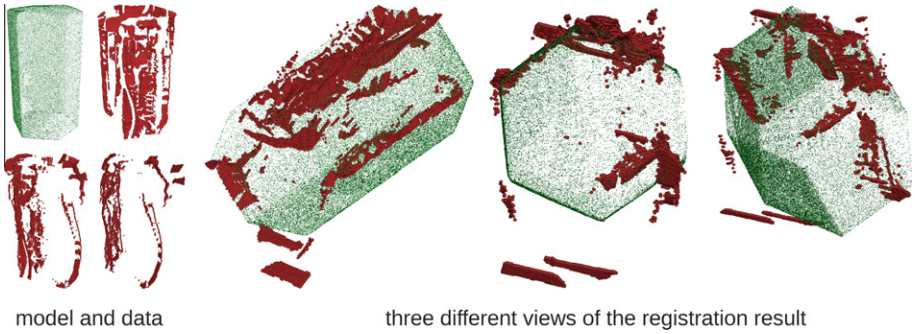

Fig. 11. (Left) The complete model of a box (shown in green; 236,089 points) and three views of the very low quality data set (shown in red; 5000 out of 9623 points were randomly sampled and used for the registration). The data was obtained with a correlation based stereo algorithm under poor lighting conditions. (Right) Our method robustly achieved the right alignment in 10 out of 10 trials. Each registration trial took about 30 s. The high amount of noise and outliers which almost completely destroy the shape of the object makes this a challenging example. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

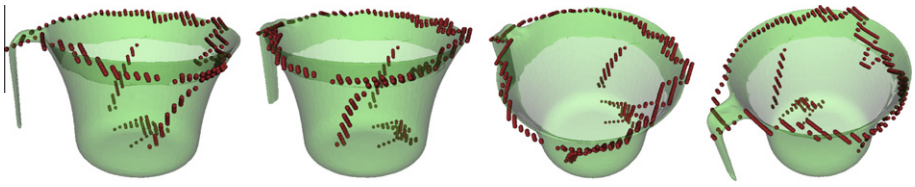

Fig. 12. Registration result in the case of a noisy and very sparsely reconstructed data set (shown by the red "curve") and a complete noise-free model (transparent green mesh). Note that in this case the state-of-the-art integral volume descriptor (used in [6]) will fail since the curve which represents the data set does not enclose a volume in $\mathbb{R}^3$. Local descriptors which use surface normals like, e.g., spin images [5] will fail as well since in general the normal of a curve which lies on a surface does not match the surface normal. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

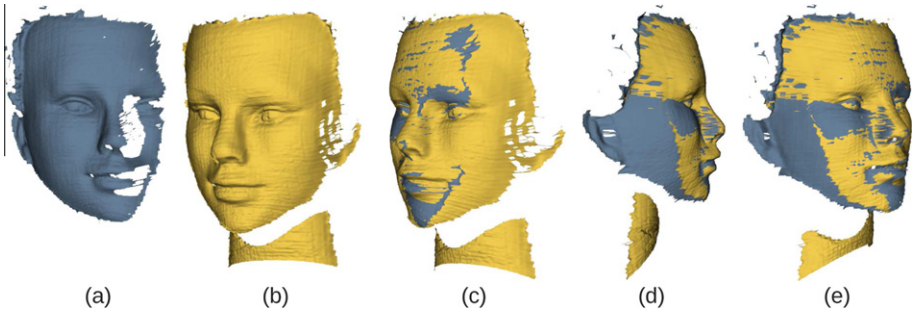

Fig. 13. Registration of noisy point sets with low overlap. Although rendered as meshes only points are used for the registration. Note that the input scans, (a) and (b), represent different parts of the face and the model set, shown in (a), contains no parts of the neck. (c)–(e) A typical registration result obtained with our method shown from three different viewpoints.

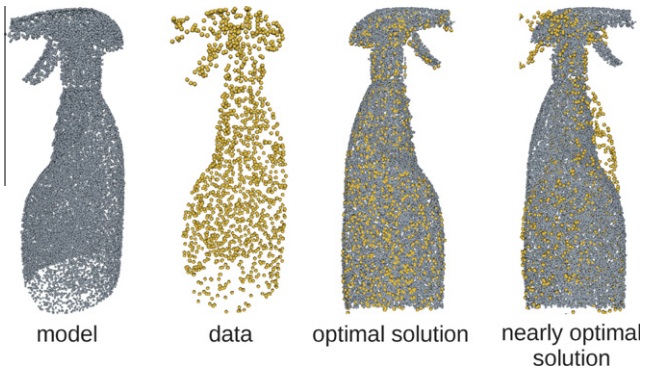

Fig. 14. Point sets leading to a cost function which has two almost equally low minima. The nearly optimal solution differs from the optimal one by a rotation of the data set by 180° about the axis which corresponds to the upright orientation of the bottle.

results with the ones reported in [7] which are obtained with the robust 4PCS algorithm and a state-of-the-art local descriptor based approach. We perform the tests on the same point sets which are used in [7]. This allows for a precise comparison without the need of re-implementing neither of the two algorithms. The model and data sets together with some typical registration results obtained with our method are shown in Fig. 8. In the Fig. 7c and d, we plot our results together with the ones reported in [7].

In the third test scenario, we measure the computation time of our algorithm and compare it with the one of the deterministic registration method of Li and Hartley [19]. Since we run the tests on a similar (i.e., *not* more powerful) hardware as the one used in [19] an accurate comparison is possible. The results are summarized in Fig. 9.

Next, we measure the performance of our method for varying cooling speed v defined in (22). We report the results in Fig. 10. Our algorithm achieves a success rate of 100% and an RMS error below 0.5 for less than 2.5 s (for point sets consisting of 100 points).

Finally, we demonstrate the ability of our method to deal with partially overlapping and very sparsely sampled point sets corrupted by noise and outliers which are not artificially generated but originate in scan device imprecision. In Fig. 11, we show that our method successfully computes the right registration even in the case of an extremely degraded data set which represents only a subset of the model. Fig. 12 illustrates the stability of our algorithm when dealing with very sparsely sampled data sets. Figs. 1 and 13 show typical registration results for partially overlapping points sets.

Note that our registration method could lead to incorrect results for a class of shapes for which several almost equally good alignments exist and the registration ambiguity can be dissolved by small scale features only. An example of such a shape is a large cup with a small handle. In this case, the corresponding point sets lead to a cost function with several local minima which are almost as "good" as the global one (see Fig. 14).

6. Conclusions

We introduced a new technique for pairwise rigid registration of point sets. Our method is based on a noise robust and outlier resistant cost function which itself is based on an inverse distance kernel. One of the main messages of the paper is that a registration method which minimizes an objective function based on an unbounded kernel will be sensitive to outliers in the point sets. This was fully validated by comparisons between our kernel and the Huber kernel which were presented in the experimental part of the paper.

A further property of our algorithm is that it does not rely on any initial estimation of the globally optimal rigid transform. This was achieved by employing a new stochastic algorithm for global optimization. In order to minimize efficiently over complex shaped search spaces like the space of rotations we generalized the BSP trees and introduced a new technique for hierarchical rotation space decomposition. Furthermore, we derived a new procedure for uniform point sampling from spherical boxes.

Tests on a variety of point sets showed that the proposed method is insensitive to noise and outliers and can cope very well with sparsely sampled and incomplete data sets. Comparisons showed that our algorithm is by three orders of magnitude faster than a deterministic branch-and-bound method and that it outperforms a recently proposed generate-and-test approach and a state-of-the-art local descriptor based method in terms of accuracy and robustness.

Acknowledgments

This work has been funded by the European Commission's Seventh Framework Programme as part of the Project GRASP (IST-FP7-IP-215821).

References

[1] P. Besl, N. McKay, A method for registration of 3-D shapes, IEEE Transactions on PAMI 14 (1992).
[2] Y. Hecker, R. Bolle, On geometric hashing and the generalized hough transform, IEEE Transactions on Systems, Man, and Cybernetics 24 (1994).
[3] H. Wolfson, I. Rigoutsos, Geometric hashing: an overview, IEEE Computational Science & Engineering 4 (1997) 10–21.
[4] G. Stockman, Object recognition and localization via pose clustering, Computer Vision, Graphics, and Image Processing 40 (1987) 361–387.
[5] A. Johnson, M. Hebert, Using spin images for efficient object recognition in cluttered 3D scenes, IEEE Transactions on PAMI 21 (1999) 433–449.
[6] N. Gelfand, N. Mitra, L. Guibas, H. Pottmann, Robust global registration, Eurographics Symposium on Geometry Processing (2005) 197–206.
[7] D. Aiger, N. Mitra, D. Cohen-Or, 4-Points congruent sets for robust pairwise surface registration, ACM Transactions on Graphics 27 (2008).
[8] Y. Chen, G. Medioni, Object modeling by registration of multiple range images, in: Proceedings of IEEE International Conference on Robotics and Automation, vol. 3, 1991, pp. 2724–2729.
[9] S. Rusinkiewicz, M. Levoy, Efficient variants of the ICP algorithm, 3DIM, 2001, pp. 145–152.
[10] S. Granger, X. Pennec, Multi-scale EM-ICP: a fast and robust approach for surface registration, in: Proceedings of ECCV, 2002, pp. 418–432.
[11] Y. Tsin, T. Kanade, A correlation-based approach to robust point set registration, in: Proceedings of ECCV, 2004, pp. 558–569.
[12] B. Jian, B.C. Vemuri, A robust algorithm for point set registration using mixture of gaussians, in: Proceedings of ICCV, 2005, pp. 1246–1251.
[13] T. Tamaki, M. Abe, B. Raytchev, K. Kaneda, Softassign and EM-ICP on GPU, in: Proceedings of the 2nd Workshop on Ultra Performance and Dependable Acceleration Systems (UPDAS), 2010.
[14] B. Ma, R.E. Ellis, Surface-based registration with a particle filter, in: Proceedings of MICCAI, 2004, pp. 566–573.
[15] M.H. Moghari, P. Abolmaesumi, Point-based rigid-body registration using an unscented Kalman filter, IEEE Transactions on Medical Imaging 26 (2007) 1708–1728.
[16] R. Sandhu, S. Dambreville, A. Tannenbaum, Point set registration via particle filtering and stochastic dynamics, IEEE Transactions on PAMI 32 (2010) 1459–1473.
[17] T.M. Breuel, Implementation techniques for geometric branch-and-bound matching methods, Computer Vision and Image Understanding 90 (2003) 258–294.
[18] C. Olsson, F. Kahl, M. Oskarsson, Branch-and-bound methods for Euclidean registration problems, IEEE Transactions on PAMI 31 (2009) 783–794.
[19] H. Li, R.I. Hartley, The 3D–3D registration problem revisited, in: Proceedings of ICCV, 2007, pp. 1–8.
[20] N. Mitra, N. Gelfand, H. Pottmann, L. Guibas, Registration of point cloud data from a geometric optimization perspective, in: Symposium on Geometry Processing, 2004, pp. 23–32.
[21] H. Pottmann, Q.-X. Huang, Y.-L. Yang, S.-M. Hu, Geometry and convergence analysis of algorithms for registration of 3D shapes, International Journal of Computer Vision 67 (2006) 277–296.
[22] A.W. Fitzgibbon, Robust registration of 2D and 3D point sets, Image Vision Computing 21 (2003) 1145–1153.
[23] N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller, E. Teller, Equation of state calculations by fast computing machines, The Journal of Chemical Physics 21 (1953) 1087–1092.
[24] V. Cerny, Thermodynamical approach to the traveling salesman problem: an efficient simulation algorithm, Journal of Optimization Theory and Applications 45 (1985) 41–51.
[25] S. Kirkpatrick, C. Gelatt, M. Vecchi, Optimization by simulated annealing, Science 220 (1983) 671–680.
[26] P. Pardalos, E. Romeijn (Eds.), Handbook of Global Optimization 2, Nonconvex Optimization and Its Applications, Kluwer Academic Publishers, 2002.
[27] D. Bulger, G. Wood, Hesitant adaptive search for global optimisation, Mathematical Programming 81 (1998) 89–102.
[28] G. Bilbro, W. Snyder, Optimization of functions with many minima, IEEE Transactions on Systems, Man, and Cybernetics 21 (1991) 840–849.
[29] C. Papazov, D. Burschka, Stochastic optimization for rigid point set registration, in: Proceedings of ISVC, 2009, pp. 1043–1054.
[30] H. Samet, The Design and Analysis of Spatial Data Structures, Addison-Wesley, 1990.
[31] K. Kanatani, Group-Theoretical Methods in Image Understanding, Springer Series in Information Sciences, Springer, 1990.
[32] A. Watt, M. Watt, Advanced Animation and Rendering Techniques, Addison-Wesley, 1992.
[33] R.I. Hartley, F. Kahl, Global optimization through rotation space search, International Journal of Computer Vision 82 (2009) 64–79.
[34] N. Madras, Lectures on Monte Carlo Methods, American Mathematical Society, 2002.