Stochastic Optimization for Rigid Point Set Registration

Chavdar Papazov and Darius Burschka

Machine Vision and Perception Group (MVP)
Department of Computer Science
Technische Universität München, Germany
email: {papazov|burschka}@in.tum.de

Abstract. In this paper we propose a new method for pairwise rigid point set registration. We pay special attention to noise robustness, outlier resistance and global optimal alignment. The problem of registering two point clouds in space is converted to a minimization of a nonlinear cost function. We propose a cost function that aims to reduce the impact of noise and outliers. Its definition is based on the input point sets and is directly related to the quality of a concrete rigid transform between them. In order to achieve a global optimal registration, without the need of a good initial alignment, we develop a new stochastic approach for global minimization. Tests on a variety of point sets show that the proposed registration algorithm performs very well on noisy, outlier corrupted and incomplete data.

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 $\mathbf{M} = \{\mathbf{x}_1, \dots, \mathbf{x}_m\} \subset \mathbb{R}^3$ and $\mathbf{D} = \{\mathbf{y}_1, \dots, \mathbf{y}_n\} \subset \mathbb{R}^3$ find a mapping $T : \mathbb{R}^3 \to \mathbb{R}^3$ such that the point set $T(\mathbf{D}) = \{T(\mathbf{y}_1), \dots, T(\mathbf{y}_n)\}$ 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 R and a translation $\mathbf{t}$, we have the problem of rigid point set registration. The problem is especially hard when no initial pose estimation is available and the data point set is noisy, outlier corrupted and incomplete.

Point Set Registration. Algorithms for the rigid registration problem belong to two general classes. One class consists of methods designed to solve the initial pose estimation problem. 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. Johnson and Hebert introduce in their work [1] local geometric descriptors, called spin images, and use them for pose estimation and object recognition. The presented results are impressive, but no tests with noisy or outlier corrupted data are performed. Gelfand *et al.* [2] develop a local descriptor which performs well under noisy conditions, but still, defining robust local descriptors in the presence of significant noise and a great amount of outliers remains a difficult task. A more recent approach to the initial pose estimation problem is the 4PCS algorithm introduced by Aiger *et al.* [3]. It is an efficient randomized generate-and-test approach: For an appropriate quadruple $\mathbf{B}$ (called a basis) of nearly coplanar points from the model set $\mathbf{M}$, compute the optimal rigid transform between $\mathbf{B}$ and each of the potential bases in the data set $\mathbf{D}$ and choose the optimal one. In order to achieve high probability for finding the global optimal transform, 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 supported by the experimental results in Section 4 of our 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 build the second class of registration approaches. The most popular one is the Iterative Closest Point (ICP) algorithm. Since its introduction by Chen and Medioni [4], and Besl and McKay [5], a variety of improvements have been proposed in the literature. A good summary as well as new results on acceleration of ICP algorithms has been given by Rusinkiewicz and Levoy [6]. A major drawback of these ICP variants is that they assume a good initial guess for the orientation of the data point set (with respect to the model point set). This orientation is improved in an iterative fashion until an optimal rigid transform is found. The quality of the solution depends heavily on the initial guess. Another disadvantage of the methods compared by Rusinkiewicz and Levoy [6] is that they use local surface features like surface normals which cannot be computed very reliably in the presence of noise.

The approach we develop is most related to the ones proposed by Mitra *et al.* [7] and Pottmann *et al.* [8]. They also express the registration problem as a minimization of a cost function. Its definition is based on the distance of the data points to the surface defined by the model points. For its minimization, however, a local optimization method is used. This results in the already mentioned strong dependence on a good initial transform estimation.

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) as a minimization technique [9], [10], [11]. A comprehensive overview of this field is given in [12]. A major property of SA algorithms is their “willingness” to explore regions around points in search space at which the objective function takes values greater than the current minimum. This is what makes SA algorithms able to escape from local minima and makes them suitable for the task of global minimization. A known drawback of SA algorithms is the fact that they waste a lot of iterations generating candidate points, evaluating the objective function at these points, and finally rejecting them [12]. In order to reduce the number of rejections, Bilbro and Snyder [13] 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 spatial data structure (an n-dimensional binary tree) to the objective function each time a new candidate point is accepted. If, however, the current point is not accepted, the tree remains unchanged. This is—in the case of candidate rejection—a considerable waste of computation time, since the information gained by the (expensive) evaluation of the objective function is not used at all. In contrast to that, our algorithm adapts the n-dimensional tree at every iteration and thus uses all the information collected during the minimization.

Contributions and Overview. Our registration algorithm aims to solve the initial pose estimation problem with a sufficient accuracy, so that no additional refinement is necessary. Our main contributions are (i) the introduction of a new *noise and outlier resistant* cost function and (ii) a new stochastic approach for its *global* minimization.

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, we introduce a stochastic approach for global minimization. Section 4 presents experimental results obtained by our registration algorithm. Conclusions are drawn in the final Section 5 of this paper.

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 $S : \mathbb{R}^3 \rightarrow \mathbb{R}$, called the model scalar field, which takes small values when evaluated at (or near) 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. The model scalar field S will be precisely defined in Section 2.1. Consider for now it is given and it has the above mentioned property. Our aim is to find a rigid transform $T : \mathbb{R}^3 \rightarrow \mathbb{R}^3$ of the form $T(\mathbf{x}) = R \cdot \mathbf{x} + \mathbf{t}$ for a rotation matrix $R \in \mathbb{R}^{3\times 3}$ and a translation vector $\mathbf{t} \in \mathbb{R}^3$ such that the functional

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

gets minimized. This definition of $\mathcal{F}$ is based on the following quite natural 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 to have a function $S : \mathbb{R}^3 \rightarrow \mathbb{R}$ which takes its minimal value at the model points, i.e.,

$$S(\mathbf{x}_j) = s_{\min} \in \mathbb{R}, \quad \forall j \in \{1, \ldots, m\}, \tag{2}$$

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

$$S(\mathbf{x}) > s_{\min}, \quad \forall \mathbf{x} \in \mathbb{R}^3 \setminus \{\mathbf{x}_1, \ldots, \mathbf{x}_m\}. \tag{3}$$

Define

$$\mathrm{d}_{\mathbf{M}}(\mathbf{x}) := \min_{\mathbf{x}_j \in \mathbf{M}} \|\mathbf{x} - \mathbf{x}_j\| \tag{4}$$

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

$$S(\mathbf{x}) := \mathrm{d}_{\mathbf{M}}(\mathbf{x}), \tag{5}$$

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

Mitra *et al.* [7] and Pottmann *et al.* [8] consider in their work more sophisticated scalar fields. They assume that the model point set $\mathbf{M}$ consists of points sampled from some 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 [7] for details on computing the squared distance function and its approximation for point sets.

The version of S given in (5) and the one used by Mitra *et al.* [7] are both essentially distance fields. This means that $\lim_{\|\mathbf{x}\| \to \infty} S(\mathbf{x}) = \infty$, i.e., $S(\mathbf{x})$ approaches to infinity as the point $\mathbf{x}$ gets infinitely far from the point set. This has the practical consequence that a registration technique based on an unbounded scalar field S will be sensitive to outliers in the data set, because data points lying far away from the model point set will have great impact on the functional value in Eq. (1) and thus will prevent the minimization algorithm from converging towards the global optimal alignment.

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

$$\lim_{\|\mathbf{x}\| \to \infty} S(\mathbf{x}) = 0. \tag{6}$$

We set

$$S(\mathbf{x}) := -\varphi\left(\mathrm{d}_{\mathbf{M}}(\mathbf{x})\right), \tag{7}$$

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

$$\max_{x \in \mathbb{R}^+} \varphi(x) = \varphi(0) \ \text{ and} \tag{8}$$

$$\lim_{x \to \infty} \varphi(x) = 0. \tag{9}$$

In our implementation we use a rational function of the form $1/(1+\alpha x^2)$ because it is computationally efficient to evaluate and can be controlled by a single parameter α. This results in the following scalar field:

$$S_\alpha^{\mathbf{M}}(\mathbf{x}) = -\frac{1}{1 + \alpha\left(\mathrm{d}_{\mathbf{M}}(\mathbf{x})\right)^2}, \quad \alpha > 0. \tag{10}$$

It is easy to see that (2), (3) and (6) hold. Different α's in Eq. (10) lead to different scalar fields. The greater the value the faster $S_\alpha^{\mathbf{M}}(\mathbf{x})$ convergences to zero as $\|\mathbf{x}\| \to \infty$. In the next Section, we will discuss how to choose a suitable value for α.

2.2 Cost Function Definition

At the beginning of Section 2, we formulated the rigid point set registration problem as a functional minimization problem: Minimize $\mathcal{F}$ (see Eq. (1)) over the set of rigid transforms. We convert $\mathcal{F}$ to a real-valued scalar field $F : \mathbb{R}^6 \to \mathbb{R}$ of the form

$$F(\theta, \phi, \psi, x, y, z) = \sum_{i=1}^{n} S_\alpha^{\mathbf{M}}(R_{\theta,\phi,\psi} \cdot \mathbf{y}_i + (x, y, z)), \tag{11}$$

for the data points $\mathbf{y}_1, \ldots, \mathbf{y}_n$ and for $S_\alpha^{\mathbf{M}}$ defined in Eq. (10). $R_{\theta,\phi,\psi}$ is a rotation matrix describing a rotation by θ about the x-axis, followed by a rotation by ϕ about the y-axis and a rotation by ψ about the z-axis. 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 Eq. (11). More precisely, given a positive real number d, we can compute a value for α, such that $|S_\alpha^{\mathbf{M}}(\mathbf{x})|$ is less than an arbitrary $\delta > 0$, if $\mathrm{d}_{\mathbf{M}}(\mathbf{x}) > d$ holds. In this way the contribution of an outlier point to the sum in Eq. (11) can be made arbitrary close to zero, hence F behaves like an outlier rejector. Too large values for α, however, will lead to the rejection of data points which do not have exact counterparts in the model set, but still are not outliers. In our implementation we set $d = \frac{1}{5} diag(BB(\mathbf{M}))$ and $\delta = 0.1$, where $diag(BB(\mathbf{M}))$ denotes the diagonal length of the axis-aligned minimum bounding box of the model point set. Using the absolute value of the right side of Eq. (10) and solving for α yields

$$\alpha = \frac{1 - \delta}{\delta d^2}. \tag{12}$$

The cost function given in (11) is nonlinear and nonconvex. This results in a great number of local minima of F over the search space. Using a local optimization procedure—common for the most registration methods in the literature—will lead in most cases to a local minimizer of F and thus will not give the best alignment between model and data.

We employ a new stochastic approach for global minimization, described in the next Section of this paper. We seek the global minimum of F over the search space

$$\mathbf{X} := [-\pi/2, \pi/2] \times [-\pi, \pi] \times [-\pi, \pi] \times BB(\mathbf{M}), \tag{13}$$

where $BB(\mathbf{M})$ denotes the axis-aligned minimum bounding box of the model point set. The first three intervals in (13) build the search space for the rotational part and the bounding box for the translational part of the rigid transform.

3 Adaptive Search for Global Minimization

Our stochastic minimization approach is inspired by the work of Bilbro and Snyder [13]. The algorithm shares two properties with the one presented in [13]: (i) we use the same data structure (an n-dimensional binary tree) to represent the search space and (ii) we adapt the tree during the minimization process to the objective function. In contrast to [13], where the tree is updated only when a new candidate point is accepted, we update it at every iteration, so we use *all* the information gained by the evaluation of the objective function. This apparently minor modification leads to a rather different algorithm (than [13]) and enables a faster rejection of regions in which the objective function is likely to have high (i.e., poor) values and thus speeds up the convergence.

3.1 Problem Definition

We call a set $\mathbf{X} \subset \mathbb{R}^n$ an n-dimensional (or n-d) box if there are n intervals $[a_i, b_i] \subset \mathbb{R}$ such that

$$\mathbf{X} = [a_0, b_0] \times \ldots \times [a_{n-1}, b_{n-1}]. \tag{14}$$

Given an n-dimensional box $\mathbf{X}$ and a bounded continuous function $f : \mathbf{X} \rightarrow \mathbb{R}$ our aim is to find an $\mathbf{x}^* \in \mathbf{X}$ with $f(\mathbf{x}^*) \leq f(\mathbf{x})$ for all $\mathbf{x} \in \mathbf{X}$.

3.2 Overall Algorithm Description

We use an n-dimensional binary tree to represent the search space $\mathbf{X}$. The root η_0^0 is at the 0th level of the tree and represents the whole box $\mathbf{X}_0 := \mathbf{X}$. η_0^0 has two children η_{00}^1 and η_{01}^1, which are at the next level of the tree. They represent the n-d boxes $\mathbf{X}_{00}$ respectively $\mathbf{X}_{01}$ resulting from bisecting the 0th interval (this is $[a_0, b_0]$ in (14)) of $\mathbf{X}_0$ and assigning the first half to $\mathbf{X}_{01}$ and the second half to $\mathbf{X}_{11}$. In general, a node η_s^k (where $k \geq 0$ 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 child nodes represent the same n-d box as the one represented by η_s^k (this is $\mathbf{X}_s$) except for that the $(k \bmod n)$th interval of $\mathbf{X}_s$ is bisected and the first and second half is assigned to η_{s0}^{k+1} and η_{s1}^{k+1} respectively.

During the minimization the tree is built in an iterative fashion beginning with 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 has low values. The overall procedure can be outlined as follows:

1. Initialize the tree (see Section 3.3) and set an iteration counter $j := 0$.
2. Select a "promising" leaf according to a probabilistic selection scheme (see Section 3.4).
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 within the n-d box of one of the two children (see Section 3.5).
4. If a stopping criterion is not met, increment the iteration counter j and go to step 2, otherwise terminate the algorithm (see Section 3.6).

3.3 Initializing the Tree

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

3.4 Selecting a Leaf

At every iteration the search for a global minimum begins at the root and proceeds down the tree until a leaf (node without children) 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 in a way 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 respectively p_1. We make these left/right decisions until we encounter a leaf.
Computing the Probabilities. The idea is to compute the probabilities in a way, 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} respectively η_{s1}^{k+1}. The following criterion should be fulfilled:

$$f_{s0} < f_{s1} \quad \Leftrightarrow \quad p_0 > p_1. \tag{15}$$

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

$$p_0 = (t+1)/(1+2t), \quad p_1 = t/(1+2t), \tag{16}$$

for a parameter $t \geq 0$. 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 great portion of the search space and in general prevents the algorithm from finding a global minimum. For $f_{s1} < f_{s0}$ we set

$$p_0 = t/(1+2t), \quad p_1 = (t+1)/(1+2t). \tag{17}$$

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 [9], so we will refer to t as temperature as well. Like in simulated annealing, the search begins on a high temperature level (large t), so the algorithm samples the cost function quite uniformly. The temperature is decreased gradually during the search 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:

$$t = t_{\max} \exp(-vj). \tag{18}$$

$j \in \mathbb{N}$ 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.

3.5 Expanding the Tree

After reaching a leaf η_s^k, the n-d box $\mathbf{X}_s$ associated with it gets bisected in the way described at the beginning of Section 3.2. This results in the creation of two n-d boxes $\mathbf{X}_{s0}$ and $\mathbf{X}_{s1}$ associated with two new children η_{s0}^{k+1} and η_{s1}^{k+1} respectively. In this way, we add more resolution in this region 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 node η_s^k stores a pair $(\mathbf{x}_s, f(\mathbf{x}_s))$. Since we have $\mathbf{X}_s = \mathbf{X}_{s0} \cup \mathbf{X}_{s1}$ 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

$$(\mathbf{x}_{s0}, f(\mathbf{x}_{s0})) := (\mathbf{x}_s, f(\mathbf{x}_s)) \;\; \text{if}\, \mathbf{x}_s \in \mathbf{X}_{s0} \;\; \text{or} \tag{19}$$

$$(\mathbf{x}_{s1}, f(\mathbf{x}_{s1})) := (\mathbf{x}_s, f(\mathbf{x}_s)) \;\; \text{if}\, \mathbf{x}_s \in \mathbf{X}_{s1}. \tag{20}$$

To compute the other pair we sample a point uniformly over the appropriate n-d box ($\mathbf{X}_{s0}$ or $\mathbf{X}_{s1}$) and evaluate the function at this point.

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—leafs 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 other 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, i.e., if $f(\mathbf{x}_{s0}) \geq f(\mathbf{x}_u)$.

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 n-d box 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.

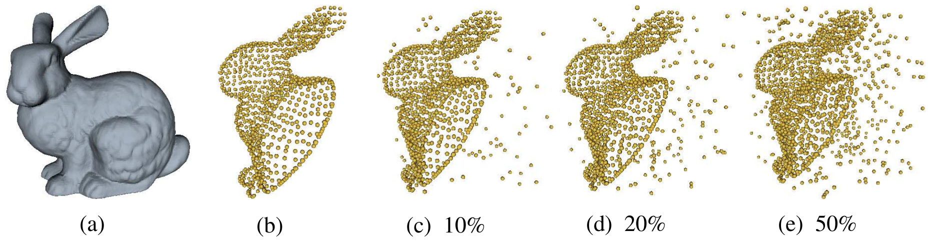

Fig. 1. (a) The bunny model point set. Although it is shown as a mesh, no surface information is used for registration. (b) The data point set without outliers. Note that the data set is incomplete and very sparsely sampled (compared to the model). (c)–(e) Contaminated data point sets. The number of outliers as percentage of the original number of points are shown below each figure. Note that local descriptors, like spin images [1] or integral invariants [2], are very difficult to compute for such sparsely sampled and outlier corrupted point sets.

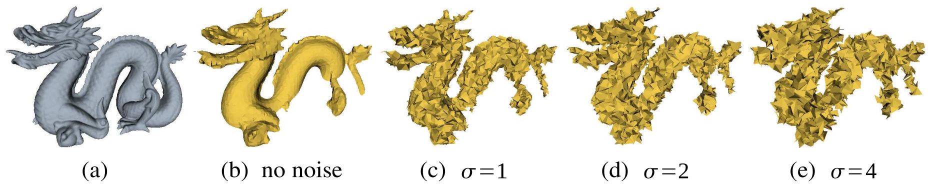

Fig. 2. The dragon model and data sets. Although all of them are shown as meshes, only points are used for registration. (a) The dragon model. (b) Noiseless data set. Note that it has a lower level of detail compared to the model and parts of the dragon are missing. (c)–(e) Data point sets corrupted by zero-mean additive Gaussian noise with variance σ which is expressed in percentage of the bounding box diagonal length of the noiseless data set. Again, computing reliable local descriptors for the point sets (d) and (e) is a very challenging task.

3.6 Stopping rule

We break the search, if for the last N iterations the absolute difference between the last sample of the objective function and the sample before is less than a predefined $\epsilon > 0$.

4 Experimental Results

In this Section, we test our registration method on several point sets. Since the algorithm 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 every pair of inputs. We measure the success rate and the accuracy of the algorithm under varying amount of noise and outliers in the data point sets. The success rate gives the percentage of registration trials in

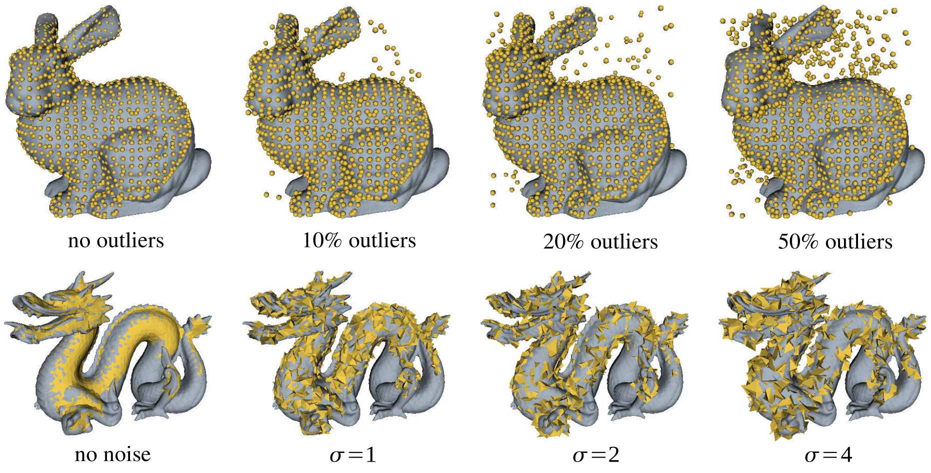

Fig. 3. (Upper row) Typical registration results for the incomplete and outlier corrupted data point sets shown in Fig. 1. The amount of outliers is indicated below the corresponding figure. (Lower row) Typical registration results for the incomplete and noise contaminated data point sets shown in Fig. 2. The value for σ of the Gaussian noise added to the data point sets is shown below each figure.

which a transform which is close to the global optimal one is found. The accuracy is measured using the RMS error between the point sets after alignment [2]. The type of noise added to some of the data sets is Gaussian and the outliers are simulated by drawing points from a uniform distribution within the bounding box of the corresponding data set. We also measure the number of cost function evaluations and the computation time for varying cooling speed v (defined in (18)). In the following, we describe each test scenario in detail.

First, we use our algorithm to register four data point sets to a noiseless model of the Stanford bunny. The data sets are at a lower level of detail (compared to the model), contain only parts of the bunny and three of them are contaminated by a significant amount of outliers (see Fig. 1). We examine each of the 100 registration results. The upper row in Fig. 3 shows exemplary one result for each data point set.

In the second test case, we register several versions of the Stanford dragon under varying noisy conditions. We use a noiseless point set as the model. The data sets have lower resolution, do not contain all parts of the dragon and three of them are corrupted by Gaussian noise (see Fig. 2). As in the bunny test case, we inspect all registration results. Four of them are shown in the lower row in Fig. 3.

We compute the success rate and the mean RMS error based on all 800 registration results in the bunny and in the dragon test cases. For comparison, we show how the newly proposed 4PCS registration algorithm [3] performs under similar conditions. Note that 4PCS has been tested on different point sets, so an

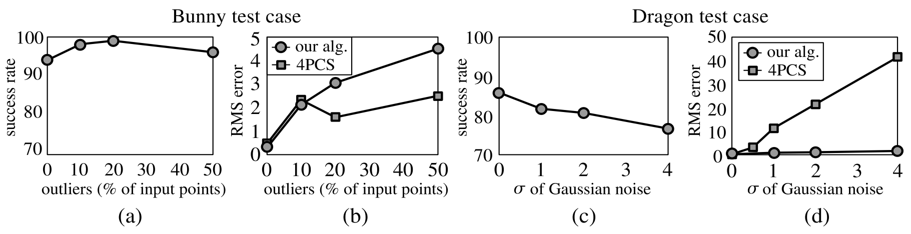

Fig. 4. Success rate and mean RMS error computed from the registration results in the bunny test case (a), (b) and in the dragon test case (c), (d). In (a) and (b) the success rate and the mean RMS error are shown as a function of the number of outliers, whereas in (c) and (d) they are a function of σ of Gaussian noise. In (b) and (d), we compare the accuracy of our method with the accuracy of the 4PCS algorithm [3]. One RMS error unit equals 1% of the bounding box diagonal length of the data point set.

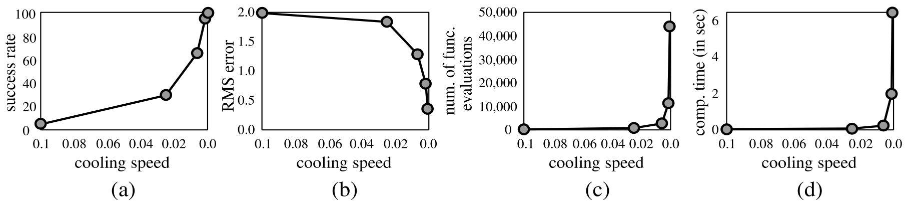

Fig. 5. Success rate (a), mean RMS error (b), mean number of cost function evaluations (c) and mean computation time (d) of our registration algorithm as a function of the cooling speed v. All tests presented in this paper run on a low-cost computer with a 2.2 GHz CPU. For all registration trials we set $t_{\max} = 40$ (see Eq. (18)). Model and data set are copies of the point set shown in Fig. 1(b). One RMS error unit equals 1% of the bounding box diagonal length of the point set.

exact comparison is not possible. In Fig. 4, we plot our results together with the ones reported in [3]. Observe that the success rate of our algorithm is immune against outliers and shows low sensitivity to noise. For outlier corrupted point sets, the 4PCS algorithm is apparently more accurate than ours (see Fig. 4(b)). Note, however, that Aiger *et al.* [3] add outliers to *both* data and model set. Thus, outliers from both point sets are close to each other and contribute little to the RMS error. In contrast to this, we corrupt only the data point set, so its outliers do not have close counterparts in the model point set, hence we get a greater RMS error. According to Fig. 4(d), our method is far more accurate on noisy point sets than the 4PCS algorithm.

Finally, we measure the performance of our algorithm for varying cooling speed v. We report the results in Fig. 5. Our algorithm achieves a success rate of 100% and a mean RMS error less than 0.5 for 6.5 seconds. For comparison, the best success rate (for similar point sets) achieved by the registration algorithms studied in [8] is 15.951% (see last column of Table 3 in [8]).

5 Conclusions

In this paper 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 and on a new stochastic approach for global minimization. Characteristic to the proposed algorithm is (i) that it does not rely on an initial estimation of the globally optimal rigid transform and (ii) that it has low sensitivity to outliers, noise and missing data. Both claims were further supported by a variety of experiments on noisy, outlier corrupted and incomplete point sets.

Acknowledgement. The research leading to these results has received funding from the European Community's Seventh Framework Programme FP7/2007-2013 under grant agreement 215821 (GRASP project).

References

1. Johnson, A., Hebert, M.: Using Spin Images for Efficient Object Recognition in Cluttered 3D Scenes. IEEE Trans. PAMI **21** (1999) 433–449
2. Gelfand, N., Mitra, N., Guibas, L., Pottmann, H.: Robust Global Registration. In: Eurographics Symposium on Geometry Processing. (2005) 197–206
3. Aiger, D., Mitra, N., Cohen-Or, D.: 4-Points Congruent Sets for Robust Pairwise Surface Registration. ACM Trans. Graph. **27** (2008)
4. Chen, Y., Medioni, G.: Object Modeling by Registration of Multiple Range Images. Robotics and Automation, Proceedings., IEEE International Conference on **3** (1991) 2724–2729
5. Besl, P., McKay, N.: A Method for Registration of 3-D Shapes. IEEE Trans. PAMI **14** (1992)
6. Rusinkiewicz, S., Levoy, M.: Efficient Variants of the ICP Algorithm. In: 3DIM. (2001) 145–152
7. Mitra, N., Gelfand, N., Pottmann, H., Guibas, L.: Registration of Point Cloud Data from a Geometric Optimization Perspective. In: Symposium on Geometry Processing. (2004) 23–32
8. Pottmann, H., Huang, Q.X., Yang, Y.L., Hu, S.M.: Geometry and Convergence Analysis of Algorithms for Registration of 3D Shapes. International Journal of Computer Vision **67** (2006) 277–296
9. Metropolis, N., Rosenbluth, A., Rosenbluth, M., Teller, A., Teller, E.: Equation of State Calculations by Fast Computing Machines. The Journal of Chemical Physics **21** (1953) 1087–1092
10. Cerny, V.: Thermodynamical Approach to the Traveling Salesman Problem: An Efficient Simulation Algorithm. Journal of Optimization Theory and Applications **45** (1985) 41–51
11. Kirkpatrick, S., Gelatt, C., Vecchi, M.: Optimization by Simmulated Annealing. Science **220** (1983) 671–680
12. Pardalos, P., Romeijn, E., eds.: Handbook of Global Optimization 2. Nonconvex Optimization and Its Applications. Kluwer Academic Publishers (2002)
13. Bilbro, G., Snyder, W.: Optimization of Functions with Many Minima. IEEE Trans. on Systems, Man, and Cybernetics **21** (1991) 840–849