Eurographics Symposium on Geometry Processing (2004)
R. Scopigno, D. Zorin, (Editors)

Registration of Point Cloud Data from a Geometric Optimization Perspective

Niloy J. Mitra†, Natasha Gelfand†, Helmut Pottmann‡, and Leonidas Guibas†

† Computer Graphics Laboratory, Stanford University
‡ Geometric Modeling and Industrial Geometry, Vienna University of Technology

Abstract

We propose a framework for pairwise registration of shapes represented by point cloud data (PCD). We assume that the points are sampled from a surface and formulate the problem of aligning two PCDs as a minimization of the squared distance between the underlying surfaces. Local quadratic approximants of the squared distance function are used to develop a linear system whose solution gives the best aligning rigid transform for the given pair of point clouds. The rigid transform is applied and the linear system corresponding to the new orientation is build. This process is iterated until it converges. The point-to-point and the point-to-plane Iterated Closest Point (ICP) algorithms can be treated as special cases in this framework. Our algorithm can align PCDs even when they are placed far apart, and is experimentally found to be more stable than point-to-plane ICP. We analyze the convergence behavior of our algorithm and of point-to-point and point-to-plane ICP under our proposed framework, and derive bounds on their rate of convergence. We compare the stability and convergence properties of our algorithm with other registration algorithms on a variety of scanned data.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modelling

1. Introduction

Registration plays an important role in 3D model acquisition, object recognition, and geometry processing. Given as input two shapes, often called the model and the data, each in its own coordinate system, the goal of registration is to find a transformation that optimally positions that data with respect to the model. In this paper, we consider the registration problem when both the model and data inputs are given as point cloud data (PCD). This is a common problem in 3D scanning, where multiple views of an object need to be brought into a common coordinate system, and in geometry processing, where point cloud representations need to be aligned for applications such as texture transfer, morphing, or watermarking [CWPG04].

A popular method for aligning two point clouds is the Iterated Closest Point (ICP) algorithm [BM92, CM91]. This algorithm starts with two point clouds and an estimate of the aligning rigid body transform. It then iteratively refines the transform by alternating the steps of choosing corresponding points across the point clouds, and finding the best rotation and translation that minimizes an error metric based on the distance between the corresponding points.

Despite a large amount of work on registration, convergence behavior of many registration algorithms, under different starting conditions, and error metrics, is poorly understood. Experimentally, it has been shown that the rate of convergence of ICP heavily depends on the choice of the corresponding point-pairs, and the distance function that is being minimized [RL01]. Many enhancements of ICP-style

† The authors wish to acknowledge support from NSF grant CARGO-0138456, a Max-Planck-Institut Fellowship, and a Stanford Graduate Fellowship.
‡ Part of this research has been carried out within the Competence Center Advanced Computer Vision and has been funded by the Kplus program. This work was also supported by the Austrian Science Fund under grant P16002-N05.

algorithms for registration propose different error metrics, and point selection strategies, to improve ICP's convergence behavior [GIRL03, Fit01, JH03, CLSB92, RL01].

Two distance metrics are commonly used in ICP and its variants. The point-to-point distance of Besl [BM92] uses the Eucledian distance between the corresponding points. This leads to an ICP algorithm that converges slowly for certain types of input data and initial positions. Another error metric is the point-to-plane distance of Chen and Medioni [CM91], which uses the distance between a point and a planar approximation of the surface at the corresponding point. When the initial position of the data is close to the model, and when the input has relatively low noise, ICP with point-to-plane error metric has faster convergence than the point-to-point version. However, when the shapes start far away from each other, or for noisy point clouds, point-to-plane ICP tends to oscillate and fails to converge [GIRL03].

Another reason behind the slow convergence of registration algorithms based on ICP, is the local nature of the minimization. The only information used by the algorithm is the point correspondences. As a result, the minimized error function only approximates the squared distance between the two point clouds up to first order.

In this paper, we propose an optimization framework for studying registration algorithms. We pose registration between two point clouds as an optimization over the space of rigid transforms. We develop an objective function that is a second order approximant to the squared distance between the model and the data. Higher order information about the surfaces represented by the point clouds, such as local curvatures, are incorporated into the quadratic approximant. Using such approximant to the squared distance function, we develop a registration algorithm. When the model and the data PCDs are close, our algorithm has a rate of convergence similar to ICP with point-to-plane error metric. Moreover, our method has a stable behavior even when the initial displacement is large. We also explain the convergence properties of the point-to-point and point-to-plane ICP variants, in terms of the accuracy of the distance function that they use during minimization.

2. Registration of Point Cloud Data

Let $P = \{\mathbf{p_1}, \mathbf{p_2}, \ldots, \mathbf{p_n}\}$ and $Q = \{\mathbf{q_1}, \mathbf{q_2}, \ldots, \mathbf{q_m}\}$ be two point clouds in $\mathbb{R}^d$. The goal of the registration algorithm is to find a rigid body transform α composed of a rotation matrix $\mathbf{R}$ and a translation vector $\mathbf{t}$ that best aligns the *data* PCD Q to match the *model* PCD P.

Registration algorithms based on ICP work as follows. Given the initial position of the data with respect to the model, the algorithm chooses a set of k point pairs $(\mathbf{p}_i, \mathbf{q}_i)$ from the model and the data. The distance between the model and data PCDs is approximated by the sum of distances between the point pairs. The algorithm then searches for the rigid transform that minimizes the residual distance, ε, between the model and the transformed data:

$$\varepsilon(\alpha) = \sum_{i=1}^{k} d^2(\alpha(\mathbf{q}_i), \mathbf{p}_i), \tag{1}$$

where d can be point-to-point distance of Besl or point-to-plane distance of Chen and Medioni. Notice, however, that the basic assumption is that the sum of squared distances between pairs of points is a good approximation for the distance between two PCDs.

In the point cloud setting, we actually know that the model and data PCDs are not arbitrary collections of points, but are sampled from some underlying surfaces Φ_P and Φ_Q. In this case, it is more appropriate to minimize the distance from the data PCD to the *surface* represented by the model PCD. Pottmann and Hofer showed that when the data and the model are close, the point-to-plane distance is a good approximation to the distance between a data point and the surface represented by the model PCD. On the other hand, when the model and the data are far apart, the point-to-point distance is a better choice [PLH02].

The goodness of a given error metric is determined by two properties. First, we would like the error metric to accurately reflect the distance between a data point $\mathbf{q}$ and the surface represented by the model PCD. Second, we would like the distance approximation to be valid not just at a point $\mathbf{q}$, but in a neighborhood around it. Both point-to-point and point-to-plane error metrics are based only on first order information about the underlying input surfaces. As a result, they do not provide a good approximation to the distance when we move around in the neighborhood of a data point $\mathbf{q}$.

In this paper, we are concerned with developing a good approximation to the distance function between two point clouds. In order to show theoretical bounds on the convergence behavior of registration algorithms based on our distance function, we pose the problem of registration of two point clouds as an optimization problem over the space of rigid transforms. This leads to the following optimization problem: we are searching for the best rigid transform $\alpha = (\mathbf{R}, \mathbf{t})$ that minimizes the error given by

$$\varepsilon(\alpha) = \sum_{i=1}^{m} d^2(\mathbf{R}\mathbf{q}_i + \mathbf{t}, \Phi_P), \tag{2}$$

where $\mathbf{R}$ is the rotation matrix and $\mathbf{t}$ is the translation vector. The function $d^2(\mathbf{R}\mathbf{q}_i + \mathbf{t}, \Phi_P)$ is the distance from the transformed data point $\mathbf{q}_i$ to the surface represented by the PCD P. Given the optimization problem of Equation 2, it is clear that the convergence behavior of any such optimization procedure depends on the accuracy of the function d^2. We call d^2 the *squared distance function* to the surface Φ_P. Since the surface Φ_P from which the point model was sampled is generally complicated and unknown, a good approximant to the squared distance function is required.

Contributions of the present paper

We develop a quadratic approximant to the squared distance function to the surface represented by a point cloud, and use this approximant in a registration algorithm. Our approximant has the desired property of being valid not only at the query point where it is computed, but also in a neighborhood around the query point. This property allows us to pose the registration problem in an optimization framework, and use methods such as Newton iteration, that depend on computing accurate derivatives of the objective function. In our optimization framework using the squared distance function, point-to-point and point-to-plane ICP variants are reduced to two special cases of the general minimization problem.

Our distance function approximates the squared distance from a data query point to the surface represented by the model PCD up to second order. We develop two methods for computing such local quadratic approximants. The first, uses local curvature of the surface to incorporate second order information into the squared distance function. The second method, approximates the global error landscape by locally fitting quadric patches to the squared distance function to the surface. The quadric patches are stored in a special octree like structure called the *d2tree*. For any point $\mathbf{q}$, the registration algorithm queries this special structure for the corresponding approximant to the squared distance to the surface. Unlike common ICP variants, the d2tree data structure allows us to perform registration without explicitly using nearest neighbors for correspondence.

Both of the above techniques incorporate information about the shape of the neighborhoods of the input surface into the error function. As a result we get better convergence behavior than purely local methods of ICP. Using our distance function, we develop a registration algorithm for point clouds that has fast convergence and is more stable behavior than standard ICP variants. When two point clouds are close to each other, our algorithm has quadratic convergence, similar to point-to-plane ICP. However, unlike point-to-plane ICP, our algorithm has more stable convergence behavior and is less prone to oscillations when the initial distance between the model and the data is large.

3. Registration using the squared distance function

In this section, we assume the existence of a function d^2 that for any point $\mathbf{x} \in \mathbb{R}^d$, gives the squared distance to the model PCD surface Φ_P. Such a squared distance function defines the error landscape for our objective function as indicated by Equation 2. So this function d^2 is important for registration algorithms. Later in Section 4, we describe how to generate local quadratic approximants F^+ of this function $d^2(\mathbf{x}, \Phi_P)$. Assuming these approximants are available, we now show how the registration problem can be solved in a least squared sense by a gradient descent search. Simply put, we try to place one point cloud in the squared distance field of the other, in order to minimize the placement error. Given Φ_P, we expect the points near its *medial axis* $MA(\Phi_P)$ to have bad quadratic approximants, since locally, the squared distance function is not smooth. If we detect such points, we leave them out of our optimization procedure.

We employ an iterative scheme to solve the nonlinear optimization problem over the complex error landscape. At each stage, we solve for a rigid transform composed of a rotation $\mathbf{R}$ followed by a translation $\mathbf{t}$. We use F^+ to solve for the rigid transform $\alpha = (\mathbf{R}, \mathbf{t})$ that brings the data PCD Q to the model PCD P. We apply this transform and repeat.

3.1. Registration in 2D

We first explain the process in 2D. At any point $\mathbf{x} = [x \;\; y] \in \mathbb{R}^2$, we assume the availability of an approximant F^+ such that $F^+(\mathbf{x}) \approx d^2(\mathbf{x}, \Phi_P)$. Let F^+ be specified in the form

$$F^+(\mathbf{x}) = Ax^2 + Bxy + Cy^2 + Dx + Ey + F,$$

where A, B, C, D, E, F are the coefficients of the approximant. More compactly, we can write F^+ in a quadratic form as

$$F^+(\mathbf{x}) = \begin{bmatrix} x & y & 1 \end{bmatrix} \quad \mathcal{Q}_{\mathbf{x}} \quad \begin{bmatrix} x & y & 1 \end{bmatrix}^T, \qquad (3)$$

where $\mathcal{Q}_{\mathbf{x}}$ is a symmetric matrix that depends on $\mathbf{x}$. Importantly, the approximant $F^+(\mathbf{x})$ is a valid locally around $\mathbf{x}$.

We denote any point $\mathbf{q}_i$ of the data PCD Q by $[x_i \;\; y_i]$. Let a matrix $\mathbf{R}$ corresponds to a rotation by angle θ around the origin. Our goal is to solve for a rigid transform, which consists of $\mathbf{R}$ followed by a translation vector $\mathbf{t} = [t_x \;\; t_y]$, that minimizes $\sum_{i=1}^{m} F^+(\mathbf{R}\mathbf{q}_i + \mathbf{t})$. For small θ, we can linearize the rotation by using $\sin\theta \approx \theta$ and $\cos\theta \approx 1$. So after each iteration step, we get, $[x_i \;\; y_i] \mapsto [x_i - \theta y_i + t_x \quad \theta x_i + y_i + t_y]$. If locally $\mathcal{Q}_{\mathbf{q}_i}$ approximately stays fixed, the residual error between the transformed data and the model PCDs is given by

$$\begin{aligned} \varepsilon(\theta, t_x, t_y) = \sum_{i=1}^{m} & \begin{bmatrix} x_i - \theta y_i + t_x & \theta x_i + y_i + t_y & 1 \end{bmatrix} \mathcal{Q}_{\mathbf{q}_i} \\ & \begin{bmatrix} x_i - \theta y_i + t_x & \theta x_i + y_i + t_y & 1 \end{bmatrix}^T, \end{aligned} \qquad (4)$$

where $\mathcal{Q}_{\mathbf{q}_i}$ denotes the matrix representing the approximant of the squared distance function to Φ_P around the point $\mathbf{q}_i$. Our goal is to find values of θ, t_x, and t_y that minimizes this residual error. Setting the respective partial derivatives of the error ε to zero, we get the following linear system

$$\begin{aligned} \begin{bmatrix} \sum_{i=1}^{m} \begin{pmatrix} \mathcal{I}_i & \mathcal{J}_i & \mathcal{K}_i \\ \mathcal{J}_i & 2A_i & B_i \\ \mathcal{K}_i & B_i & 2C_i \end{pmatrix} \end{bmatrix} \begin{bmatrix} \theta \\ t_x \\ t_y \end{bmatrix} = \\ - \begin{bmatrix} \sum_{i=1}^{m} \begin{pmatrix} \mathcal{L}_i \\ 2A_i x_i + B_i y_i + D_i \\ B_i x_i + 2C_i y_i + E_i \end{pmatrix} \end{bmatrix}, \end{aligned} \qquad (5)$$

where $\mathcal{I}_i = 2(C_i x_i^2 - B_i x_i y_i + A_i y_i^2)$, $\mathcal{J}_i = B_i x_i - 2A_i y_i$, $\mathcal{K}_i = 2C_i x_i - B_i y_i$, $\mathcal{L}_i = B_i(x_i^2 - y_i^2) + 2(C_i - A_i)x_i y_i + E_i x_i - D_i y_i$ and, $A_i, B_i, C_i, D_i, E_i, F_i$ denotes the entries of the matrix $\mathcal{Q}_{\mathbf{q}_i}$. The transformation resulting from solving Equation 5 is applied to Q. This completes one iteration of our gradient descent process. Next we use the approximants corresponding to the new positions of $\mathbf{q}_i$ to get another linear system whose solution is again applied to the data PCD. This process is iterated until the residual error falls below a pre-defined error threshold or a maximum number of iteration steps is reached.

Since we do not make any assumption about the initial alignment of P and Q, the rigid transform computed at any step, can be large. In such cases, we can only take small steps in the direction of the transform because its computation is based on approximants that are valid locally. This issue of applying a $1/\eta$-fraction of a rigid transform is an important problem and has been studied in depth in other places [Ale02].

We propose a simple way for computing fractional transforms. In our notation, the computed transform vector $[\theta \;\; t_x \;\; t_y]$ denotes a rigid transform composed of a rotation matrix $\mathbf{R}$ followed by a translation vector $\mathbf{t}$. This maps $\mathbf{q}$ to a point $\mathbf{Rq}+\mathbf{t}$. Let the fractional transform be composed of a rotation matrix $\mathbf{R}'$ and a translation vector $\mathbf{t}'$. We define $(\mathbf{R}',\mathbf{t}')$ to be a $1/\eta$ fraction of $(\mathbf{R},\mathbf{t})$ if the following relation holds,

$$(..(\mathbf{R}'(\mathbf{R}'\mathbf{q}+\mathbf{t}')+\mathbf{t}')...\eta \text{ times}) \equiv \mathbf{Rq}+\mathbf{t}. \tag{6}$$

From this relation, we can get $\mathbf{R}' = \mathbf{R}^{1/\eta}$, and $\mathbf{t}' = (\mathbf{R}-\mathbf{I})^{-1}(\mathbf{R}'-\mathbf{I})\mathbf{t}$ where, $\mathbf{I}$ is the identity matrix. By $\mathbf{R}^{1/\eta}$, we mean a rotation around the origin by angle θ/η. We defer the important issue of choosing a suitable value for η to Section 5.

3.2. Registration in 3D

In this section, we extend the results from the previous section to point cloud surface data in $\mathbb{R}^3$. For any point $\mathbf{x} = [x \;\; y \;\; z] \in \mathbb{R}^3$, let the corresponding local quadratic approximant F^+ be specified in the form

$$F^+(\mathbf{x}) = Ax^2 + Bxy + Cy^2 + Dxz + Eyz + Fz^2 + Gx + Hy + Iz\,, \tag{7}$$

where A through I are the coefficients of the quadratic approximant. With slight abuse of notation, this equation can be written in a quadratic form as $F^+(\mathbf{x}) = \mathbf{x}\mathcal{Q}_{\mathbf{x}}\mathbf{x}^T$, where $\mathbf{x}$ now denotes $[x \;\; y \;\; z \;\; 1]$.

Once again, our goal is to find the rigid transform which brings Q in best alignment with P. Let the rigid transform be composed of a rotation matrix, $\mathbf{R}$, that is parameterized by three angles (α,β,γ) in the *X-Y-Z fixed angle orientation* convention, followed by a translation vector $\mathbf{t} = [t_x \;\; t_y \;\; t_z]$.

Under small motion, the rotation matrix can be linearized as,

$$\begin{aligned}\mathbf{R} &= \begin{bmatrix}\cos\alpha & -\sin\alpha & 0\\ \sin\alpha & \cos\alpha & 0\\ 0 & 0 & 1\end{bmatrix}\begin{bmatrix}\cos\beta & 0 & \sin\beta\\ 0 & 1 & 0\\ -\sin\beta & 0 & \cos\beta\end{bmatrix}\\ &\quad\begin{bmatrix}1 & 0 & 0\\ 0 & \cos\gamma & -\sin\gamma\\ 0 & \sin\gamma & \cos\gamma\end{bmatrix}\\ &\approx \begin{bmatrix}1 & -\alpha & \beta\\ \alpha & 1 & -\gamma\\ -\beta & \gamma & 1\end{bmatrix}.\end{aligned} \tag{8}$$

Hence $\mathbf{q} \mapsto \mathbf{Rq}+\mathbf{t}$.

Now the registration problem reduces to finding values of $\alpha,\beta,\gamma,t_x,t_y,t_z$ that minimize the residual error

$$\varepsilon(\alpha,\beta,\gamma,t_x,t_y,t_z) = \sum_{i=1}^{m}(\mathbf{Rq}_i+\mathbf{t})\mathcal{Q}_{\mathbf{q}_i}(\mathbf{Rq}_i+\mathbf{t})^T. \tag{9}$$

This least square problem can be solved by setting the respective partial derivatives to zero. The resulting linear system is given by

$$\begin{aligned}&\left[\sum_{i=1}^{m}\begin{pmatrix}\mathcal{P}_i & \mathcal{S}_i\\ \mathcal{S}_i^T & \mathcal{R}_i\end{pmatrix}\right]\begin{bmatrix}\alpha & \beta & \gamma & t_x & t_y & t_z\end{bmatrix}^T =\\ &\qquad -\begin{bmatrix}\sum_{i=1}^{m}\begin{pmatrix}\mathcal{U}_i\\ \mathcal{V}_i\\ \mathcal{W}_i\\ 2A_ix_i+B_iy_i+D_iz_i+G_i\\ B_ix_i+2C_iy_i+E_iz_i+H_i\\ D_ix_i+E_iy_i+2F_iz_i+I_i\end{pmatrix}\end{bmatrix},\end{aligned} \tag{10}$$

where,

$$\begin{aligned}\mathcal{P}_i &= \begin{bmatrix}\mathcal{J}_i & \mathcal{M}_i & \mathcal{N}_i\\ \mathcal{M}_i & \mathcal{K}_i & \mathcal{T}_i\\ \mathcal{N}_i & \mathcal{T}_i & \mathcal{L}_i\end{bmatrix},\\ \mathcal{S}_i &= \begin{bmatrix}B_ix_i-2A_iy_i & 2C_ix_i-B_iy_i & E_ix_i-D_iy_i\\ -D_ix_i+2A_iz_i & -E_ix_i+B_iz_i & -2F_ix_i+D_iz_i\\ D_iy_i-B_iz_i & E_iy_i-2C_iz_i & 2F_iy_i-E_iz_i\end{bmatrix},\\ \mathcal{R}_i &= \begin{bmatrix}2A_i & B_i & D_i\\ B_i & 2C_i & E_i\\ D_i & E_i & 2F_i\end{bmatrix},\\ \mathcal{J}_i &= 2(C_ix_i^2 - B_ix_iy_i + A_iy_i^2),\\ \mathcal{K}_i &= 2(F_ix_i^2 - D_ix_iz_i + A_iz_i^2),\\ \mathcal{L}_i &= 2(F_iy_i^2 - E_iy_iz_i + C_iz_i^2),\end{aligned}$$

$$
\begin{aligned}
\mathcal{M}_i &= -E_i x_i^2 + D_i x_i y_i + B_i x_i z_i - 2A_i y_i z_i, \\
\mathcal{N}_i &= -D_i y_i^2 + E_i x_i y_i - 2C_i x_i z_i + B_i y_i z_i, \\
\mathcal{T}_i &= -B_i z_i^2 - 2F_i x_i y_i + E_i x_i z_i + D_i y_i z_i, \\
\mathcal{U}_i &= B_i(x_i^2 - y_i^2) + 2(C_i - A_i) x_i y_i + E_i x_i z_i - D_i y_i z_i + H_i x_i - G_i y_i, \\
\mathcal{V}_i &= D_i(z_i^2 - x_i^2) - E_i x_i y_i + 2(A_i - F_i) x_i z_i + B_i y_i z_i - I_i x_i + G_i z_i, \\
\mathcal{W}_i &= E_i(y_i^2 - z_i^2) + D_i x_i y_i - B_i x_i z_i + 2(F_i - C_i) y_i z_i + I_i y_i - H_i z_i,
\end{aligned}
$$

and A_i through I_i correspond to the entries of the matrix $\mathcal{Q}_{\mathbf{q}_i}$ that represents the local quadratic approximant around the point $\mathbf{q}_i$.

As in the 2D case, whenever the computed transform $(\mathbf{R},\mathbf{t})$ is large, we utilize a fractional transform given by $\mathbf{R}' = \mathbf{R}^{1/\eta}$ and $\mathbf{t}' = (\mathbf{R}-\mathbf{I})^{-1}(\mathbf{R}'-\mathbf{I})\mathbf{t}$ where, $\mathbf{I}$ denotes the identity matrix. A $1/\eta$-fraction of the rotation matrix $\mathbf{R}$ can be computed by the techniques proposed by Alexa [Ale02].

4. Squared Distance Function

Given a 3D point cloud P, we describe two methods for constructing a quadratic approximant F^+ to the squared distance function d^2 from any point $\mathbf{x} \in \mathbb{R}^3 \setminus MA(\Phi_P)$ to Φ_P. At any point $\mathbf{x}$, our goal is to construct an approximant F^+ such that $F^+(\mathbf{x}) \approx d^2(\mathbf{x},\Phi_P)$ is second order accurate. Points on the medial axis $MA(\Phi_P)$ have non-differentiable squared distance function and hence, their second order accurate approximants do not exist. In the construction phase, we ensure that the approximants are non-negative over $\mathbb{R}^3$, since F^+ is used as an objective function in a minimization process as shown in Section 3. In 2D, similar approximants can be easily computed.

Before we describe how to compute F^+ for a given PCD, we summarize a few basic results on the squared distance function of a surface as observed by Pottmann and Hofer [PH03]. For each point on a given surface, we assume that the unit normal $\vec{n}$ along with the principal curvature directions $\vec{e}_1, \vec{e}_2$ are given. These three unit vectors combine to form a local coordinate system called the *principal frame*. At *umbilical points*, where the principal curvature directions are not well-defined, any two orthogonal unit vectors on the tangent plane may be used as $\vec{e}_1, \vec{e}_2$. Let ρ_i be the *principal radius of curvature* in the direction $\vec{e}_i$. The *normal footpoint* $\mathbf{y}$ denotes the closest point on the surface from $\mathbf{x}$. Let x_1, x_2, x_3 represent the coordinates of $\mathbf{x}$ in the principal frame at $\mathbf{y}$. The signed distance from $\mathbf{x}$ to its normal footprint is denoted by d. The sign of d is positive if $\mathbf{x}$ and the centers of the osculating circles at $\mathbf{y}$, lie on the same side of the surface around $\mathbf{y}$.

The second order Taylor approximant [PH03] of the squared distance function to the surface at a point $\mathbf{x}$ can be expressed in the principal frame at $\mathbf{y}$ as

$$F_d(\mathbf{x}) = F_d(x_1,x_2,x_3) = \frac{d}{d-\rho_1}x_1^2 + \frac{d}{d-\rho_2}x_2^2 + x_3^2. \quad (11)$$

We shall use δ_j, $j = 1,2$, to denote $d/(d-\rho_j)$.

Let us look at two important special cases.

- For $d = 0$ we obtain
$$F_d(x_1,x_2,x_3) = x_3^2.$$
Thus, if we are close to the surface, i.e. in its *'near-field'*, the squared distance function to the tangent plane at the normal footpoint, is a quadratic approximant.
- For $d = \infty$ we obtain
$$F_d(x_1,x_2,x_3) = x_1^2 + x_2^2 + x_3^2,$$
which is the squared distance from $\mathbf{x}$ to its footpoint $\mathbf{y}$. So the distances to normal footpoints are second order accurate if we are in the *'far-field'* of the surface.

In order to use F_d as an objective function for a minimization, we want the approximant to be non-negative over $\mathbb{R}^3$. To this end, we replace δ_j with

$$\hat{\delta}_j = \begin{cases} d/(d-\rho_j) & \text{if } d < 0, \\ 0 & \text{otherwise.} \end{cases}$$

The resulting approximant F^+ is positive definite and is given by

$$F^+(\mathbf{x}) = \hat{\delta}_1 x_1^2 + \hat{\delta}_2 x_2^2 + x_3^2. \quad (12)$$

This quadratic approximant F^+ of d^2 is simply a weighted sum of the squared distance functions x_1^2, x_2^2, x_3^2 to three planes: the two principal planes and the tangent plane at the normal footpoint. Based on this observation, we transform Equation 12 to the global coordinate system as,

$$
\begin{aligned}
F^+(\mathbf{x}) &= \hat{\delta}_1 (\vec{e}_1 \cdot (\mathbf{x}-\mathbf{y}))^2 + \hat{\delta}_2 (\vec{e}_2 \cdot (\mathbf{x}-\mathbf{y}))^2 + \\
&\quad (\vec{n} \cdot (\mathbf{x}-\mathbf{y}))^2. \quad (13)
\end{aligned}
$$

We can now express this equation in the form given by Equation 7 to get values for the coefficients A through I.

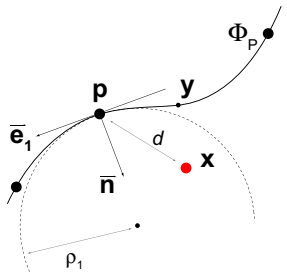

Figure 1: *A query point $\mathbf{x} \in \mathbb{R}^2$ has a footpoint $\mathbf{y} \in \mathbb{R}^2$ on the surface Φ_P represented by a PCD P. We approximate the footpoint by $\mathbf{p}$, the closest point in P from $\mathbf{x}$. The principal frame at the footpoint is spanned by $\vec{e}_1$ and $\vec{n}$. The osculating circle to Φ_P at $\mathbf{p}$ has a radius of curvature ρ_1. In the figure, the signed distance d from $\mathbf{x}$ to the footpoint $\mathbf{p}$ is positive.*

4.1. On-Demand Computation

Given a point **x**, in our first method for computing a second order accurate squared distance field F^+ for a given PCD P, we perform an *on-demand* computation of Equation 13. For this method we first need to compute the normal footpoint of **x** to P. As an approximation, we treat **p**, the closest point to **x** in P, as the normal footpoint. This point is found using an approximate nearest neighbor data structure [AMN*98]. Figure 1 shows the scenario in 2D. When the P is a sparse sampling of Φ_P, we can use the underlying moving least square (MLS) surface to get a better approximation for the normal footpoint [AK04]. We further need to evaluate local curvatures at points of P in order to use Equation 13. These quantities are computed in the preprocessing step of our algorithm.

At each point of a given PCD, we first determine the principal frame using a local covariance analysis as detailed in [CP03, MNG04]. If the the underlying surface Φ_P is *regular*, at each of point **p** of P, a local parametrization exists. In the principal frame at **p**, we estimate the local surface by least square fitting a quadratic function of the form $ax^2 + bxy + cy^2 + dx + ey$ to the neighboring points in P. Once we estimate the coefficients a through e, we can use facts from differential geometry to get the Gaussian curvature K and the mean curvature H using

$$K = \frac{4ac - b^2}{(1+d^2+e^2)^2} \qquad (14)$$

$$H = \frac{a(1+e^2) - bde + c(1+d^2)}{(1+d^2+e^2)^2}.$$

Finally we evalute the principal radii of curvature ρ_i as $1/(H \pm \sqrt{H^2 - K})$.

The correctness of these estimates depends on the sampling density of the given PCD and on the measurement noise. Further, the neighborhood size used for the least square fits can be adapted to the local shape [MNG04]. In low noise scenarios, when the local estimates of the differential properties can be reliably computed, the approximants F^+ given by this method are good.

4.2. Quadratic Approximants using d2Tree

Our second method for computing approximate quadratic approximants involves least square fitting of quadratic patches to a sampled squared distance function. For a given PCD, these quadratic patches are pre-computed and stored in a special data structure called the *d2Tree*[LPZ03]. Given any point **x**, in this method we do a point location in the cells of the d2Tree and return the quadratic approximant stored in the corresponding cell.

Simply put, the d2Tree is an octree-like (quad-tree in 2D) data structure, where each cell stores a quadratic fit to the squared distance function, correct to some maximum error threshold. The approximants are stored in the form as given in Equation 7 (Equation 3 in 2D). Details of a top-down construction of *d2Tree* can be found in [LPZ03]. Here we describe a bottom-up construction, which is computationally more efficient.

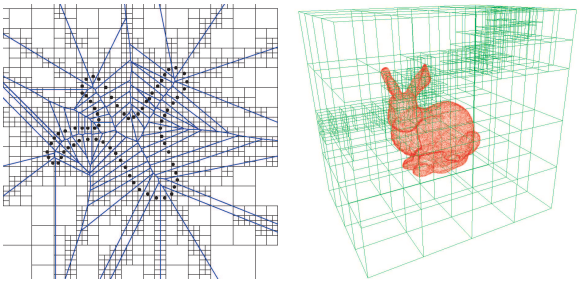

Figure 2: *d2Tree can be used in 2D (left) and in 3D (right) to store quadratic approximants of the squared distance fields correct to some error threshold. The maximum number of levels and the error threshold, which are parameters used during the construction of this quad-tree like data-structure, determine the size of the cells. In the 2D case, we overlay the Voronoi diagram of the PCD on top of the d2Tree, to illustrate that small cells are created around the medial axis.*

As a first step, a sampled squared distance field is build for an input PCD by sweeping the space starting from the PCD P and propagating the squared distance information [LPZ03]. Depending on the number of levels, which is an input to the algorithm, the space is divided into smallest allowable cells (see Figure 2). In each cell, a quadric patch, that best fits the sampled squared distance field, is computed. The fitting error and the matrices used to compute the coefficients of the fit are saved in each cell. At the next level, the neighboring cells (four in 2D and eight in 3D) are merged to form a larger quadric patch, only if the resulting fitting error is below the given error threshold. The larger quadric patches can be efficiently fitted by re-using the matrices stored in the smaller cells. The quadratic matrix stored in any cell is made positive semi-definite during construction. The maximum number of levels of the tree and the error threshold are the required parameters for the construction of this data structure. Notice that there exists a tradeoff between the size of the cells and the accuracy of the quadratic approximants.

Unlike the on-demand method for computing quadratic approximants described before, the d2Tree approach does not need estimates of the local curvature or any nearest neighbor structure. Quadratic approximants computed by d2Tree, implicitly learn the local curvature information by fitting quadrics to the sampled squared distance field. We find this method to be robust to noisy or under-sampled PCD. Given a query point **x**, computing $F^+(\mathbf{x})$ simply involves a point location in this d2Tree structure, and does not re-

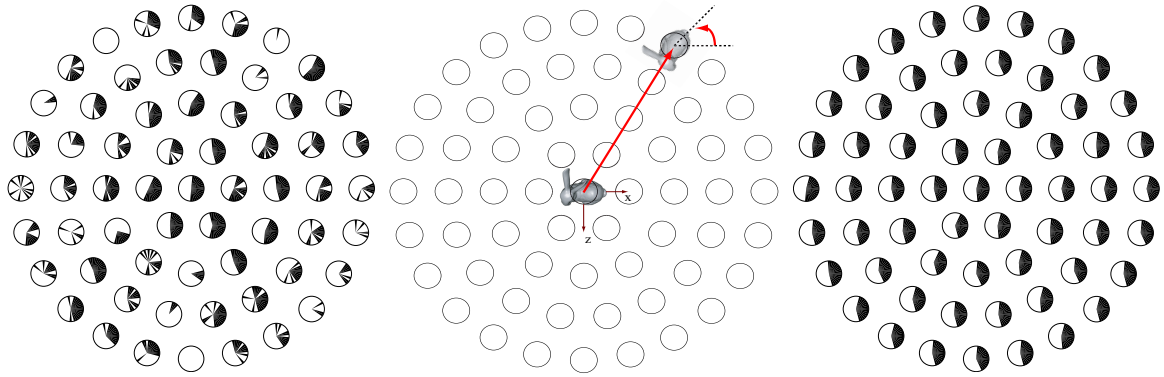

Figure 3: *Funnel of Convergence for aligning a bunny with itself. The bunny is rotated (around the y-axis) and translated (along the x-z plane) to generate different initial positions for the* data PCD. *The figure in the middle denotes the sampling pattern used to get the initial positions. The rotation angle is sampled at* 10° *intervals, while the maximum radial translation of the bunny is around* $5\times$ *the height of the bunny. Regions in black denote convergence to the correct solution. The convergence funnel of the point-to-plane ICP (left) is found to be quite narrow and unstable. Under similar conditions, our on-demand algorithm (right) is found to have a significantly broader, and much more stable convergence funnel. The shape of the convergence funnel corresponding to our d2Tree based approach is similar.*

quire any explicit correspondence between points of the input PCDs.

4.3. Point-to-point and point-to-plane ICP error metrics as special cases of quadratic approximant

In our framework, the standard ICP algorithms can be reduced to special cases by selecting suitable approximants to the squared distance function. Basic point-to-point ICP uses squared distance to the closest point as its approximant, i.e. $F^+(\mathbf{x}) = \|\mathbf{x}-\mathbf{y}\|^2$ while the Chen–Medioni point-to-plane ICP uses $F^+(\mathbf{x}) = (\vec{n}\cdot(\mathbf{x}-\mathbf{y}))^2$ as the quadratic approximant (Equation 7). In the form given by Equation 13, point-to-point ICP has $\hat{\delta}_i = 1$, and point-to-plane ICP has $\hat{\delta}_i = 0$. However, as pointed out in Section 4, such approximants are second order accurate only in the 'far field' and 'near field' of the PCD, respectively, and hence neither of these algorithms is well-behaved for all relative placements of the model and data PCDs.

5. Convergence Issues

In this section, we discuss the convergence behavior of point-to-point and point-to-plane ICP algorithms, and then give bounds on the convergence rates for our algorithm. In contrast to ICP algorithms, our scheme uses second order accurate square distance approximants at all point in space, and hence, exhibits better convergence properties.

Experimentally, the point-to-point ICP algorithm converges linearly. In a recent result, Pottmann has provided theoretical justification for this behavior [Pot04]. We define a *low residual* problem as one where the data shape fits the model shape well, and a *zero residual* problem as one where the fit is exact. For a low residual problem, when the minimizer is approached tangentially, point-to-point ICP has a very slow convergence[Pot04, RL01].

We recall that the Chen-Medioni approach iteratively minimizes the sum of squared distances to the tangent planes at the normal footpoints of the current data point locations. This implies a gradient descent in the error landscape, where the squared distance to the tangent plane is used to define the objective function. From an optimization perspective, this process corresponds to Gauss–Newton iteration. For a zero residual problem, and a sufficiently good initial position, this algorithm converges quadratically [Pot04]. In practice, the Chen-Medioni method also works well for low residual problems. Notice that in the 'near-field', the squared distance to the tangent plane is a second order accurate approximant to d^2. So point-to-plane ICP performs much better for fine registration than the point-to-point ICP algorithm. However, there is no reason to expect convergence when the two PCDs are sufficiently apart in the transform space, and in practice, this is found to be true.

For low residue problems, our algorithm also exhibits quadratic convergence, which means that the error reduction is of the form

$$\varepsilon_+ \leq C\varepsilon_c^2 \tag{15}$$

where, $C \in (0,1)$ denotes the convergence constant, ε_c denotes the residual error in the current step, and ε_+ denotes the residual error after application of the computed rigid transform. For each point $\mathbf{q}_i$, our algorithm computes a second order approximant F^+ to the squared distance function of the surface Φ_P represented by PCD P. Using these approximants, we derive the best aligning transform for

Algorithm 1 Brings a data PCD Q in alignment to a model PCD P using *on-demand* computation or *d2Tree* based approach or point-to-point ICP or point-to-plane ICP.

if using *on-demand* method **then**
  Build an approximate nearest neighbor structure for P.
  Pre-compute principal frame and radii of curvature ρ_j, $j = 1,2$, at each point $\mathbf{p}_i \in P$ as described in Section 4.1.
else if using *d2Tree* method **then**
  Build *d2Tree* with a suitable error threshold. (see Section 4.2)
end if
count $\leftarrow$ MAXCOUNT
repeat
  for each point $\mathbf{q}_i \in Q$ **do**
    Compute $F^+(\mathbf{q}_i)$ using method described in Section 4.1 for *on-demand* approach. For *d2Tree* based approach refer to Section 4.2. For point-to-point or point-to-plane ICP refer to Section 4.3.
  end for
  Using $F^+(\mathbf{q}_i)$, build and solve the linear system given in Section 3.
  if Armijo condition not satisfied **then**
    Take $1/\eta$ fraction of the computed rigid transform (see Section 3). Value of η chosen via line-search to satisfy Armijo rule (see Section 5).
  end if
  if (residual error $<$ ERRORTHRESHOLD) **then**
    break
  end if
  count $\leftarrow$ *count* -1
until *count* $\neq 0$

Q by following a gradient descent with Newton iteration steps [Kel99] in the rigid transform group (see Section 3). We continue until the residual error falls below a pre-defined threshold or a maximum number of iteration steps has been reached. Since the presented method is a Newton algorithm, it converges quadratically [Kel99, Pot04].

As mentioned in Section 3, if the residue ε is large, we apply only $1/\eta$ fraction of the computed transform to prevent oscillations, or even divergence. Various line search strategies exist for choosing good values for η [Kel99]. In our implementation, we used the *Armijo condition* [Kel99] to select η. This results in a *damped Gauss–Newton algorithm*. It is well known in optimization, that algorithms which uses the Armijo rule converge linearly. Hence, to ensure faster convergence for large residue problems, it may be better to select a quadratic approximant of the motion, instead of a linear one [Pot04].

Our gradient descent based optimization can get stuck at a local minimum. We bound the maximum residual error for a given PCD pair, and use it to detect a local minimum. A point cloud P, sampled from a surface Φ_P, is said to be *sampled r-dense*, if any sphere with radius r centered on Φ_P contains at least one sample point in P [HDD*92]. Suppose that the model PCD P is an r_P-dense sampling of Φ_P. Further, assume the measurement noise only perturbs any point by a maximum amount of σ_P and σ_Q respectively for the given PCDs P, Q. Under this restrictive sampling model, when Φ_Q represents a subset of Φ_P, a final residual matching error ε greater than $M(r_P + \sigma_P + \sigma_Q)^2$ indicates the algorithm has been stuck at a local minima during the search process. When such a situation happens, we may randomly perturb Q to a new orientation, and try to align the two PCDs starting from that position.

To further study this global convergence property, we define the *funnel of convergence* for a registration algorithm, as the set of all initial poses of a PCD Q, which can be successfully aligned with P, using the given algorithm. Notice that the funnel only measures global convergence and not speed. A *broad funnel* indicates that the algorithm can successfully handle a wide range of initial positions. An algorithm is said to have a *stable funnel* if the convergence zones, in the transform space, are clustered and not arbitrarily distributed. A stable funnel is desirable, since this can enable a systematic way of generating positions for random re-starts, using some branch and bound approach. Experimentally, we observe that our algorithm has a broader and more stable funnel of convergence as compared to the point-to-plane ICP variant. This can be explained by the fact, that our algorithm makes use of higher order surface properties.

6. Results

We test our algorithm on a variety of data sets with different amounts of noise, and compare its performance against point-to-point and point-to-plane ICP algorithms.

A brief summary of our registration framework is given in Algorithm 1. We compare the performance of approaches based on the choice of the approximant F^+ of the squared distance function at any point $\mathbf{x}$:

1. on-demand computation of quadratic approximant (Section 4.1)
2. quadratic approximant using d2Tree (Section 4.2)
3. squared point-to-point distance (point-to-point ICP)
4. squared distance to the tangent point at the footpoint of $\mathbf{x}$ (point-to-plane ICP)

In our implementation, we test for Armijo condition to ensure stability of the algorithms.

On the bunny model, which consists of $50,282$ points, we compare the convergence funnel of point-to-plane ICP and that of our algorithm based on on-demand computation. A copy of the same PCD is rotated around the y-axis and translated to different positions along the x-z plane. Figure 3

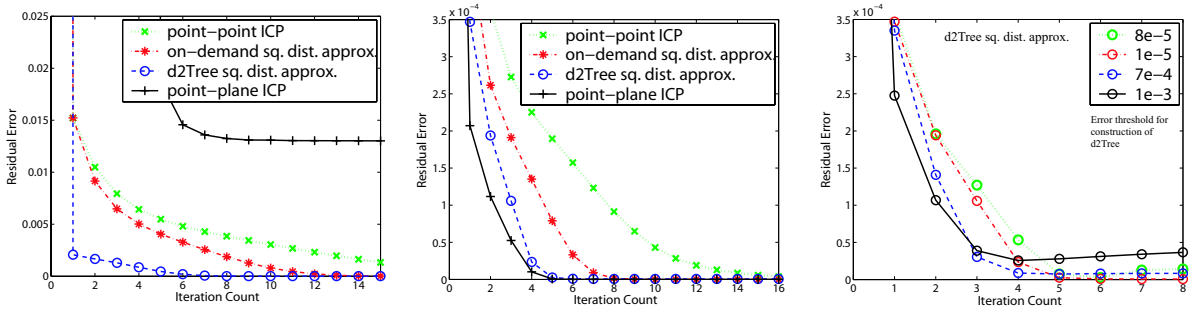

Figure 4: *Plots of residual error vs iteration count for bunny PCD. When the* model *and* data *PCDs, both corrupted with noise, start far apart in the transform space, the point-to-plane ICP fails to converge to the correct solution (left). However, algorithms using any of the other square distance approximants do converge, with the d2Tree based approach converging fastest. Middle: For good initial position and small residual problem (the two PCDs align well), the point-to-point ICP algorithm has a slow convergence, while optimization based on any of the other squared distance approximants, converges quadratically. The figure to the right shows the effect of changing the error threshold value used for constructing the d2Tree. As the threshold is increased, a larger neighborhood of the squared distance function is captured by each of the cells of the tree and hence, the algorithm converges faster. However, for sufficiently high error threshold values, the distance approximants get too crude, and the method starts to deteriorate.*

shows that the convergence funnel of point-to-plane ICP is quite narrow when the initial displacement is large. Under similar conditions, our algorithm is found to have a much broader convergence funnel. Our convergence funnel is also more stable. Experimentally, the initial translation is found to have little effect on the convergence of our algorithm.

Next we compare the convergence rates for the four variants listed before. For both on-demand and d2Tree based approaches, the pre-computation time depends on the size n of the model PCD. For point-to-point, point-to-plane and on-demand computation, at each iteration, F^+ for a point $\mathbf{x}$ can be computed in O(1) time after the nearest neighbor query has been answered. For d2Tree, the nearest neighbor query is replaced by point location in the d2Tree cells. The solution of the linear system involves an inversion of a 6×6 matrix. Since the amount of work in each iteration step for any of the algorithms is roughly same, we simply count the number of iterations for comparing speed.

In Figure 4, we plot the residual error vs iteration count for four approaches. In the presence of noise and for large residues, point-to-plane ICP often fails to converge. In such noisy scenarios, since the estimates of local principal radii are bad, our on-demand algorithm is found to be marginally worse than point-to-point ICP. The d2Tree based method still converges fast, since the cell-sizes automatically get adjusted during their construction phase, to partially average out the effect of noise. However, in low residual cases, for reasons explained in Section 5, all algorithms except for point-to-point ICP converge quadratically. The threshold value used for constructing the d2Tree is also varied. As the threshold is increased, a larger neighborhood of the squared distance function is captured by each of the cells of the tree and hence, the algorithm converges faster. However, for sufficiently high error threshold values, the distance approximants get too crude, and the method starts to deteriorate.

Our algorithm is able to handle the case when the data PCD is a subset of the model PCD. We take a partial scan of the bunny consisting of $17,600$ points. This scan is from scanned data and is corrupted with measurement noise. We used the on-demand algorithm to match the partial scan to the complete bunny model PCD. The starting arrangement and the final match are shown in Figure 5.

Finally we test the robustness of our approach in presence of noise and varying sampling density.We try to align a part (consisting of $14,519$ points) of a ball-joint with the socket of a hip-bone represented by $132,538$ points. Note the sampling density and sampling pattern are vastly different across the two models. The ball-joint is much densely sampled compared to the hip-bone. Even in this case, for

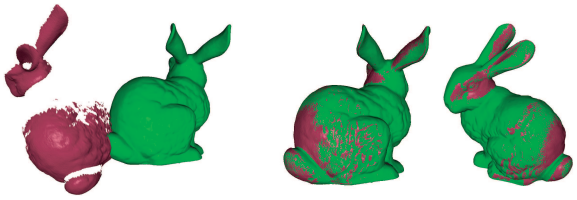

Figure 5: *Partial Match: A partial scan of the bunny (shown in purple) is registered to the bunny, the* model *PCD. The initial arrangement of the PCDs is shown to the left. Our algorithm found the correct match (middle,right) in six iterations.*

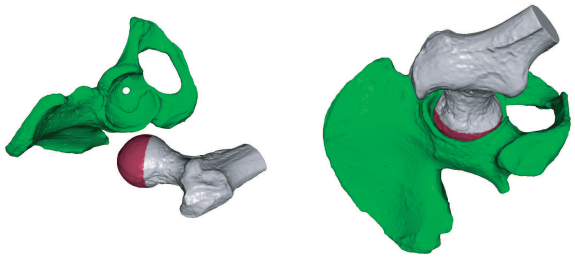

Figure 6: *Partial Match: The goal is to fit a part (shown in purple) of the ball-joint to the hip-bone (shown in green). The starting arrangement (left) and the final alignment (right) are shown. The whole ball-joint is shown to help the reader judge the correctness of the match. Our algorithm is robust enough to handle varying sampling density and noise in the given PCDs.*

reasonable starting positions, we got a good final alignment (see Figure 6). The whole ball-joint is shown just to illustrate the goodness of the alignment. We manually selected a part of the ball-joint to satisfy our constraint that Φ_Q represents a subset of Φ_P.

7. Conclusion and Future Work

We have developed a framework for pairwise registration of point cloud data. In this framework, registration is treated as a distance minimization between the surfaces represented by the PCDs. We develop quadratic approximants of the squared distance function to a point cloud and use the approximants to perform a minimization. Since our approximants are second order accurate, we can use them for a Gauss-Newton optimization. As a result, compared to other commonly used registration algorithms, our algorithm is stabler and has a faster convergence rate.

Extending the framework to compute partial matches will be very useful. Finally, we plan to extend our framework to solve the problem of simultaneously aligning more than two PCDs.

References

[AK04] AMENTA N., KIL Y.: Defining point-set surfaces. In *ACM SIGGRAPH* (to appear 2004). 6

[Ale02] ALEXA M.: Linear combination of transformations. In *Computer graphics and interactive techniques* (2002), pp. 380–387. 4, 5

[AMN*98] ARYA S., MOUNT D. M., NETANYAHU N. S., SILVERMAN R., WU A. Y.: An optimal algorithm for approximate nearest neighbor searching. In *Journal of the ACM* (1998), vol. 45, pp. 891–923. 6

[BM92] BESL P. J., MCKAY N. D.: A method for registration of 3d shapes. In *IEEE Transactions on PAMI* (1992), vol. 14, pp. 239–256. 1, 2

[CLSB92] CHAMPLEBOUX G., LAVALLEE S., SZELISKI R., BRUNIE L.: From accurate range imaging sensor calibration to accurate model-based 3d object localization. In *IEEE Conference of CVPR* (1992), pp. 83–89. 2

[CM91] CHEN Y., MEDIONI G.: Object modeling by registration of multiple range images. In *IEEE Conference on Robotics and Automation* (1991). 1, 2

[CP03] CAZALS F., POUGET M.: Estimating differential quantities using polynomial fitting of osculating jets. In *Eurographics/ACM SIGGRAPH symposium on Geometry processing* (2003), pp. 177–187. 6

[CWPG04] COTTING D., WEYRICH T., PAULY M., GROSS M.: Robust watermarking of point-sampled geometry. In *Shape Modeling International* (2004). 1

[Fit01] FITZGIBBON A. W.: Robust registration of 2d and 3d point sets. In *British Machine Vision Conference* (2001). 2

[GIRL03] GELFAND N., IKEMOTO L., RUSINKIEWICZ S., LEVOY M.: Geometrically stable sampling for the ICP algorithm. In *3D Digital Imaging and Modeling* (2003). 2

[HDD*92] HOPPE H., DEROSE T., DUCHAMP T., MCDONALD J., STUETZLE W.: Surface reconstruction from unorganized points. In *ACM SIGGRAPH* (1992), pp. 71–78. 8

[JH03] JOST T., HUGLI H.: A multi-resolution ICP with heuristic closest point search for fast and robust 3d registration of range images. In *3D Digital Imaging and Modeling* (2003). 2

[Kel99] KELLEY. C. T.: *Iterative Methods for Optimization*. SIAM, Philadelphia, 1999. 8

[LPZ03] LEOPOLDSEDER S., POTTMANN H., ZHAO H.: The d^2-tree: A hierarchical representation of the squared distance function. In *Tech.Rep. 101, Institut of Geometry, Vienna University of Technology* (2003). 6

[MNG04] MITRA N. J., NGUYEN A., GUIBAS L.: Estimating surface normals in noisy point cloud data. In *International Journal of Computational Geometry and Applications* (to appear 2004). 6

[PH03] POTTMANN H., HOFER M.: Geometry of the squared distance function to curves and surfaces. In *Visualization and Mathematics III* (2003), pp. 221–242. 5

[PLH02] POTTMANN H., LEOPOLDSEDER S., HOFER M.: Registration without ICP. In *Technical Report 91, Institut für Geometrie, TU Wien* (2002). 2

[Pot04] POTTMANN H.: Geometry and convergence analysis of registration algorithms. In *Tech.Rep. 117, Geometry Preprint Series, Vienna University of Technology* (2004). 7, 8

[RL01] RUSINKIEWICZ S., LEVOY M.: Efficient variants of the ICP algorithm. In *3D Digital Imaging and Modeling* (2001). 1, 2, 7