Quasi-Newton Solver for Robust Non-Rigid Registration

Yuxin Yao1 Bailin Deng2 Weiwei Xu3 Juyong Zhang1*

1University of Science and Technology of China 2Cardiff University 3Zhejiang University

Abstract

Imperfect data (noise, outliers and partial overlap) and high degrees of freedom make non-rigid registration a classical challenging problem in computer vision. Existing methods typically adopt the $\ell_p$ type robust estimator to regularize the fitting and smoothness, and the proximal operator is used to solve the resulting non-smooth problem. However, the slow convergence of these algorithms limits its wide applications. In this paper, we propose a formulation for robust non-rigid registration based on a globally smooth robust estimator for data fitting and regularization, which can handle outliers and partial overlaps. We apply the majorization-minimization algorithm to the problem, which reduces each iteration to solving a simple least-squares problem with L-BFGS. Extensive experiments demonstrate the effectiveness of our method for non-rigid alignment between two shapes with outliers and partial overlap, with quantitative evaluation showing that it outperforms state-of-the-art methods in terms of registration accuracy and computational speed. The source code is available at https://github.com/Juyong/Fast_RNRR.

1. Introduction

With the popularity of depth acquisition devices such as Kinect, PrimeSense and the depth sensors on smartphones, techniques for 3D object tracking and reconstruction from point clouds have enabled various applications. Non-rigid registration is a fundamental problem for such techniques, especially for the reconstruction of dynamic objects. Since depth maps obtained from structured light or time-of-flight cameras often contain outliers and holes, a robust non-rigid registration algorithm is needed to handle such data. Moreover, real-time applications require high computational efficiency for non-rigid registration.

Given two point clouds sampled from a source surface and a target surface respectively, the aim of non-rigid registration is to find a transformation field for the source point cloud to align it with the target point cloud. This problem is typically solved via optimization. The objective energy often includes alignment terms that measure the deviation between the two point clouds after the transformation, as well as regularization terms that enforce smoothness of the transformation field. Existing methods often formulate these terms using the $\ell_2$-norm, which penalizes alignment and smoothness errors across the whole surface [1, 21, 20]. On the other hand, the ground-truth alignment can potentially induce large errors for these terms in some localized regions of the point clouds, due to noises, outliers, partial overlaps, or articulated motions between the point clouds. Such large localized errors will be inhibited by the $\ell_2$ formulations, which can lead to erroneous alignment results. To improve the alignment accuracy, recent works have utilized sparsity-promoting norms for these terms, such as the $\ell_1$-norm [36, 22, 17] and the $\ell_0$-norm [12]. The sparsity optimization enforces small error metrics on most parts of the point cloud while allowing for large errors in some local regions, improving the robustness of the registration process. However, these sparsity terms can lead to non-smooth problems that are more challenging to solve. Existing methods often employ first-order solvers such as the alternating direction method of multipliers (ADMM), which can suffer from slow convergence to high-accuracy solutions [6].

In this paper, we propose a new approach for robust non-rigid registration with fast convergence. The key idea is to enforce sparsity using the Welsch’s function [14], which has been utilized for robust processing of images [13] and meshes [37]. We formulate an optimization that applies the Welsch’s function to both the alignment error and the regularization, to achieve robust registration. Unlike the $\ell_p$-norms, the Welsch’s function is smooth and does not induce non-smooth optimization. We solve the optimization problem using the majorization-minimization (MM) algorithm [19]. It iteratively constructs a surrogate function for the target energy based on the current variable values and minimizes the surrogate function to update the variables, and is guaranteed to converge to a local minimum. The Welsch’s function enables us to derive a surrogate function in a simple least-squares form, which can be solved efficiently using L-BFGS.

*Corresponding author

17600
Experimental results verify the robustness of our method as well as its superior performance compared to existing robust registration approaches.

In summary, the main contributions of this paper include:

- We formulate an optimization problem for non-rigid registration, using the Welsch’s function to induce sparsity for alignment error and transformation smoothness. The proposed formulation effectively improves the robustness and accuracy of the results.
- We propose an MM algorithm to solve the optimization problem, using L-BFGS to tackle the sub-problems. The combination of MM and L-BFGS greatly improves the computational efficiency of robust non-rigid registration compared to existing approaches.

2. Related work

Non-rigid registration has been widely studied in computer vision and image processing. The reader is referred to [31] for a recent survey on rigid and non-rigid registration of 3D point clouds and meshes. In the following, we focus on works that are closely related to our method.

Various optimization-based methods have been proposed for non-rigid registration. Chui et al. [8] utilized a Thin Plate Spline (TPS) model to represent non-rigid mappings, and alternately update the correspondence and TPS parameters to find an optimized alignment. Following this approach, Brown et al. [7] used a weighted variant of iterative closest point (ICP) to obtain sparse correspondences, and warped scans to global optimal position by TPS. Extending the classical ICP algorithm for rigid registration, Amberg et al. [1] proposed a non-rigid ICP algorithm that gradually decreases the stiffness of regularization and incrementally deforms the source model to the target model. Li et al. [21] adopted an embedded deformation approach [29] to express a non-rigid deformation using transformations defined on a deformation graph, and simultaneously optimized the correspondences between source and target scans, confidence weights for measuring the reliability of correspondence, and a warping field that aligns the source with the target. Later, Li et al. [20] combined point-to-point and point-to-plane metrics for the more accurate measure of correspondence.

Other methods tackle the problem from a statistical perspective. Considering the fitting of two point clouds as a probability density estimation problem, Myronenko et al. [2] proposed the Coherent Point Drift algorithm which encourages displacement vectors to point into similar directions to improve the coherence of the transformation. Hontani et al. [15] incorporated a statistical shape model and a noise model into the non-rigid ICP framework, and detected outliers based on their sparsity. Jian et al. [16] represented each point set as a mixture of Gaussians treated point set registration as a problem of aligning two mixtures. Also with a statistical framework, Wand et al. [34] used a meshless deformation model to perform the pairwise alignment. Ma et al. [24] proposed an $L_2E$ estimator to build more reliable sparse and dense correspondences.

Many non-rigid registration algorithms are based on $\ell_2$-norm metrics for the deviation between source and target models and the smoothness of transformation fields, which can lead to erroneous results when the ground-truth alignment induces large deviation or non-smooth transformation due to noises, partial overlaps, or articulate motions. To address this issue, various sparsity-based methods have been proposed. Yang et al. [36] utilized the $\ell_1$-norm to promote regularity of the transformation. Li et al. [22] additionally introduced position sparsity to improve robustness against noises and outliers. For robust motion tracking and surface reconstruction, Guo et al. [12] proposed an $\ell_0$-based motion regularizer to accommodate articulated motions.

Depending on the application, different regularizations for the transformation have been proposed to improve the robustness of the results. In [10], an as-rigid-as-possible energy was introduced to avoid shrinkage and keep local rigidity. Wu et al. [35] introduced an as-conformal-as-possible energy to avoid mesh distortion. Jiang et al. [17] applied a Huber-norm regularization to induce piecewise smooth transformation.

3. Problem Formulation

Let $\mathcal{S} = \{\mathcal{V}, \mathcal{E}\}$ be a source surface consisting of sample points $\mathcal{V} = \{v_1, v_2, \ldots, v_n \in \mathbb{R}^3\}$ connected by a set of edges $\mathcal{E}$. Let $\mathcal{T}$ be a target surface with sample points $\mathcal{U} = \{u_1, u_2, \ldots, u_m \in \mathbb{R}^3\}$. We seek to apply affine transformations to the source points $\mathcal{V}$ to align them with the target surface. In this paper, we adopt the embedded deformation approach proposed in [29] to model the transformations. Specifically, we construct a deformation graph $\mathcal{G}$ with its vertices $\mathcal{V}_D = \{p_1, \ldots, p_p\}$ being a subset of the source points $\mathcal{V}$, and with its edges $\mathcal{E}_G$ connecting vertices that are nearby on the target surface. On each deformation graph vertex $p_j$, we define an affine transformation represented with a transformation matrix $A_j \in \mathbb{R}^{3\times3}$ and a displacement vector $t_j \in \mathbb{R}^3$. Each vertex $p_j$ influences a localized region that contains any source point $v_i$ whose geodesic distance $D(v_i, p_j)$ to $p_j$ on the source surface is smaller than a user-specified radius $R$. If $p_j$ influences a source point $v_i$, the affine transformation $\{A_j, t_j\}$ associated with $p_j$ induces a transformed position $A_j(v_i - p_j) + p_j + t_j$ for $v_i$. Then the final transformed position $\hat{v}_i$ for $v_i$ is a convex combination of all positions induced by the vertices in graph $\mathcal{G}$ [20]:

$$\hat{v}_i = \sum_{p_j \in \mathcal{E}(v_i)} w_{ij} \cdot (A_j(v_i - p_j) + p_j + t_j),$$

(1)

where $\mathcal{E}(v_i) = \{p_j \mid D(v_i, p_j) < R\}$ denotes the set of vertices that influence $v_i$, and $w_{ij}$ is a distance-dependent
normalized weight:

$$w_{ij} = \frac{(1 - D^2(v_i, p_j)/R^2)^3}{\sum_{p_k \in I(v_i)} (1 - D^2(v_i, p_k)/R^2)^3}.$$  

Compared to formulations that attach a different transformation to each source point such as [22], a major benefit of using the deformation graph is that the deformation of the target surface can be defined with a much smaller number of variables, enabling more efficient optimization.

In the following, we first present an optimization formulation to determine the affine transformations associated with the deformation graph, and then explain how to construct the deformation graph. For the ease of presentation, we use $X$ to denote the affine transformation $(A_j, t_j)$ at vertex $p_j$, whereas $\hat{X}$ denotes the set of all transformations.

### 3.1. Optimization Formulation

For robust alignment between the source and the target surfaces, we determine the affine transformations $X$ via the following optimization:

$$\min_X E_{\text{align}}(X) + \alpha E_{\text{reg}}(X) + \beta E_{\text{rot}}(X).$$  

Here the terms $E_{\text{align}}$, $E_{\text{smooth}}$, and $E_{\text{rot}}$ measures the alignment error, the regularization of transformations across the surface, and the deviation between the transformation matrices and rotation matrices, respectively. $\alpha$ and $\beta$ are positive weights that control the tradeoff between these terms. The definition for each term is explained in the following.

**Alignment Term.** For each transformed source point $\hat{v}_i$, we can find the closest target point $u_{p(i)} \in U$. The alignment term should penalize the deviation between $\hat{v}_i$ and $u_{p(i)}$. A simple approach is to define it as the sum of squared distances between all such pairs of points. This is indeed the alignment error metric used in the classical ICP algorithm [3] for rigid registration [25]. On the other hand, such $\ell_2$-norm of pointwise distance can lead to erroneous alignment on real-world data, where the two surfaces might only overlap partially and their point positions might be noisy. This is because partial overlaps and noisy data can induce large distance from some source points to their corresponding target points under the ground-truth alignment, which would be prohibited by the $\ell_2$-norm minimization.

Some previous work, such as the Sparse ICP algorithm from [5], adopts the $\ell_p$-norm ($0 < p < 1$) as the error metric. It is less sensitive to noises and partial overlaps, since $\ell_p$-norm minimization allows for large distances at some points. However, numerical minimization of the $\ell_p$-norm can be much more expensive than the $\ell_2$-norm. For example, the problem is solved in [5] with an iterative algorithm that alternately updates the point correspondence and the transformation which is similar with classical ICP. However, its transformation update has to be done with an inner ADMM solver which is much slower than the closed-form update in classical ICP, and also lack convergence guarantee due to the non-convexity of the problem.

Inspired by the recent work from [13] on robust image filtering and [37] on robust mesh filtering, we adopt the following robust metric for the alignment error:

$$E_{\text{align}}(X) = \sum_{i=1}^n \psi_{\nu_a}(\|\hat{v}_i - u_{p(i)}\|),$$  

where $\psi_{\nu_a}(\cdot)$ is the Welsch’s function [14] (see Fig. 1 left):

$$\psi_{\nu_a}(x) = 1 - \exp\left(-\frac{x^2}{2\nu_a^2}\right),$$  

and $\nu_a > 0$ is a user-specified parameter. $\psi_{\nu_a}$ is monotonically increasing on $[0, +\infty)$, thus $\psi_{\nu_a}(\|\hat{v}_i - u_{p(i)}\|)$ penalizes the deviation between $\hat{v}_i$ and $u_{p(i)}$. On the other hand, as $\psi_{\nu_a} \leq 1$, the deviation only induces a bounded influence on the metric $E_{\text{align}}$. Moreover, if $\nu_a \to 0$, then $E_{\text{align}}$ approaches the $\ell_0$-norm of the pointwise distance from the source points to their closest points on the target surface. Therefore, this error metric is insensitive to noisy data and partial overlaps.

**Regularization Term.** Ideally, the transformation induced by two neighboring vertices $p_i, p_j$ of the deformation graph should be consistent on their overlapping influenced regions. In [29], such consistency is measured at $p_i$ using the difference of deformation induced by the transformation $X$ at $p_i$ and the transformation $X_j$ at $p_j$:

$$D_{ij} = A_j(p_i - p_j) + p_j + t_j - (p_i + t_i).$$  

Ideally, $D_{ij}$ should be small across the deformation graph. On the other hand, in some cases the optimal deformation may induce large values of $D_{ij}$ in some regions, such as the joints of a human body. To reduce the magnitudes of $D_{ij}$ across the graph while allowing for large magnitudes in some regions, we define the regularization term using the Welsch’s function on $\|D_{ij}\|$:

$$E_{\text{reg}}(X) = \sum_{i=1}^n \sum_{p_j \in N(p_i)} \psi_{\nu_1}(\|D_{ij}\|),$$  

Figure 1. Left: the Welsch’s function with different $\nu$ values. Right: different surrogate functions for the Welsch’s function with $\nu = 1$. 

where \( r \) is the number of nodes in \( G \), \( \nu_k > 0 \) is a user-specified parameter, and \( \mathcal{N}(p_i) \) denotes the set of neighboring vertices for \( p_i \) in \( G \).

**Rotation Matrix Term.** To preserve rigidity on local surface regions during registration, we would like each transformation \( \mathbf{X}_i \) to be as close to a rigid transformation as possible. We measure this property using the deviation between the transformation matrix \( \mathbf{A}_i \) and its closest projection onto the rotation matrix group \( \mathcal{R} = \{ \mathbf{R} \in \mathbb{R}^{3 \times 3} \mid \mathbf{R}^T \mathbf{R} = \mathbf{I}, \det(\mathbf{R}) > 0 \} \), and define \( E_{\text{rot}} \) as

\[
E_{\text{rot}}(\mathbf{X}) = \sum_{i=1}^{r} \| \mathbf{A}_i - \text{proj}_{\mathcal{R}}(\mathbf{A}_i) \|^2_F, \tag{6}
\]

where \( \text{proj} \) is the projection operator, and \( \| \cdot \|_F \) is the Frobenius norm.

**3.2. Construction of Deformation Graph**

To construct the deformation graph \( G \), we first extract its vertices \( V_G \) by iteratively adding source points as follows. \( V_G \) is initialized as empty. Then we perform PCA on all source points, and sort them based on their projections onto the axis with the largest eigenvalue of the covariance matrix. According to the sorted order, we go through each source point \( v_i \), and add it to \( V_G \) if \( V_G \) is empty or the shortest geodesic distance from \( v_i \) to the points in \( V_G \) is no smaller than the radius parameter \( R \). After determining the vertex set \( V_G \), we construct the edge set \( E_G \) by connecting any two vertices whose geodesic distance is smaller than \( R \). We compute the geodesic distance using the fast marching method [18]. \( R \) is set to \( \bar{R} \) by default, where \( \bar{R} \) is the average edge length on the source surface. Compared with alternative sampling strategies such as the farthest point sampling method [26], our approach results in fewer sample points for the same radius parameter, which reduces the computational cost while achieving similar accuracy. A comparison example is provided in the supplementary material.

**4. Numerical Algorithm**

The target function \( E \) for the optimization problem (2) is non-linear and non-convex. Thanks to the use of the Welsch’s function, it can be solved efficiently using the MM algorithm [19]. Specifically, given the variable values \( \mathbf{X}^{(k)} \) in the current iteration, the MM algorithm constructs a surrogate function \( \mathcal{E}^{X^{(k)}} \) for the target function \( E \) such that

\[
\mathcal{E}^{X^{(k)}}(\mathbf{X}^{(k)}) = E(\mathbf{X}^{(k)}),
\]

\[
\mathcal{E}^{X^{(k)}}(\mathbf{X}) \geq E(\mathbf{X}), \quad \forall \mathbf{X} \neq \mathbf{X}^{(k)}. \tag{7}
\]

Then the variables are updated by minimizing the surrogate function

\[
\mathbf{X}^{(k+1)} = \arg \min_{\mathbf{X}} \mathcal{E}^{X^{(k)}}(\mathbf{X}). \tag{8}
\]

In this way, each iteration is guaranteed to decrease the target function, and the iterations are guaranteed to converge to a local minimum regardless of the initialization [19]. In comparison, existing solvers for minimizing the non-convex \( \ell_p \)-norm such as ADMM [5] or iteratively reweighted least squares [9] either lack convergence guarantee or rely on strong assumptions for convergence. In the following, we explain the construction of the surrogate function, and present a numerical algorithm for its minimization in each iteration.

**4.1. Surrogate Function**

To construct the surrogate function \( \mathcal{E}^{X^{(k)}} \), we note that there is a convex quadratic surrogate function for the Welsch’s function \( \psi_\nu \) at \( y \) [13] (see Fig. 1 right):

\[
\overline{\psi}_\nu(x) = \psi_\nu(y) + \frac{1 - \psi_\nu(y)}{2\nu^2}(x^2 - y^2).
\]

This function bounds the Welsch’s function from above, and the two function graphs touch at \( x = y \). Applying this surrogate to all relevant terms, we obtain a surrogate function for \( E_{\text{reg}} \) at \( \mathbf{X}^{(k)} \). Moreover, we can ignore all constant terms as they do not affect the minimum, resulting in the following convex quadratic function

\[
\mathcal{E}^{X^{(k)}}_{\text{reg}} = \frac{1}{2\nu^2} \sum_{i=1}^{r} \gamma(\mathbf{X}) \sum_{p_j \in \mathcal{N}(p_i)} \exp\left(-\frac{\| \mathbf{D}_{ij}^{(k)} \|^2}{2\nu^2}\right)\| \mathbf{D}_{ij} \|^2, \tag{9}
\]

where \( \mathbf{D}_{ij}^{(k)} \) is evaluated using Eq. (4) at \( \mathbf{X}^{(k)} \). Similarly, we can obtain the following function as a surrogate for \( E_{\text{align}} \) at \( \mathbf{X}^{(k)} \) up to a constant:

\[
\frac{1}{2\nu^2_a} \sum_{i=1}^{n} \exp\left(-\frac{\| \mathbf{v}_i^{(k)} - \mathbf{u}_{p(i)}^{(k)} \|^2}{2\nu^2_a}\right)\| \mathbf{v}_i - \mathbf{u}_{p(i)} \|^2, \tag{10}
\]

where \( \mathbf{v}_i^{(k)} \) is the transformed position of \( v_i \) according to \( \mathbf{X}^{(k)} \), and \( \mathbf{u}_{p(i)}^{(k)} \) is the closest target point for \( v_i^{(k)} \). Eq. (10) is not a quadratic function of \( \mathbf{X} \), as \( \mathbf{u}_{p(i)}^{(k)} \) depends non-linearly on \( \mathbf{X} \). To obtain a more simple form, we note that the term \( \| \mathbf{v}_i - \mathbf{u}_{p(i)} \|^2 \) has a quadratic surrogate function \( \| \mathbf{v}_i - \mathbf{u}_{p(i)}^{(k)} \|^2 \) at \( \mathbf{X}^{(k)} \). Applying it to Eq. (10), we obtain the following convex quadratic surrogate function for \( E_{\text{align}} \) at \( \mathbf{X}^{(k)} \) up to a constant:

\[
\mathcal{E}^{X^{(k)}}_{\text{align}} = \frac{1}{2\nu^2_a} \sum_{i=1}^{n} \exp\left(-\frac{\| \mathbf{v}_i^{(k)} - \mathbf{u}_{p(i)}^{(k)} \|^2}{2\nu^2_a}\right)\| \mathbf{v}_i - \mathbf{u}_{p(i)}^{(k)} \|^2. \tag{11}
\]
Replacing $E_{\text{align}}$ and $E_{\text{reg}}$ in the target function with their surrogates, we arrive at the following MM iteration scheme:

$$X^{(k+1)} = \min_X E^{X^{(k)}}(X),$$

where $E^{X^{(k)}} = E_{\text{align}}^{(k)} + \alpha E_{\text{reg}}^{(k)} + \beta E_{\text{rot}}^{(k)}$.

### 4.2. Numerical Minimization

The target function $E^{X^{(k)}}$ in Eq. (12) still contains a non-linear term $E_{\text{rot}}$, as the projection onto the rotation matrix group depends non-linearly on $A_i$. On the other hand, as explained later, the special structure of $E_{\text{rot}}$ leads to a simple form of its gradient, allowing us to evaluate the gradient of $E^{X^{(k)}}$ efficiently. Therefore, we solve the sub-problem (12) using an L-BFGS solver for fast convergence. In each iteration, L-BFGS utilizes the gradients of $E^{X^{(k)}}$ at the latest $m + 1$ iterates $X^{(j)}, X^{(j-1)}, \ldots, X^{(j-m)}$ to implicitly approximate its inverse Hessian and derive a descent direction $d_{(j)}$, followed by a line search along $d_{(j)}$ for a new iterate $X^{(j+1)}$ with sufficient decrease of the target function [27]. In the following, we present the details of the solver.

**Gradient Computation.** For the function $E_{\text{rot}}$ defined in Eq. (6), each term $\|A_i - \text{proj}_R(A_i)\|^2$ is the squared Euclidean distance from the matrix $A_i$ to the manifold $R$ of rotation matrices. Even though $\text{proj}_R(A_i)$ depends non-linearly on $A_i$, it can be shown that the squared distance has a simple form of gradient [11]:

$$\frac{\partial \|A_i - \text{proj}_R(A_i)\|^2}{\partial A_i} = 2(A_i - \text{proj}_R(A_i)).$$

Thus, we can write the gradient of $E_{\text{rot}}$ as:

$$\nabla E_{\text{rot}} = 2(A_i - \text{proj}_R(A_i)), \quad \nabla E_{\text{rot}} = 0.$$  

Since $E_{\text{align}}^{X^{(k)}}$ and $E_{\text{reg}}^{X^{(k)}}$ are quadratic functions, their gradients have simple linear forms. To facilitate presentation, we first rewrite the functions in matrix forms. In the following, we assume all 3D vectors are $3 \times 1$ matrices. The transformation at $p_i$ is represented as $X_i = [A_i, t_i]^T \in \mathbb{R}^{4 \times 3}$, and $X = [X_1^T, \ldots, X_n^T]^T \in \mathbb{R}^{4n \times 3}$. Then we have:

$$E_{\text{align}}^{X^{(k)}} = \|W_a(FX + P - U)\|^2_F,$$

where $W_a = \text{diag}(\sqrt{u_1}, \ldots, \sqrt{u_n}) \in \mathbb{R}^{n \times n}$ with $u_i = \frac{1}{2\sigma^2} \exp\left(-\frac{\|v_i - u_i\|^2}{2\sigma^2}\right)$. $F$ is a block matrix

$$F_{ij} = \begin{cases} w_{ij} \cdot [v_i^T - p_j^T, 1] & \text{if } p_j \in \mathcal{F}(v_i) \\ 0 & \text{otherwise} \end{cases},$$

and $P = \left[\sum_{p_j \in \mathcal{F}(v_1)} P_j, \ldots, \sum_{p_j \in \mathcal{F}(v_n)} P_j\right]^T \in \mathbb{R}^{n \times 3}$.

### Algorithm 1: Two-loop recursion for computing descent direction $d_{(j)}$

```python
Q = -G(X_{(j)});
for i = j - 1, \ldots, j - m do
    S_i = X_{i+1} - X_i; \quad T_i = G(X_{i+1}) - G(X_i);
    \rho_i = \text{Tr}(T_i^T S_i);
    \xi_i = \text{Tr}(S_i^T Q) / \rho_i;
    Q = Q - \xi_i T_i;
end
R = H_0^{-1} Q;
for i = j - m, \ldots, j - 1 do
    \eta = \text{Tr}(T_i^T R) / \rho_i;
    R = R + S_i(\xi_i - \eta);
end
d_{(j)} = R;
```

and $P = \left[\sum_{p_j \in \mathcal{F}(v_1)} P_j, \ldots, \sum_{p_j \in \mathcal{F}(v_n)} P_j\right]^T \in \mathbb{R}^{n \times 3}$. Similarly, we have

$$E_{\text{reg}}^{X^{(k)}} = \|W_r(BX - Y)\|^2_F,$$

where the matrices $B \in \mathbb{R}^{2|\mathcal{F}| \times 4}$ and $Y \in \mathbb{R}^{2|\mathcal{F}| \times 3}$ encode the computation of a term $D_{ij}$ in each row, and the diagonal matrix $W_r$ stores the weight $\sqrt{\frac{1}{2\sigma^2} \exp\left(-\frac{\|D_{ij}\|^2}{2\sigma^2}\right)}$ at the corresponding row. In the row corresponding to $D_{ij}$, the elements in $Y$ are $[p_i^T - p_j^T]$, whereas the elements in $B$ for $X_i$ and $X_j$ are $[p_i^T - p_i^T, 1]$ and $[0, 0, 0, 1]$, respectively. We can further write the gradient of $E_{\text{rot}}$ in matrix form as

$$\frac{\partial E_{\text{rot}}}{\partial X} = 2(JX - Z),$$

where $Z = [\text{proj}_R(A_1), 0, \ldots, \text{proj}_R(A_n), 0]^T \in \mathbb{R}^{4n \times 3}$, and $J = \text{diag}(1, 1, 1, 0, 1, 1, 1, 0, \ldots, 1, 1, 1, 0) \in \mathbb{R}^{4n \times 4}.

We can then derive the gradient function of $E^{X^{(k)}}$ as

$$G(X) = 2F^T W_a^2 (FX + P - U) + \alpha B^T W_r^2 (BX - Y) + \beta (JX - Z).$$

**Computing $d_{(j)}$ and $X^{(j+1)}$.** We adopt the L-BFGS implementation from [23] to utilize the special structure of $E^{X^{(k)}}$. Specifically, given an initial approximation $H_0$ for the Hessian of $E^{X^{(k)}}$ at $X_{(j)}$, we compute the descent direction $d_{(j)}$ using a two-loop recursion explained in [23] (see Algorithm 1). Following [23], we derive $H_0$ by assuming a fixed projection $\text{proj}_R(A_i)$ for the term $E_{\text{rot}}$, resulting in the following approximation:

$$H_0 = 2(F^T W_a^2 F + \alpha B^T W_r^2 B + \beta J).$$
The new iterate $X_{(j+1)}$ is then computed as

$$X_{(j+1)} = X_{(j)} + \lambda d_{(j)},$$

using a line search to determine the step size $\lambda > 0$ that achieve sufficient decrease of $E^{X^{(k)}}$:

$$E^{X^{(k)}}(X_{(j+1)}) \leq E^{X^{(k)}}(X_{(j)}) + \gamma \lambda \operatorname{Tr}(\langle (G(X_{(j)}))^T d_{(j)} \rangle),$$

with $\gamma \in (0, 1)$. The iterative L-BFGS solver is terminated if $E^{X^{(k)}}(X_{(j)}) - E^{X^{(k)}}(X_{(j+1)}) < \epsilon_1$ where $\epsilon_1$ is a user-specified threshold. And the outer MM iteration is terminated if $\max_i \|\tilde{\nu}_i^{(k+1)} - \tilde{\nu}_i^{(k)}\| < \epsilon_2$ with a user-specified threshold $\epsilon_2$ or the number of outer iterations reaches a threshold $I_{\max}$. In all experiments, we set $m = 5$, $\gamma = 0.3$, $\epsilon_1 = 10^{-3}$, $\epsilon_2 = 10^{-3}$, and $I_{\max} = 100$.

The matrix $H_0$ is sparse symmetric positive definite and remains unchanged in each iteration of an L-BFGS solver. To solve the linear equation $R = H_0^{-1} Q$ efficiently in the two-loop recursion, we compute a sparse Cholesky factorization for $H_0$ at the beginning of an L-BFGS run, and reuse it in each iteration to solve the linear system with different right-hand-sides. In addition, the columns of the right-hand-side matrix $Q$ are independent, thus we solve them in parallel. Moreover, although the matrix $H_0$ may change between different L-BFGS runs, its sparsity pattern remains the same. Therefore, we pre-compute a symbolic factorization of $H_0$ and only perform numerical factorization subsequently.

Choosing $\nu_a$ and $\nu_r$. To achieve robust registration, the values of $\nu_a$ and $\nu_r$ play an important role. Each of them acts as the standard deviation parameter for the Gaussian weight in the surrogate function (9) or (11). The Gaussian weight becomes effectively zero for error terms whose current magnitude is much larger than the standard deviation. Both $\nu_a$ and $\nu_r$ need to be sufficiently small at the final stage of the solver, to exclude the influence of large error terms that arise from partial overlaps etc. On the other hand, at the initial stage of the solver, their values need to be larger in order to accommodate more error terms and achieve coarse alignment. Therefore, we initialize the variables $X$ with a rigid transformation (see Sec. 5 for details), and solve the problem (2) with relatively large values $\nu_a = \nu_a^{\text{max}}$ and $\nu_r = \nu_r^{\text{max}}$. The solution is then used as initial variable values to re-solve the problem (2) with the values of $\nu_a$ and $\nu_r$ decreased by half. We repeat this process until the value of $\nu_a$ reaches a lower bound $\nu_a^{\text{min}}$ and take the final solution as the registration result. Algorithm 2 summarizes our overall registration algorithm with gradually decreased values of $\nu_a$ and $\nu_r$. By default, we set $\nu_a^{\text{max}} = 40\bar{l}$, $\nu_r^{\text{max}} = 100\bar{l}$ and $\nu_a^{\text{min}} = 0.5\bar{l}$, where $\bar{l}$ is the medium distance from source points to their corresponding target points under the initial rigid transformation, and $\bar{l}$ is the average edge length on the source surface. In the supplementary material, we compare our strategy of gradually decreasing $\nu_a$ and $\nu_r$ with registration using fixed $\nu_a$ and $\nu_r$, which shows that our strategy achieves a more accurate result.

Setting of $\alpha$ and $\beta$. To account for the number of terms in each component of the target function in Eq. (2), we set the weights as $\alpha = k_\alpha(|V|/|E_a|)$ and $\beta = k_\beta(|V|/|E_r|)$, with $k_\alpha = k_\beta = 1$ by default. We decrease the values of $k_\alpha$ and $k_\beta$ on problem instances with reliable initialization to achieve better alignment, and increase their values on problem instances with less reliable initialization to avoid converging to an undesirable local minimum. Moreover, the value of $k_\alpha$ can be increased for smoother deformation of the source surface, or decreased for faster convergence.

5. Results

In this section, we evaluate the effectiveness of our approach, and compare its performance with state-of-the-art methods including N-ICP [11], RPTS [22], and SVR-\textit{\ell}_0 [12]. The evaluations are performed on the MPI Faust dataset [4] and the human motion datasets from [33]. We only show representative results in this section, with more comparisons provided in the supplementary material.
Table 1. Average computation time and RMSE (in millimeters) on the MPI Faust dataset. We set $\alpha = 50$ for N-ICP, $\alpha = 100$, $\beta = 0.1$ for RPTS, and $k_\alpha = k_\beta = 0.001$ for our method.

<table>
<thead>
<tr>
<th>Pose Pair</th>
<th>N-ICP</th>
<th>RPTS</th>
<th>Ours</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time (s) RMSE</td>
<td>Time (s) RMSE</td>
<td>Time (s) RMSE</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>7.43</td>
<td>59.7</td>
<td>61.83</td>
</tr>
<tr>
<td>2</td>
<td>4.46</td>
<td>20.8</td>
<td>46.52</td>
</tr>
<tr>
<td>3</td>
<td>7.95</td>
<td>80.1</td>
<td>35.25</td>
</tr>
<tr>
<td>4</td>
<td>5.45</td>
<td>45.4</td>
<td>22.99</td>
</tr>
<tr>
<td>5</td>
<td>4.13</td>
<td>14.2</td>
<td>14.04</td>
</tr>
<tr>
<td>6</td>
<td>8.61</td>
<td>59.6</td>
<td>28.72</td>
</tr>
<tr>
<td>7</td>
<td>12.57</td>
<td>208</td>
<td>43.26</td>
</tr>
<tr>
<td>8</td>
<td>6.05</td>
<td>70.4</td>
<td>26.27</td>
</tr>
<tr>
<td>Mean</td>
<td>7.08</td>
<td>69.7</td>
<td>34.86</td>
</tr>
<tr>
<td>Median</td>
<td>6.74</td>
<td>59.6</td>
<td>31.99</td>
</tr>
</tbody>
</table>

5.1. Effectiveness of the Welsch’s Function

We perform the optimization (2) with the Welsch’s functions in $E_{\text{align}}$ and $E_{\text{reg}}$ replaced by square functions. Fig. 2 compares the registration accuracy of such $\ell_2$-norm formulation with our approach, on two problem instances from the “crane” dataset of [33]. We can see that the Welsch’s function leads to more accurate results than the $\ell_2$-norm.

5.2. Comparison with Other Methods

In Tab. 1 and Fig. 3, we compare our method with N-ICP and RPTS on the MPI Faust dataset. We select 10 subjects with 16GB of RAM and a 6-core CPU at 3.60GHz. Each

Implementation Details

We implement our method in C++, and all results and comparisons are tested on a PC with 16GB of RAM and a 6-core CPU at 3.60GHz. Each pair of surfaces are pre-processed by aligning their centroids and scaling them to achieve unit-length diagonal for their combined bounding box. We initialize the transformation variables using a rigid transformation computed via 15 iterations of the ICP algorithm to align the source and target points, with rejection of point pairs whose distance is larger than a threshold $\epsilon_d$ or whose normal vectors deviate by more than an angle $\theta$ [28]. We set $\epsilon_d = 0.3$ and $\theta = 60^\circ$ by default. The ICP iterations are initialized by aligning corresponding feature points, determined either using the closest point pairs between the pre-processed surfaces coupled with the above rejection criteria, or using the SHOT feature [32] with diffusion pruning [30], or through manual labeling. In the figures, we visualize the initial correspondence using blue lines connecting the corresponding points. We evaluate the registration accuracy via the root mean square error compared with the ground truth:

$$\text{RMSE} = \sqrt{\frac{1}{|V|} \sum_{i \in V} e_i^2},$$  \hspace{1cm} (18)$$

where $e_i = \|v_i^* - v_i^p\|$ is the deviation between the transformed positions $v_i^*$, $v_i^p$ of a source point $v_i$ under the computed and the ground-truth transformations respectively. In the figures, we show the RMSE for each registration result, and use color-coding to visualize the error values $\{e_i\}$ across the surface. Unless stated otherwise, all RMSE values and color-coded registration errors are in the unit of meters. We use $\#v$ to denote the number of sample points on a surface, and $\#n$ for the number of deformation graph vertices. Similar to our formulation (2), the optimization target functions of N-ICP, RPTS, and SVR-$\ell_2$ all involve a regularization term and/or a term for the rigidity of transformations. To ease presentation, for all methods we use $\alpha$ and $\beta$ to denote the weights for the regularization term and the rigidity term, respectively. We tune the weights for each method to obtain their best results for comparison. In addition, N-ICP and RPTS require dropping unreliable correspondence between source and target points in each iteration, and we adopt the distance and normal deviation thresholds $\epsilon_d$ and $\theta$ as used in the ICP iterations for initialization.
We can see that our method requires significantly less computational time while achieving similar or better accuracy. A major reason for the efficiency of our method is the adoption of a deformation graph, which only requires optimizing one affine transformation per graph vertex. In comparison, N-ICP and RPTS require optimizing one affine transformation per mesh vertex, which significantly increases the number of variables as well as computational time.

Fig. 4 compares our method with N-ICP and RPTS on models from the “jumping” dataset of [33], with added Gaussian noise on vertex positions of the target surface along their normal directions. In the first two rows of Fig. 4, we add noise with standard deviation $\sigma = \bar{l}$ to 5% or 50% of the vertices on the target surface respectively, where $\bar{l}$ is the average edge length of the target surface. In the last two rows, we add noise to all vertices of the target surface with stand deviation $\sigma = 0.3\bar{l}$ and $\sigma = 0.7\bar{l}$, respectively. The comparison shows that our method is more robust to noisy data.

In Fig. 5, we compare our method with N-ICP, RPTS and SVR-$\ell_0$ on partially overlapping data, which is synthesized from the “bouncing” dataset of [33] by removing some vertices from the target surface. For SVR-$\ell_0$, we use the squared distance from source points to their closest target points as the data fitting term. We can see that our method is significantly faster than other methods, and is more robust to such partial overlapping model.

6. Conclusion

In this paper, we proposed a robust non-rigid registration model based on the Welsch’s function. Applying the Welsch’s function to the alignment term and the regularization term makes the formulation robust to the noise and partial overlap. To efficiently solve this problem, we apply majorization-minimization to transform the nonlinear and non-convex problem into a sequence of simple sub-problems that are efficiently solved with L-BFGS. Extensive experiments demonstrate the effectiveness of our method and its efficiency compared to existing approaches.

Acknowledgement We thank the authors of [12] for providing their implementation. This work was supported by the National Natural Science Foundation of China (No. 61672481), and Youth Innovation Promotion Association CAS (No. 2018495).
References


[12] Kaiven Guo, Feng Xu, Yangang Wang, Yebin Liu, and Qionghai Dai. Robust non-rigid motion tracking and surface reconstruction using l0 regularization. In IEEE International Conference on Computer Vision (ICCV), pages 3083–3091, 2015. 1, 2, 6, 8


