Fast and Robust Non-Rigid Registration Using Accelerated Majorization-Minimization

Yuxin Yao, Bailin Deng, Member, IEEE, Weiwei Xu, Member, IEEE, Juyong Zhang†, Member, IEEE

Abstract—Non-rigid 3D registration, which deforms a source 3D shape in a non-rigid way to align with a target 3D shape, is a classical problem in computer vision. Such problems can be challenging because of imperfect data (noise, outliers and partial overlap) and high degrees of freedom. Existing methods typically adopt the $\ell_p$ type robust norm to measure the alignment error and regularize the smoothness of deformation, and use a proximal algorithm to solve the resulting non-smooth optimization problem. However, the slow convergence of such algorithms limits their wide applications. In this paper, we propose a formulation for robust non-rigid registration based on a globally smooth robust norm for alignment and regularization, which can effectively handle outliers and partial overlaps. The problem is solved using the majorization-minimization algorithm, which reduces each iteration to a convex quadratic problem with a closed-form solution. We further apply Anderson acceleration to speed up the convergence of the solver, enabling the solver to run efficiently on devices with limited compute capability. Extensive experiments demonstrate the effectiveness of our method for non-rigid alignment between two shapes with outliers and partial overlaps, 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/yaoyx689/AMM_NRR.

Index Terms—Non-rigid registration, Robust Estimator, Welsch’s function, Anderson acceleration.

1 INTRODUCTION

With the popularity of depth acquisition devices such as Kinect, PrimeSense and 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 reconstructing 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 a source surface and a target surface, each represented as a point cloud or a mesh, non-rigid registration aims to find a deformation field for the source surface to align it with the target surface. This problem is typically solved via optimization. The objective energy often includes alignment terms that measure the deviation between the target surface and deformed source surface, as well as regularization terms that enforce the smoothness of the deformation field. Many existing methods formulate these terms using the $\ell_2$-norm, which penalizes alignment and smoothness errors across the whole surface [1], [2], [3]. On the other hand, the ground-truth alignment may induce large errors for these terms in some local regions due to noise, outliers, partial overlaps, or articulated motions. The $\ell_2$ formulations can inhibit such large localized errors and lead to erroneous alignment. To improve the alignment accuracy, recent works have utilized sparsity-promoting norms for these terms, such as the $\ell_1$-norm [4], [5], [6] and the $\ell_0$-norm [7]. The sparsity optimization enforces small error values on most parts of the surface while allowing for large errors in some local regions, improving the robustness of the registration process. However, the resulting optimization problem can be non-smooth and more challenging to solve. Existing methods often use proximal algorithms such as the alternating direction method of multipliers (ADMM), which can suffer from slow convergence to high-accuracy solutions [8].

In this paper, we propose a new approach to robust non-rigid registration with fast convergence. The key idea is to enforce sparsity using Welsch’s function [9], which has been utilized for robust filtering of images [10] and meshes [11], as well as robust rigid registration [12]. We formulate an optimization that applies Welsch’s function to both the alignment term and the regularization term. Unlike the $\ell_p$-norms, Welsch’s function is globally smooth and avoids non-smooth optimization. We solve the optimization problem using the majorization-minimization (MM) algorithm [13]. It iteratively constructs a surrogate function for the objective 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. This leads to a convex quadratic problem in each iteration, which can be solved efficiently in closed form. To speed up the convergence of the MM algorithm, we regard it as a fixed-point iteration and apply Anderson acceleration [14], [15], a well-established technique to accelerate the convergence of fixed-point iterations. Experimental results verify the robustness of our method as well as its superior performance.
compared to existing robust registration approaches.

In summary, our main contributions include:

- We formulate an optimization problem for non-rigid registration, using 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, which only involves minimizing a convex quadratic surrogate function in each iteration; we also apply Anderson acceleration to speed up its convergence. The simple form of the surrogate function, together with the effectiveness of Anderson acceleration, helps to improve the computational efficiency of robust non-rigid registration compared to existing approaches.
- A preliminary version of this work appeared in [16]. The main additions in the current version include: (1) a more robust method for the deformation graph construction; (2) a more efficient optimization solver that combines the MM algorithm with Anderson acceleration; (3) a more extensive evaluation of the performance.

2 RELATED WORK

Non-rigid registration has been extensively studied in computer vision, medical image processing and computer graphics. The reader can refer to the surveys in [17] and [18] for a more complete overview of past research. In the following, we focus on works that are closely related to our method.

Non-rigid Registration Models. Various optimization-based methods perform non-rigid registration by minimizing the alignment error, often in conjunction with other constraints. Extending the classical iterative closest point (ICP) algorithm for rigid registration, Amberg et al. [1] proposed a non-rigid ICP algorithm that incrementally deforms the source model to align with the target model. Li et al. [2] adopted an embedded deformation approach [19] to express a non-rigid deformation using a deformation graph, and simultaneously optimized the correspondence between source and target scans, the confidence weights for the correspondence, and a warping field that aligns the source with the target. In addition, various geometric constraint terms have been introduced to preserve the local shape of the source model during deformation. Examples include preservation of Laplacian operators [20], [21], [22], as-conformal-as-possible deformations [23], and as-rigid-as-possible deformations [24].

Other methods tackle the registration problem from a statistical perspective. By treating point cloud alignment as a probability density estimation problem, Myronenko et al. [25] proposed the Coherent Point Drift (CPD) algorithm, which encourages displacement vectors to point in similar directions to improve the coherence of the transformation. Hirose [26] reformulated CPD in a Bayesian setting to improve robustness with a better guarantee of convergence. Hontani et al. [27] 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. [28], [29] represented each point set as a mixture of Gaussians and treated point set registration as a problem of aligning two mixtures. Also with a statistical framework, Wand et al. [30] used a meshless deformation model to perform the pairwise alignment. Ma et al. [31] proposed an $L_2E$ estimator to build more reliable sparse and dense correspondences. These methods typically require calculations for each possible pair of points from the source and the target, which is time-consuming for problems with a large number of points.

Some dynamic reconstruction and tracking frameworks involve non-rigid registration between a template model and the live frames. The registration is performed by alternately updating the correspondences and solving the transformation on the GPU to produce real-time results [32], [33], [34], [35], [36]. For such applications, the non-rigid registration tasks are usually less challenging due to the small deviation between the source and target models, and real-time performance is favored over high-precision solutions.

When the transformation between the source and target models is nearly isometric, some approaches utilize the geodesic distance of the surface as an invariant metric to directly optimize the global correspondences [37], [38], [39]. However, these methods may not be suitable for noisy mesh surfaces, as the noise can distort the geodesic distance and lead to erroneous alignment.

With the growing popularity of deep learning in computer graphics and vision tasks, it has also been adopted for non-rigid registration problems. Shimada et al. [40] represented the source and target models using regular voxel grids, and learned the displacement field using datasets of deformable objects with known correspondence. For non-rigid 3D reconstruction, Božič et al. [41] extracted features from the input RGB-D frames and matched them to obtain sparse correspondences, and used these initial correspondences to improve the accuracy of registration. For non-rigid tracking, Li et al. [42] proposed an alignment term based on deep features learned through CNN to improve the robustness of the optimization formulation, and introduced a learning-based pre-conditioner to speed up the convergence of the PCG method used in the optimization solver. Božič et al. [43] combined a dense correspondences prediction module based on an optical flow network of RGB images, and a differentiable alignment module to achieve more accurate real-time non-rigid tracking. These methods improve the robustness and/or speed compared with optimization-based methods, by training on carefully constructed datasets. Feng et al. [44] represented the non-rigid transformation as a point-wise combination of several rigid transformations. They proposed an unsupervised framework to learn the alignment, using a differentiable loss that measures similarity between 3D shapes via their depth images from multiple views. However, this method cannot handle partial overlaps.

Robust Metrics. Since the data generated from a 3D acquisition process often contains noise, outliers and partial overlaps, various robust metrics have been applied in optimization-based approaches to better handle such cases. Unlike the classical $\ell_2$ error metrics that require the error to be small across the whole shape, such robust metrics promote the sparsity of the error and allow for large errors in some local regions to accommodate data imperfections. Yang et al. [4] introduced an $\ell_1$ term for the smoothness of the
transformation, which is suitable for models with articulated motions. Li et al. [5] additionally utilized an $\ell_1$-norm for the alignment error, to improve robustness against noises and outliers. Guo et al. [7] proposed an $\ell_0$-based motion regularizer for robust motion tracking and surface reconstruction. Zampogianiis et al. [45] applied the Huber-$L_1$ loss to the difference between the transformation parameters at neighboring deformation graph nodes, to promote piecewise smooth deformation fields. Other works used the Geman-McClure function for the non-rigid alignment term [35], [46], [47]. In this paper, we adopt Welsch’s function for robust non-rigid registration, which results in a smooth optimization problem that allows for an efficient numerical solver.

Anderson Acceleration. Anderson acceleration [14] was initially designed for the iterative solution of nonlinear integral equations. In recent years, it has been shown effective in accelerating the convergence of fixed-point iterations [15], [48], [49]. In each iteration, Anderson acceleration utilizes $m$ previous iterates to construct a new iterate that converges faster, which can be considered as a quasi-Newton method for finding a root of the residual function [50]. It has been adopted to accelerate first-order solvers in computer graphics [51], [52], [53], as well as ICP-based methods for rigid registration [12], [54]. In this paper, we apply Anderson acceleration to speed up the convergence of our non-rigid registration solver.

3 Model

Given a source surface and a target surface, non-rigid registration computes a deformation field on the source surface to align it with the target surface. We assume that the source surface is represented with a set of sample points $V = \{v_1, \ldots, v_{|V|} \in \mathbb{R}^3\}$ connected by a set of edges $E$ that define their neighboring relations, and $U = \{u_1, \ldots, u_{|U|} \in \mathbb{R}^3\}$ is a set of sample points on the target surface. In this paper, we focus on the case where the source surface is a triangle mesh, so that $V$ and $E$ are the mesh vertices and edges, respectively. However, the formulation is also applicable when the input surface is a point cloud and $V$ are the points: in this case, the edge set $E$ can be constructed by connecting each point with its nearest neighbors. Following [19], we define the deformation field using an embedded deformation graph $G = (V_G, E_G)$, where $V_G = \{p_1, \ldots, p_{|V_G|} \in \mathbb{R}^3\} \subset V$ is the set of graph nodes that lie on the source surface, and $E_G$ is the set of edges. Each graph node is associated with an affine transformation $X_j = [A_j, t_j]$, where $A_j \in \mathbb{R}^{3 \times 3}$ and $t_j \in \mathbb{R}^3$. In the following, we use $X = [X_1, X_2, \ldots, X_{|V_G|}] \in \mathbb{R}^{4|V_G| \times 3}$ to denote the concatenation of all $X_j$. Using the deformation graph, each sample point $v_i$ on the source surface is influenced by a set $I(v_i)$ of nearby deformation graph nodes:

$$I(v_i) = \{p_j \in V_G \mid D(v_i, p_j) < R\},$$

where $D(\cdot, \cdot)$ denotes the geodesic distance on the source surface, and $R$ is a given radius parameter. The new position of the sample point $v_i$ is computed as

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

where $w_{ij}$ is a distance-dependent normalized weight [3]:

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

From the given source and target surfaces, we first construct a deformation graph, and then optimize the transformations associated with the graph nodes to deform the source surface and align it with the target surface. Compared with existing non-rigid registration methods that define the transformation field on each source vertex such as [5], the use of a deformation graph can effectively reduce the number of variables for the optimization problem and significantly reduce the computational time. The remainder of this section explains each step of our methods in detail.

3.1 Construction of deformation graph

To avoid excessive degrees of freedom for the deformation, we would like to reduce the number of deformation graph nodes while ensuring that for each source point $v_i$, the set of influencing nodes $I(v_i)$ (defined in Eq. (1)) is not empty. To this end, we simultaneously construct the set $V_G$ of deformation graph nodes and the set $I(v_i)$ of influencing nodes for each point as follows. We first initialize $V_G$ and all $I(v_i)$ to empty sets. Then we perform PCA on all source points $\{v_i\}$, and sort them based on their projections onto the axis corresponding to the largest eigenvalue of the covariance matrix. We go through all the points according to this order, and add a source point $v_j$ to the deformation graph nodes $V_G$ if either $V_G$ or $I(v_j)$ is empty. Once a point $v_j$ is added to the deformation graph nodes, we also add $v_j$ to the influencing node set for any point $v_i$ that is within geodesic distance $R$ from $v_j$ (i.e., $D(v_i, v_j) < R$). To locate all such points efficiently, we first start from $v_j$ and perform a breadth-first search on the surface to identify the largest neighborhood $B(v_j)$ of points whose Euclidean distance to $v_j$ is smaller than $R$. Since the Euclidean distance is no larger than the geodesic distance, any point with a geodesic distance smaller than $R$ must belong to $B(v_j)$. Hence we compute the geodesic distance from $v_j$ to each point $q \in B(v_j)$, and add $v_j$ to $I(q)$ if $D(v_j, q) < R$. To compute the geodesic distance, we construct a sub-mesh consisting of all faces on the source surface mesh that are incident with the vertices from $B(v_j)$, and apply the exact geodesic algorithm of [55] (VTP) on this sub-mesh. We use VTP for the computation because it can produce accurate results while being efficient on small meshes. After the set of deformation graph nodes is constructed, we connect two nodes with an edge if there exists a point that is influenced by both of them. Algorithm 1 summarizes our method for constructing the deformation graph. Note that in the preliminary version [16] of this paper, the deformation graph construction method needs to compute the geodesic distance from each graph node to all the source points; in comparison, our new method only requires the geodesic distance to the source points within the sphere centered at the graph node and with a radius $R$, which involves a significantly smaller number of geodesic distance values. This allows us to compute the exact geodesic distance using VTP to achieve more robust results, whereas the method in [16] needs to use the fast marching method (FMM) to compute an approximate geodesic distance.
3.2 Problem formulation

Similar to existing non-rigid registration methods, we would like to align the source and target surfaces using a smooth transformation field that is locally close to rigid deformation. Therefore, we minimize the following target function to determine the transformation \( X \) that is defined over the deformation graph nodes:

\[
E(X) = E_{\text{align}}(X) + \alpha E_{\text{reg}}(X) + \beta E_{\text{rot}}(X).
\]  

Here \( E_{\text{align}} \) measures the alignment error, \( E_{\text{reg}} \) is a smoothness regularization term for the transformation field, and \( E_{\text{rot}} \) penalizes the deviation between the transformation \( X_j \) on each node and a rigid transformation. \( \alpha \) and \( \beta \) are user-specified weights. The definition for each term is explained below.

**Alignment term.** For each transformed source point \( \hat{v}_i \), we find its closest point \( u_{\rho(i)} \) on the target surface as the corresponding point:

\[
u_{\rho(i)} := \arg \min_{u \in \Omega} \| u - \hat{v}_i \|.
\]

Traditional methods such as [1] measure the alignment error using the sum of squared distance from all the transformed source points to their corresponding points. However, outliers and partial overlaps can result in large distances between some point pairs for the ground-truth...
transformation, which can lead to a large value of such $\ell_2$ alignment measures and prevent the optimization from finding the ground-truth alignment. Recently, robust metrics such as the $\ell_p$-norm ($p \leq 1$) [4], [5], [7], [58] have been utilized instead for the alignment error. It is less sensitive to noises and partial overlaps, since such $\ell_p$-norm minimization allows for large distances at some points. However, due to the non-smoothness of such $\ell_p$-norm formulations, their numerical minimization can be much more expensive than the $\ell_2$-norm. For example, the problem is solved in [58] with an iterative algorithm that alternately updates the point correspondence and the transformation; the transformation update step minimizes a target function that involves the correspondence and the transformation; the transformation

\[ E_{\text{align}}(X) = \sum_{v_i \in V} \psi_{\nu_a}(\|\bar{v}_i - u_{\rho_1(i)}\|), \quad (5) \]

where $\nu_a > 0$ is a user-specified parameter, and

\[ \psi_{\nu_a}(x) = 1 - \exp(-\frac{x^2}{2\nu_a^2}). \quad (6) \]

As shown in Fig. 2, the function $\psi_{\nu_a}$ is monotonically increasing on $[0, +\infty)$ while being bounded from above by 1. In addition, when $\nu_a \to 0$, the error measure in Eq. (5) approaches the $\ell_0$-norm of the distance values between the corresponding points. Thus it penalizes the distance between corresponding points, without being overly sensitive to large distances caused by outliers and partial overlaps. Moreover, as shown later in Section 3.3, the use of Welsch’s function enables us to design an efficient optimization solver with guaranteed convergence.

Regularization term. Ideally, the transformation induced by two neighboring deformation graph nodes $p_i$ and $p_j$ should be consistent on their overlapping influenced region. We measure such consistency at $p_i$, using the following term:

\[ D_{ij} = r_{ij}(A_j(p_i - p_j) + p_j + t_j - (p_i + t_i)). \quad (7) \]

Here the term $A_j(p_i - p_j) + p_j + t_j - (p_i + t_i)$ is the difference between transformations induced by $X_j$ and $X_i$ at $p_i$, which is commonly used for the regularization of transformation fields defined on a deformation graph [3], [19]. $r_{ij}$ is a normalization weight that accounts for the distance between $p_i$ and $p_j$:

\[ r_{ij} = \frac{2|E_c| \cdot \|p_i - p_j\|^{-\frac{1}{\nu}}}{\sum_{p_i \in V_G} \sum_{p_j \in N(p_i)} \|p_i - p_j\|^{-1}}, \quad (8) \]

where $N(p_i)$ denotes the set of neighboring deformation graph nodes for the node $p_i$.

For non-rigid registration, the magnitude of $\|D_{ij}\|$ should be small for most parts of the deformation graph. On the other hand, the optimal transformation field may induce large magnitudes of $\|D_{ij}\|$ in some local regions such as the joints of a human body due to articulated motion. Therefore, we apply the Welsch’s function to $\|D_{ij}\|$ to penalize its magnitude across the deformation graph while allowing for large magnitudes in some regions, resulting in the following regularization term:

\[ E_{\text{reg}} = \sum_{p_i \in V_G} \sum_{p_j \in N(p_i)} \psi_{\nu_r}(\|D_{ij}\|), \quad (9) \]

where $\nu_r > 0$ is a user-specified parameter.

Rotation Matrix term. To preserve local surface shapes during the deformation, we would like each transformation $X_i$ to be close to a rigid transformation. We enforce this property by penalizing the difference between the affine transformation matrix $A_i$ and its closest projection onto the rotation matrix group $R = \{ R \in \mathbb{R}^{3 \times 3} | RR^T = I, \det(R) > 0 \}$, and define the term $E_{\text{rot}}$ as

\[ E_{\text{rot}} = \sum_{p_i \in V_G} \|A_i - \text{proj}_R(A_i)\|_F^2, \quad (10) \]

where $\text{proj}_R(\cdot)$ is the projection operator:

\[ \text{proj}_R(A) = \arg \min_{R \in R} \|R - A\|. \quad (11) \]

3.3 Numerical optimization

The target function of the optimization problem (4) is non-linear and non-convex. Thanks to the use of Welsch’s function, it can be solved efficiently using the majorization-minimization (MM) algorithm [13]. Specifically, given the variable values $X^{(k)}$ in the current iteration, the MM algorithm constructs a surrogate function $E(X|X^{(k)})$ for the target function $E$ such that

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

\[ E(X|X^{(k)}) \geq E(X) \quad \forall X \neq X^{(k)}. \quad (11) \]

Then the variables are updated by minimizing the surrogate function

\[ X^{(k+1)} = \arg \min_X E(X|X^{(k)}). \quad (12) \]

This guarantees the decrease of the target function in each iteration, since Equations (11) and (12) imply that

\[ E(X^{(k+1)}) \leq E(X^{(k+1)}|X^{(k)}) \leq E(X|X^{(k)}) = E(X^{(k)}). \]

As a result, the iterations are guaranteed to converge to a local minimum [13]. In comparison, existing solvers for minimizing non-convex $\ell_p$-norms such as ADMM [58] or iteratively reweighted least squares [59] either lack a convergence guarantee or rely on additional strong assumptions to guarantee convergence. In the following, we explain the construction of the surrogate function and its numerical minimization.
3.3.1 Surrogate function

To construct the surrogate function $E(X|X^{(k)})$, we note that there is a convex quadratic surrogate function for the Welsch’s function $\psi$, at $y$ [10] (see Fig. 2 right):

$$\psi_{\nu}(x|y) = \psi_{\nu}(y) + \frac{1 - \psi_{\nu}(y)}{2\nu^2} (x^2 - y^2).$$

(13)

This function bounds the Welsch’s function from above, and the two function graphs touch at $x = y$. Applying this to $E_{\text{reg}}$ in Eq. (9), we obtain a convex quadratic surrogate function

$$E_{\text{reg}}(X|X^{(k)}) = \frac{1}{2\nu^2} \sum_{p_i \in V} \sum_{p_j \in N(p_i)} \exp\left(-\frac{||D^{(k)}_{ij}||^2}{2\nu^2}\right) ||D_{ij}||^2,$$

(14)

where $D^{(k)}_{ij}$ is evaluated using Eq. (7) at $X^{(k)}$. Note that in Eq. (14) we have ignored some constant terms as they do not affect the optimization. Similarly, we can apply Eq. (13) to $E_{\text{align}}$ and ignore some constant terms, to obtain a surrogate function

$$\frac{1}{2\nu^2} \sum_{v_i \in V} \exp\left(-\frac{||\hat{v}_i^{(k)} - u^{(k)}_{\rho(i)}||^2}{2\nu^2}\right) ||\hat{v}_i - u_{\rho(i)}||^2,$$

(15)

where the point $\hat{v}_i^{(k)}$ is calculated with Eq. (2) at $X^{(k)}$, and $u^{(k)}_{\rho(i)}$ is the closest point to $\hat{v}_i^{(k)}$ on the target surface. However, this is not a quadratic function of the new position $v_i$, since the point $u_{\rho(i)}$ depends non-linearly on $v_i$. To obtain a more simple form, we note that the term $||v_i - u_{\rho(i)}||^2$ has a quadratic surrogate function $||\hat{v}_i - u_{\rho(i)}||^2$ at $X^{(k)}$, since by definition the closest point $u_{\rho(i)}$ satisfies $||\hat{v}_i - u_{\rho(i)}||^2 \leq ||\hat{v}_i - u||$ for all $u \in \mathbb{R}^3$. Applying this to Eq. (15), we obtain the following convex quadratic surrogate function for $E_{\text{align}}$ at $X^{(k)}$ up to an additional constant:

$$E_{\text{align}}(X|X^{(k)}) = \frac{1}{2\nu^2} \sum_{v_i \in V} \exp\left(-\frac{||\hat{v}_i^{(k)} - u^{(k)}_{\rho(i)}||^2}{2\nu^2}\right) ||\hat{v}_i - u^{(k)}_{\rho(i)}||^2,$$

(16)

For the rotation term $E_{\text{rot}}$, we can similarly derive a quadratic surrogate function as

$$E_{\text{rot}}(X|X^{(k)}) = \sum_{p_i \in V} ||A_i - \text{proj}_R(A_i^{(k)})||.$$

(17)

In total, the surrogate function of $E(X)$ can be written as

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

(18)

This surrogate function is minimized to obtain the updated variables $X^{(k+1)}$.

3.3.2 Numerical minimization

Since $E(X|X^{(k)})$ is a convex quadratic function, we can minimize it by solving a linear system. To derive the system, we first rewrite $E(X|X^{(k)})$ in a matrix form. The term $E_{\text{align}}(X|X^{(k)})$ can be written as

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

(19)

where $W_a = \text{diag}(\sqrt{w_1^2}, \ldots, \sqrt{w_n^2}) \in \mathbb{R}^{|V| \times |V|}$ with

$$w_i^a = \frac{1}{2\nu^2} \exp\left(-\frac{||\hat{v}_i^{(k)} - u^{(k)}_{\rho(i)}||^2}{2\nu^2}\right),$$

$F$ is a block matrix $\{F_{ij}\}_{1 \leq i \leq |V|, 1 \leq j \leq |V|}$ with

$$F_{ij} = \begin{cases} u_{ij} \cdot [v_i^T - p_{ij}^T, 1] & \text{if } p_{ij} \in I(v_i) \\ 0 & \text{otherwise} \end{cases},$$

and $Q = [Q_1, Q_2, \ldots, Q_{|V|}]^T \in \mathbb{R}^{|V| \times 3}$ with

$$Q_i = u^{(k)}_{\rho(i)} - \sum_{p_j \in I(v_i)} u_{ij}p_j.$$ Similarity, the term $E_{\text{reg}}(X|X^{(k)})$ can be written as

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

(20)

where the matrices $H \in \mathbb{R}^{2|E| \times 4|V|}$ and $Y \in \mathbb{R}^{2|E| \times 3}$ encode the computation of a term $D_{ij}$ in the same way: the row in $H$ has two non-zero blocks $[r_{ij}(p_i^T - p_j^T), r_{ij}]$ and $[0, 0, 0, -r_{ij}]$ corresponding to $X_i$ and $X_j$ respectively, whereas the row in $Y$ has elements $r_{ij}(p_i^T - p_j^T)$. The diagonal matrix $W_r \in \mathbb{R}^{2|E| \times 2|E|}$ has an element

$$\sqrt{\frac{1}{2\nu^2} \exp\left(-\frac{||D^{(k)}_{ij}||^2}{2\nu^2}\right)}$$

in the row corresponding to $D_{ij}$. Finally, the term $E_{\text{rot}}(X|X^{(k)})$ can be written as

$$E_{\text{rot}}(X|X^{(k)}) = ||JX - Z||^2_F,$$

where

$$J = \text{diag}(1, 1, 1, 0, 1, 1, 0, \ldots, 1, 1, 0) \in \mathbb{R}^{4|V| \times 4|V|},$$

$$Z = [\text{proj}_R(A_1), 0, \ldots, \text{proj}_R(A_{|V|}), 0]^T \in \mathbb{R}^{4|V| \times 3}.$$

Using the matrix forms, the updated variable $X^{(k+1)}$ is computed via the linear system

$$K^{(k)}X^{(k+1)} = B^{(k)},$$

(21)

where

$$K^{(k)} = F^TW_a^TW_aF + \alpha H^TW_a^TW_aH + \beta J^TJ,$$

$$B^{(k)} = F^TW_a^TW_aQ + \alpha H^TW_a^TW_aY + \beta J^TJZ.$$ We repeat this step until either the maximum number of iterations $I_{\text{max}}$ is reached, or the deformed source model converges (indicated by the condition $max_i ||\hat{v}_i^{(k+1)} - \hat{v}_i|| < \epsilon$ where $\epsilon$ is a user-specified threshold).

3.3.3 Choice of parameters $\nu_a$ and $\nu_r$

The parameters $\nu_a$ and $\nu_r$ play an important role in the robustness of our method. The surrogate functions in Eqs. (14) and (16) show that our solver actually minimizes a quadratic error function, where the weights for the alignment error terms and the regularization terms are updated dynamically as a Gaussian function of the current error norms with variance $\nu_a^2$ and $\nu_r^2$ respectively. On one hand, we would like $\nu_a$ and $\nu_r$ to be sufficiently small in the final stage of the optimization, such that the Gaussian weights can effectively filter out the influence of alignment and rotation terms with large errors due to outliers, partial overlaps and articulated motions. On the other hand, fixing $\nu_a^2$ and $\nu_r^2$ to small values can reduce the effectiveness of the optimization,
as many terms would have large errors at the beginning of the optimization and be excluded due to their small Gaussian weights. Therefore, we gradually decrease the values of $\nu_a$ and $\nu_r$ during the optimization. Specifically, we initialize $\nu_a$ and $\nu_r$ to values $\nu_a^0$ and $\nu_r^0$ that are large enough to accommodate a large number of terms, and use these values $\nu_a$, $\nu_r$ to perform optimization until convergence. Then we reduce $\nu_a$ and $\nu_r$ by half, and use the previous optimized variables as initial values to continue the optimization until convergence. We repeat this process until the $\nu_a$ reaches a user-specified lower bound $\nu_a$. We choose $\nu_a^0 = \bar{d}$ and $\nu_r^0 = \bar{l}/\sqrt{3}$ following [12], and set $\nu_r = 3\bar{l}$, where $\bar{d}$ is the median value of the initial distance between each source point and its closest point on the target surface, and $\bar{l}$ denotes the average edge length of the source model.

### 3.3.4 Anderson acceleration

The MM solver monotonically decreases the target function and converges quickly to an approximate solution, but it can take a long time to converge to a high-accuracy solution. To speed up the final convergence, we adopt Anderson acceleration [15], a well-established approach for accelerating fixed-point iterations. To do so, we consider our MM solver as a fixed point iteration

\[
X^{(k+1)} = G(X^{(k)})
\]

with a residual function

\[
F(X) = G(X) - X.
\]

If the MM solver converges to a solution $X^*$, then $X^*$ is a fixed point of the iteration (22) and $F(X^*) = 0$. In each iteration, Anderson acceleration uses the current iterate $X^{(k)}$ and $m$ previous iterates $X^{(k-1)}, \ldots, X^{(k-m)}$ to compute an accelerated iterate $X_{AA}$ as an affine combination of $G(X^{(k)}), G(X^{(k-1)}), \ldots, G(X^{(k-m)})$:

\[
X_{AA} = G(X^{(k)}) - \sum_{j=1}^{m} \theta_j^s (G(X^{(k)+1}) - G(X^{(k)+j})),
\]

with the coefficients $\{\theta_j^s\}$ computed by solving a linear least-squares problem:

\[
(\theta_1^s, \ldots, \theta_m^s) = \arg\min_{\theta_1, \ldots, \theta_m} \| F(k) - \sum_{j=1}^{m} \theta_j (F(k+j+1) - F(k+j)) \|^2,
\]

where $F(k) = F(G(k))$. Since the accelerated iterate $X_{AA}$ may suffer from instability and lead to stagnation, we follow the stabilization approach from [51] and only accept it as the new iterate $X^{(k+1)}$ if it decreases the target function; otherwise, we revert to the original MM update $G(X^{(k)})$ which is guaranteed to decrease the target function, i.e.,

\[
X^{(k+1)} = \begin{cases} 
X_{AA} & \text{if } E(X_{AA}) < E(X^{(k)}), \\
G(X^{(k)}) & \text{otherwise}.
\end{cases}
\]

This approach guarantees a monotonic decrease of the target function. Similar to [51], some intermediate results that are computed during the evaluation of $E(X_{AA})$ — in particular the closest points $\{u_{\rho(i)}\}$ and the rotation matrices $\{\text{proj}_R(A_i)\}$ — can be reused for the MM step in the next iteration if $X_{AA}$ is accepted. This helps to alleviate the computational overhead of the acceptance mechanism.

---

Algorithm 2: Robust non-rigid registration using Welsh’s function and Anderson acceleration.

**Input:** $\{V, E\}, U$: source and target models; $X^*$ converging to a local minimum of $E(X)$.

1. Initialize deformation graph $\{V_{\rho}, E_{\rho}\}$ by Alg. 1.
2. Initialize $X^{(0)}$ with identity transformations;
3. $\nu_a = \nu_a^0, \nu_r = \nu_r^0$;
4. $k = 1$;
5. while TRUE do
6. $k_{\text{start}} = k - 1$; $E_{\text{prev}} = +\infty$;
7. repeat
8. Find corresponding point $u_{\rho(i)}^{(k)}$ for each $q_{\rho(i)}^{(k)}$;
9. Update the projection $\text{proj}_R(A_{\rho(i)}^{(k)})$;
10. Calculate objective function $E$ by Eq. (4);
11. if $E > E_{\text{prev}}$ then
12. $X^{(k)} = X^{(k)}$;
13. continue;
14. end
15. $E_{\text{prev}} = E$;
16. Update weight matrices $W_a$ and $W_r$;
17. $X' = \arg\min E(X|X^{(k)})$ by solving Eq. (21);
18. // Anderson acceleration
19. $G^{(k)} = X'$; $F^{(j)} = G^{(k)} - X^{(j)}$;
20. $m_k = \min(k - k_{\text{start}}, m)$;
21. $(\theta_1^s, \ldots, \theta_{m_k}^s) = \arg\min \| F(k) - \sum_{j=1}^{m} \theta_j (F(k)+j+1) - F(k)+j) \|^2$;
22. $X^{(k+1)} = G^{(k)} - \sum_{j=1}^{m} \theta_j (G^{(k)+j+1} - G^{(k)+j})$;
23. $k = k + 1$;
24. until $\max_i \| q_{\rho(i)}^{(k)} - q_{\rho(i)}^{(k)} \| < \epsilon$ or $(k - k_{\text{start}}) = I_{\text{max}}$;
25. if $\nu_a = +\infty$ then return $X^{(k)}$;
26. $\nu_a = \max(\nu_a/2, \nu_r)$; $\nu_r = \nu_r/2$;
27. $k = k + 1$;
28. end

**Remark.** In the preliminary version [16] of the current paper, the same optimization problem is solved using a different approach, which iteratively constructs a non-quadratic surrogate function and minimizes it using an inner L-BFGS solver. One benefit of our new approach is that it involves fewer parameters for the solver. In particular, the inner L-BFGS solver in [16] requires a threshold parameter for the sufficient decrease condition used in its line search, as well as two parameters for its termination criteria. These parameters need to be chosen properly to achieve the best performance of the solver. In comparison, our new solver does not involve a line search or an inner solver, and hence does not require the tuning of such parameters. More numerical comparisons between the two approaches can be found in Section 4.3.

### 4 Results

We conduct extensive experiments to evaluate the effectiveness of our method and compare its performance with...
existing approaches.

### 4.1 Implementation and metrics

We implement our method in C++, using Eigen [60] for linear algebra operations and OpenMP for parallelization. Unless stated otherwise, the experiments are run on a PC with 16GB of RAM and a 6-core CPU at 3.60GHz. Each pair of surfaces are pre-processed by scaling them to achieve a unit-length diagonal for their bounding box. By default we set $\epsilon = 10^{-5}$, $I_{\text{max}} = 100$, and the sampling radius $R = 5\ell$ where $\ell$ is the average edge length of the source model. The source code of our method is available at https://github.com/yaoyx689/AMM_NRR.

To evaluate the registration accuracy of a result, we first compute the error for each source point $v_i$ using the distance from its transformed position $\hat{v}_i$ to its ground-truth corresponding point $u_{\rho^*}(i)$ on the target surface:

$$D_i = \|\hat{v}_i - u_{\rho^*}(i)\|. \quad (24)$$

The registration accuracy is then measured by the root mean square error (RMSE):

$$r = \sqrt{\frac{\sum_{v_i \in V} D_i^2}{|V|}}, \quad (25)$$

where a lower value of $r$ indicates higher accuracy. The error metrics $D_i$ and $r$ are measured in the original scale of the models, and are in the unit of meters unless stated otherwise.

In the following, we use $\#S$ and $\#T$ to denote the number of sample points on the source surface and the target surface, respectively. We render the initial source model, the target

---

Fig. 3. Comparison on partially overlapping data constructed from the “crane” dataset [57]. We set $k_\alpha = 100, k_\beta = 10$ for all variants.
model, and the deformed source model in red, blue, and orange respectively, and use color-coding to visualize the point-wise error \( D_i \) across the surface.

### 4.2 Effectiveness of our method

**Welsch’s function and parameters update.** We first evaluate the effectiveness of our formulation with Welsch’s function and our dynamic update strategies for its parameter. To do so, we consider the following variants of our algorithm:

- We replace the Welsch’s functions in the target function in Eq. (4) with the squared \( \ell_2 \)-norm, such that the target function becomes

\[
E_{\ell_2} = \sum_{v_i \in V} w_i^d \| \hat{v}_i - u_{p(i)} \|^2 + \alpha \sum_{p_j \in V_{\partial}^s} \sum_{p_j \in N(p_i)} \| D_{ij} \|^2 + \beta E_{\text{rem}}
\]

where \( \{ w_i^d \} \) are weights. We consider two variants with different weighting schemes:

- All weights \( w_i^d \) are set to 1. In the following examples, we refer to this variant as \( \ell_2 \)-norm.

- The weight \( w_i^d \) is set to 1 only if \( \hat{v}_i \) is close enough to \( u_{p(i)} \). Otherwise, it is set to zero. To mimic the exclusion mechanism of our Welsch formulation, we use the empirical three-sigma rule for the Gaussian function and set \( w_i^d = 0 \) if \( \| \hat{v}_i - u_{p(i)} \| \geq \nu_a \) where \( \nu_a \) is the Welsch’s function parameter used in our alignment term, i.e., the weight \( w_i^{d(k)} \) at the \( k \)-th iteration is chosen as

\[
w_i^{d(k)} = \begin{cases} 1 & \| \hat{v}_i - u_{p(i)}^{(k)} \| < 3\nu_a, \\ 0 & \text{others}, \end{cases}
\]

We refer to this variant as \( \ell_2 \)-norm-r.

- Instead of dynamically updating \( \nu_a \) and \( \nu_r \), we fix them throughout the optimization. We consider two variants:

- Both parameters are fixed to their maximum values during our dynamic update scheme. We refer to this variant as Welsch-fmax.

- Both parameters are fixed to their minimum values during our dynamic update scheme. We refer to this as Welsch-fmin.

- Both parameters are fixed to their average values between the minimum values and the maximum values during our dynamic update scheme. We refer to this as Welsch-fmed.

In Fig. 3, we compare different variants on problems where the source and target models overlap partially. To this end, we choose two pairs of meshes from the “crane” dataset of [57], each consisting of two consecutive frames from the sequence. For each pair, we choose a view direction and one of the meshes, and use the visible part of the mesh from the view direction as the source model. For the other mesh, we choose three view directions different from the source model, and use its visible parts from these directions to create three target models that overlap partially with the source model. Fig. 3 presents the registration results from different variants on these models. To compute the RMSE (Eq. (25)) in this scenario, we note that the point correspondence lies on the target model. Fig. 3 shows that the \( \ell_2 \)-based variants result in lower accuracy than other variants, due to the sensitivity of \( \ell_2 \)-norm to large errors, while the Welsch-based variants with fixed \( \nu_a \) and \( \nu_r \) are less effective in distinguishing non-overlapping points. Our Welsch-based formulation with a dynamic update of \( \nu_a, \nu_r \)
achieves better accuracy than these variants.

In Fig. 4, we further compare the variants on two pairs of meshes from the “samba” dataset [57] with a relatively large deformation between the source and the target. Our approach also achieves better accuracy than the variants in this scenario, thanks to its use of Welsch’s function which helps to attenuate the influence of incorrect correspondence obtained from the closest point, as well as its dynamic parameter update strategy that helps to avoid undesirable local minima of the optimization.

Radius parameter for deformation graph. The radius parameter \( R \) for the deformation graph construction affects the number of graph nodes. It can be used to trade off the degrees of freedom and computational efficiency. Fig. 5 shows results using different values of \( R \) on a pair of models from the MPI Faust dataset [61]. We set \( k_a = 0.1 \), \( k_b = 0.001 \). Here \#n denotes the number of the deformation graph nodes, \( t_1 \) and \( t_0 \) denote the computational time for the deformation graph construction and the numerical optimization respectively, and \#i denotes the number of iterations required for the numerical optimization solver to converge.

The optimization to converge to a suboptimal solution, as it becomes more difficult to maintain the local shape (e.g., see the right leg of the result from \( R = 3 \tilde{t} \)). In our experiments, our default choice \( R = 5 \tilde{t} \) usually achieves a good balance between accuracy and efficiency.

<table>
<thead>
<tr>
<th>( R )</th>
<th>#n</th>
<th>( t_1 )</th>
<th>#i</th>
<th>( t_0 )</th>
<th>#n</th>
<th>( t_1 )</th>
<th>#i</th>
<th>( t_0 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>558</td>
<td>0.11s</td>
<td>135</td>
<td>1.09s</td>
<td>229</td>
<td>0.13s</td>
<td>52</td>
<td>0.31s</td>
</tr>
<tr>
<td>5</td>
<td>122</td>
<td>0.15s</td>
<td>43</td>
<td>0.003</td>
<td>7</td>
<td>0.14s</td>
<td>61</td>
<td>0.26s</td>
</tr>
<tr>
<td>10</td>
<td>59</td>
<td>0.14s</td>
<td>61</td>
<td>0.004</td>
<td>10</td>
<td>0.14s</td>
<td>61</td>
<td>0.004</td>
</tr>
</tbody>
</table>

4.3 Comparison with the solver in [16]
A preliminary version of our method was presented in [16], where the same optimization problem is solved using a quasi-Newton approach instead. Similar to our solver, the solver in [16] constructs a surrogate for the target function and minimizes it to update the variables, with a dynamic update of the parameters \((\nu_a, \nu_r)\). However, their surrogate function is not quadratic and requires an inner L-BFGS solver for its minimization. The two solvers achieve similar results but differ in their efficiency, as shown in Fig. 6. Here we choose two pairs of adjacent frame data from the “jumping” dataset of [57], and perform registration with the same optimization problem on the “jumping” dataset of [57]. We set \( k_a = 10 \), \( k_b = 100 \) for both solvers.

4.4 Comparison with existing methods
We compare our method with the \( \ell_2 \)-based approach from [1] (N-ICP), the \( \ell_1 \)-based approach from [5] (RPTS), and the
and control $\alpha, \beta$ using the parameters $k_\alpha$ and $k_\beta$. For N-ICP, we ignore the parameter adjustment for the stiffness and landmark terms for a fair comparison. For SVR-$\ell_0$, we extract the $\ell_0$-optimization part and replace the corresponding points with the closest points. For both N-ICP and RPTS, we terminate the solver if it reaches 100 iterations or $\max_i \| \hat{v}_i^{(k+1)} - \hat{v}_i^{(k)} \| < 1 \times 10^{-3}$. RPTS also requires an inner ADMM solver to compute all affine transformations $X$, and it is terminated when reaching 20 iterations or $\| X^{(k+1)} - X^{(k)} \|_F < 1 \times 10^{-3}$. For SVR-$\ell_0$, we terminate the solver when it reaches 100 iterations or $\| X^{(k+1)} - X^{(k)} \|_F < 1 \times 10^{-3}$. We compare the methods on the human motion datasets from [57], the MPI Faust dataset [61], the TOSCA dataset [62], the DeepDeform dataset [41], and real-world data acquired by a Kinect camera. The results are presented in detail in the following paragraphs. The iterative registration process of different optimization-based methods on some problem instances is also demonstrated in the supplementary video.

**Clean data with small deformation.** In Figs. 7 and 8 and Tab. 1, we test the methods on the “handstand” and “march1” datasets from [57]. Each dataset contains a sequence of meshes for the motion of a human performer. We register the $i$-th frame to the $(i+2)$-th frame for each dataset. In this setting, the deformation between the source and the target models is relatively small. Fig. 7 shows the computational time and RMSEs (Eq. (25)) of 50 pairs of such models, while Fig. 8 shows the results on one example pair for each dataset. Tab. 1 further shows the mean and median values of RMSE for each dataset. The results show that overall our method can achieve higher accuracy with a shorter computational time.

**Clean data with large deformation.** In Tab. 2 and Fig. 9, we compare the methods on the MPI Faust dataset [61]. We select 10 subjects with nine poses, and use the first pose of each subject as the source surface and the other poses of the same subject as its targets. Due to the large difference between the source and the target models, we use the SHOT feature [63] with diffusion pruning [64] to obtain initial corresponding points. In Fig. 9, we visualize the initial correspondences using blue lines connecting the corresponding points. Fig. 9 shows a visualization of the computational time and RMSE for each pose pair using different methods, as well as the results for a particular pair on one subject. Tab. 2 further shows the average RMSE and computational time among all subjects for each pose.

### Table 1

Mean/median RMSE ($\times 10^{-2}$) and average computational time (s) using different methods on 50 pairs of models from the “handstand” and “march1” datasets [57]. We set $\alpha = 10$ for N-ICP, $\alpha = 100, \beta = 1$ for RPTS, $\alpha = 1, \beta = 1$ for SVR-$\ell_0$, and $k_\alpha = 100, k_\beta = 1$ for our method.

<table>
<thead>
<tr>
<th></th>
<th>N-ICP</th>
<th>RPTS</th>
<th>SVR-$\ell_0$</th>
<th>Ours</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>handstand</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mean RMSE</td>
<td>4.33</td>
<td>3.30</td>
<td>6.45</td>
<td>1.21</td>
</tr>
<tr>
<td>Median RMSE</td>
<td>3.09</td>
<td>1.17</td>
<td>4.95</td>
<td>0.90</td>
</tr>
<tr>
<td>Average Time</td>
<td>5.31</td>
<td>215.47</td>
<td>96.61</td>
<td>1.70</td>
</tr>
<tr>
<td><strong>march1</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mean RMSE</td>
<td>1.78</td>
<td>0.93</td>
<td>2.49</td>
<td>0.75</td>
</tr>
<tr>
<td>Median RMSE</td>
<td>1.46</td>
<td>0.60</td>
<td>2.51</td>
<td>0.33</td>
</tr>
<tr>
<td>Average Time</td>
<td>2.86</td>
<td>103.12</td>
<td>55.36</td>
<td>1.06</td>
</tr>
</tbody>
</table>

2 https://github.com/Juyong/Fast_RNRR
Dense Noise (0.7\text{l}) where each pair consists of the per mesh vertex, which significantly increases the number pair. We can see that our method requires significantly less noise data.

Fig. 9. Comparisons on the MPI Faust dataset [61]. The plot in the upper right visualizes the computational time and RMSE using different methods on 80 pairs of models from the dataset, while the rendered images show the results on a particular pair.

**TABLE 2**

Mean RMSE ($\times 10^{-3}$) and average computational time (s) using different methods on models from the MPI Faust dataset [61]. We set $\alpha = 0.1$ for N-ICP, $\alpha = 0.1$, $\beta = 0.001$ for RPTS, $\alpha = 0.001$, $\beta = 1$ for SVR-$\ell_0$, and $k_\alpha = 0.1$, $k_\beta = 0.001$ for our method.

<table>
<thead>
<tr>
<th>Pose pair</th>
<th>N-ICP</th>
<th>RPTS</th>
<th>SVR-$\ell_0$</th>
<th>Ours</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time RMSE</td>
<td>Time RMSE</td>
<td>Time RMSE</td>
<td>Time RMSE</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1.47</td>
<td>21.30</td>
<td>67.18</td>
<td>43.77</td>
</tr>
<tr>
<td>2</td>
<td>0.97</td>
<td>9.69</td>
<td>35.52</td>
<td>14.54</td>
</tr>
<tr>
<td>3</td>
<td>1.44</td>
<td>26.29</td>
<td>88.58</td>
<td>65.32</td>
</tr>
<tr>
<td>4</td>
<td>1.85</td>
<td>20.69</td>
<td>86.69</td>
<td>23.83</td>
</tr>
<tr>
<td>5</td>
<td>0.84</td>
<td>5.60</td>
<td>29.64</td>
<td>6.96</td>
</tr>
<tr>
<td>6</td>
<td>1.59</td>
<td>19.85</td>
<td>56.75</td>
<td>23.68</td>
</tr>
<tr>
<td>7</td>
<td>2.05</td>
<td>110.15</td>
<td>125.90</td>
<td>132.49</td>
</tr>
<tr>
<td>8</td>
<td>1.53</td>
<td>22.87</td>
<td>66.29</td>
<td>47.80</td>
</tr>
</tbody>
</table>

Mean 1.47 29.56 69.57 44.80 38.94 300.14 0.58 11.44
Median 1.50 21.00 66.74 33.80 35.49 288.49 0.58 5.98

**Fig. 10.** Visualization of computational time and RMSE resulting from different methods on 50 pairs of models constructed from the “jumping” dataset [57] with different profiles of added noise.

**TABLE 3**

Mean/median RMSE $(\times 10^{-2})$ and average computational time (s) using different methods on 50 pairs of models constructed from the “jumping” dataset [57] with different noise profiles. We set $\alpha = 10$ for N-ICP,

<table>
<thead>
<tr>
<th>Data</th>
<th>N-ICP</th>
<th>RPTS</th>
<th>SVR-$\ell_0$</th>
<th>Ours</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dense Noise (0.37)</td>
<td>Median RMSE</td>
<td>6.39</td>
<td>4.47</td>
<td>8.67</td>
</tr>
<tr>
<td></td>
<td>Average Time</td>
<td>4.88</td>
<td>206.29</td>
<td>82.87</td>
</tr>
<tr>
<td>Dense Noise (0.77)</td>
<td>Median RMSE</td>
<td>6.62</td>
<td>5.06</td>
<td>8.77</td>
</tr>
<tr>
<td></td>
<td>Average Time</td>
<td>4.74</td>
<td>224.94</td>
<td>83.50</td>
</tr>
<tr>
<td>Sparse Noise (5%)</td>
<td>Median RMSE</td>
<td>6.41</td>
<td>4.47</td>
<td>8.66</td>
</tr>
<tr>
<td></td>
<td>Average Time</td>
<td>5.09</td>
<td>270.34</td>
<td>89.42</td>
</tr>
<tr>
<td>Sparse Noise (50%)</td>
<td>Median RMSE</td>
<td>6.96</td>
<td>5.37</td>
<td>8.83</td>
</tr>
<tr>
<td></td>
<td>Average Time</td>
<td>4.95</td>
<td>206.29</td>
<td>82.87</td>
</tr>
</tbody>
</table>

Fig. 11 shows the results on a particular model pair. Tab. 3 shows for each method the average RMSE and computational time for all model pairs under each noise profile. We can see that our method achieves better overall performance in

as the source model, and the (i+2)-th frame with added Gaussian noise as the target model. For each pair, we generate four noisy variants of the target model as follows:  
- Dense Noise (0.37): we add noises with a standard deviation $\sigma = 0.37$ to all vertices;  
- Dense Noise (0.77): we add noises with a standard deviation $\sigma = 0.77$ to all vertices;  
- Sparse Noise (5%): we add noises with a standard deviation $\sigma = I$ to 5% of the vertices;  
- Sparse Noise (50%): we add noises with a standard deviation $\sigma = I$ to 50% of the vertices.

Noisy data. To evaluate the effectiveness of our method on noisy data, we synthesize noisy models using the “jumping” dataset from [57]. Specifically, we collect 50 pairs of models, where each pair consists of the $i$-th frame from the dataset and the (i+2)-th frame with Gaussian noise as the target model. For each pair, we generate four noisy variants of the target model as follows:
terms of both accuracy and efficiency.

**Partial overlaps.** We also evaluate different methods on models that overlap partially. We choose three pairs of models from the “march2” dataset and three pairs of models from the “squat1” dataset from [57], where the models in each pair are two frames apart. Then for each pair, we construct two different view directions for the two models respectively, and use their visible parts from these directions to construct source and target models that overlap partially. Then for each pair, we construct source and target models that overlap partially, and use their visible parts from these directions to construct source and target models that overlap partially.

We also construct five pairs of models from the TOSCA dataset [62] in a similar way. Figs. 12 and 13 show the results from different methods on these partially overlapping models. N-ICP performs poorly on such models, since its \( \ell_2 \)-based optimization can be sensitive to target function terms with large residuals. The \( \ell_1 \)-based formulation in RTPS is more robust than an \( \ell_2 \) formulation, but it still attempts to reduce large residual terms, which can lead to incorrect alignments. SVR-\( \ell_0 \) applies an \( \ell_0 \)-based regularization term but its alignment term is still based on the \( \ell_2 \)-norm, which can affect its robustness. In comparison, our use of the Welsch’s function on both the alignment term and the regularization term helps to improve robustness and achieve overall better results in these examples.

**Real scanned data.** Our method is also effective on real-world data. To demonstrate this, we conduct experiments on the real scanned data from the DeepDeform dataset [41]. This dataset provides RGB-D frame pairs with estimated optical flows and scene flows and manually annotated sparse corresponding point pairs. To evaluate the accuracy of the result, we adopt the scene flows provided by the dataset as the ground-truth deformation, and compute an error measure as follows:

\[
 r_s = \sqrt{\frac{1}{|S|} \sum_{i \in S} \| \hat{v}_i - (v_i + t_i^s) \|^2 },
\]

where \( S \) is the set of scene flows for the source surface points.
We use 349 pairs in the validation set for testing, as the validation set includes the scene flows required for computing $r_s$. Besides N-ICP, RPTS and SVR-$\ell_0$, we also compare with four recent deep learning methods, Lepard+NICP [65], NDP [66], RMA-Net [44], and SyNoRiM [67] using their open-source implementations. For each frame, we first downsample the depth map and obtain a point cloud according to the intrinsic camera parameters, then remove the background using the ground-truth segmentation masks provided by the dataset. Finally, we construct a triangular mesh from the point cloud according to their adjacency in the depth map, and extract the largest connected part. We then use either the resulting mesh or its vertices as the input to a method depending on its requirement. Since SyNoRiM requires 8192 points on each input shape, when the number of points on an input shape is different from 8192 we either downsample the points or add repeated points to achieve the required number. To improve the performance of RMA-Net, we fine-tune the model on the 2120 data pairs from the training set of the DeepDeform dataset. For other learning-based methods, since they already test their performance on the DeepDeform dataset, we directly evaluate their performance using their pre-trained models. We perform the same processing on the training data as mentioned above for the testing data. The training and testing of the

---

5. https://github.com/huangjh-pub/synorim
learning-based methods are run on an NVIDIA GeForce RTX 3090 GPU. For both SVR-ℓ₀ and our method, we set the sampling radius \( R = 8l \) for the deformation graph construction for better efficiency. For the optimization-based methods, we first perform a coarse alignment using the rigid registration method from [12] before running each method for non-rigid registration. Tab. 4 shows the mean and median error measures and average computational time for each method, while Fig. 14 shows some example results from the methods. Overall, the optimization-based methods achieve lower registration errors than the learning-based methods, which is likely due to the capability of optimization-based methods to locally refine the alignment. RMA-Net has worse accuracy than other learning-based methods, because its training does not consider partial overlaps between the input shapes. Our method achieves the lowest mean value of \( r_s \) and the third lowest median value of \( r_k \) (and is only 3% higher than the lowest median \( r_k \) value), which shows the good accuracy of our method. On the other hand, the learning-based methods are overall faster than the optimization-based methods, thanks to the efficiency of running a neural network model on the GPU. Nevertheless, our method is only slower than SyNoRiM and is significantly faster than other optimization-based methods.

Performance on mobile devices. Furthermore, due to the use of the deformation graph and our efficient solver, our method can be deployed on devices with limited compute capabilities. To this end, we compare our method and other optimization-based methods on a Raspberry Pi 4B with 8GB of RAM and a quad-core Cortex-A72 at 1.5GHz running Ubuntu 20.04. In Fig. 15, we show the comparison on a problem instance from the the DeepDeform dataset. In Fig. 16, we compare the methods on data pairs obtained by scanning a self-rotating human subject with a Kinect camera; the scans are converted to triangle meshes for registration, by removing isolated points and connecting the remaining points according to their adjacency in the depth map. Due to the lower compute power of Raspberry Pi, we relax the termination condition of our method to \( \epsilon = 3 \times 10^{-4} \), which causes no notable change to the result while still being tighter than other methods. We can see that our method is significantly faster than other methods and achieves similar or better accuracy.

5 Conclusion

In this paper, we proposed a robust non-rigid registration model based on Welsch’s function. Applying the Welsch’s
our method achieves good results on many examples, it still has some limitations. First, although our use of the robust metric helps to mitigate the impact of incorrect correspondence, our non-convex formulation still requires proper initialization to avoid converging to an undesirable local minimum. In particular, when there is a significant difference between the source and target shapes, the closest-point correspondence may be erroneous on a large part of the surface and lead to inaccurate alignment (see Fig. 17 for an example of such failure cases). The issue can potentially be addressed by adopting a more sophisticated point correspondence method that also accounts for the local shape features. Secondly, our deformation graph construction method relies on a global radius parameter that leads to an approximately uniform density of graph nodes. Such a deformation graph may not be suitable for meshes with highly non-uniform samples. This can be improved using an adaptive radius parameter that depends on the local curvatures and sampling densities, which we will leave as future work.

**ACKNOWLEDGMENTS**

This work was supported by the National Natural Science Foundation of China (No. 62122071), the Youth Innovation Foundation of China (No. 62122071), the Youth Innovation Promotion Association CAS (No. 2018495), and the Fundamental Research Funds for the Central Universities (No. WK3470000021).

**REFERENCES**


Yuixin Yao is currently working toward a PhD degree in the School of Mathematical Sciences, University of Science and Technology of China. Her research interests include computer vision, computer graphics and numerical optimization.

Bailin Deng is a Senior Lecturer in the School of Computer Science and Informatics at Cardiff University. He received the BEng degree in computer software (2005) and the MSc degree in computer science (2008) from Tsinghua University (China), and the PhD degree in technical mathematics (2011) from Vienna University of Technology (Austria). His research interests include geometry processing, numerical optimization, computational design, and digital fabrication. He is a member of the IEEE.

Weiwei Xu is a researcher with the State Key Lab of CAD & CG, College of Computer Science, Zhejiang University, awardee of NSFC Excellent Young Scholars Program in 2013. His main research interests include the digital geometry processing, physical simulation, computer vision and virtual reality. He has published around 70 papers on international graphics journals and conferences, including 20 papers on ACM TOG. He is a member of the IEEE and ACM.

Juyong Zhang is a professor in the School of Mathematical Sciences at University of Science and Technology of China. He received the BS degree from the University of Science and Technology of China in 2006, and the PhD degree from Nanyang Technological University, Singapore. His research interests include computer graphics, computer vision, and numerical optimization. He is an associate editor of IEEE Transactions on Multimedia and The Visual Computer.