DEPARTMENT OF INFORMATICS

TECHNICAL UNIVERSITY OF MUNICH

Master's Thesis in Informatics: Games Engineering

Improving Deformation Graph Based Non-Rigid Registration

Angela Denninger

DEPARTMENT OF INFORMATICS

TECHNICAL UNIVERSITY OF MUNICH

Master's Thesis in Informatics: Games Engineering

Improving Deformation Graph Based Non-Rigid Registration

Verbesserung von nicht-rigider Registrierung basierend auf der Verwendung von Deformationsgraphen

Author:	Angela Denninger
Supervisor:	Prof. Dr. Matthias Nießner
Submission Date:	15.10.2019

I confirm that this master's thesis is my own work and I have documented all sources and material used.

Munich, 15.10.2019 Angela Denninger

Abstract

We propose *Adaptive Rigidity at Edge with Smoothness Reduction*, an adaptive non-rigid registration objective function in order to improve the accuracy of non-rigid registration algorithms based on deformation graphs. The objective function simultaneously optimizes for non-rigid deformations and optimal deformation graph edge rigidity weights in combination with a simple, but effective smoothness reduction strategy. We show that our method achieves better results than other smoothness reduction and deformation graph refinement strategies. Our method is able to represent the underlying kinematics of the deforming object, which leads to a more robust and accurate non-rigid registration.

Contents

1 Introduction

A common computer vision problem is to create accurate digital 3D models from complex real-world objects. For static objects algorithms like SLAM or dense reconstruction systems provide promising results. However, these algorithms are based on the assumption that the scene is mostly static, which leads to failures when considering deforming objects. Therefore, in real-world applications it is often necessary to reconstruct non-rigid objects. For example, when scanning a human with a handheld camera we can ask an adult to hold still, but the same task will most probably fail if we ask a child. While it is possible to reconstruct non-rigid scenes with multiview reconstruction, by reconstructing the object with images from different angles taken at the same time, such algorithms require a professional setup for creating those images. In such cases, where a professional setup is not available or it is impossible to get the target object into the professional setup an algorithm which can handle non-rigid objects is required.

Non-rigid objects of known types may be handled by using static shape template models, which are fit to the deforming object. For example, a static shape template model of a face can be fit in a first step to a specific face. In a second step, the model is adapted in order to fit and track deformations of the face. Although the usage of a shape template model may yield amazing results, depending on the object it is often not feasible to construct a static shape template model. Moreover, due to it's structure it is restricted to concrete types of objects. Template free methods on the other hand jointly build the shape model along with tracking the non-rigid deformations.

Another goal of non-rigid registration is to not only reconstruct an accurate 3D model of an object, but to also track the deformation of the object over time. A common application is movement capturing for character animations, to make them more realistic. To do so, not only the 3D reconstruction is interesting, but also the deformation from the reconstructed object to all possible deformed poses of the object. While algorithms like DynamicFusion [1] and VolumeDeform [2] are able to reconstruct non-rigid objects and track their deformation over time, they sometimes fail if the deformation from one frame to another is too big or the subset of overlapping surface areas is too small. The difficulty of this problem arises from the high number of unknown parameters and the ambiguity of the problem as many different deformations can lead to the same shape. Another challenge are changes in topology, for instance if two hands that are at first combined, separate.

Many template free non-rigid registration methods are based on the construction and fitting of a deformation graph, which is able to deform general surface representations. The

accuracy of the fitting depends on the characteristics of the deformation graph and the used registration method. In the scope of this thesis, different deformation graph construction and registration methods will be introduced and analyzed in order to find the method which most improves the robustness and accuracy of non-rigid registration under high deformation. We will first give an overview over state-of-the-art methods, followed by an introduction of the general concept of rigid and non-rigid registration. Subsequently we will define different adaptions of the non-rigid registration and compare them again each other.

2 Previous Work

This chapter gives an overview over different rigid and non-rigid registration methods. Rigid registration methods as iterative closest point and multiview reconstruction will be explained as well as real-time variants. Also non-rigid registration methods, starting from shape deformation frameworks over template based and template free approaches and recent real-time non-rigid registration methods will be explained.

2.1 Rigid Registration

Rigid registration denotes the process of matching surfaces or parts of surfaces in 3D by finding the rotation and translation, which places the surfaces at the same location and orientation in space. The combination of rotation and translation is called rigid transformation. A typical application of rigid registration is the reconstruction of detailed 3D objects from complex static real-world scenes, by registering a series of depth images or point clouds taken from the same object with different angles.

The iterative closest point algorithm was introduced by Besl and McKay [3] and Chen and Medioni [4] and solves for frame to frame registration by iteratively solving closest point correlations and minimizing a distance metric with respect to the searched rigid transformation. While Besl and McKay use a simple point-to-point distance metric, Chen and Medioni introduced the point-to-plane metric which leads to a faster convergence than the point-to-point metric [5]. Different improvement on the iterative closest point algorithm were made over time. Pulli et al. [6], Turk et al. [7], Masuda et al. [8] and Gelfand et al. [9] improved the performance and stability by using different subsets of the available points and removing outliers by different strategies. Simon [10] sped up correspondence search by using kd-trees. Blais and Levine [11] introduced an even faster correspondence search approach based on depth images, finding correspondences by projecting source points into the target image. Low [12] approximated the nonlinear optimization problem with a more efficient to solve linear least-squares problem. Zhou et al. [13] stabilize the rigid registration algorithm by preserving detailed geometry using points of interest and running a global optimization algorithm which reduces the distributed residual registration error over the environment.

2.1.1 Multiview Reconstruction

Multiview reconstruction denotes the reconstruction of 3D models given multiple images from the same object under different angles at the same time. Cremers and Kolev [14] proposed a convex formulation for 3D reconstruction from multiple images using silhouette

consistency. To restrict the domain to a solvable problem, it is relaxed to a convex silhouette consistent reconstruction first, followed by a projection onto the original non-convex set, leading to an optimal solution. The algorithm was stabilized by Oswald et al. [15] by adding additional connectivity constraints to ensure consistent reconstructions of thin objects. The algorithm introduced by Leroy et al. [16] improves the accuracy of multiview reconstruction of sequences, by using temporal redundancy in the sequences for precision refinement of 4D reconstruction. Therefore a frame is reconstructed using information of neighboring frames and warping local and reliable geometric regions of neighboring frames to the reconstructed frame.

2.1.2 Real-time Mapping of Static Scenes Using Implicit Surface Models

Newcombe et al. [17] present *KinectFusion* a framework for real-time mapping and tracking of complex static scenes using a single handheld depth camera. Instead of reconstructing the scene frame-by-frame, Newcombe et al. construct an implicit surface representation of the scene based on a truncated signed distance function (TSDF). For each depth frame the rigid transformation is obtained by minimizing the difference between the transformed frame and the current implicit surface representation of the scene. After obtaining the rigid transformation the implicit surface representation of the observed scene is updated by fusing the current depth frame into the representation. This makes the algorithm robust against noise in the depth data.

While Newcombe et al. [17] are able to reconstruct scenes in real-time, the quality of the reconstructed scenes is limited by the scale of the underlying memory inefficient regular voxel grids. Nießner et al. [18] solve this problem by introducing *voxel hashing*. The approach represents a virtual space of small voxels, whose locations are addressed with a spatial hash function. By only storing surface data of observed geometric data, the memory consumption is significantly reduced and the scene size not restricted. As a result, voxel hashing allows real-time access and updates of implicit surface data, allowing high-quality reconstructions at scale.

Kerl et al. [19] introduced in *Dense Visual SLAM for RGB-D Cameras* a simultaneous localization and mapping (SLAM) algorithm. To track motions of a camera directly image alignment of color and depth frames are used. The algorithm optimizes for pose graph consistency [20] and loop closure to determine globally consistent camera motions.

Dai et al. [21] extended in *BundleFusion* the work of Nießner et al. [18] with an efficient global pose optimization algorithm, which enables globally consistent surface reconstructions at real-time. To reach real-time performance, the algorithm organizes the input frames in a hierarchy of chunks and initially solves the global alignment of all frames in each chunk before solving the global alignment between chunks. The data-parallel bundle adjustment is solved on the GPU in real-time. By updating the reconstructed model based on the refined rigid transformation for each frame, the algorithm is able to correct errors in the 3D reconstruction. Due to this, the algorithm is robust against tracking failure that usually occurs from occlusions, fast frame-to-frame motions or featureless regions.

2.2 Non-Rigid Registration

Non-rigid registration denotes the tracking and reconstruction of dynamic motions. As modelling non-rigid motions of general deforming scenes requires a lot more parameters as modeling rigid motions, non-rigid registration is a much more challenging problem. Finding an optimal solution is an ill-posed problem with infinitely many solutions which deform one shape to another. If no template is given beforehand, non-rigid registration also needs to solve for object deformation and object shape simultaneously. Existing non-rigid registration approaches typically employ an as-rigid-as-possible or an embedded deformation regularization. Approaches that purely rely on implicit surfaces also use the as-killing-as-possible regularization.

2.2.1 Non-Rigid Shape Manipulation Methods

In *As-Rigid-As-Possible Surface Modeling* Sorkine et al. [22] introduce a shape deformation framework, which allows manual shape deformations that deform the surface of an object while maintaining as much rigidity as possible. To deform an object some of the objects vertices can manually moved to a new location. The algorithm then solves for an as-rigid-as-possible deformation. This is done by minimizing an objective function consisting of a data term, which fits the manually moved vertex to the new position and a rigidity term which penalizes deviation from rigid transformations. Adding weights to the rigidity term for each edge, which compensates for non-uniformly shapes, allows the algorithm to be mesh independent.

In *Embedded deformation for shape manipulation*, Sumner et al. [23] use an embedded deformation graph to allow intuitive shape deformations while preserving surface features. A surface is deformed by smoothly blending the affine deformations of the individual deformation graph nodes and applying the blended deformations to the vertices of the surface. Given a deformation graph, a user can manipulate the embedded surface by selecting nodes of the deformation graph and adjust their location. To solve for an optimal deformation while preserving surface details an objective function is used, which simultaneously optimizes the fit of the moved vertices and the rigidity of the deformation. The rigidity term is optimized by enforcing the affine transformations of adjacent graph nodes to be rigid and similar.

Instead of solving for an as-rigid-as-possible deformation, as done in [23] and [22], Solomon et al. [24] solve in *As-Killing-As-Possible Vector Fields for Planar Deformation* for as-killing-as-possible deformations of 2D objects by using approximate killing vector fields. Killing vector fields enforce the deformation to be locally isometric. Approximate killing vector fields were introduced by Ben-Chen et al. [25], to find the best approximation of killing vector fields for non isometric surfaces.

2.2.2 Non-Rigid Registration Using Shape Templates

Most shape template based non-rigid registration methods focus on human body parts where specialized templates are learned or manually designed. Chen et al. [26] and Li et al. [27] demonstrate high accuracy for tracking and reconstructing faces. Iason et al. [28] and Qian et al. [29] focus on the reconstruction of hands. Complete human poses were reconstructed and tracked by Taylor et al. [30]. Schmidt et al. [31] and Ye et al. [32] introduced frameworks for tracking general articulated objects. Yu et al. [33] track and reconstruct in *BodyFusion* non-rigid surface motions from humans in real-time, by using a human motion prior consisting of a skeleton model with attached graph node deformations. The algorithm jointly solves for skeleton and graph node deformations. Because of the simplified skeleton model the algorithm is able to robustly reconstruct and track humans.

2.2.3 Template-Free Non-Rigid Reconstruction

In *Optimal Step Nonrigid ICP Algorithms for Surface Registration* Amberg et al. [34] extend iterative closest point to non-rigid registration by assigning an affine transformation to each vertex and minimizing the cost function with a stiffness regularization. Similar to Amberg et al., Li et al. [35] extend in *Global Correspondence Optimization for Non-Rigid Registration of Depth Scans* the work of Sumner et al. [23] to a registration framework, by combining non-rigid registration with iterative closest point. Their algorithm simultaneously solves for correspondences between points, non-rigid deformation graph transformations, the global deformation, and regions of overlap. Li et al. extended their work in *Robust Single-View Geometry and Motion Reconstruction* [36] by using a smooth initial template without high-frequency details to register a sequence of frames. The algorithm starts with a coarse deformation graph and is refined in regions with high smoothness error. To handle non-uniformly sampled graph nodes, the node weighting as introduced by Sumner et al. is adapted to be based on geodesic distances and regions of influence of the deformation graph nodes. To create an accurate fit of the final registration the smooth template is augmented with detail information from the previous frame.

2.2.4 Real-Time Non-Rigid Registration

Zollhofer et al. [37] introduced in *Real-time Non-rigid Reconstruction using an RGB-D Camera* the first template-based non-rigid tracking approach working for general objects in real-time. Given a template model the algorithm tracks non-rigid object motions from sequences of color and depth images. The real-time performance is achieved using a hierarchical coarse to fine GPU optimization strategy. While the algorithm enables real-time tracking, it requires a pre-computed template mesh of the object, which is depending on the object a cumbersome if not infeasible process.

In *DynamicFusion: Reconstruction and Tracking of Non-rigid Scenes in Real-Time* Newcombe et al. [1] generalize KinectFusion [17] to reconstruct implicit surface representations and track

deformations of non-rigid scenes in real-time. The implicit surface representation, a truncated signed distance field within a voxel domain, is therefore deformed to each dynamic scene of the motion with a warp field. The warp field is similar to the deformation graph used by Sumner et al. [23] but has a global deformation additionally to the deformation per node. The warp field of the current scene is estimated by applying an iterative closest point non-rigid registration algorithm. For each scene the implicit surface representation is updated by fusing the current target scene given the estimated warp field into the implicit representation. Additionally the warp field structure is extended when new geometry is captured. For more robustness the edges used for the rigidity regularization of the warp field are connected in a hierarchical manner. To achieve real-time performance the algorithm is solved on the GPU.

Innmann et al. [2] improved in *VolumeDeform: Real-time Volumetric Non-rigid Reconstruction* DynamicFusion [1] to higher reconstruction quality and robustness while maintaining real-time performance. Additionally to the input depth information Innmann et al. use sparse color features to find correspondences between images which leads to a more robust tracking. The color information is also fused into the truncated signed distance fields. VolumeDeform also provides higher reconstruction quality using a dense volumetric representation, where the scene geometry and the motion is stored in the same resolution. To achieve real-time performance they employ a hierarchical coarse-to-fine data-parallel GPU optimization strategy.

In *Real-time Geometry, Albedo and Motion Reconstruction Using a Single RGBD Camera*, Guo et al. [38] solve jointly for geometry, motion, surface reflectance and illumination. This leads to a more robust tracking under large motions due to the dense reflectance constancy assumption which is more generally valid than the color constancy assumption.

Instead of minimizing the error between two meshes Slavcheva et al. [39] optimize in *KillingFusion: Non-rigid 3D Reconstruction without Correspondences* the error between two signed distance functions, while enforcing the deformation field to be approximately killing as done in [24] and [25]. The approximately killing term serves the same purpose as the rigidity term in [22]. The killing regularization is a trade-off between the killing property and the volume distortion penalization. Also a level set regularization is used to ensure geometric correctness during the deformation. To fit two truncated signed distance functions with a deformation field, the difference between the deformed truncated signed distance function and the cumulative one is minimized. This leads to an alignment without explicit point correspondence search. Also the algorithm is able to robustly handle topological changes.

Dou et al. [40] introduce in *Fusion4D: real-time performance capture of challenging scenes* a key volume strategy which allows high quality registration for challenging sequences and is robust to topological changes. As done by Innmann et al. [2] Dou et al. fuse the acquired depth maps into the reconstruction volume. But instead of using the initial frame as fixed template volume, the template is periodically reset to a fused local data volume also called key volume. The algorithm automatically detects tracking failures and refreshes misaligned

voxels. But in contrast to other non-rigid registration method it loses its global tracking information due to the reset of the template.

Dou et al. [41] achieve in *Motion2Fusion: Real-time Volumetric Performance Capture* high speed reconstruction by using machine learning to estimate dense correspondences between input data and reconstruction. In addition topological changes are handled using backward and forward alignment and geometrical details are preserved by the use of a detail layer with a finer resolution then the voxel size.

3 Method Overview

State-of-the-art non-rigid registration algorithms as presented in Chapter 2 allow impressive reconstructions of deformed objects when sufficiently high frame rates can be used. However, intense deformations reduce the robustness of the algorithms. The main objective of this thesis is to improve the robustness and quality of the registration of a given template to a sequence of deformed point clouds. While current methods usually simultaneously improve the template model and the registration, in this thesis we will only focus on the registration itself, as an improvement here, will also lead to improvements of more complex non-rigid registration frameworks. Our goal is to improve the non-rigid registration by introducing and analyzing different strategies for fitting and creating deformation graphs, which are used to describe the deformation of the template model.

In Chapter 4 and 5 the general non-rigid registration pipeline for a series of point clouds will be described, followed by an introduction of our adaptations to the non-rigid registration algorithm in Chapter 6. In Chapter 7 the impact of different strategies on the accuracy of the registration will be analyzed.

4 Iterative Closest Point

The iterative closest point algorithm (ICP) minimizes the difference between two point clouds by finding the rigid transformation that optimally transforms a *source* point cloud to a *target* point cloud. An example ICP registration is visualized in Figure 4.1.

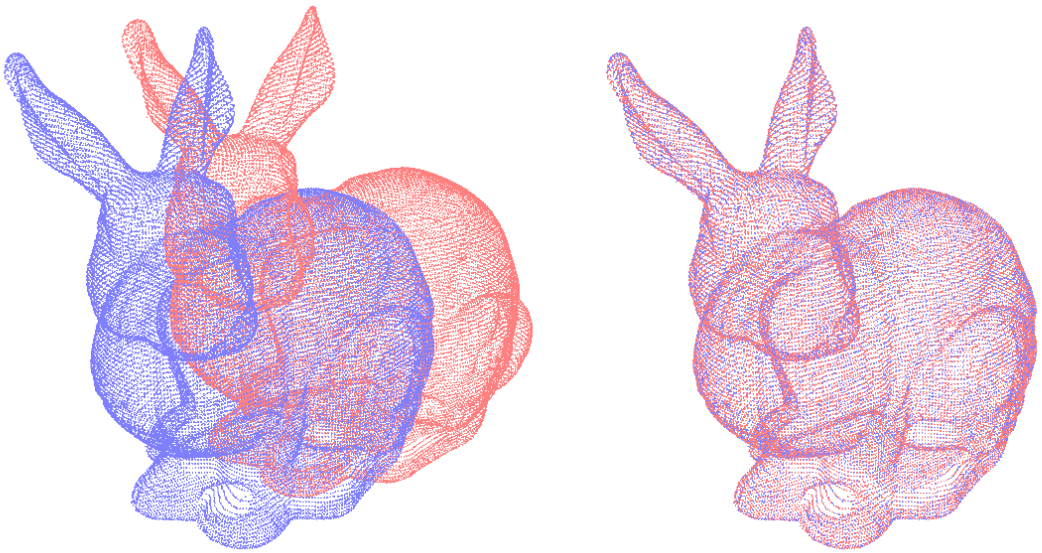


Figure 4.1: The iterative closest point algorithm. On the left a source (red) and a target (blue) point cloud. On the right the two point clouds aligned with a rigid deformation found by the ICP algorithm.

The ICP algorithm illustrated in Algorithm 1 converges to a optimal rigid transformation by iteratively finding closest point correspondences and minimizing distance metric for the optimal rigid transformation. The distance metric (Section 4.4) is optimized by minimizing the sum of distances between corresponding point pairs (Section 4.2). For more robustness, correspondences between target points and transformed source points that heuristically lead to suboptimal performance are excluded from the optimization step (Section 4.3). The runtime of the registration is usually improved by using a subset of the available points in the two point clouds, as described in Section 4.1. The algorithm receives the source $S = \{s_i \mid i = 1...n_s,\ s_i \in \mathbb{R}^3\}$ and target point cloud $T = \{t_j \mid j = 1...n_t,\ t_j \in \mathbb{R}^3\}$ as input, with n_s and n_t denoting the number of points in the point clouds, and returns the optimal rigid transformation RT typically initialized with the identity transformation I. The algorithm

terminates after converging to an optimal rigid transformation or after exceeding a predefined number of iterations.

Algorithm 1: Iterative closest point

Input : Source and target point cloud
Output: Best fitting rigid transformation RT

$RT \leftarrow I$
while *not converged* **do**
  Select points from the source point cloud
  for *each of the selected points* **do**
    Find corresponding closest *target* point to the deformed *selected source* point.
    Remove outliers.
  end
  $RT \leftarrow$ minimize error metric for corresponding point pairs
end

The algorithm was first introduced by Chen and Medioni [4] and Besl and McKey [3] and improved among others by [6][7][8][5]. ICP is today a common algorithm to solve for rigid transformations between point clouds. In the following we will explain each step of the algorithm in detail.

4.1 Source Point Selection

Different source point selection strategies lead to different runtime and accuracy of the ICP algorithm. By selecting *all* points [3], the algorithm will converge to the best fitting rigid transformation but will suffer under long runtimes for large numbers of points. Instead of using all source points, the points can be selected by *uniformly* [7] or *randomly* [8] sampling a predefined amount of points. If the selected subset is representative for the two point clouds the algorithm will converge to an optimal rigid transformation, which best aligns the source to the target point cloud. Another method is to select a small subset of points in the first iteration and increase the number of points in each iteration. As the algorithm iteratively converges to the best fitting rigid transformation, it is sufficient to obtain an approximation of the rigid transformation in the first steps. The used subset should be representative for the point clouds as otherwise wrong alignments can occur and the algorithm may diverge. To find a representative subset Gelfand et al. [9] introduced *stable sampling* choosing samples that constrain all degrees of freedom of the rigid transformation and therefore minimizing the uncertainty. We define the set of all selected points as $P = \{p_i \mid i = 1...n_p,\ p_i \in S\}$ where n_p is the number of selected points.

4.2 Correspondence Finding

Closest point correspondences are searched between the selected source points P deformed with the rigid transformation RT^k of the current iteration k and the target points T. A naive approach would be to calculate for each selected source point the distance to all target points and select the nearest one, which leads to a performance of $O(mn)$ where $m = n_p$ is the number of selected source points and $n = n_t$ is the number of target points. As point clouds usually are very large, often containing 10000 and more points, large number of distance calculations are performed. A well known way to speed up closest point search is to use a kd-tree constructed on the target points [10], this will speed up the average search time to $O(m \log n)$. More formally, the corresponding point to a deformed source point $RT(p)$, $p \in P$, is the point in the target space T with minimal distance to the deformed source point $RT(p)$ and defined as

$$c(p) = \arg\min_{t \in T} d_E(RT(p), t) \tag{4.1}$$

where d_E denotes the euclidean distance.

If the input data is given by depth images and small rigid motions between point clouds can be assumed, the approach *Protective Data Association* introduced by Blais and Levine [11] can be used to further accelerate the corresponding finding. Protective Data Association finds correspondences by projecting the source points into the depth image of the target points using the currently assumed rigid transformation. The corresponding target point is obtained by using the target point where the source point is project to as corresponding point. This will only work if the rigid transformation between the two frames is small but it will also lead to the big advantage of very fast point cloud correspondence in $O(m)$ as points are associated directly. In most real-world applications the input to the algorithm is a recorded stream of depth images taken from a rgb-d camera with a sufficient frame rate which implies that the taken assumptions are usually true.

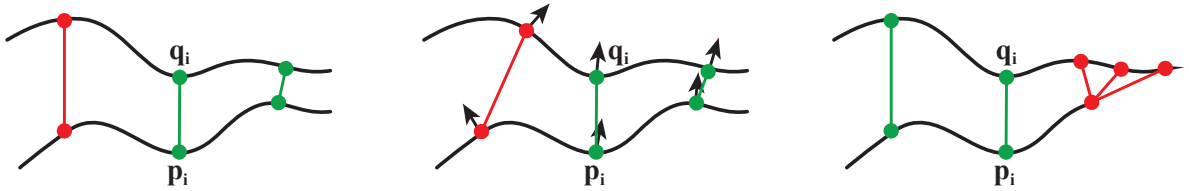
We define a set C of corresponding deformed source and target points as

$$C = \{\ (RT(p),\ c(p)) \mid p \in P,\ c(p) \in T\}. \tag{4.2}$$

4.3 Outlier Removal and Pruning

Suboptimal correlations are excluded from the optimization step as they can prevent the algorithm from converging. A correlation between two points is suboptimal if the distance between the two points is too large or if the angle between their normals exceeds a cutoff value. A common distance pruning is to prune all corresponding points where the distance is more than k times the mean distance of all corresponding points [8]. The pruning condition for distances and normal angles are visualized in Figure 4.2a and Figure 4.2b. Correspondences to boundary points are also pruned as they can lead to a systematic bias of the alignment [7].

Boundary points are often matched when two point clouds do not overlap completely and therefore no corresponding points exist in this area, as visualized in Figure 4.2c. Boundary points can be derived by constructing the implicit surface from the point cloud.



(a) Prune corresponding point pairs based on the distance between them.

(b) Prune corresponding point pairs based on normal compatibility.

(c) Prune corresponding point pairs that contain boundary points.

Figure 4.2: Correspondence rejection. Good corresponding point pairs are kept (green) while ouliers (red) are pruned.

In summary we reject a corresponding point pair $cp_p = (RT(p), c(p))$, $cp_p \in C$ from the set of selected points if

$$\text{prune}(cp_p) = \begin{cases} 1, & RT(p) \cdot c(p) > \text{angle threshold} \\ 1, & d_E(RT(p), c(p)) > k \, \text{mean distance} \\ 1, & \text{boundary}(c(p)) \\ 0, & \text{otherwise.} \end{cases} \tag{4.3}$$

We define the selected corresponding point set Q as the set of all corresponding points that are not pruned

$$Q = \{\, cp_p \mid cp_p \in C,\ \text{prune}(cp_p) = 0\}. \tag{4.4}$$

4.4 Distance Metric

To find a rigid transformation that optimally transforms a source point cloud to a target point cloud, we need a distance metric minimizable with respect to the rigid transformations. Given a source point set P and a target point set Q we search for a rigid transformation $RT = \{r, t\}$ that best fits the two point sets. The rigid transformation consists of a rotation $r \in so(3)$ and a translation $t \in \mathbb{R}^3$. There are two common distance metrics, the point-to-point and the point-to-plane metric. Often a weighted combination of both is used.

The most intuitive distance metric is the *point-to-point* error metric introduced by Besl and McKey [3] which sums up the euclidean distances between the rigidly transformed source points and the target points. The point-to-point metric is defined as

$$E_{\textit{point-to-point}} = \sum_{(p_i, q_i) \in Q} (Rp_i + t - q_i)^2 \tag{4.5}$$

where $R \in SO(3)$ is the rotation matrix in the Lie group corresponding to the Lie algebra $r \in so(3)$. The mapping from $so(3)$ to $SO(3)$ is given by the exponential map, the inverse by the logarithmic map.

One problem of the point-to-point error metric lies in the assumption that the points in the two point clouds are evenly distributed which is not always fulfilled. As visible in Figure 4.3 the error does not minimize the distance between the two surfaces abstracted by the point clouds instead it minimizes the distance between discrete points. Therefore the gradient resulting from point-to-point metric does not point in the optimal direction which leads to slow convergence.

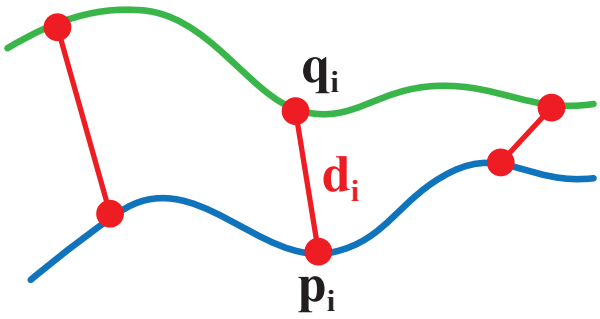


Figure 4.3: Visualizing of the point-to-point metric. The red line represents the distances between corresponding points.

A more robust error metric is the *point-to-plane* metric introduced by Chen and Medioni [4], which allows flat regions to slide along each other. The point-to-plane error metric minimizes the distance between the deformed source point and the plane spanned by the target point q_i and the target normal n_i. The point-to-plane error metric is defined as

$$E_{\textit{point-to-plane}} = \sum_{(p_i,q_i)\in Q} ((Rp_i + t - q_i) \cdot n_i)^2. \tag{4.6}$$

If the two point clouds are close to each other, the planes spanned by the target points are a good approximation of the true underlying surface. The gradient of the point-to-plane error points into the best direction to minimize the distance between the two surfaces defined by the point clouds as visualized in Figure 4.4. In general the point-to-plane error metric performs significantly better than the point-to-point metric [5].

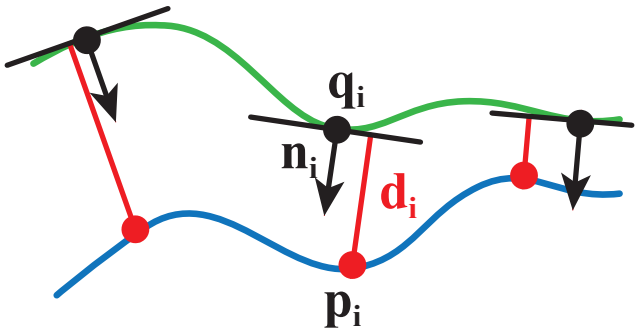


Figure 4.4: Visualizing of the point-to-plane metric. The red line represents the distances between the source point and the target plane.

One drawback of the point-to-plane error metric lies therein that it is not robust to featureless regions. It may fail due to the lack of constraints in flat regions which leads to poor convergence and a suboptimal registration resulting from too much sliding. For more robustness in large featureless regions the point-to-plane and the point-to-point error metric can be combined by using a weighted combination of both [36], leading to the following fit cost

$$E_{fit} = \alpha_{plane} E_{point\text{-}to\text{-}plane} + \alpha_{point} E_{point\text{-}to\text{-}point} \tag{4.7}$$

where α_{plane} and α_{point} are used to impose weights on the two error terms. A common weighting for the point-to-plane and point-to-point error metric is $\alpha_{plane} = 0.9$ and $\alpha_{point} = 0.1$.

5 Non-Rigid Surface Registration

This section describes how the ICP algorithm can be extended to the more general case of non-rigidly deforming surfaces. To do so we first introduce a surface deformation model which is able to deform any template surfaces model into arbitrary shapes. Then we describe the adjusted objective function which leads to an as rigid as possible deformation and conclude with a explanation how the objective function is solved. An example non-rigid surface registration is visualized in Figure 5.1

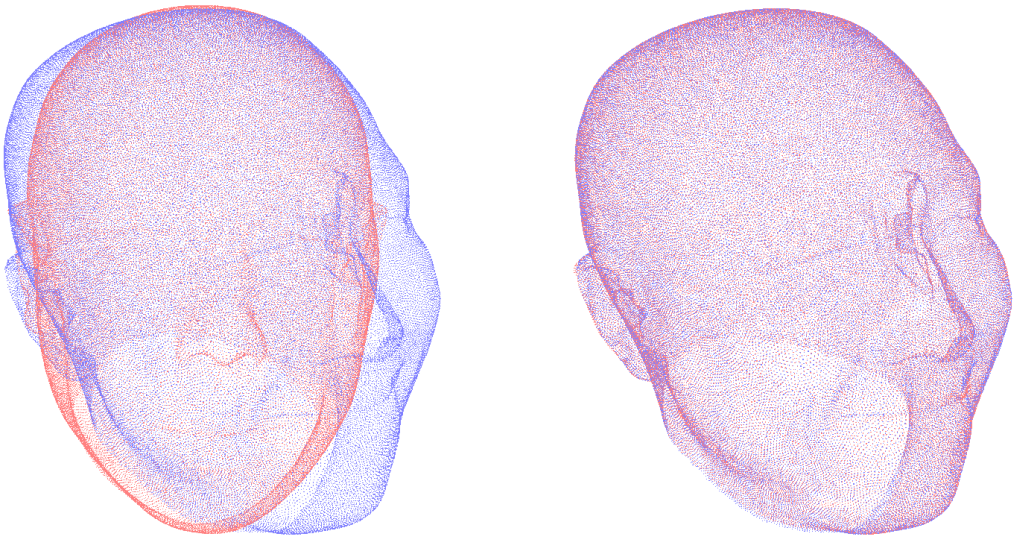


Figure 5.1: Non-rigid surface registration between two point clouds. On the left a source (red) and a target (blue) point cloud. On the right the two point clouds non-rigidly aligned.

5.1 Surface Deformation Model

The goal of non-rigid registration is to find a deformation that deforms a template surface model in a way that best fits a non-rigidly deformed target surface or point cloud. A template surface model can be any 3D model where we can sample points from. It is called template as we can create every deformed object from it by applying the surface deformation model on

it. By applying the identity deformation model on the template surface model the result will have the same form as the surface of the template surface model. We used in our implementation a simple point cloud as template surface model, however the representation could also be a volume model as done by Innmann et al. [2]. While it is possible to find for each vertex of a given template surface model an affine transformation that best fits the target surface, such an approach would lead to a high number of unknowns and therefore to a hard to solve and poor performing algorithm. To solve the problem we need a surface deformation model that leads to natural shape deformations and can express complex deformations of a variety of shapes. Also it should provide high-quality results, be robust to noise and efficient to fit.

To deform a template surface model we use the *deformation graph* as introduced by Sumner et al. [23] which is able to deform general objects in space by expressing complex deformations with smoothly blended affine transformations of the deformation graph nodes. The nodes of the deformation graph are distributed sparsely over the template surface model which leads to a small number of unknowns and therefore to an efficient algorithm. The collection of all node transformations expresses a non-rigid deformation of the template surface model and the deformation graph itself. In order to capture rigid deformations of the template surface model, which influence all nodes, a global rigid deformation is modeled separately to the non-rigid deformations. This enhances the performance of the algorithm as proven by Li et al. [35]. To deform a vertex first the non-rigid deformations and then the global rigid deformation are applied to the vertex. The unknowns of the deformation model are the global rigid transformation and the non-rigid transformations for each deformation graph node.

The global rigid transformation $x_{global} = \{g, r, t\}$ consists of a position $g \in \mathbb{R}^3$ denoting the center of mass of the template surface model and a rigid transformation consisting of a rotation $r \in so(3)$ and a translation $t \in \mathbb{R}^3$. The deformation graph $G = \{x_j \mid j = 1...n_g\}$ consists of n_g nodes, where each node x induces an affine transformation on the nearby space. A node $x = \{g_x, r_x, t_x\}$ consists of a node position $g \in \mathbb{R}^3$ a rotation $r \in so(3)$ and a translation $t \in \mathbb{R}^3$. To deform a point cloud with a given deformation graph each vertex $v_i \in \mathbb{R}^3$ of the point cloud $V = \{v_i \mid i = 1...n_v\}$ is deformed with a smooth blending of the deformations of all deformation graph nodes that have influence on the vertex. The deformed vertex v_i' is defined as

$$
\begin{aligned}
v_i' &= \phi_{global} \circ \phi_{local}(v_i) \\
\phi_{global}(v_i) &= R(v_i - g) + g + t \\
\phi_{local}(v_i) &= \sum_{j=1}^{n_g} w_j(v_i)[R_j(v_i - g_j) + g_j + t_j]
\end{aligned}
\tag{5.1}
$$

where $R \in SO(3)$ corresponds to the Lie group of the Lie algebra $r \in so(3)$ and $w_j(v_j)$ denotes to the influence of node x_j on the vector v_i. The deformation of the template surface model's

normals is defined analogous by

$$
\begin{aligned}
n_i' &= \phi_{global} \circ \phi_{local}(n_i) \\
\phi_{global}(n_i) &= {R^{-1}}^{\top} n_i \\
\phi_{local}(n_i) &= \sum_{j=1}^{n_g} w_j(v_i)[R_j^{-1\top} n_i].
\end{aligned}
\tag{5.2}
$$

As the function describes a linear blending of the deformations, the weights related to one vertex need to sum up to one

$$
\sum_{j=1}^{m} w_j(v_j) = 1. \tag{5.3}
$$

As the influence of a node on the deformation of a vertex depends on the weighting function $w_j(v_i)$, the choice of the weighting function has a relevant influence on the smoothness of the deformation. In the original paper of Sumner et al. [23] a regular deformation graph is used where each vertex deformation is influenced by its k-nearest nodes. The used weighting function is $w_j(v_i) = (1 - d_E(v_i, g_i)/d_{max})^2$ where d_{max} is the distance to the $k+1$-nearest node. As this weighting function results in some artifacts for extremely coarse graphs or graphs with irregular node distances Li et al. [36] changed the weighting function to a radius of influence, which is more robust to irregular deformation graphs. The weighting function of Li et al. which is also used in our implementation is

$$
w_j(v_i) = \max\left(0, \left(1 - \frac{d(v_i, g_j)^2}{{r_j}^2}\right)^3\right) \tag{5.4}
$$

where d denotes a distance metric between v_i and g_j and r_j denotes the radius of the influence region of node x_i. While Li et al. use a geodesic distance on the template surface model, we used the euclidean distance as done by Sumner et al. [23]. The radius of influence for uniformly distributed nodes is usually chosen as the average distance between nodes and is set to the same value for all nodes. In general the influence radius can differ between nodes, which is relevant for non-uniform node distributions.

Besides the nodes itself, the deformation graph also consists of connections between nodes also called edges which are used for the regularization of the deformation. There exist different strategies how the nodes are connected. A common approach is to connect all nodes with influence on the same vertex of the template surface model. The nodes can also be connected by triangulation of the nodes or by creating hierarchical connections between nodes [1]. The initialization of the deformation graph is described in Section 5.3.

5.2 Non-Rigid Registration

Given the template surface model, the related deformation graph and a target point cloud of a deformed object, we want to solve for all unknown rigid and non-rigid transformations of an objective function that fits the template surface model with the corresponding deformation graph to the target point cloud. Therefor we need a data term similar to the data term used in the ICP algorithm, as described in Chapter 4, which fits the template surface model to the target point cloud. Additionally a regularization term is used which encourages the transformations to be as rigid as possible. This is necessary due to the high degree of freedom of the non-rigid deformation, which would otherwise lead to unexpected deformations. Without the regularization the algorithm would find a deformation that fits the template to the target in an unexpected way, by squeezing and distorting the template. This would also make it impossible to track smooth realistic deformations of the object. We solve for an optimal non-rigid deformation by iteratively solving for closest point correspondences with the current deformation graph and updating the deformation graph by solving the non-rigid objective functions.

5.2.1 Correspondence Finding

Correspondence finding for non-rigid registration is very similar to the rigid registration case, as described in Section 4.1, 4.2 and 4.3. We first select a subset of points from the template surface, find the corresponding points in the target point cloud and remove all corresponding pairs that are classified as outliers. But instead of finding the closest target point to the rigidly deformed source point we search for the closest target point to the non-rigidly deformed point of the template surface model. The corresponding point for the non-rigid case is defined as

$$c_{non\text{-}rigid}(v) = \arg\min_{t \in T} d_E(v', t) \tag{5.5}$$

where d_E as before denotes the euclidean distance. The set $C_{non\text{-}rigid}$ of corresponding non-rigidly deformed template surface model points and target points is defined as

$$C_{non\text{-}rigid} = \{\ (v',\ q_i \mid v \in P,\ q_i \in T,\ \text{prune}(q_i) = 0\ \} \tag{5.6}$$

with $q_i = c_{non\text{-}rigid}(v)$ the corresponding target point.

5.2.2 Data Term

The data term is similar to the ICP data term of Equation 4.7 but instead of deforming the vertices of the template surface model with a rigid deformation, we deform it with a non-rigid deformation according to Equation 5.1. The resulting fit cost is

$$E_{fit} = \alpha_{plane} \sum_i ((v_i' - q_i) \cdot n_i)^2 + \alpha_{point} \sum_i (v_i' - q_i)^2. \tag{5.7}$$

where n_i is the normal of the corresponding target point q_i.

5.2.3 Regularization Term

Our goal is to find a surface deformation that deforms the object in an as rigid as possible way. Given two connected node positions in space g_i and g_j a deformation of both positions is rigid if the edge between the two points deforms in a rigid way. It follows that if the deformation of two connected positions is rigid there exists a rotation matrix R_i such that

$$g'_i - g'_j = R_i(g_i - g_j), \quad \forall j \in \mathcal{N}(i) \tag{5.8}$$

where $\mathcal{N}$ is the neighborhood of i. The deformation does not necessarily need to be rigid, however the rigidity condition can be formulated as a minimization problem. The as-rigid-as-possible minimization problem as introduced by Sorkine et al. [22] which minimizes the deviation from rigid deformation is defined as

$$E_{arap} = \sum_i \sum_{j \in \mathcal{N}(i)} ||R_i(g_i - g_j) - (g'_i - g'_j)||^2. \tag{5.9}$$

The problem can be solved by finding the optimal rotation matrix R_i. As not all matrices lead to a valid rotation matrix the regularization term is solve for a valid rotation by finding the corresponding Lie algebra $r_i \in so(3)$ to the Lie group $R_i \in SO(3)$. By doing so the dimensionality of unknowns is reduced from 9 to 3 for each matrix. The regularization term is often called smoothness term as it describes the smoothness of the deformation.

An alternative version of the smoothness energy is to solve the term for a general matrix and use an additional regularization term to minimize the deviation of the matrix structure from a valid rotation matrix. This type of regularization term was introduced by Sumner et al. [23] and is usually referred as *embedded deformation* regularization. The term is defined as

$$E_{ed} = \sum_i \sum_{j \in N(i)} ||M_i(v_i - v_j) - (v'_i - v'_j)||^2 \tag{5.10}$$

where $M_i \in \mathbb{R}^{3x3}$ refers to a affine transformation matrix. To represent a valid rotation matrix each of the three columns in the matrix must be unit length and all columns must be orthogonal to one another [42]. The deviation from this condition is given by

$$\begin{aligned} Rot(M) =& (c_1 \cdot c_2)^2 + (c_1 \cdot c_3)^2 + (c_2 \cdot c_3)^2 + \\ & (c_1 \cdot c_1 - 1)^2 + (c_2 \cdot c_2 - 1)^2 + (c_3 \cdot c_3 - 1)^2 \end{aligned} \tag{5.11}$$

where c_1, c_2 and c_3 denotes to the column vectors of M. The rotational error over all rotations is minimized using

$$E_{rot} = \sum_i Rot(M_i). \tag{5.12}$$

The regularization term for embedded deformation is defined as

$$E_{reg} = E_{ed} + \alpha_{rot} E_{rot} \tag{5.13}$$

the combination of both terms. As regularization term either the as rigid as possible regularization $E_{smooth} = E_{arap}$ or the embedded deformation regularization $E_{smooth} = E_{reg}$ can be used.

5.2.4 Optimization

The optimal non-rigid deformations of the deformation graph are solved by using the non-rigid ICP algorithm. The used objective function consists of the data term which fits the model with the point-to-point and point-to-plane term and the regularization term which leads to an as rigid as possible deformation. The energy to minimize is therefore

$$E = \alpha_{fit} E_{fit} + \alpha_{smooth} E_{smooth} \tag{5.14}$$

where α_{smooth} is the weighting of the data term E_{fit} and E_{smooth} correspond to the used regularization term with α_{smooth} to the weighting of the regularization term.

For each template surface to target alignment, the registration is initialized with an identity deformation graph. Then the algorithm alternates between correspondence finding and solving for the objective function. If the relative total energy stagnates between two iterations

$$\frac{|E_{k+1} - E_k|}{E_k} < \mu, \tag{5.15}$$

with k the number of iterations and μ the minimum change between iterations, or the number of iterations is higher than a maximum value the algorithm terminates. The non-rigid ICP algorithm is described in Algorithm 2.

Algorithm 2: Non-rigid Iterative closest point

Input : Template surface model
Corresponding deformation graph
Target point cloud
Output: Best fitting rigid x_{global} and non-rigid deformations G
. **while** *not converged* **do**
  Select points from the template surface model
  for *each of the selected points* **do**
    Find corresponding closest *target* points to the deformed template surface model.
    Remove outliers.
  end
  x_{global}, $G \leftarrow$ minimize objective function for corresponding point pairs
end

5.3 Initialization of the Deformation Graph

The structure of the deformation graph should be similar to the underlying template surface model, to be able to correct express and deform the template surface model. Therefore the

nodes of the deformation graph are created by sampling node positions from the template surface model with a given sample distance. The intuitive assumption that the deformation graph that best fits the curvature of the surface is optimal is not generally true as some parts of non-rigid objects usually deform more than others independent from the curvature of the surface e.g. the ears of a human have fine structures with high curvature but the deformations are small in contrast to the mouth of a human which performs many different deformations to show facial expressions.

Often poisson disk sampling [43] is used for the sampling of nodes. Poisson disk sampling produces points that are at least a specified minimum distance apart but densely packed. Another possibility is to create the deformation graph by approximating the surface model through remeshing. To create a uniformly distributed deformation graph with an equal edge length we used isotropic remeshing as introduced by Botsch and Kobbelt [44].

5.4 Sequence of Frames Registration

To apply non-rigid registration to a series of point clouds we use the deformation graph introduced in Section 5.1 to deform a given template surface model to each frame of a sequence of deformed target point clouds. As a result the algorithm remembers for each frame the optimal deformation of the deformation graph. To fit the template model to each point cloud of the sequence, we perform for each target point cloud first a rigid surface fitting followed by a non-rigid surface fitting. We initialized the non-rigid surface fitting with the rigid deformation of the rigid registration of the current and the optimized non-rigid deformation of the previous frame. The rigid registration is also initialized with the rigid part of the non-rigid registration. The deformation graph and the rigid transformation of the first point cloud is initialized with the identity deformation. An overview of the non-rigid reconstruction algorithm is given in Algorithm 3.

Algorithm 3: Non-rigid reconstruction

Input : Template surface model
Sequence of target point clouds
Output: Best fitting rigid x_{global} and non-rigid deformations G

$x_{global} \leftarrow$ initialize with identity transformation.
$G \leftarrow$ create deformation graph using template surface model.
for *each frame in sequence of frames* **do**
  $x_{global} \leftarrow$ optimize using iterative closest point.
  $x_{global}, G \leftarrow$ optimize using non-rigid iterative closest point
end

6 Variants of Non-Rigid Surface Registration

In this section we will introduce different approaches to improve the robustness and fitting of non-rigid surface registration by adapting the underlying deformation graph in various ways. As regions with high smoothness energy E_{smooth} indicate areas with contradicting or intense surface deformation, we use this information and adapt the graph to better fit the surface. Different approaches for improving the deformation graph are explained in the following.

6.1 Deformation Graph Refinement

To make the deformation graph more adaptive to different degrees of freedom, Li et al. [36] replaced the uniform graph with an adaptive node distribution. Therefore they start with a coarse deformation graph which is refined in regions with intense deformations.

To create a refinable graph a hierarchical deformation graph is pre-computed where each level corresponds to a different level of detail representation of the underlying template surface model. The finest level L_{max} of the hierarchical deformation graph is created by densely and uniformly sampling of positions on the template surface model. The deformation graph nodes of the finest level are initialized with the sampled positions and the identity deformation. To create the next coarser level $l_i \in L_{max}...L_{min}$ the nodes of the current level l_{i-1} are subsampled by increasing the average sampling distance $r_l = 2r_{l-1}$. Li et al. used in contrast to us an increasing average sampling distance of $r_l = 4r_{l-1}$. Poisson disk sampling is used to create an uniform node distribution in each level. Each of the nodes x_j^l from level l form a cluster C_j^l which contains every node from the next finer level x_j^{l+1} that is not closer to any other cluster from l. The deformation graph can be refined at a node x_j^l by inserting all nodes of the corresponding cluster C_j^l. This leads to a deformation graph which can be adaptively refined.

The initial deformation graph starts with the coarsest level L_{min}. To dynamically adapt the graph in regions with high deformations, the deformation graph is refined by inserting nodes in regions with high regularization residual E_{smooth}. High regularization residuals indicate regions with contradicting or intense surface deformations and occur in regions where the deformation graph is not able to fit the target surface sufficiently. The nodes of an edge in the graph are refined if the edge regularization residual is larger then a percentage ρ of the highest regularization residual of all edges and the regularization term is bigger then a

minimal threshold λ. Formally a node x is refined if

$$\text{residual}(e_x) > \rho \max_{e \in edges(g)} (\text{residual}(e)) \ \wedge \ E_{smooth} > \lambda \tag{6.1}$$

where e_x is one of the adjacent edges of the node x. Li et al. set the value of ρ to 0.1 and the value of λ to 0.01.

The graph is refined by inserting for each node x_j^l, which should be refined, all nodes of the corresponding clusters C_j^l into the graph. The deformations of all new nodes are updated based on the influence of the surrounding nodes. To calculate the deformation of a new node, the weights of all neighbor nodes with influence on the node are calculated using Equation 5.4. To calculate the rotation, the rotations of all nodes with influence are linear interpolated. The new node rotation is achieved by

$$r_{\text{new node}} = \sum_{i \in G} w_i r_i. \tag{6.2}$$

For the translation the node is simply deformed and the vector from the original node to the deformed node is used as transformation. The transformation is updated by

$$t_{\text{new node}} = g'_{\text{new node}} - g_{\text{new node}}. \tag{6.3}$$

After inserting all new nodes the edge connections of the deformation graph are recalculated.

We introduce a variant of the mesh refinement strategy of Li et al.. Instead of refining nodes based on the adjacent edge residual, nodes are refined if the average regularization residual of all adjacent edges $avg\_res(x)$ of the node are higher than a percentage ρ of the highest average regularization residual per node

$$\text{avg residual}(x) > \rho \max_{x \in G}(\text{avg residual}(x)) \ \wedge \ E_{smooth} > \lambda. \tag{6.4}$$

All other parts of the refinement algorithm of Li et al. stay the same.

6.2 Smoothness Reduction

One key element of optimal non-rigid deformation fitting lies in the used smoothness coefficient of the objective function. The used α_{smooth} coefficient needs a good balance between forcing the deformation to be as rigid as possible and allowing the necessary deformation to fit the target. A heuristic introduced by Li et al. [35] finds an optimal weight by simply starting with a high smoothness weight and reducing the weight accompanied with the convergence of the fitting. This allows the deformation graph to first find the roughly correct deformation with a more rigid constraint which is relaxed over time to fit the true deformations.

This very simple but efficient heuristic automatically adapts the rigidity of the graph by reducing the smoothness coefficient α_{smooth} by halve of its value whenever the error converges. More formally α_{smooth} is defined as

$$\alpha_{smooth}{}^{k} = \begin{cases} 0.5\ \alpha_{smooth}^{k-1} & , \frac{|E_k - E_{k-1}|}{E_k} < \gamma\ \wedge\ a_{smooth} > \tau \\ \alpha_{smooth}^{k-1} & , \text{otherwise} \end{cases} \tag{6.5}$$

where k is the current iteration of the ICP algorithm and E_k is the residual of the current objective function at iteration k. The weighting coefficient for the data term α_{fit} stays constant. The values for γ and τ are set by Li et al. to $\gamma = 10^{-5}$ and $\tau = 0.005$. This heuristic leads to an initial globally more rigid alignment where the stiffness of the object is subsequently lowered leading to an increase of the deformability of the deformed surface template and therefore to a better fit.

6.3 Rigidity Reduction

We introduce a new strategy which is similar to the reducement of the smoothness factor as described in section 6.2 but instead of reducing the smoothness coefficient, we reduce a rigidity coefficient per edge. This allows the strategy to model the kinematics of the deforming object. To do so we introduce a new factor for each edge called $e_{rigidity}$ which describes the rigidity of the edge. We adapt the regularization term of Equation 5.9 to

$$E_{rigidity\ reduction} = \sum_i \sum_{j \in N(i)} e_{rigidity_i}{}^{k} ||R_i(g_i - g_j) - (g_i' - g_j')||^2 \tag{6.6}$$

were $e_{rigidity_i}{}^{k}$ refers to the edge e_i of the ICP iteration k. If the objective function stagnates

$$\frac{|E_k - E_{k-1}|}{E_k} < \nu \tag{6.7}$$

where ν denotes the minimum change between iterations,the edge rigidity for all edges with a smoothness residual below a given threshold η and an edge rigidity larger than a given minimal value ζ

$$e_{rigidity_i}{}^{k} = \begin{cases} 0.5\ e_{rigidity_i}{}^{k-1} & , \text{residual}(e_i)^k > \eta\ \wedge\ e_{rigidity_i}^{k} > \zeta \\ e_{rigidity_i}{}^{k-1} & , \text{otherwise} \end{cases} \tag{6.8}$$

is reduced. For sequence registration we reset the edge rigidity coefficient for each frame as we want for each frame a more rigid registration first, followed by an relaxation of the rigidity constrain to fit deformations.

6.4 Rigidity Adaption

Instead of just reducing a rigidity weight per edge, we want to solve for an optimal rigidity weight per edge. To do so we introduce *Adaptive Rigidity* where we simultaneously solve

for deformations and optimal rigidity weights per edge. We adapt the objective function by changing the smoothness term to be scaled with the edge rigidity weight $e_{rigidity}$. The adapted regularization term of the objective function is

$$E_{adaptive\ rigidity} = \sum_i \sum_{j \in N(i)} ||{e_{rigidity}}^{ij}[R_i(g_i - g_j) - (g'_i - g'_j)]||^2 \tag{6.9}$$

where ${e_{rigidity}}^{ij}$ is the rigidity edge weight of the edge connecting the nodes i and j.

As the minimal energy of this term would lead to a rigidity weight of zero we need a regularization term which minimizes its energy when the rigidity weight of each edge is near one leading to an as rigid as possible deformation of the graph edges. As rigid weight regularization we use the simple quadratic term per edge

$$E_{rigidity\ per\ edge}(e) = \begin{cases} ||1 - e_{rigidity}||^2 & , e_{rigidity} \leq 1 \\ 0 & , \text{otherwise} \end{cases} \tag{6.10}$$

the edge rigidity regularization energy is therefore

$$E_{rigidity} = \sum_{e \in edges} E_{rigidity\ per\ edge}(e). \tag{6.11}$$

The adapted objective function with the adaptive rigidity smoothness term and an additional rigidity weight regularization is

$$E = \alpha_{fit} E_{fit} + \alpha_{smooth}(E_{adaptive\ rigidity} + \alpha_{rigidity} E_{rigidity}) \tag{6.12}$$

where $\alpha_{rigidity}$ denotes the edge rigidity weight coefficient. As the edge rigidity weight $e_{rigidity}$ occurs in both the smoothness and the edge weighting regularization term, the edge weighting regularization term needs to scale with the dimensions of the smoothness weighing α_{smooth}. We pay attention to this by scaling $\alpha_{rigidity}$ with the smoothness coefficient α_{smooth}.

Instead of searching for the optimal rigidity values per edge, we can also search for the optimal rigidity values per node. As an edge is always defined by the connection between two nodes the edge weight can be defined as

$${e_{rigidity}}^{ij} = \frac{({x_{rigidity}}^i + {x_{rigidity}}^j)}{2} \tag{6.13}$$

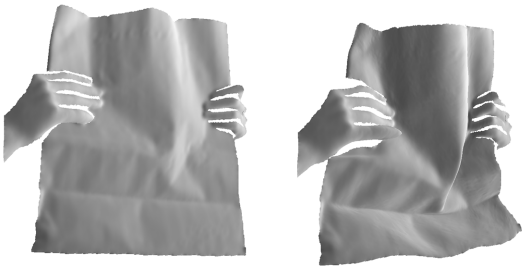
where ${x_{rigidity}}^i$ is the rigidity weight of node i that we optimize instead of the rigidity edge weight itself, we simply replace the edge rigidity in Equation 6.10 with Equation 6.13.

7 Results

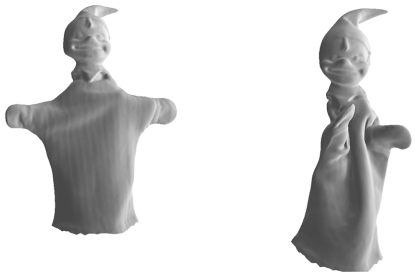
In this section we report the results of the previously introduced non-rigid surface registration variants as described in Chapter 6 and compare them against each other. Also we report the achieved improvement of combinations of the introduced variants.

7.1 Data Set

We apply the non-rigid surface registration variants to the datasets from Li et al. [36] which consists of four point cloud sequences, the *puppet*, the *paperbag*, the *head* and the *hand*. The datasets are visualized in Figure 7.1.

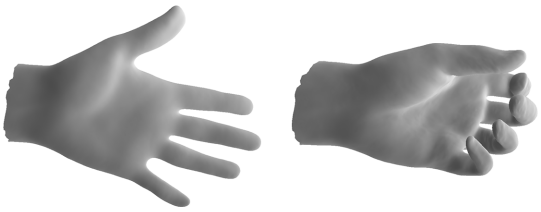


(a) Screenshots from the paperbag sequence.



(b) Screenshots from the puppet sequence.

(c) Screenshots from the head sequence.



(d) Screenshots from the hand sequence.

Figure 7.1: Sample Images of the dataset from Li et al. [36].

For simplicity we only compare the methods on the reconstructed datasets and not on the more challenging depth scan datasets.

7.2 Implementation Details

We implemented the non-rigid registration algorithm and all variants in C++ using the *Ceres Solver* [45] library for solving the different introduced objective functions. In addition we used CGAL [46] as mesh and point cloud processing library and *mLib* [47] for input and visualization handling. The algorithm ran on a common hardware with a Ryzen 5 1600 X CPU and 16 GB Memory. We published the implementation on GitHub [48].

7.3 General Setup

Our *baseline* is the non-rigid registration algorithm with the as-rigid-as-possible regularization term $E_{smooth} = E_{arap}$ as described in Algorithm 3. The template surface model is initialized with the first frame of the sequence. The nodes and edges of the graph are created by the result of the isotropic remeshed template surface model as described in Section 5.3. As edge length for the remeshing we used

$$\text{edge length} = l\sqrt{\text{area}(V)} \tag{7.1}$$

where $\text{area}(V)$ is the surface area of the template surface model V and l a hyperparameter which is set for all datasets to 0.095. This makes the edge length relative to the area of the template model and leads to an edge length which takes the dimension of the dataset into account. We used random sampling as point selection strategy as described in Section 4.1 and the euclidean distance for correspondence finding as introduced in Section 5.2.1. Corresponding point pairs are pruned if the euclidean distance is larger than 0.1, the normals differ more than 45 degrees or one of the corresponding points is a boundary point, a more detailed explanation can be found in Section 4.3. For all experiments and datasets we set α_{fit} to 1, α_{smooth} to 3, the term for $\alpha_{point\text{-}to\text{-}point}$ to 0.1 and $\alpha_{point\text{-}to\text{-}plane}$ to 0.9.

The configuration of the non-linear least squares solver *Ceres Solver* is the same in all our experiments. We used the trust region minimizer levenberg-marquardt with the conjugate gradient solver on the normal equation and Jacobi as preconditioner.

7.4 Non-Rigid Registration Variants

We test the baseline against all variants introduced in Chapter 6. The first variant *Deformation Graph Refinement* as described in Section 6.1, refines the deformation graph by inserting new nodes in regions with high deformations. We implemented two variants, either we refine both adjacent vertices to an edge with high smoothness residual or we refine a vertex based on the average smoothness residual cost of the adjacent edges. We will call the strategy that refines the two adjacent vertices to an edge in the following *Refinement at Edge* and the other variant *Refinement at Vertex*. To create the initial hierarchical mesh we used for $r_{L_{max}}$ a radius of $0.02\sqrt{\text{area}(V)}$ and created 3 hierarchical levels. For both variants the threshold for

refinement is set to $\lambda = 0.01$ and the percentage ρ of the highest regularization residual to 0.9. In addition we implemented *Smoothness Reduction* as described in Section 6.2 where the smoothness coefficient α_{smooth} is reduced whenever the objective residual stagnates. In our experiments we set γ to 0.01 and τ to 0.01. The next variant is our *Rigidity Reduction* algorithm as introduced in Section 6.3 where the rigidity of an edge is reduced if the edge smoothness residual is high. We reduce the rigidity of an edge if the difference of the objective function residual between two iterations is below $\nu = 0.1$, the edge residual residual$(e) > 0.01$ and the edge rigidity $e_{rigidity} > 0.001$. Furthermore we compare the baseline against the *Adaptive Rigidity* variant as introduced in Section 6.4, where the objective function simultaneously optimizes for optimal deformation and edge rigidity. We used as hyperparameter for the rigidity regularization coefficient $\alpha_{rigidity} = 0.01$. We have here also two variants, either we optimize a rigidity value per edge or per vertex. In the following, the two variants are called *Adaptive Rigidity at Edge* and *Adaptive Rigidity at Vertex*.

To compare the different variants against each other we used the normalized chamfer distance of the registered frames. The normalized chamfer distance [49] between two point clouds is defined as

$$d_{\text{chamfer distance}}(V', T) = \frac{1}{n_{V'} + n_T} \left(\sum_{v' \in V'} \min_{t \in T} ||v' - t||_2^2 + \sum_{t \in T} \min_{v' \in V'} ||v' - t||_2^2 \right) \tag{7.2}$$

where V' is the deformed template model with $n_{V'}$ vertices and T is the target point cloud of the current frame with n_T vertices.

Figures 7.2, 7.3, 7.4 and 7.5 show the mean chamfer distance of the different variants over the sequence of point clouds of the four datasets *paperbag*, *puppet*, *head* and *hand*. The baseline strategy as-rigid-as-possible without any adaption is visualized in red in each plot. To quantify the performance of the variants we use the relative error over the as-rigid-as-possible baseline. The relative error is defined as

$$\text{relative error}(v, d) = \frac{\text{mean}(v, d)}{\text{mean}(\text{baseline}, d)} \tag{7.3}$$

with

$$\text{mean}(v, d) = \frac{1}{n_f} \sum_{T \in Sequence} d_{\text{chamfer distance}}(V'_T, T) \tag{7.4}$$

the average chamfer distance over all frames of a dataset d and a given variant v, where n_f is the number of frames in the sequence and V'_T is the template model deformed to frame T of the sequence. The relative error of the different variants over the baseline are shown in Tables 7.1, 7.2, 7.3 and 7.4.

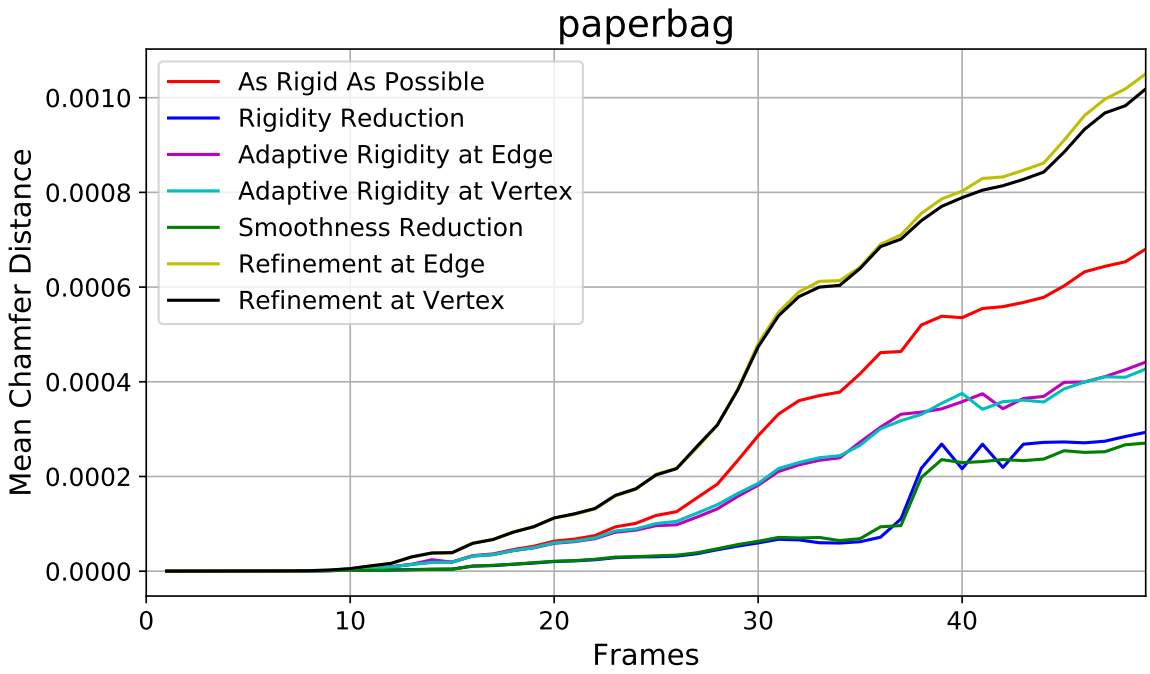


Figure 7.2: Mean chamfer distance over paperbag sequence.

Variant	relative error	variance in 10^{-4}	relative error per node	n_g
As-Rigid-As-Possible	1.000	0.9296	1.000	198
Rigidity Reduction	0.3520	0.9976	0.3520	198
Adaptive Rigidity at Edge	0.6694	0.9133	0.6694	198
Adaptive Rigidity at Vertex	0.6680	0.9061	0.6680	198
Smoothness Reduction	**0.3371**	**0.9483**	**0.3371**	**198**
Refinement at Edge	1.5579	1.2254	1.6444	111 - 209
Refinement at Vertex	1.5289	1.1851	1.2200	111 - 158

Table 7.1: Relative error of different variants on the paperbag sequence.

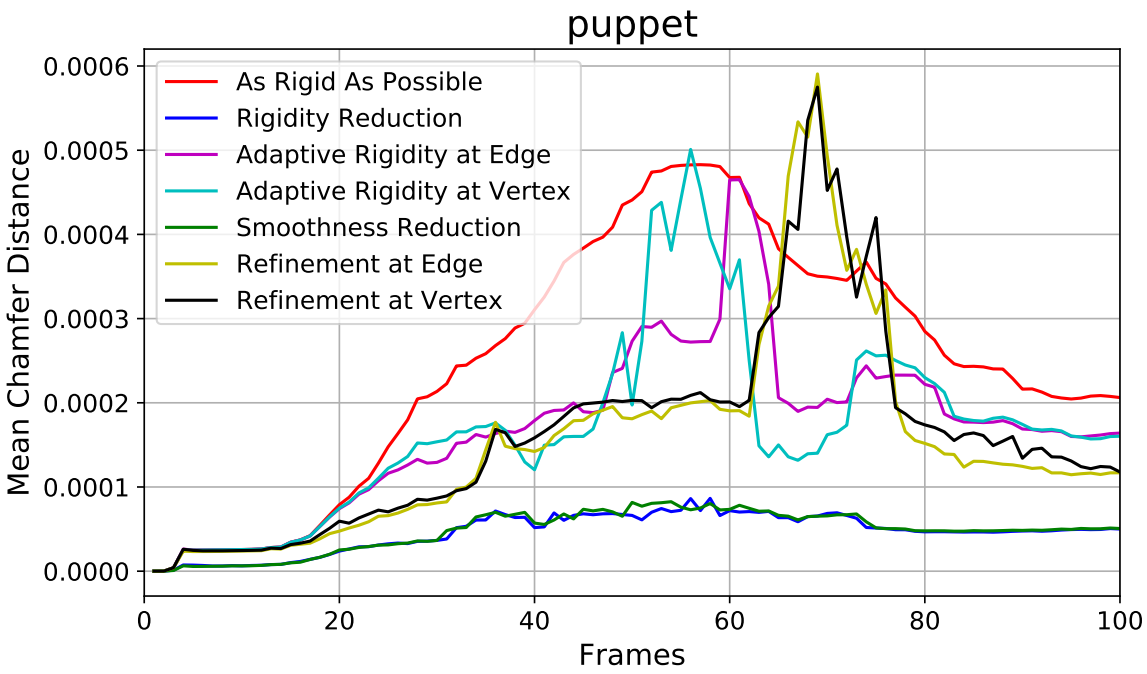


Figure 7.3: Mean chamfer distance over puppet sequence.

Variant	relative error	variance in 10^{-5}	relative error per node	n_g
As-Rigid-As-Possible	1.000	7.4026	1.000	134
Rigidity Reduction	**0.1840**	**1.5414**	**0.1840**	**134**
Adaptive Rigidity at Edge	0.6742	5.4682	0.6742	134
Adaptive Rigidity at Vertex	0.6698	5.3529	0.6698	134
Smoothness Reduction	0.1897	1.6378	0.1897	134
Refinement at Edge	0.6101	4.5540	5.0032	90 - 1421
Refinement at Vertex	0.6399	4.4431	3.8480	90 - 987

Table 7.2: Relative error of different variants on the puppet sequence.

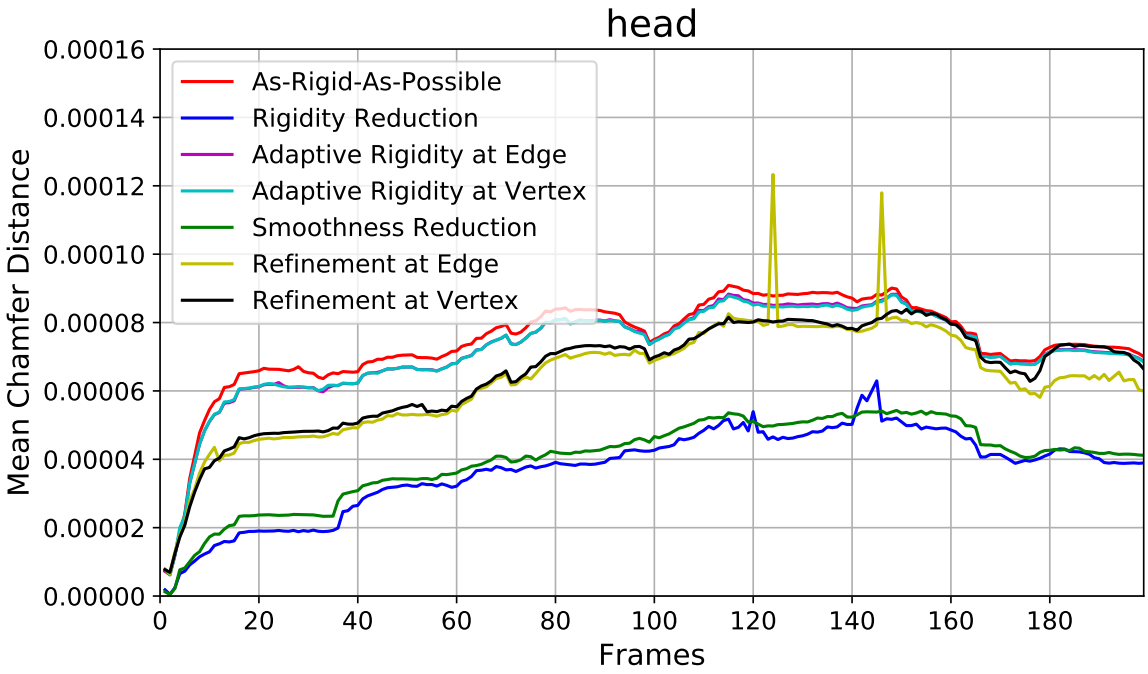


Figure 7.4: Mean chamfer distance over head sequence.

Variant	relative error	variance in 10^{-5}	relative error per node	n_g
As-Rigid-As-Possible	1.000	2.2185	1.000	145
Rigidity Reduction	**0.4991**	**1.3001**	**0.4991**	**145**
Adaptive Rigidity at Edge	0.9668	2.1298	0.9668	145
Adaptive Rigidity at Vertex	0.9652	2.1206	0.9652	145
Smoothness Reduction	0.5371	1.3247	0.5371	145
Refinement at Edge	0.8521	1.8067	2.9831	97 - 1192
Refinement at Vertex	0.8758	1.7403	2.4175	97 - 528

Table 7.3: Relative error of different variants on the head sequence.

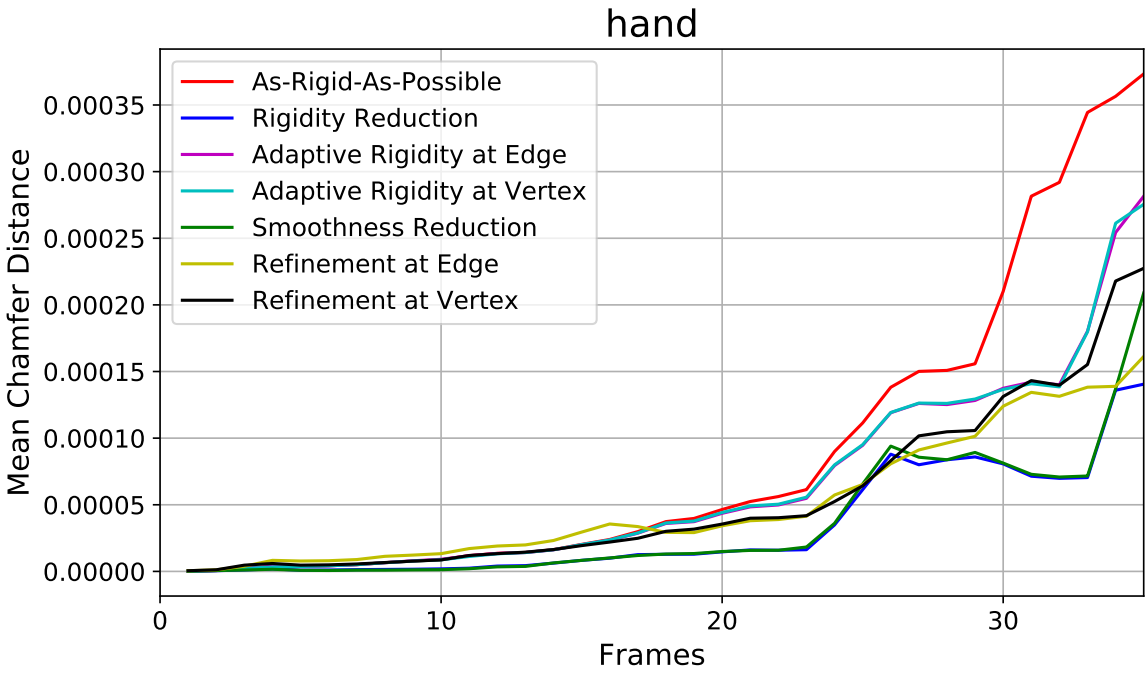


Figure 7.5: Mean chamfer distance over hand sequence.

Variant	relative error	variance in 10^{-5}	relative error per node	n_g
As-Rigid-As-Possible	1.000	4.5523	1.000	148
Rigidity Reduction	**0.3689**	**2.9684**	**0.3689**	**148**
Adaptive Rigidity at Edge	0.7209	3.4296	0.7209	148
Adaptive Rigidity at Vertex	0.7216	3.4653	0.7216	148
Smoothness Reduction	0.3984	3.1008	0.3984	148
Refinement at Edge	0.5720	2.3321	2.4034	88 - 741
Refinement at Vertex	0.6149	2.8023	3.1610	88 - 790

Table 7.4: Relative error of different variants on the hand sequence.

To quantify the performance of the variants over all datasets we define the total relative error over the baseline as

$$\text{total relative error}(v) = \frac{1}{N} \sum_{d \in datasets} \text{relative error}(v, d) \tag{7.5}$$

with N the number of datasets, in our case four. The total relative errors of all variants are shown in Table 7.5.

Variant	relative error	relative error per node
As Rigid As Possible	1.0	1.0
Rigidity Reduction	**0.3510**	**0.3510**
Adaptive Rigidity at Edge	0.7578	0.7578
Adaptive Rigidity at Vertex	0.7562	0.7562
Smoothness Reduction	0.3656	0.3656
Refinement at Edge	0.8980	3.0085
Refinement at Vertex	0.9149	2.6616

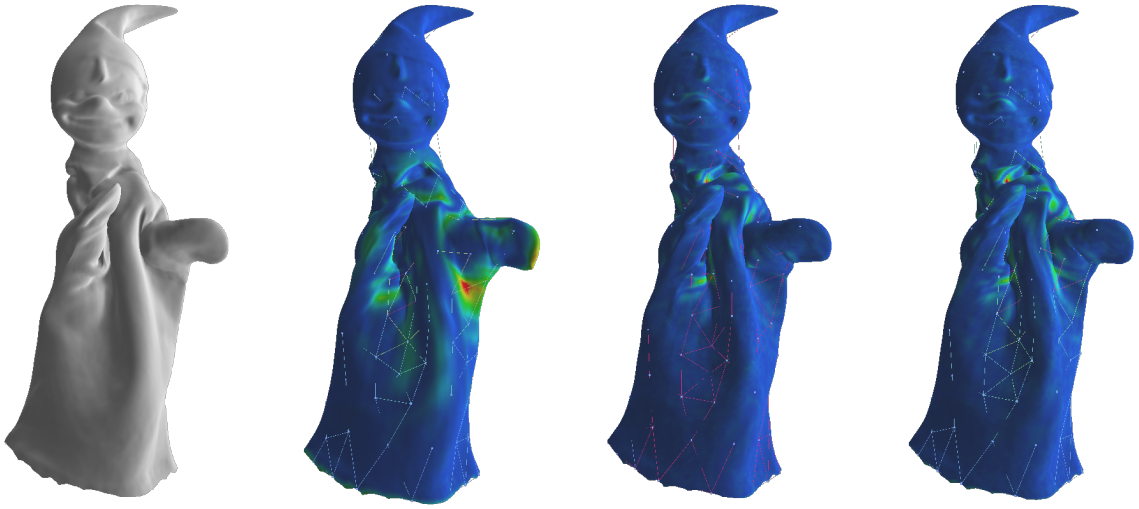
Table 7.5: Total relative error of different variants over all datasets.

Table 7.5 shows that *Rigidity Reduction* is the overall best performing variant with the best fitting in all four datasets. The images in Figure 7.6 and Figure 7.7 reflect the same statement. The puppet registered with Rigidity Reduction best represents the true deformation and Rigidity Reduction is also the only variant able to kink the small finger of the hand in the correct direction.

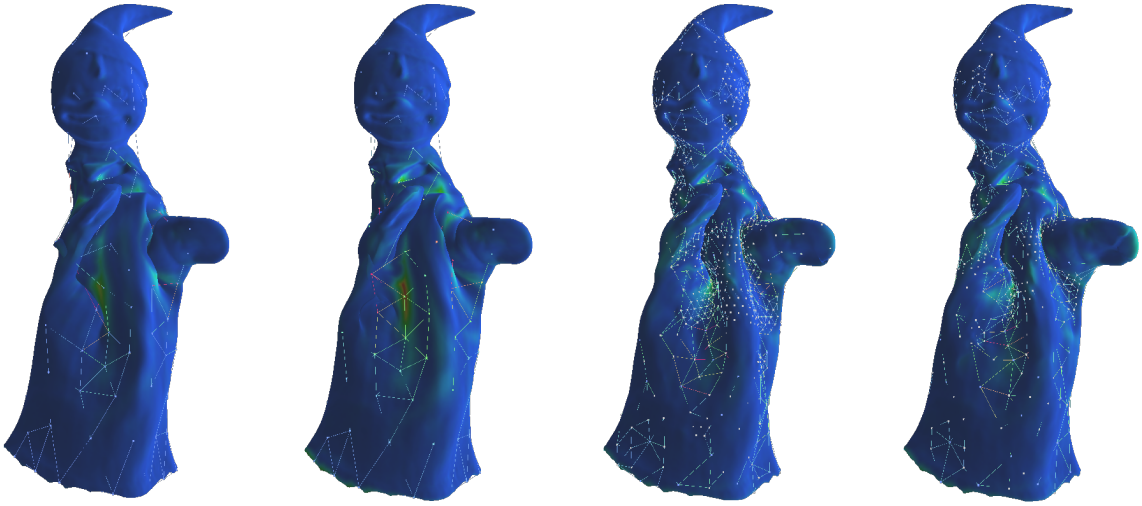
Smoothness Reduction is the second best performing variant and performs almost as good as Rigidity Reduction, for single frames sometimes even better. This is for example visible in the paperbag chamfer distance plot shown in Figure 7.2. In contrast to Rigidity Reduction where the edge weights are reduced for single edges with high smoothness loss, Smoothness Reduction reduces the smoothness coefficient for the complete graph. This supports our assumption that finding an optimal rigidity weight per edge results in a deformation graph which is more rigid in regions without or with less deformations and therefore leads to a fitting that better represents the true kinematics of the deformed object. By comparing Figure 7.7c and Figure 7.7d showing the registration results of the last frame of the hand, we see that the small finger of the hand in the smoothness reduction variant tends to an curvature while the rigidity reduction performs more a kink.

The two *Adaptive Rigidity* strategies *on Edge* and *on Vertex* provide relevant improvement over the baseline. In contrast to the two reduction variants, Adaptive Rigidity directly optimizes for the optimal edge rigidity which is more difficult to solve than starting with a more rigid optimization and relaxing the smoothness over time. Additionally, due to the initialization of the edge weights with the previous frames it occurs that the edge weights are initialized with smaller values than necessary. To avoid divergence of the algorithm due to an initialization with too small edge weights, the rigidity regularization weight should not be too small. The plot visualized in Figure 7.3 and the images in Figure 7.6e and 7.6f show the drawback if the rigidity regularization coefficient is too small. The algorithm stays in a local optimum and therefore leads to a sub-optimal fitting. We assume these two points are the main reason why the reduction strategies perform better than Adaptive Rigidity. However the results show that Adaptive Rigidity is able to improve the baseline variant by optimizing for optimal edge rigidity weights. Whether the variant optimizes for an optimal weight per edge or per vertex makes no relevant difference, in all experiments the two variants perform similar.

The two refinement strategies *Refinement at Edge* and *Refinement at Vertex* lead to different performances for each dataset. Overall these variants lead to an improvement over the baseline for most datasets. However for the dataset paperbag the performance is worse than the performance of the baseline. If we look at the relative error per node the performance is the worst of all variants. As the algorithm only inserts nodes, the algorithm tends to use a lot more nodes in later frames that are not even needed for a good representation, an example is visible in Figure 7.6h and Figure 7.6g. As we use euclidean distance instead of the geodesic distance as done by Li et al. [36] nodes of the deformation graph tend to influence the wrong source vertices, which leads to a wrong deformation and therefore to a bad alignment to the target dataset, an example is visible in 7.7h and Figure 7.7g.

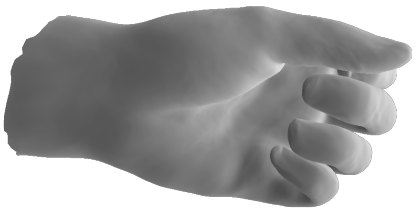


(a) Ground truth. (b) As-rigid-as-possible registration. (c) Reduce Rigidity registration. (d) Reduce Smoothness registration.

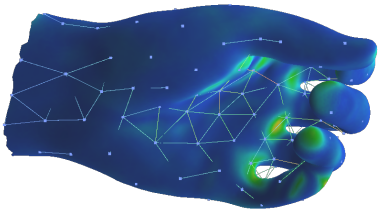


(e) Adaptive Rigidity at Edge registration. (f) Adaptive Rigidity at Vertex registration. (g) Refinement at Edge registration. (h) Refinement at Vertex registration.

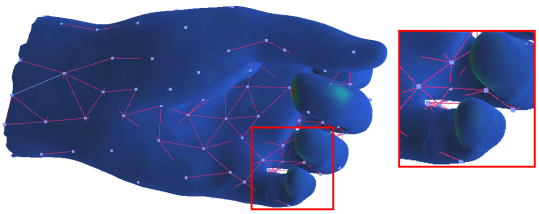
Figure 7.6: Non-rigid registration results of different variants on the puppet sequence at frame 50.



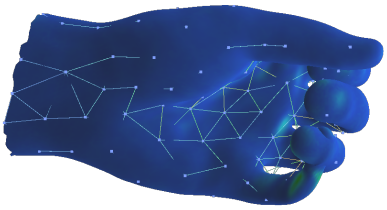
(a) Ground truth.



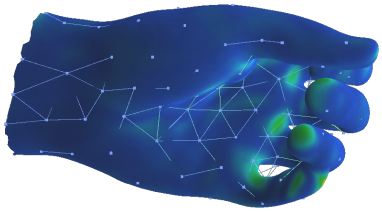
(b) As-rigid-as-possible registration.



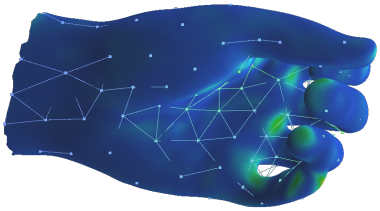
(c) Reduce Rigidity registration.



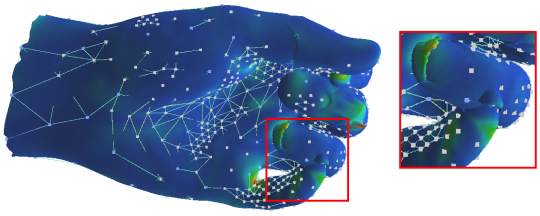
(d) Reduce Smoothness registration.



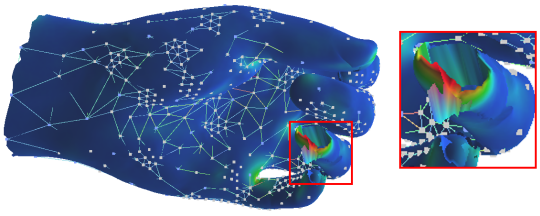
(e) Adaptive Rigidity at Edge registration.



(f) Adaptive Rigidity at Vertex registration.



(g) Refinement at Edge registration.



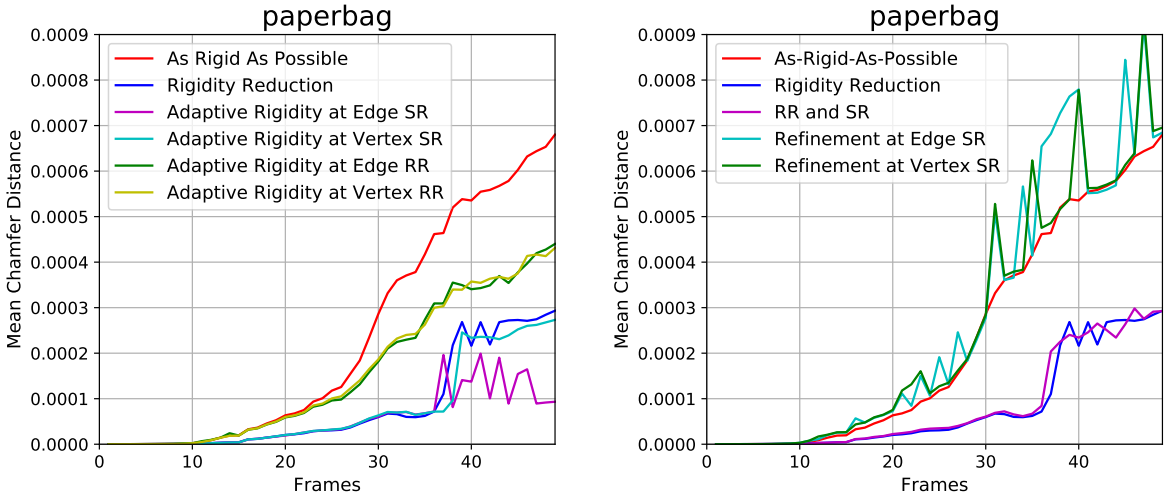
(h) Refinement at Vertex registration.

Figure 7.7: Non-rigid registration results of different variants on the hand sequence at frame 35.

7.5 Combination of Non-Rigid Registration Variants

To find the overall best performing strategy we compared different combinations of non-rigid registration variants against each other. We therefore tested the variants Rigidity Reduction combined with Smoothness Reduction (RR and SR), the Adaptive Rigidity variants combined with Rigidity Reduction (Adaptive Rigidity at Edge RR, Adaptive Rigidity at Vertex RR) and Smoothness Reduction (Adaptive Rigidity at Edge SR, Adaptive Rigidity at Vertex SR) and the combination of the Refinement variants with the Smoothness Reduction (Refinement at Edge SR, Refinement at Vertex SR). We didn't combine the refinement variants with the rigidity reduction as the refinement variants reconnect the nodes after each refinement step, leading to new edges and a loss of previous edge information. The results of the different combinations are visualized in Figures 7.8, 7.9, 7.10 and 7.11.

Table 7.10 shows the total relative errors of the different combined variants. The overall best performing algorithm is *Adaptive Rigidity at Edge with Smoothness Reduction*. The result supports our assumption that a combination of a good rigidity weight per edge with a smoothness reduction strategy leads to the best performance. This time the variant *at Edge* performs better than the variant *at Vertex*, we assume the reason for this is that the variant at Edge better represent the kinematics of the deforming object. While Adaptive Rigidity in combination with Smoothness Reduction achieve the best results, Adaptive Rigidity in combination with Rigidity Reduction does not even achieve the performance of Rigidity Reduction itself. As both strategies adapt rigidity weights per edge, the combination of both suffers from the dependency of the Adaptive Rigidity objective function on the Rigidity Reduction edge weighting, which prevents the objective function from a stable convergence. This problem is not present in the Smoothness Reduction combination as the relative edge rigidity to all other edges stays the same. The best performing variant combinations are visualized in Figure 7.12 and Figure 7.13. The Refinement variants profit from the combination with the Smoothness Reduction, however the performance is not as good as the Smoothness Reduction itself.



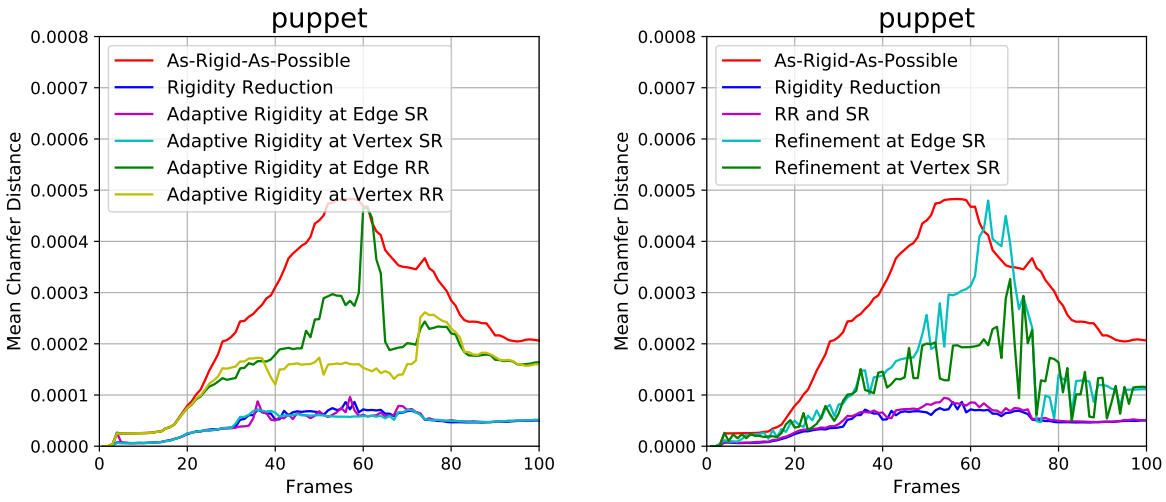
(a) Adaptive Rigidity in combination with Rigidity Reduction and Smoothness Reduction.

(b) Refinement in combination with Smoothness Reduction and Rigidity Reduction with Smoothness Reduction.

Figure 7.8: Mean chamfer distance of different variant combinations over paperbag sequence.

Variant	relative error	variance in 10^{-4}	relative error per node	n_g
As-Rigid-As-Possible	1.000	0.9296	1.000	198
Rigidity Reduction	0.3520	0.9976	0.3520	198
Smoothness Reduction	0.3371	0.9483	0.3371	198
Adaptive Rigidity at Edge SR	**0.2252**	**0.2782**	**0.2252**	**198**
Adaptive Rigidity at Vertex SR	0.3272	0.9036	0.3272	198
Adaptive Rigidity at Edge RR	0.6635	0.8942	0.6635	198
Adaptive Rigidity at Vertex RR	0.6686	0.9054	0.6686	198
RR and SR	0.3651	1.0297	0.3651	198
Refinement at Edge SR	1.2014	1.1471	1.2802	111 - 211
Refinement at Vertex SR	1.1196	1.1387	0.8426	111 - 149

Table 7.6: Relative error of different variants combinations on the paperbag sequence



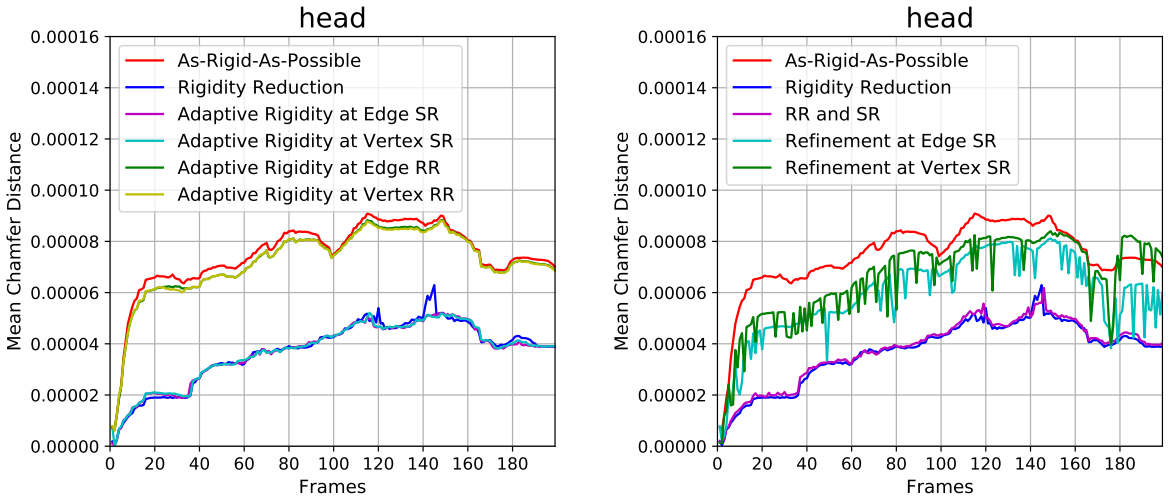
(a) Adaptive Rigidity in combination with Rigidity Reduction and Smoothness Reduction.

(b) Refinement in combination with Smoothness Reduction and Rigidity Reduction with Smoothness Reduction.

Figure 7.9: Mean chamfer distance of different variant combinations over puppet sequence.

Variant	relative error	variance in 10^{-5}	relative error per node	n_g
As-Rigid-As-Possible	1.000	7.4026	1.000	134
Rigidity Reduction	0.1840	1.5414	0.1840	134
Smoothness Reduction	0.1897	1.6378	0.1897	134
Adaptive Rigidity at Edge SR	0.1782	1.2859	0.1782	134
Adaptive Rigidity at Vertex SR	**0.1764**	**1.3121**	**0.1764**	**134**
Adaptive Rigidity at Edge RR	0.6741	5.4975	0.6741	134
Adaptive Rigidity at Vertex RR	0.5503	3.7645	0.5503	134
RR and SR	0.2003	1.7033	0.2003	134
Refinement at Edge SR	0.5748	4.6078	4.2896	90 - 1310
Refinement at Vertex SR	0.4467	2.9757	2.2439	90 - 940

Table 7.7: Relative error of different variants combinations on the puppet sequence



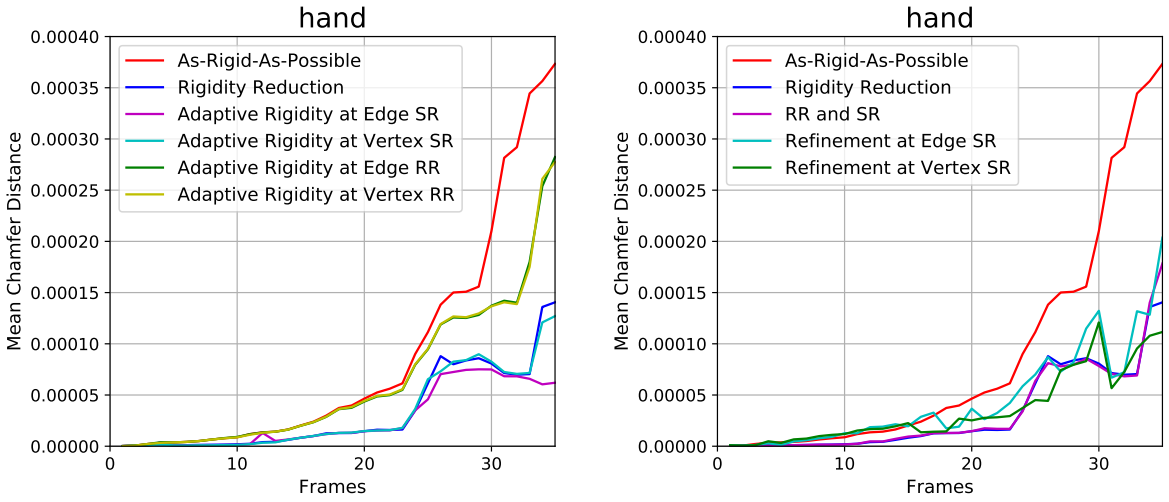
(a) Adaptive Rigidity in combination with Rigidity Reduction and Smoothness Reduction.

(b) Refinement in combination with Smoothness Reduction and Rigidity Reduction with Smoothness Reduction.

Figure 7.10: Mean chamfer distance of different variant combinations over head sequence.

Variant	relative error	variance in 10^{-5}	relative error per node	n_g
As-Rigid-As-Possible	1.000	2.2185	1.000	145
Rigidity Reduction	0.4991	1.3001	0.4991	145
Smoothness Reduction	0.5371	1.3247	0.5371	145
Adaptive Rigidity at Edge SR	**0.4955**	**1.2420**	**0.4955**	**145**
Adaptive Rigidity at Vertex SR	0.4972	1.2330	0.4972	145
Adaptive Rigidity at Edge RR	0.9679	2.1314	0.9679	145
Adaptive Rigidity at Vertex RR	0.9656	2.1219	0.9656	145
RR at SR	0.5114	1.3041	0.5114	145
Refinement at Edge SR	0.8000	1.6182	2.9433	97 - 1259
Refinement at Vertex SR	0.8827	1.9160	2.3433	97 - 601

Table 7.8: Relative error of different variants combinations on the head sequence.



(a) Adaptive Rigidity in combination with Rigidity Reduction and Smoothness Reduction.

(b) Refinement in combination with Smoothness Reduction and Rigidity Reduction with Smoothness Reduction.

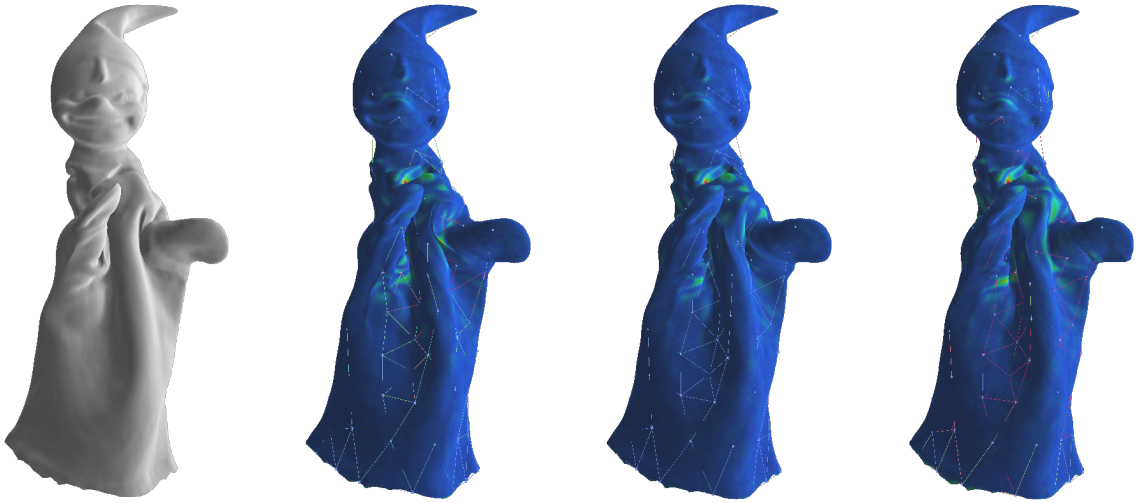
Figure 7.11: Mean chamfer distance of different variant combinations over hand sequence.

Variant	relative error	variance in 10^{-5}	relative error per node	n_g
As-Rigid-As-Possible	1.000	4.5523	1.000	148
Rigidity Reduction	0.3689	2.9684	0.3689	148
Smoothness Reduction	0.3984	3.1008	0.3984	148
Adaptive Rigidity at Edge SR	**0.2982**	**2.3398**	**0.2982**	**148**
Adaptive Rigidity at Vertex SR	0.3594	2.8799	0.3594	148
Adaptive Rigidity at Edge RR	0.7209	3.4297	0.7209	148
Adaptive Rigidity at Vertex RR	0.7211	3.4652	0.7211	148
RR and SR	0.3783	2.9454	0.3783	148
Refinement at Edge SR	0.5110	2.3709	2.4312	88 - 876
Refinement at Vertex SR	0.4028	1.8911	1.5011	88 - 700

Table 7.9: Relative error of different variants combinations on the hand sequence.

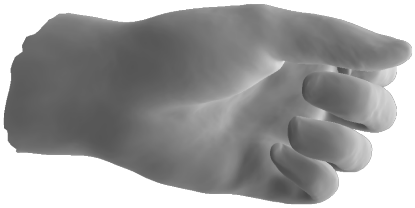
Variant	relative error	relative error per node
As-Rigid-As-Possible	1.000	1.000
Rigidity Reduction	0.3510	0.3510
Smoothness Reduction	0.3656	0.3656
Adaptive Rigidity at Edge SR	**0.2993**	**0.2993**
Adaptive Rigidity at Vertex SR	0.3401	0.3401
Adaptive Rigidity at Edge RR	0.7566	0.7566
Adaptive Rigidity at Vertex RR	0.7264	0.7264
RR and SR	0.3638	0.3638
Refinement at Edge SR	0.7718	2.7361
Refinement at Vertex SR	0.7130	1.7327

Table 7.10: Total relative error of different combined variants.

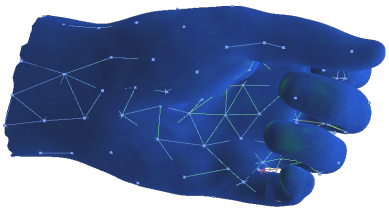


(a) Ground truth. (b) Adaptive Rigidity at Edge with Smoothness Reduction. (c) Adaptive Rigidity at Vertex with Smoothness Reduction. (d) Reduce Rigidity with Smoothness Reduction.

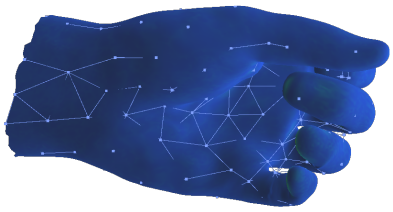
Figure 7.12: Non-rigid registration results of different combinations of variants on the puppet sequence at frame 50.



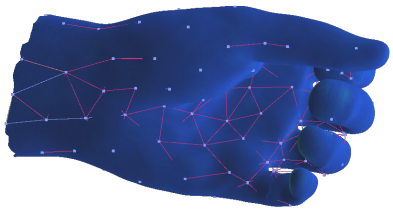
(a) Ground truth.



(b) Adaptive Rigidity at Edge with Smoothness Reduction.



(c) Adaptive Rigidity at Vertex with Smoothness Reduction.



(d) Reduce Rigidity combined with Smoothness Reduction.

Figure 7.13: Non-rigid registration results of different combinations of variants on the hand sequence at frame 50.

7.6 Limitations

While we are able to show that Rigidity Reduction, as well as the combination of Adaptive Refinement with Smoothness Reduction improve the results of non-rigid registration there are still limitations of our improvements. The Adaptive Rigidity approach only takes the edge rigidity of the previous frame into account instead of learning an edge rigidity model representing the true kinematics of the object. This problem is even more relevant for Rigidity Reduction which takes no edge rigidity information at all from other frames into account. The algorithms where also only tested on the reconstructed datasets and not on the depth scan datasets, therefore we have no quantification how the different variants perform for part in part registration.

A limitation of the introduced deformation graph refinement strategy lies therein that the algorithm only inserts new nodes for better fitting but doesn't find the optimal deformation graph with minimal number of nodes. This leads to an increase of the number of nodes for each new frame the algorithm fits.

8 Conclusion

Non-rigid tracking and reconstruction of arbitrary objects and motions, especially in real-time, provides many new possibilities for interesting 3D scanning applications in dynamic environments. Nevertheless even after significant developments in the last years, non-rigid registration remains a challenging problem when dealing with fast motions, occlusions and topological changes.

We proposed *Adaptive Rigidity at Edge with Smoothness Reduction* an adapted non-rigid registration objective function which leads to a more stable and more accurate non-rigid fitting. The key contribution of our method is to optimize for an adaptive rigidity weight for each edge. In contrast to one smoothness coefficient for all edges, this allows our method to incorporate the underlying kinematics of the object.

8.1 Future Work

Instead of optimizing the edge rigidity weights of each frame separately it might be beneficial to learn a rigidity bias for each edge of the deformation graph. By doing so the bias rigidity for each edge can represent the kinematics of the deforming object. Finding an optimal bias could be implemented by using a confidence rigidity term as bias and updating it over time or by learning the rigidity bias with machine learning. Additionally it would be interesting to use machine learning to learn the optimal edge weights for frames over time and based on the current iteration of the algorithm. Another topic for further research is to create an adaptively refinable deformation graph. The deformation graph refinement introduced by Li et al. [36] leads to a high number of nodes and therefore to an increase of unknowns due to the fact that nodes are only inserted and never deleted. Instead of only adding new nodes it would be more promising to find the optimal minimal graph structure which best represents the object.

List of Figures

List of Tables

Bibliography

[1] R. A. Newcombe, D. Fox, and S. M. Seitz. "DynamicFusion: Reconstruction and tracking of non-rigid scenes in real-time". In: June 2015, pp. 343–352. DOI: 10.1109/CVPR.2015.7298631.

[2] M. Innmann, M. Zollhöfer, M. Nießner, C. Theobalt, and M. Stamminger. "VolumeDeform: Real-time Volumetric Non-rigid Reconstruction". In: (Mar. 2016).

[3] P. Besl and H. McKay. "A method for registration of 3-D shapes. IEEE Trans Pattern Anal Mach Intell". In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 14 (Mar. 1992), pp. 239–256. DOI: 10.1109/34.121791.

[4] Y. Chen and G. Medioni. "Object Modeling by Registration of Multiple Range Images". In: *Image Vision Comput.* 10 (Jan. 1992), pp. 145–155. DOI: 10.1109/ROBOT.1991.132043.

[5] S. Rusinkiewicz and M. Levoy. "Efficient variants of the ICP algorithm". In: *Proceedings Third International Conference on 3-D Digital Imaging and Modeling*. May 2001, pp. 145–152. DOI: 10.1109/IM.2001.924423.

[6] K. Pulli and L. G. Shapiro. "Surface Reconstruction and Display from Range and Color Data". In: *Graph. Models* 62.3 (May 2000), pp. 165–201. ISSN: 1524-0703. DOI: 10.1006/gmod.1999.0519. URL: http://dx.doi.org/10.1006/gmod.1999.0519.

[7] G. Turk and M. Levoy. "Zippered Polygon Meshes from Range Images". In: *Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH '94. New York, NY, USA: ACM, 1994, pp. 311–318. ISBN: 0-89791-667-0. DOI: 10.1145/192161.192241. URL: http://doi.acm.org/10.1145/192161.192241.

[8] T. Masuda, K. Sakaue, and N. Yokoya. "Registration and integration of multiple range images for 3-D model construction". In: *Proceedings of 13th International Conference on Pattern Recognition*. Vol. 1. Aug. 1996, 879–883 vol.1. DOI: 10.1109/ICPR.1996.546150.

[9] N. Gelfand, L. Ikemoto, S. Rusinkiewicz, and M. Levoy. "Geometrically Stable Sampling for the ICP Algorithm". In: (Aug. 2003).

[10] D. A. Simon. "Fast and Accurate Shape-based Registration". AAI9838226. PhD thesis. Pittsburgh, PA, USA, 1996. ISBN: 0-591-91885-4.

[11] G. Blais and M. Levine. "Registering multiview range data to create 3D computer objects". In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 17 (Sept. 1995), pp. 820–824. DOI: 10.1109/34.400574.

[12] K.-L. Low. "Linear Least-Squares Optimization for Point-to-Plane ICP Surface Registration". In: 2004.

[13] Q.-Y. Zhou and V. Koltun. "Dense Scene Reconstruction with Points of Interest". In: *ACM Trans. Graph.* 32.4 (July 2013), 112:1–112:8. ISSN: 0730-0301. DOI: 10.1145/2461912.2461919. URL: http://doi.acm.org/10.1145/2461912.2461919.

[14] D. Cremers and K. Kolev. "Multiview Stereo and Silhouette Consistency via Convex Functionals over Convex Domains". In: *IEEE Trans. Pattern Anal. Mach. Intell.* 33.6 (June 2011), pp. 1161–1174. ISSN: 0162-8828. DOI: 10.1109/TPAMI.2010.174. URL: http://dx.doi.org/10.1109/TPAMI.2010.174.

[15] M. Oswald, J. Stuhmer, and D. Cremers. "Generalized Connectivity Constraints for Spatio-temporal 3D Reconstruction". In: Sept. 2014, pp. 32–46. DOI: 10.1007/978-3-319-10593-2_3.

[16] V. Leroy, J. Franco, and E. Boyer. "Multi-view Dynamic Shape Refinement Using Local Temporal Integration". In: *2017 IEEE International Conference on Computer Vision (ICCV)*. Oct. 2017, pp. 3113–3122. DOI: 10.1109/ICCV.2017.336.

[17] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohli, J. Shotton, S. Hodges, and A. W. Fitzgibbon. "KinectFusion: Real-time dense surface mapping and tracking". In: *2011 10th IEEE International Symposium on Mixed and Augmented Reality* (2011), pp. 127–136.

[18] M. Nießner, M. Zollhöfer, S. Izadi, and M. Stamminger. "Real-time 3D reconstruction at scale using voxel hashing". In: *ACM Trans. Graph.* 32 (2013), 169:1–169:11.

[19] C. Kerl, J. Sturm, and D. Cremers. "Dense visual SLAM for RGB-D cameras". In: *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. Nov. 2013, pp. 2100–2106. DOI: 10.1109/IROS.2013.6696650.

[20] F. Lu and E. Milios. "Globally Consistent Range Scan Alignment for Environment Mapping". In: *Auton. Robots* 4.4 (Oct. 1997), pp. 333–349. ISSN: 0929-5593. DOI: 10.1023/A:1008854305733. URL: https://doi.org/10.1023/A:1008854305733.

[21] A. Dai, M. Nießner, M. Zollhöfer, S. Izadi, and C. Theobalt. "BundleFusion: Real-time Globally Consistent 3D Reconstruction using On-the-fly Surface Re-integration". In: *CoRR* abs/1604.01093 (2016). arXiv: 1604.01093. URL: http://arxiv.org/abs/1604.01093.

[22] O. Sorkine and M. Alexa. "As-rigid-as-possible Surface Modeling". In: *Proceedings of the Fifth Eurographics Symposium on Geometry Processing*. SGP '07. Barcelona, Spain: Eurographics Association, 2007, pp. 109–116. ISBN: 978-3-905673-46-3. URL: http://dl.acm.org/citation.cfm?id=1281991.1282006.

[23] R. W. Sumner, J. Schmid, and M. Pauly. "Embedded Deformation for Shape Manipulation". In: *ACM Trans. Graph.* 26.3 (July 2007). ISSN: 0730-0301. DOI: 10.1145/1276377.1276478. URL: http://doi.acm.org/10.1145/1276377.1276478.

[24] J. Solomon, M. Ben-Chen, A. Butscher, and L. Guibas. "As-Killing-As-Possible Vector Fields for Planar Deformation". In: *Comput. Graph. Forum* 30 (Aug. 2011), pp. 1543–1552. DOI: 10.1111/j.1467-8659.2011.02028.x.

[25] M. Ben-Chen, A. Butscher, J. Solomon, and L. Guibas. "On Discrete Killing Vector Fields and Patterns on Surfaces". In: *Comput. Graph. Forum* 29 (July 2010), pp. 1701–1711. DOI: 10.1111/j.1467-8659.2010.01779.x.

[26] C. Cao, Y. Weng, S. Lin, and K. Zhou. "3D Shape Regression for Real-time Facial Animation". In: *ACM Transactions on Graphics (TOG)* 32 (July 2013). DOI: 10.1145/2461912.2462012.

[27] H. Li, J. Yu, Y. Ye, and C. Bregler. "Realtime Facial Animation with On-the-fly Correctives". In: *ACM Transactions on Graphics (TOG)* 32 (July 2013). DOI: 10.1145/2461912.2462019.

[28] I. Oikonomidis, N. Kyriazis, and A. Argyros. "Efficient model-based 3D tracking of hand articulations using Kinect". In: vol. 1. Jan. 2011. DOI: 10.5244/C.25.101.

[29] C. Qian, X. Sun, Y. Wei, X. Tang, and J. Sun. "Realtime and Robust Hand Tracking from Depth". In: *2014 IEEE Conference on Computer Vision and Pattern Recognition*. June 2014, pp. 1106–1113. DOI: 10.1109/CVPR.2014.145.

[30] J. Taylor, J. Shotton, T. Sharp, and A. Fitzgibbon. "The Vitruvian manifold: Inferring dense correspondences for one-shot human pose estimation". In: *2012 IEEE Conference on Computer Vision and Pattern Recognition*. June 2012, pp. 103–110. DOI: 10.1109/CVPR.2012.6247664.

[31] T. Schmidt, R. A. Newcombe, and D. Fox. "DART: dense articulated real-time tracking with consumer depth cameras". In: *Autonomous Robots* 39 (2015), pp. 239–258.

[32] M. Ye, Y. Shen, C. Du, Z. Pan, and R. Yang. "Real-Time Simultaneous Pose and Shape Estimation for Articulated Objects Using a Single Depth Camera". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 38.8 (Aug. 2016), pp. 1517–1532. DOI: 10.1109/TPAMI.2016.2557783.

[33] T. Yu, K. Guo, F. Xu, Y. Dong, Z. Su, J. Zhao, J. Li, Q. Dai, and Y. Liu. "BodyFusion: Real-Time Capture of Human Motion and Surface Geometry Using a Single Depth Camera". In: *2017 IEEE International Conference on Computer Vision (ICCV)*. Oct. 2017, pp. 910–919. DOI: 10.1109/ICCV.2017.104.

[34] B. Amberg, S. Romdhani, and T. Vetter. "Optimal Step Nonrigid ICP Algorithms for Surface Registration". In: June 2007. DOI: 10.1109/CVPR.2007.383165.

[35] H. Li, R. W. Sumner, and M. Pauly. "Global Correspondence Optimization for Non-rigid Registration of Depth Scans". In: *Proceedings of the Symposium on Geometry Processing*. SGP '08. Copenhagen, Denmark: Eurographics Association, 2008, pp. 1421–1430. URL: http://dl.acm.org/citation.cfm?id=1731309.1731326.

[36] H. Li, B. Adams, L. J. Guibas, and M. Pauly. "Robust Single-view Geometry and Motion Reconstruction". In: *ACM Trans. Graph.* 28.5 (Dec. 2009), 175:1–175:10. ISSN: 0730-0301. DOI: 10.1145/1618452.1618521. URL: http://doi.acm.org/10.1145/1618452.1618521.

[37] M. Zollhöfer, M. Nießner, S. Izadi, C. Rehmann, C. Zach, M. Fisher, C. Wu, A. Fitzgibbon, C. Loop, C. Theobalt, and M. Stamminger. "Real-time Non-rigid Reconstruction Using an RGB-D Camera". In: *ACM Trans. Graph.* 33.4 (July 2014), 156:1–156:12. ISSN: 0730-0301. DOI: 10.1145/2601097.2601165. URL: http://doi.acm.org/10.1145/2601097.2601165.

[38] K. Guo, F. Xu, Y. Tao, X. Liu, Q. Dai, and Y. Liu. "Real-time geometry, albedo and motion reconstruction using a single RGBD camera". In: *ACM Transactions on Graphics* 36 (July 2017), p. 1. DOI: 10.1145/3072959.3126786.

[39] M. Slavcheva, M. Baust, D. Cremers, and S. Ilic. "KillingFusion: Non-rigid 3D Reconstruction without Correspondences". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2017), pp. 5474–5483.

[40] M. Dou, S. Khamis, Y. Degtyarev, P. L. Davidson, S. R. Fanello, A. Kowdle, S. Orts, C. Rhemann, D. Kim, J. Taylor, P. Kohli, V. Tankovich, and S. Izadi. "Fusion4D: real-time performance capture of challenging scenes". In: *ACM Trans. Graph.* 35 (2016), 114:1–114:13.

[41] M. Dou, P. Davidson, S. R. Fanello, S. Khamis, A. Kowdle, C. Rhemann, V. Tankovich, and S. Izadi. "Motion2Fusion: Real-time Volumetric Performance Capture". In: *ACM Trans. Graph.* 36.6 (Nov. 2017), 246:1–246:16. ISSN: 0730-0301. DOI: 10.1145/3130800.3130801. URL: http://doi.acm.org/10.1145/3130800.3130801.

[42] F. S. Grassia. "Practical Parameterization of Rotations Using the Exponential Map". In: *J. Graphics, GPU, and Game Tools* 3 (1998), pp. 29–48.

[43] R. Bridson. "Fast Poisson disk sampling in arbitrary dimensions". In: *ACM SIGGRAPH* (Aug. 2007). DOI: 10.1145/1278780.1278807.

[44] M. Botsch and L. Kobbelt. "A Remeshing Approach to Multiresolution Modeling". In: vol. 71. Jan. 2004, pp. 189–196. DOI: 10.1145/1057432.1057457.

[45] S. Agarwal, K. Mierle, et al. *Ceres Solver*. http://ceres-solver.org.

[46] The CGAL Project. *CGAL User and Reference Manual*. 4.14.1. CGAL Editorial Board, 2019. URL: https://doc.cgal.org/4.14.1/Manual/packages.html.

[47] M. Nießner. *mLib*. https://github.com/niessner/mLib.

[48] A. Denninger. *Variants of Non-Rigid Registration*. https://https://github.com/Angela-Oo/MasterThesis.

[49] G. Borgefors. "Distance transformations in arbitrary dimensions". In: *Computer Vision, Graphics, and Image Processing* 27.3 (1984), pp. 321–345. ISSN: 0734-189X. DOI: https://doi.org/10.1016/0734-189X(84)90035-5. URL: http://www.sciencedirect.com/science/article/pii/0734189X84900355.