Hierarchical mesh-to-points as-rigid-as-possible registration

Pierre Bourquat, David Coeurjolly, Guillaume Damiand, Florent Dupont

Hierarchical mesh-to-points as-rigid-as-possible registration

Pierre Bourquat[1,*], David Coeurjolly[1], Guillaume Damiand[1], Florent Dupont[1]

Univ Lyon, UCBL, CNRS, INSA Lyon, LIRIS, UMR5205, 69622 Villeurbanne, France

ARTICLE INFO

Article history:
Received 3 July 2021
Received in final form 20 October 2021
Accepted 22 October 2021
Available online 26 November 2021

Keywords: Surface registration, Non-rigid deformation, Point cloud, As-Rigid-As-Possible, Hierarchical approach

ABSTRACT

Surface registration is a fundamental problem in computer graphics and computer-aided design. The problem consists in finding a deformation from one surface to another that preserves some properties. For instance, in our inverse engineering context, we aim at finding the best, as isometric as possible, map between an input triangular model, and a large point cloud acquired on the actual mechanical part being processed. Existing solutions are not able to handle very large models with a good level of precision. We propose a method which is accurate and fast. Our solution combines an efficient iterative energy minimization scheme on a hierarchical decomposition of the problem geometry. Our experiments show that we obtain a fast and efficient algorithm compared to the state-of-the art method, while keeping its numerical accuracy.

1. Introduction

Surface registration is a fundamental problem in computer graphics and computer-aided design. Given two surfaces, called *source* and *target*, the objective of the registration is to find a deformation that maps from the *source* into the *target* satisfying some properties (isometric, conformal, one-to-one, partial...). This is a common problem, for example in 3D scanning, where multiple datasets captured from different viewpoints must be registered. In geometry processing, a mapping between the *source* and the *target* is for example needed in post-processing such as reconstruction, morphing or information transfer.

This is also a key problem in inverse engineering where CAD models are compared to measure their real counterparts. In this context, *real* objects are captured live using laser range or probing devices. Hence, the *source* is a CAD model (*e.g.* a clean triangular mesh), and the *target* a large, possibly noisy, point cloud. In sheet metal manufacturing, which motivates the present research, the *springback* problem [1], caused by the elastoplastic material behavior of sheet metals, falls into registration problems where the CAD model *source* must be registered into the measure of the real model *target* before being processed by the device (*e.g.* for machining). The *springback* being mainly bending, a classical assumption is that tangential shearing or stretching are neglected and therefore the deformation can be considered as non-rigid and almost isometric. This defines the class of maps between the *source* and the *target* we are looking for.

Since time efficiency is central for engineering production, the registration method must be time efficient on very large datasets. In the following, we will assume that the *source* is a mesh with more than a million of vertices and the *target* is a point cloud with more than 5 million samples.

The computer graphics literature contains several non-rigid registration techniques, see *e.g.* [2] for a survey. Despite the quality of the results of such approaches, they may not be suited to handle very large models as it is the case in the production engineering field. The CAD literature proposes fewer solutions (see Section 2). Even though they are more suited for our problem, they are still not fast enough to deal with very large datasets. One common remark about the non-rigid isometric registration solutions is almost all of them register mesh to mesh instead of mesh to point cloud. Therefore they all require a reconstruction preprocessing on the point cloud to extract a mesh.

In this paper, we extend computer graphics techniques and propose an accurate and efficient algorithm to compute an as-rigid-as-possible map from a 3D CAD model to a large (possibly noisy) point cloud. This approach combines an efficient iterative energy minimization scheme and a multiscale optimization using a hierarchical mesh representation of the *source*.

The paper is organized as follows: the next section reviews the state-of-the-art of non-rigid isometric registration. In Section 3 we introduce the concepts used in the paper. In Section 4 we present our registration algorithm. The method is validated on our dataset in Section 5.

*Corresponding author:
e-mail: pierre.bourquat@liris.cnrs.fr (Pierre Bourquat)

2. Related works

Almost isometric, also known as as-rigid-as-possible (arap for short), and non-rigid approaches have been studied in many fields of computer graphics: Sorkine and Alexa [3] Chao and al. [4] and Levi and al. [5] proposed surface manipulation techniques based on the arap energy, which we detail in Section 3.1. Gotsmam and al. [6] introduced an arap parametrization method and more recently Smith and al. [7] proposed optimization solutions for distortion energy based problems (such as the ones previously cited). Functional maps have been widely used to map scalar or vector functions from a given mesh onto another one while preserving some arap properties [8]. In this paper, we focus on the explicit construction of an arap map between a model (mesh) and a target (point cloud).

The NI-ICP algorithm introduced by Sacharow and al. [9] is a Non-rigid Isometric variant of ICP that registers a mesh onto another one. The method deforms the *source* into the *target* using an alternate scheme that first projects the *source* vertices onto their nearest neighbor in the *target*, and then enforces the isometry of those projections (gradient mesh editing step). Given the correspondences, the algorithm fits, for each *source* face, the best rigid transformation from the non-projected face to the projected one, transforming the *source* mesh into a soup of faces where each face is isometric to its original version. Because this last step breaks the mesh connectivity, a global stitching step is applied following the mesh editing proposed by Yu and al. [10]. A final step NURBS-based technique is applied from the resulting deformation to compute a continuous map.

While the need for this final step is questionable in our context, the method provides an actual solution to the non-rigid isometric registration for CAD models under *springback* deformations. However, it requires a well-reconstructed surface mesh from the *target* point cloud and is still not fast enough on large datasets. Authors say in the paper they had to prematurely abort their experiments on full-resolution models.

Schweinoch and al. [11] have proposed a similar non-rigid isometric registration algorithm based on a combination of ICP and arap mesh deformation. However, they use a hierarchical divide-and-conquer approach: the deformed *source* is considered as a collection of segments initially set to a single segment. The core step of the algorithm consists of subdividing each segment according to its distance from the *target* and rigidly align all those segments onto the *target*. Similarly to the NI-ICP approach, the rigid alignment of all segments results in a loss of connectivity at the segment endpoints. This connectivity is subsequently reestablished using the arap mesh deformation proposed by Sorkine and al. [3] that we detail in Section 3. Similarly to Sacharow and al. [9] algorithm, this method is an actual solution for CAD models under *springback* deformation. However it was not tested on full-resolution models. Although the method uses a hierarchical approach, the whole set of *source* vertices is used at each iteration. All vertices of each segment are used to compute the rigid alignment and only the boundary vertices are used for the stitching step. So over the iterations, the total number of vertices used for alignment remains the same and only the number of vertices used for the stitching step increases. Therefore the method does not benefit from the speed up of classical hierarchical approaches.

Klein and al. [12] propose a new method by exactly formulating the isometry registration with a proper objective function:

$$\min \sum_{ij} \left| \|\mathbf{p}_i - \mathbf{p}_j\|^2 - \|\mathbf{z}_i - \mathbf{z}_j\|^2 \right| , \qquad (1)$$

with $\mathbf{p}_i$ and $\mathbf{p}_j$ being vertices of the *source* surface mesh, $\mathbf{z}_i$ and $\mathbf{z}_j$ being their associated vertices on the *target*, and ij a set of vertex pairs not specified yet. The first step of this method consists in computing a neighborhood structure of the *source* mesh for the objective function such that close-by situated vertex pairs are conserved in the energy and distant pairs are removed. The second step is a global registration where the (non-convex) energy minimization is treated as a Quadratic Assignment Problem (QAP for short) from the neighborhood structure and the downsampled *target*. The final step is a local reoptimization where the energy is minimized by a gradient descent method initialized by the QAP solutions. The method provides an actual isometric registration energy and works on a point cloud *target*. However the computation of the neighborhood structure and the energy minimization being based on a QAP problem (whose optimal solution is NP-hard) make the method not suited for large datasets.

Finally, Huang and al. [13] proposed a non-rigid registration method under isometric assumptions that alternates between a correspondence and a deformation optimization step. The correspondences are computed by matching *source* and *target* using the nearest neighbor approach in a metric space defined by the Euclidean distance and a feature metric defined from the principal curvatures of multi-level quadric patch fitted for each vertex according to Cazals and Pouget formulation [14]. Then the correspondences are pruned based on geodesic distance consistency and the remaining ones are propagated to assure uniform correspondences. The *source* is segmented into clusters whose deformations can be described as single rigid deformations and then registration of the *source* onto the *target* is computed by minimizing a combination of point-to-point and point-to-plane energy according to the optimized correspondences. This method focuses on handling complex computer graphics geometries, leaving open its numerical accuracy on controlled shapes in the engineering context.

3. Preliminaries

In this section, we introduce the concepts used in the following.

3.1. Notations and As-Rigid-As-Possible deformation

We denote by $\mathcal{S}$ a *source* triangle mesh and by $\mathcal{T}$ a *target* point cloud. The mesh $\mathcal{S}$ has N vertices $\{\mathbf{s}_i\}$ in $\mathbb{R}^3$, its vertex normals are denoted $\{\mathbf{n}_i\}$ and its faces are denoted $\{\mathbf{f}_k\}$. The neighboring vertices to a vertex $\mathbf{s}_i$ is given by $\mathcal{N}(i)$ and Δ_{cotan} is the discrete Laplace-Beltrami operator on $\mathcal{S}$ following the *cotan* approach of Meyer and al. [15]. The *target* point cloud $\mathcal{T}$ has M vertices $\{\mathbf{t}_i\}$ and its normal vectors are denoted $\{\mathbf{m}_i\}$. When not specified otherwise, we will assume that $N < M$ (the CAD model has fewer vertices than the acquired point cloud).

We denote by $\pi : \{0 \dots N-1\} \to \{0 \dots M-1\}$ an assignment map from vertices of $\mathcal{S}$ to points of $\mathcal{T}$. We further elaborate below on the definition of such assignment.

Our objective is to determine the deformation $\mathcal{S}$ onto $\mathcal{T}$ resulting in the registered surface mesh $\mathcal{S}'$ such that $\mathcal{S}$ and $\mathcal{S}'$ are isometric and $\mathcal{S}'$ and $\mathcal{T}$ are aligned. Because true isometric deformation cannot be achieved, the deformation between $\mathcal{S}$ and $\mathcal{S}'$ must be arap instead of isometric. This objective implies two different energies: a first one to account for the proximity between $\mathcal{S}'$ and $\mathcal{T}$, and a second one for the arap evaluation.

The proximity energy is a simple quadratic fitting energy between $\mathcal{S}'$ and $\mathcal{T}$ from the assignment:

$$E^{prox}(\mathcal{S}', \mathcal{T}) := \sum_i \|\mathbf{s}'_i - \mathbf{t}_{\pi(i)}\|^2 . \tag{2}$$

The arap energy is a local rigidity energy between $\mathcal{S}$ and $\mathcal{S}'$:

$$E^{arap}(\mathcal{S}, \mathcal{S}', \mathcal{R}) := \sum_i w_i \sum_{j \in \mathcal{N}(i)} w_{ij} \|\mathbf{e}'_{ij} - R_i \mathbf{e}_{ij}\|^2 , \tag{3}$$

where $\mathbf{e}_{ij} := (\mathbf{s}_j - \mathbf{s}_i)$, resp. $\mathbf{e}'_{ij} := (\mathbf{s}'_j - \mathbf{s}'_i)$, denotes the edge vector, $\mathcal{R}$ denotes a set of rotation with R_i the rotation of the tangent space at $\mathbf{s}_i$. Weights w_i and w_{ij} are the cotangent weights of the Δ_{cotan} operator attached to $\mathcal{S}$. A more generic and continuous elasticity model following this principle can be found in [4]. In Sorkine and al. [3] arap deformation framework they define R_i as the rotation between $\mathbf{s}_i$ and $\mathbf{s}'_i$ neighborhoods and derive R_i from the singular value decomposition $S_i = U_i \Sigma_i V_i^T$ of

$$S_i := \sum_{j \in \mathcal{N}(i)} w_{ij} \mathbf{e}_{ij} \mathbf{e}'^T_{ij} , \tag{4}$$

leading to $R_i := V_i U_i^T$ (up to changing the last column of U_i if $\det(R_i) < 0$).

From the arap energy, one can define several inverse problems. For instance, given a dense set of rotations $\mathcal{R}$ and $\mathbf{s}$, find the $\mathbf{s}'$ minimizing E^{arap}. The new positions of a given vertex $\mathbf{s}'_i$ minimizing E^{arap} is given by solving the (sparse) linear system:

$$\sum_{j \in \mathcal{N}(i)} w_{ij} (\mathbf{s}'_i - \mathbf{s}'_j) = \sum_{j \in \mathcal{N}(i)} \frac{w_{ij}}{2} (R_i + R_j) \mathbf{e}_{ij} . \tag{5}$$

If $\mathbf{s}'$ denotes the $3 \times N$ matrix where each row i is $\mathbf{s}'_i$, and $\mathbf{b}$ the $3 \times N$ matrix defined from the right-hand side of (5), optimizing the positions amounts to solving the following Poisson problem:

$$\Delta_{cotan}\, \mathbf{s}' = \mathbf{b} . \tag{6}$$

3.2. Mesh simplification

Our approach relies on a hierarchical structure of the *source* mesh $\mathcal{S}$. Although many solutions exist in geometry processing and computer graphics, we rely on the simple, yet very efficient, edge collapsing simplification proposed by Garland and al. [16]. This method combines edge collapsing and quadric error metric that estimates the squared distance to the mesh. Initially an error quadric matrix Q_i is assigned to each $\mathbf{s}_i$ that encodes the total squared distance a point $\mathbf{p}$ to $\mathbf{s}_i$'s neighboring faces which is given by computing $\mathbf{p}^T Q_i \mathbf{p}$. The cost of each edge (i, j) is computed by minimizing the energy $\mathbf{p}^T (Q_i + Q_j) \mathbf{p}$ where $\mathbf{p}$ is the point that minimizes the cost (see Figure 1).

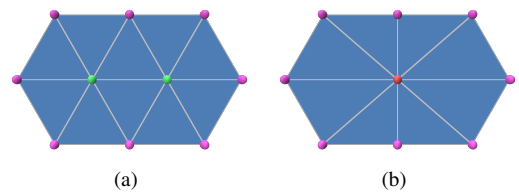


Fig. 1. Edge collapsing: the edge formed by the two green vertices in (*a*) is collapsed and the green vertices are merged into the red vertex in (*b*).

The edge with the minimal cost is collapsed into a new vertex whose position is the point $\mathbf{p}$ that minimizes the edge cost and its quadric error matrix is the sum of the quadrics of the edge extremities.

From this elementary edge collapse step, the overall hierarchical representation $\mathcal{H} : \mathcal{S} \to \mathcal{S}^{h-1} \to \mathcal{S}^{h-2} \to \dots \to \mathcal{S}^0$ of $\mathcal{S}$ with h layers can be given by considering disjoint sequences of minimal edge collapses per layer (see 4.2 for details).

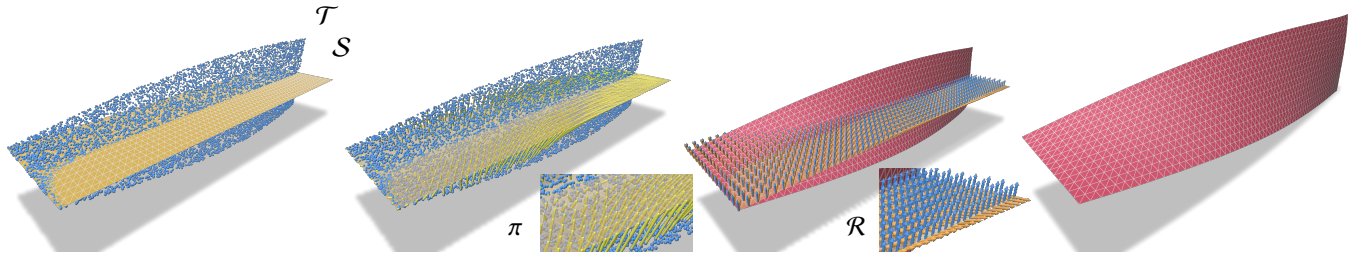
4. Fast and accurate hierarchical As-Rigid-As-Possible registration

In this section we present our solution to compute an arap registration of $\mathcal{S}$ onto $\mathcal{T}$ and then its hierarchical optimization.

4.1. Single level registration

Let us first detail the arap registration for a single layer of the hierarchical reconstruction. Algorithm 1 presents the overall algorithm to construct $\mathcal{S}'$ with the same topology as $\mathcal{S}$, such that $\mathcal{S}'$ is close to $\mathcal{T}$ and arap with respect to $\mathcal{S}$. This algorithm alternates between an assignment and an arap optimization step close to the one proposed by Sacharow and al. [9]. We extend this preminilary formulation with two main contributions: First, we propose a hierarchical optimization detailed in Section 4.2. Second, Sacharow and al. method requires that the *target* is a triangulated surface to define per face rotations needed in their formulation, leading to a mandatory preliminary surface reconstruction step from the point cloud. Our formulation has *per vertex* rotation information allowing us to directly work on the input point cloud stored in an efficient associated data structure (see below).

Assignment step (line 4 to 6). This step consists in assigning each vertex $\mathbf{s}'_i$ to a point $\mathbf{t}_{\pi(i)}$ of $\mathcal{T}$ and therefore results in a set of pairs $(i, \pi(i))$. Although the nearest neighbor operator from $\mathbf{s}'$ to $\mathbf{t}$ is not injective and may produce inconsistent pairs, our experiments and the results of Sacharow and al. [9] and Klein and al. [12] has shown that its use under small or simple deformations is a good trade-off between assignment quality and computation efficiency. We use a kdTree structure to perform the nearest neighbor requests. Since $\mathcal{T}$ is static, the kdTree is computed only once before the optimization loop. During the minimization process, geometrical information from $\mathcal{T}$ will only be given through nearest-neighbor queries on the kdTree.



(a) (b) (c) (d)

Fig. 2. Overall workflow: Starting from a *source* mesh $\mathcal{S}$ and a *target* point cloud $\mathcal{T}$ (*a*), we first start with assignation $\pi : \mathcal{S} \to \mathcal{T}$ (*b*). From rotations $\mathcal{R}$ computed from the assignment, we solve a first arap matching (*c*). In (*d*) we illustrate the final surface after 3 iterations.

We provide more technical details in Section 5.1, efficient NN queries allow us to consider very large point clouds as targets.

Algorithm 1: arap registration algorithm

Input: $\mathcal{S} = (\{\mathbf{s}_i\}, \{\mathbf{f}_k\}, \{\mathbf{n}_i\}, \Delta_{cotan})$: *source* surface mesh
$\mathcal{T} = (\{\mathbf{t}_i\}, \{\mathbf{m}_i\})$: *target* point cloud
ϵ: a stopping criterion

Output: $\mathcal{S}' = \left(\{\mathbf{s}'_i\}, \{\mathbf{f}_k\}\right)$: as-rigid-as possible deformed mesh with the same topology as $\mathcal{S}$.

1 $\mathcal{S}' \leftarrow \mathcal{S}$; $d \leftarrow +\infty$
2 $\mathbf{b}_{\mathcal{T}} \leftarrow$ BARYCENTER($\mathcal{T}$)
3 **while** $d > \varepsilon$ **do**
  /* Construct π. */
4 **foreach** $\mathbf{s}'_i \in \mathcal{S}'$ **do**
5 $\pi(i) :=$ NEAREST-NEIGHBOR($\mathbf{s}'_i$,$\mathcal{T}$)
6 **end**
  /* Optimize $\mathcal{S}'$ such as $\mathcal{S}'$ and $\mathcal{S}$ are arap. */
7 **foreach** $\mathbf{s}'_i \in \mathcal{S}'$ **do**
8 compute R_i according to (7)
9 **end**
10 compute $\mathbf{b}$ according to (5)
11 $\mathbf{s}_{tmp} \leftarrow \{\mathbf{s}'_i\}$ // warm start for the iterative solver
12 solve $\Delta_{cotan}\, \mathbf{s}_{tmp} = \mathbf{b}$
13 translate $\mathbf{s}_{tmp}$ by ($\mathbf{b}_{\mathcal{T}}$−BARYCENTER($\mathbf{s}_{tmp}$))
  /* End of iteration. */
14 $d \leftarrow \sum_i \|\mathbf{s}_{tmp} - \mathbf{s}'_i\|^2$
15 $\{\mathbf{s}'_i\} \leftarrow \mathbf{s}_{tmp}$
16 **end**
17 **return** $\mathcal{S}'$

Optimization step (line 7 to 13). We could use the $\{\mathbf{t}_{\pi(i)}\}$ as our mapping solution, but since they may not respect the arap constraint, we need to refine those pairs to ensure that $\mathbf{s}'$ minimizes E^{arap}. The solution of the arap energy minimization (5) requires the original *source* vertices $\mathbf{s}$, it's cotangent weights expressed in the Laplacian operator Δ_{cotan} and rotation matrices $\mathcal{R}$. From the assignment π, rotation matrices R_i are given by:

$$R_i := I + [\mathbf{n}_i \times \mathbf{m}_{\pi(i)}]_\times + \frac{1}{1 + \mathbf{n}_i.\mathbf{m}_{\pi(i)}} [\mathbf{n}_i \times \mathbf{m}_{\pi(i)}]^2_\times \,. \quad (7)$$

with I is the identity 3×3 matrix and $[\mathbf{v}]_\times$ is the map from a 3D vector $\mathbf{v}$ to a 3×3 skew-symmetric matrix such that $[\mathbf{v}]_\times\mathbf{q} = \mathbf{v}\times\mathbf{q}$ for any $\mathbf{q} \in \mathbb{R}^3$. Because we assume the deformation of $\mathcal{S}$ onto $\mathcal{T}$ is isometric, R_i is the rotation occurring between $\mathbf{s}_i$ and $\mathbf{s}'_i$ local plane, thus R_i can be computed from $\mathbf{s}_i$ and $\mathbf{s}'_i$ normals according to (7). With a rotation for each $\mathbf{s}_i$ (5) can be solved to get new vertices $\mathbf{s}'$ that respect the arap constraint. Solving (5) handles the rotational part of the mapping, but because it also writes off its translation part, we move the barycenter of $\mathbf{s}'$ to $\mathcal{T}$ barycenter.

Solver (line 12). The Δ_{cotan} being sparse, (5) can be efficiently solved either using factorized solvers (*e.g.* LU decomposition) or iterative schemes (*e.g.* conjugate gradient). The first one would lead to an expensive preprocessing overhead while the second approach also allows us to reuse results from previous iterations in Algorithm 1, *i.e.* vertices $\mathbf{s}'$ (see line 11), as warm start. This warm start could be problematic for the very first iteration as $\mathcal{S}'$ may be far from the target, but it becomes more and more efficient for the remaining iterations.

4.2. Hierarchical registration

When the source mesh $\mathcal{S}$ is large, the main bottleneck of Algorithm 1 is the linear system solve on line 12 (see Section 5). To improve the overall performances we use a geometrical multigrid approach which constructs a hierarchical representation of $\mathcal{S}$, and a coarse-to-fine minimization of the arap energy.

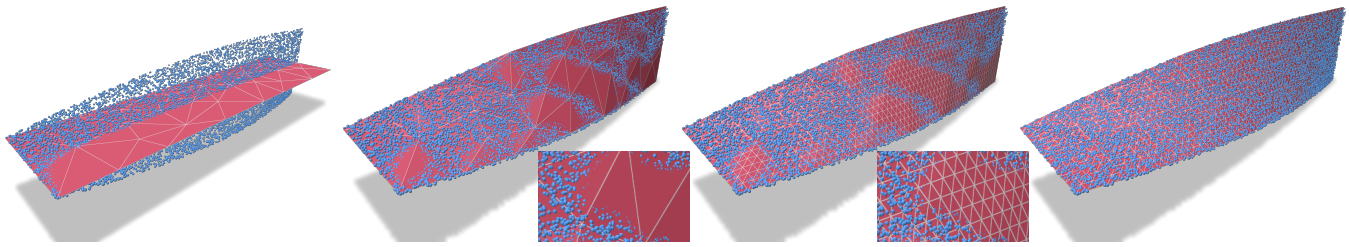
As first sketched in Section 3.2, $\mathcal{H}$ denotes a hierarchical view of $\mathcal{S}$ in which $\mathcal{S}^h$ denotes the h−th layer with vertices $\mathbf{s}^h_i$ (resp. normal vectors $\mathbf{n}^h_i$, faces $\mathbf{f}^h_k$ and Laplace-Beltrami operator Δ^h_{cotan}). Layers are sorted in ascending order (with respect to the number of vertices), the last layer being $\mathcal{S}$. Connections between two consecutive layers $\mathcal{S}^h$ and $\mathcal{S}^{h+1}$ (see Figure 4) are given by linking each vertex $\mathbf{s}^{h+1}_i$ to a face $\mathbf{f}^h_k$ and $\mathbf{s}^{h+1}_i$ local position within $\mathbf{f}^h_k$ is represented by the barycentric coordinates $(\alpha^{h+1}_i, \beta^{h+1}_i, \gamma^{h+1}_i)$ of its projection onto $\mathbf{f}^h_k$ along the face normal and the (possibly negative) height d^{h+1}_i from its projection such that:

$$d^{h+1}_i = (\mathbf{s}^{h+1}_i - \mathbf{p}_0) \cdot \mathbf{q} \quad (8)$$

$$\mathbf{s}^{h+1}_i - d^{h+1}_i \mathbf{q} = \alpha^{h+1}_i \mathbf{p}_0 + \beta^{h+1}_i \mathbf{p}_1 + \gamma^{h+1}_i \mathbf{p}_2 \,, \quad (9)$$

where $\mathbf{p}$ is one of the three vertices of $\mathbf{f}^h$, and $\mathbf{q}$ the face normal vector. With $\mathbf{s}^{h+1}_i$ local coordinates to $\mathbf{f}^h_k$ and given $\mathbf{f}^h_k$ updated vertices $\mathbf{p}'$ and normal $\mathbf{q}'$, $\mathbf{s}^{h+1}_i$ new position is simply:

$$\mathbf{s}^{h+1}_i = \alpha^{h+1}_i \mathbf{p}'_0 + \beta^{h+1}_i \mathbf{p}'_1 + \gamma^{h+1}_i \mathbf{p}'_2 + d^{h+1}_i \mathbf{q}' \,. \quad (10)$$



(a) (b) (c) (d)

Fig. 3. Overall hierarchical workflow: Starting from a coarse mesh $\mathcal{S}^0$ (*a*), we solve a first arap matching leading to the mesh $\mathcal{S}'^0$ (*b*). We then consider the mesh $\mathcal{S}^1$ with updated positions from $\mathcal{S}'^0$ (*c*), and repeat the minimization+optimization process to obtain the final surface $\mathcal{S}'$ after 3 iterations (*d*).

Since there is no inclusion between $\{\mathbf{s}^h\}$ (resp. $\{\mathbf{f}^h\}$) and $\{\mathbf{s}^{h+1}\}$ (resp. $\{\mathbf{f}^{h+1}\}$), the sparse operator Δ^{h+1}_{cotan} is simply reconstructed from $\mathcal{S}^{h+1}$ (linear time algorithm in the number of edges/vertices).

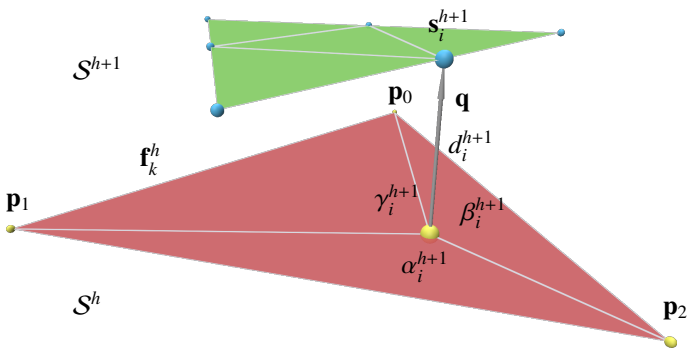


Fig. 4. Exploded view of the hierarchical structure: Vertex $\mathbf{s}_i^{h+1}$ of upper layer $\mathcal{S}^{h+1}$ is linked to face $\mathbf{f}_k^h$ of lower layer $\mathcal{S}^h$. Vertex $\mathbf{s}_i^{h+1}$ position within $\mathbf{f}_k^h$ is encoded the barycentric coordinates $(\alpha_i^{h+1}, \beta_i^{h+1}, \gamma_i^{h+1})$ of its projection and its signed distance d_i^{h+1} from $\mathbf{f}_k^h$.

Algorithm 2: Hierarchical arap registration

Input: $\mathcal{H} = \{\mathcal{S}^0 \dots \mathcal{S}^h\}$: hierarchical *source* surface mesh
$\mathcal{T} = (\{\mathbf{t}_i\}, \{\mathbf{m}_i\})$: *target* point cloud
ϵ: a stopping criterion

1 **foreach** $\mathcal{S}^h \in \mathcal{H}$ *with h ascending* **do**
  /* Register $\mathcal{S}^h$ to $\mathcal{T}$ */
2 $\mathcal{S}^{h\prime} \leftarrow$ ARAP_REGISTRATION$(\mathcal{S}^h, \mathcal{T}, \epsilon)$
3 **if** $\mathcal{S}^h$ *is last layer* **then**
4 **return** $\mathcal{S}^{h\prime}$
5 **else**
  /* Propagate $\mathcal{S}^{h\prime}$ to $\mathcal{S}^{h+1}$ */
6 **foreach** $\mathbf{s}_i^{h+1} \in \mathcal{S}^{h+1}$ **do**
7 compute $\mathbf{s}_i^{h+1}$ according to (10)
8 **end**
9 **end**
10 **end**

Given a set of desired vertices sizes, Garland and Heckbert's edge collapsing method [16] is applied to the original *source* $\mathcal{S}$ until all desired sizes are reached and results in $\mathcal{H} = \{\mathcal{S}^0 \dots \mathcal{S}^h\}$ without connection between layers. Because this method minimizes the distance between the vertices from collapsed edges and their neighboring faces, each vertex $\mathbf{s}^{h+1}$ of $\mathcal{S}^{h+1}$ is linked to $\mathcal{S}$ closest face $\mathbf{f}_k^h$.

Our hierarchical optimization of the arap registration of $\mathcal{S}$ onto $\mathcal{T}$ is presented in Algorithm 2. Given a hierarchy $\mathcal{H}$ of the *source* surface mesh $\mathcal{S}$ and the *target* point cloud $\mathcal{T}$, each layer $\mathcal{S}^h$ from the coarsest to the finest is registered to $\mathcal{T}$ (line 2) and the current registration result $\mathcal{S}'^h$ is propagated to the next layer $\mathcal{S}^{h+1}$ (line 7). As $\mathcal{S}^h$ has fewer vertices than $\mathcal{S}^k$ (for $k > h$), fewer nearest neighbor queries are required and the size of the Δ^h_{cotan} is smaller. The last iteration of Algorithm 2 is performed on the full resolution mesh $\mathcal{S}$. Note thanks to the hierarchical construction, very few optimization steps of Algorithm 1 are necessary to obtain a correct registration of this last step.

In the next section, the hierarchical optimization is experimented on data to evaluate the quality and the efficiency of the method.

5. Experiments

5.1. Dataset

The dataset (see Figure 5) is composed of three analytical models that mimic real-application deformations (*springback*, torsion and complex deformation), a car hood model (exhibiting high curvature features), and a CAD model corresponding to a real test case. To illustrate the *springback*, a hat shape whose parametrization is detailed in Appendix A is registered onto another hat. The hat shape is controlled by a bending factor and given its parametrization, two hats with different bending factor are isometric. For the torsion, a rectangle is mapped onto a helicoid twisted 90° whose parametrization is also detailed in Appendix A. The sheet case corresponds to the registration of a square to a deformed one generated by applying several rotation fields according to (5) and (6). The same method is used to generate the deformed car hood. Because the rotation fields used are manually chosen, there is no reason for them to be actual solutions of E^{arap} minimization.

Input scans for our analytical models and the hood have been generated by uniformly sampling each object and by adding Gaussian noise on the samples positions and normals. Given a triangular mesh, a face is randomly selected with an importance function proportional to the face area and the point lying on the

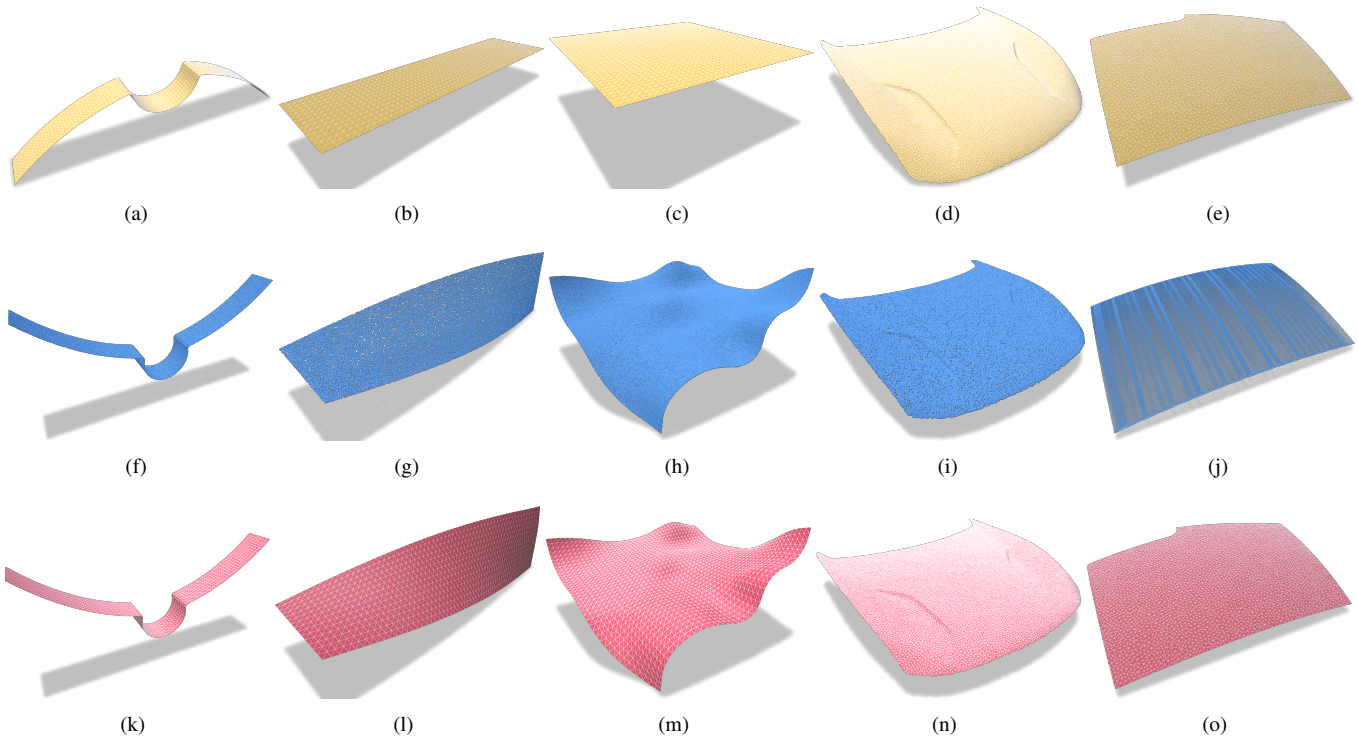


Fig. 5. Dataset: Yellow meshes are *source* models, blue point clouds correspond the *target* scans. Finally, red meshes are outputs we obtain after registration (see Table 1 for the settings). From left to right columns: two hats (a, f) under *springback* deformation, a mapping between a rectangle (b) and a twisted helicoid (g) (90° to mimic a torsion), the sheet (c, h) for a complex deformation and a car hood model (d) with simulated scan (i). The last column (e, j, o) corresponds to a real test case of a CAD model and a scan of the real metal sheet. Meshes and point clouds are downsampled for visualization purposes.

face is selected by uniformly sampling its barycentric coordinates. The normal vector of the point is set to the normal vector of the face it belongs to, and this process is repeated until the desired number of points M has been sampled. The positions are perturbed by adding to their coordinates a noise sampled from a Gaussian distribution $(\mu = 0, \sigma_{coord})$ and the normals are perturbed by a rotation (θ, ϕ) [1] with θ sampled from an uniform distribution $[0, \pi[$ and ϕ sampled from a Gaussian distribution $(\mu = 0, \sigma_{angle})$.

Finally, the scan of the CAD model is a measure of the real piece which undergoes a *springback* deformation. Because of the machining context, the actual *target* surface (which corresponds to the *source* after deformation) is yet unknown. To ensure that no part is missing, the scanned surface is bigger and so the arap registration is performed with a *source* being a sub-part of the *target* point cloud.

For all experiments, the dataset has been normalized such that the diagonal of *source* bounding box measures 1. Our algorithm has been implemented in C++ with the libraries eigen [17] and libigl [18] for the data structures and geometry processing, nanoflann [19] for the kdTree and polyscope [20] for the visualization. All tests were run on a 4.0 GHz PC with 16 GB RAM.

For each experiment (*e.g.* Table 1) we measure the total number of iterations n_{iter} which is the sum of Algorithm 1 iterations for all layers, the final E^{prox} and E^{arap}, the initialization step duration T_{init} (construction of $\mathcal{H}$ and the Laplace-Beltrami operator Δ_{cotan} computations), the iterative step duration T_{core} which is the execution time of Algorithm 2. The assignment step duration T_{NN} is the timing of all nearest-neighbor queries on $\mathcal{T}$ (accumulated for all layers), T_{opt} denotes the remaining optimization timing of $\mathcal{S}$ (also accumulated for all layers). Hence, $T_{core} = T_{NN} + T_{opt}$ and the final timing is given by $T_{total} = T_{init} + T_{core}$. We denote a_{total} (resp. a_{core}), the speedup between the hierarchical and the nonhierarchical variants of the algorithms (in total and for the core part only).

Hat and helicoid sources have both $N = 10^6$ vertices and their targets have $M = 5 \times 10^6$ points. The sheet source has $N = 4 \times 10^5$ vertices and its target has $M = 2.5 \times 10^6$ points. The CAD model has $N = 8 \times 10^5$ vertices and its target has $M = 1.4 \times 10^6$ points. The number of layers for the hierarchical registration is set to 3 with the first, second and last layer having respectively $N/100$, $N/10$ and N vertices. The stopping criterion d and the maximum number of iterations of Algorithm 1 per layer are set to 10^{-6} and 100.

5.2. Accuracy and Efficiency

In this experiment the hierarchical optimization is evaluated by comparing for each dataset case, the registration results with and without the hierarchy. The results for each case are displayed in Table 1.

For the hat object, both global and hierarchical approaches produce similar results in terms of energy. However, the hierarchical registration is 43 times faster in total and 115 times

[1] θ being the azimuthal angle and ϕ the polar angle.

Case	sizes (N, M)	n_{iter}	E^{prox}	E^{arap}	T_{total}	T_{init}	T_{core}		a_{total}	a_{core}
							T_{NN}	T_{opt}		
hat	$N = 10^6$	17	$\mathbf{1.3 \times 10^{-2}}$	5.1×10^{-6}	1734	**2 (0.1%)**	1732 (99.9%)		43	115
							1373	361		
hat hierarchical	$M = 5 \times 10^6$	**12**	1.9×10^{-2}	$\mathbf{4.6 \times 10^{-6}}$	**40**	25 (62.5%)	**15 (37.5%)**			
							12	3		
helicoid	$N = 10^6$	**100**	2.5×10^{-1}	5.0×10^{-3}	514	**2 (0.4%)**	512 (99.6%)		7	10
							288	224		
helicoid hierarchical	$M = 5 \times 10^6$	136	$\mathbf{1.0 \times 10^{-1}}$	$\mathbf{2.0 \times 10^{-6}}$	**72**	25 (34.7%)	**47 (65.3%)**			
							33	14		
sheet	$N = 4 \times 10^5$	**7**	1.3	1.8×10^{-2}	107	**1 (0.9%)**	106 (99.1%)		4	8
							51	55		
sheet hierarchical	$M = 2.5 \times 10^6$	40	$\mathbf{1.3 \times 10^{-1}}$	$\mathbf{2.7 \times 10^{-3}}$	**24**	11 (45.8%)	**13 (54.2%)**			
							3	10		
hood	$N = 8 \times 10^5$	**8**	3.7×10^{-1}	2.5×10^{-3}	311	**1 (0.3%)**	310 (99.7%)		1.5	1.7
							55	255		
hood hierarchical	$M = 5 \times 10^6$	18	$\mathbf{3.7 \times 10^{-2}}$	$\mathbf{1.0 \times 10^{-3}}$	**201**	21 (10.4%)	**180 (89.6%)**			
							7	173		
CAD	$N = 8 \times 10^5$	**100**	6.7×10^{-1}	4.5×10^{-4}	2065	**1 (0.04%)**	2065 (99.96%)		21	28
							38	2027		
CAD hierarchical	$M = 1.4 \times 10^6$	150	$\mathbf{6.3 \times 10^{-1}}$	$\mathbf{3.8 \times 10^{-6}}$	**94**	21 (22.3%)	**73 (81.7%)**			
							3	70		

Table 1. Metrics of nonhierarchical and hierarchical registration on the dataset after convergence (with stopping criterion $d = 10^{-6}$ – see Algorithm 1 – and number of iterations per layer limited to 100). Metrics are n_{iter} the sum of Algorithm 1 iterations for all layers, the final E^{prox} and E^{arap}, the total duration T_{total}, the initialization step and iterative step duration T_{init} and T_{core} ($T_{total} = T_{init} + T_{core}$), the assignment steps and optimization step T_{NN} and T_{opt} ($T_{core} = T_{NN} + T_{opt}$) and the total and iteration step speedups a_{total} and a_{core}. Times are expressed in seconds.

faster if we only consider the iterative part. More detailed measures show that iterations with a bad alignment onto $\mathcal{T}$ (*i.e.* with large E^{prox}) were performed on the smaller layers. Even though the hierarchical registrations of the helicoid, the sheet and the CAD models have smaller speed ups and required more iterations than their nonhierarchical versions, they both produce mappings with lower arap energies. This can be due to the fact that optimizing the energies in a coarse-to-fine scheme make the process unlikely to get stuck on local minima.

For the sheet and hood cases for which the deformations are not fully arap, the hierarchical method produces better E^{prox} than the nonhierarchical one. The speedup for the hierarchical car hood (1.5) is less favorable than for the other shapes as several high resolution iterations are required to cope with the high curvature features. Figure 6 shows E^{prox} and E^{arap} distributions on the sheet case.

Results in Table 2 show the impact on the *target* point cloud size to the arap registration computation time. For all analytical cases, we thus increase the sample count of the target. The assignment time T_{NN} (which is a function of N and M) scales up with M thanks to the kdTree structure used for nearest neighbor queries, while the optimization time T_{opt} which is a function of N is constant as expected.

5.3. Robustness to noise

In this experiment, we evaluate the robustness to perturbations on the positions and normal vectors of $\mathcal{T}$ on the hat case (hierarchical approach). To evaluate the robustness with respect to the perturbations, we consider the l_2-norm between the deformed surface $\mathcal{S}^*$ vertices for the the noise-free target $\mathcal{T}$ (the groundtruth for this test), and the surface obtained on the perturbed $\mathcal{T}$:

$$E^*(\mathcal{S}, \mathcal{S}^*) := \sum_i \|\mathbf{s}_i - \mathbf{s}_i^*\|^2 . \tag{11}$$

When perturbing the normal vectors, experiments (Figure 7) show that Algorithm 1 is stable for small perturbations (closer to our use-case). For extreme cases with strong perturbations (30° (*c*) and 60°), the output surface becomes highly impacted. When perturbing the sample positions, (Table 3 and Figure 8), the algorithm produces very stable solutions.

Finally, in Figure 9, we provide additional experiments when we consider non-uniform point distributions for the target point cloud $\mathcal{T}$. Note that the last case ($c - d$) exactly corresponds to the kind of point clouds we are facing in inverse engineering when $\mathcal{T}$ is given by a laser range or a probing device. Our method can efficiently handles these cases.

5.4. Failure example

In this experiment we apply our registration method on data that are slightly out-of-scope of the reverse engineering context. As illustrated in Figure 10, our method does not produce satisfactory results on complex geometries, due to the assignment step. In the bunny case (see Figure 10(c)) the two sides of the *source* left ear are assigned on the opposite sides of the *target* two ears. With this assignment the two sides of the *source* ear should be flipped. The two sides of the *source* right ear

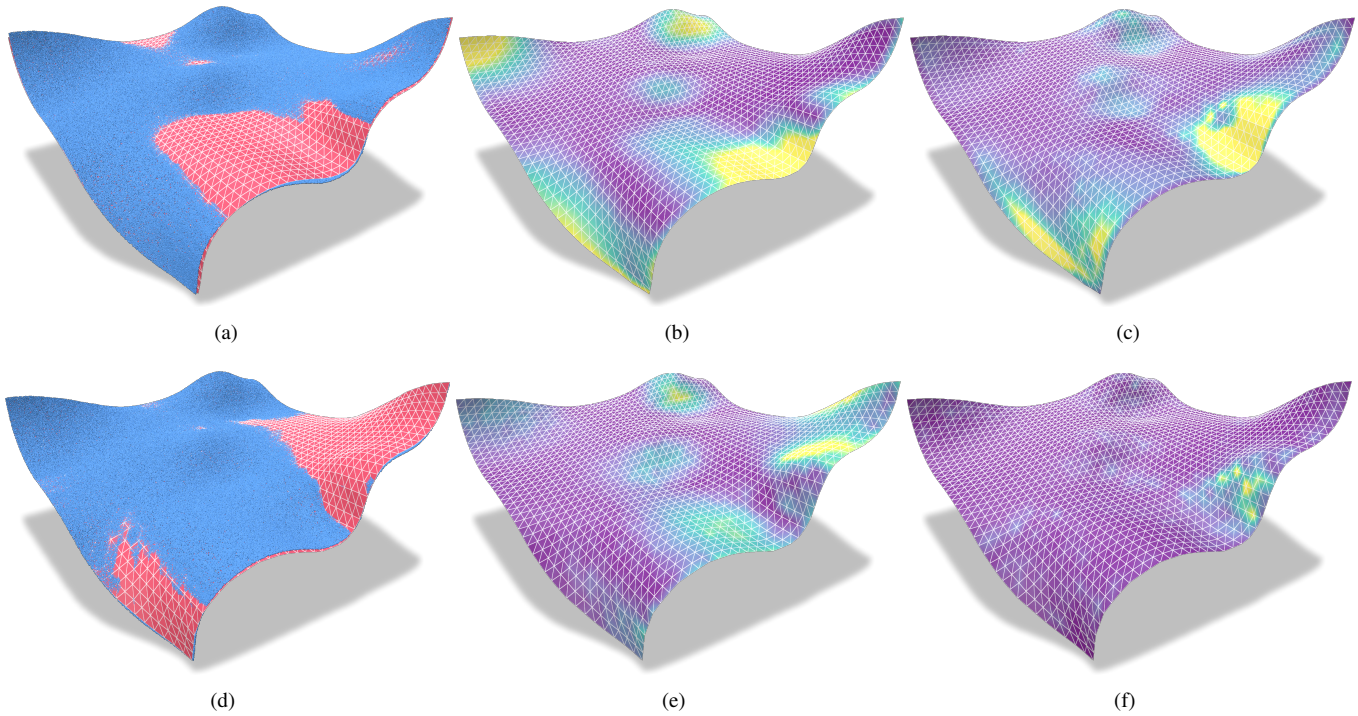


Fig. 6. Sheet registration results: Top row displays nonhierarchical results, bottom row the hierarchical results. From left to right the results are registration output (with $\mathcal{T}$ in blue and S' in red), the E^{prox} per vertex and the E^{arap} per vertex. E^{prox} (resp. E^{arap}) share the same colormap for both versions with purple (resp. yellow) as lower (resp. upper) bound.

Case	N	M	1×10^6	5×10^6	10×10^6	15×10^6	20×10^6	25×10^6
hat	$N = 10^6$	T_{NN}	3	13	20	28	34	40
		T_{opt}	4	10	11	12	11	11
helicoid	$N = 10^6$	T_{NN}	23	38	56	70	81	96
		T_{opt}	17	16	15	17	17	18
sheet	$N = 4 \times 10^5$	T_{NN}	2	6	8	11	11	12
		T_{opt}	16	17	17	16	18	17

Table 2. Scan size impact on hierarchical registration duration: For all cases the *source* vertex number N is fixed, M is the number of points in the *target* scan and T_{NN} and T_{opti} the total amounts of time spent on assignment and optimization steps (see Section 4.1 for step details).

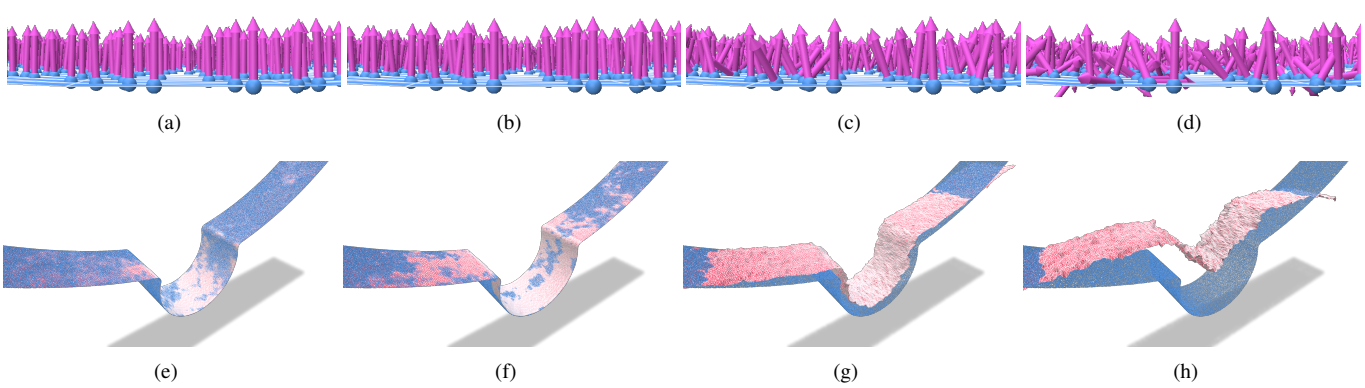


Fig. 7. Registration results with perfect position and noisy normals: The normals are perturbed using a Gaussian noise as specified in Section 5.1 with σ_{angle} equals from left to right 3° (*a*), 6° (*b*), 30° (*c*) and 60° (*d*). Top row displays the normal distribution and bottom row the registration output $\mathcal{S}'$. With small enough noises (*e.g.* 3° (*a*) and 6° (*b*)) the mapping is not smooth but consistent while with too big noises (*e.g.* 30° (*c*) and 60° (*d*)) the algorithm produces degenerated results.

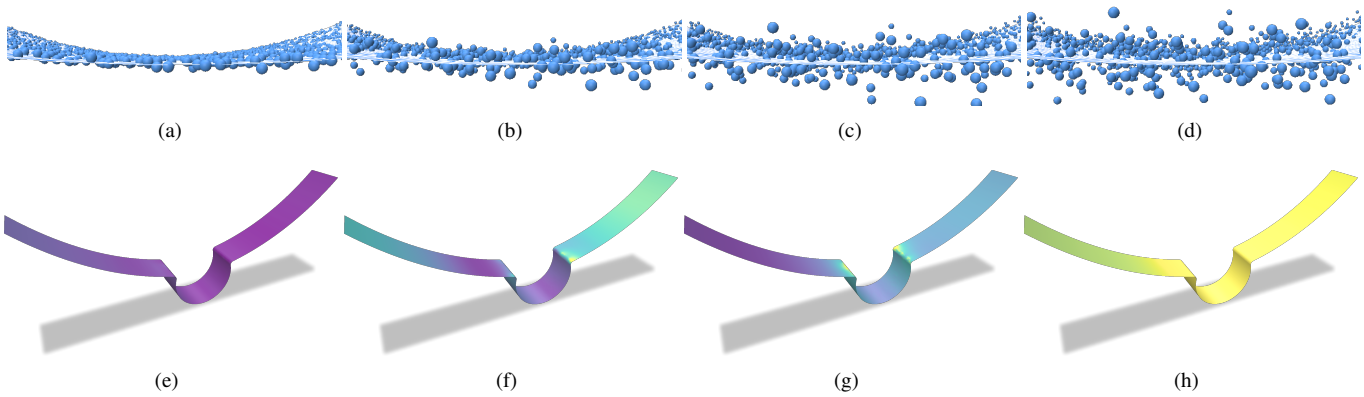


Fig. 8. Registration results with noisy positions and perfect normals: The positions are perturbed using a Gaussian noise as specified in Section 5.1 with σ_{coord} equals from left to right 0.1% (a), 0.4% (b), 0.7% (c) and 1% (d) of the diagonal length of the point cloud bounding box. Top row displays the position distribution and bottom row the registration output $\mathcal{S}'$ colored by E^* per vertex, E^* share the same colormap for all σ_{coord} with purple (resp. yellow) as lower (resp. upper) bound. See Table 1 for detailed metrics.

σ_{coord}	n_{iter}	E^{prox}	E^{arap}	E^*	T_{total} (sec)
0%	16	1.3×10^{-2}	3.6×10^{-6}	0	48
0.1%	**23**	$\mathbf{1.5 \times 10^{-2}}$	$\mathbf{7.0 \times 10^{-6}}$	4.5×10^{-3}	**43**
0.4%	81	$\mathbf{3.2 \times 10^{-2}}$	$\mathbf{4.7 \times 10^{-6}}$	3.0×10^{-2}	100
0.7%	174	4.7×10^{-2}	1.2×10^{-4}	2.3×10^{-2}	179
1%	290	5.9×10^{-2}	2.5×10^{-4}	8.5×10^{-2}	297

Table 3. Registration results with noisy positions and perfect normals after convergence: σ_{coord} is expressed as percent of the diagonal length of the point cloud bounding box (see Table 1 for column details). See Figure 8 for visualizations of E^*. Under $\sigma_{coord} = 0.4\%$ the algorithm produces solutions with similar E^{arap} and E^{prox} (which is affected by σ_{coord}). Under $\sigma_{coord} = 0.1\%$ the algorithm produces a solution quasi-equal to S^* in the same amount of time and iterations.

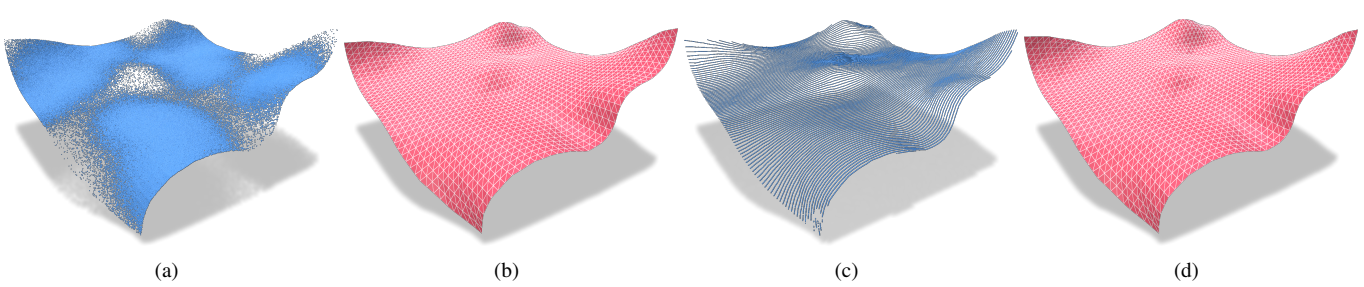


Fig. 9. Robustness to different point distributions: When the sampling density is not uniform (a) or when its has high aliasing structure when simulating probing/laser based acquisition devices (c), our approach still provides stable and accurate outputs (b) and (c).

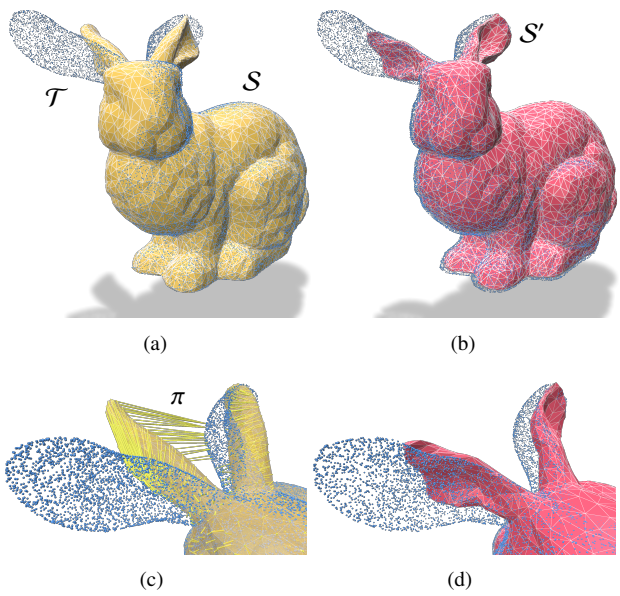


Fig. 10. Failed bunny registration: Registration between two Stanford bunnies (*a*) with one bunny having bent ears. The algorithm returns a bunny with degenerated ears (*b*). Bad assignments (*c*) produce inconsistent rotations leading to an inconsistent solution (*d*).

are assigned on the same side of one *target* ear, so one side of the *source* ear should also be flipped. Theses assignments produce areas where two neighboring vertices could have opposite normals leading to a inconsistent rotation field $\mathcal{R}$, and inconsistent **b** and a degenerated (6) solution (*e.g.* bunny result and Figure 7(h)). Note that the overall algorithm (assignment/arap alternate steps, hierarchical construction...) perfectly handles this case but providing a better assignment for these complex shapes (mimicking the one in [13] without scarfing the speed of our approach) is a challenging future work.

6. Conclusion

In this paper, we have presented a new method of non-rigid isometric registration of a source triangle mesh onto a target point cloud that is fast, accurate and can handle large geometries. Our solution is based on a hierarchy of meshes built from the source triangle mesh. Our algorithm alternates between an assignment and an arap optimization step, starting from the coarsest mesh of the hierarchy, and going up progressively each time the solution with the current level has converged. Our experiments show a very highest speed up compared to the non-hierarchical approach, while ensuring the same accuracy.

Several future works exist. As mentioned in Section 5.4, the assignment step could be improved to be able to handle more complex shapes in terms of geometry or topology. We would also consider alternative schemes for the hierarchical construction (for instance to keep some user-specified features, or to preserve some spectral properties similarly to [21]). The challenge in these future works would be to keep a solution that scales up with respect to the size of the *source* and the *target*, while keeping the accuracy of the reconstruction as required in our inverse engineering context.

Appendix A. Hat and helicoid parametrization

By definition a ruled surface can be described by a parametrization $p(u, v) = c(u) + v * r(u)$ with p the surface vertices, c its directrix curve and r its generator. Hat directrix c_{hat} is controlled by a bending factor b and its piecewise definition is given on each piece i by:

$$c_{hat}(u, b) := \begin{bmatrix} L_i(b)\, cos(\omega_i(b)\, u + \phi_i(b)) \\ L_i(b)\, sin(\omega_i(b)\, u + \phi_i(b)) \\ 0 \end{bmatrix} + P_i(b) \qquad \text{(A.1)}$$

with L_i, ω_i, ϕ_i and P_i chosen such that c_{hat} is at least C^1 and isometric for any b. Hat generator r_{hat} is a simple constant $[0, 0, W]$.

The helicoid is a well known ruled surface, like with the hat we introduce a bending factor b to control its deformation such that:

$$c_{helicoid}(u) := \begin{bmatrix} 0 \\ 0 \\ W\, u \end{bmatrix} \text{and } r_{helicoid}(u) := \begin{bmatrix} L\, cos(b\, u) \\ L\, sin(b\, u) \\ 0 \end{bmatrix}. \qquad \text{(A.2)}$$

References

[1] Kleiner, M, Tekkaya, AE, Chatti, S, Hermes, M, Weinrich, A, Ben-Khalifa, N, et al. New incremental methods for springback compensation by stress superposition. Production Engineering 2009;3(2):137–144. URL: https://doi.org/10.1007/s11740-009-0151-7. doi:10.1007/s11740-009-0151-7.

[2] Tam, G, Cheng, ZQ, Lai, YK, Langbein, F, Liu, Y, Marshall, D, et al. Registration of 3d point clouds and meshes: A survey from rigid to nonrigid. IEEE transactions on visualization and computer graphics 2013;19:1199–217. URL: https://doi.org/10.1109/TVCG.2012.310. doi:10.1109/TVCG.2012.310.

[3] Sorkine, O, Alexa, M. As-Rigid-As-Possible Surface Modeling. Symposium on Geometry Processing; 2007. URL: https://doi.org/10.1145/1281991.1282006. doi:10.1145/1281991.1282006.

[4] Chao, I, Pinkall, U, Sanan, P, Schröder, P. A simple geometric model for elastic deformations. ACM transactions on graphics (TOG) 2010;29(4):1–6.

[5] Levi, Z, Gotsman, C. Smooth rotation enhanced as-rigid-as-possible mesh animation. IEEE Transactions on Visualization and Computer Graphics 2015;21(2):264–277. URL: http://www.scopus.com/inward/record.url?scp=84920861602&partnerID=8YFLogxK. doi:10.1109/TVCG.2014.2359463; article.

[6] Gotsman, C, Liu, L, Zhang, L, Xu, Y, Gortler, S. A local/global approach to mesh parameterization. Computer Graphics Forum 2008;27. URL: https://doi.org/10.1111/j.1467-8659.2008.01290.x. doi:10.1111/j.1467-8659.2008.01290.x.

[7] Smith, B, Goes, F, Kim, T. Analytic eigensystems for isotropic distortion energies. ACM Transactions on Graphics 2019;38:1–15. URL: https://doi.org/10.1145/3241041. doi:10.1145/3241041.

[8] Ovsjanikov, M, Ben-Chen, M, Solomon, J, Butscher, A, Guibas, L. Functional maps: A flexible representation of maps between shapes. ACM Transactions on Graphics - TOG 2012;31. URL: https://doi.org/10.1145/2185520.2185526. doi:10.1145/2185520.2185526.

[9] Sacharow, A, Balzer, J, Biermann, D, Surmann, T. Non-rigid isometric icp: A practical registration method for the analysis and compensation of form errors in production engineering. Computer-Aided Design 2011;43(12):1758–1768. URL: https://www.sciencedirect.com/science/article/pii/S0010448511001849. doi:https://doi.org/10.1016/j.cad.2011.07.007.

[10] Yu, Y, Zhou, K, Xu, D, Shi, X, Bao, H, Guo, B, et al. Mesh editing with poisson-based gradient field manipulation. ACM Transactions on Graphics 2004;23. URL: https://doi.org/10.1145/1186562.1015774. doi:10.1145/1186562.1015774.

[11] Schweinoch, M, Schäfer, R, Sacharow, A, Biermann, D, Buchheim, C. A non-rigid registration method for the efficient analysis of shape deviations in production engineering applications. Production Engineering 2016;10(2):137–146. URL: https://doi.org/10.1007/s11740-016-0660-0. doi:10.1007/s11740-016-0660-0.

[12] Klein, L, Wagner, T, Buchheim, C, Biermann, D. A procedure for the evaluation and compensation of form errors by means of global isometric registration with subsequent local reoptimization. Production Engineering 2014;8(1):81–89. URL: https://doi.org/10.1007/s11740-013-0510-2. doi:10.1007/s11740-013-0510-2.

[13] Huang, Q, Adams, B, Wicke, M, Guibas, L. Non-rigid registration under isometric deformations. Comput Graph Forum 2008;27:1449–1457. URL: https://doi.org/10.1111/j.1467-8659.2008.01285.x. doi:10.1111/j.1467-8659.2008.01285.x.

[14] Cazals, F, Pouget, M. Estimating Differential Quantities Using Polynomial Fitting of Osculating Jets. In: Kobbelt, L, Schröder, P, Hoppe, H, editors. Eurographics Symposium on Geometry Processing. Aachen, Germany: Eurographics; 2003, p. 177–187. URL: https://hal.archives-ouvertes.fr/hal-00103047.

[15] Meyer, M, Desbrun, M, Schr, P, Barr, A. Discrete differential-geometry operators for triangulated 2-manifolds. Proceedings of Visualization and Mathematics 2001;3. URL: https://doi.org/10.1007/978-3-662-05105-4_2. doi:10.1007/978-3-662-05105-4_2.

[16] Garland, M, Heckbert, P. Surface simplification using quadric error metrics. Proceedings of the ACM SIGGRAPH Conference on Computer Graphics 1997;1997. URL: https://doi.org/10.1145/258734.258849. doi:10.1145/258734.258849.

[17] Guennebaud, G, Jacob, B, et al. Eigen v3. http://eigen.tuxfamily.org; 2010.

[18] Jacobson, A, Panozzo, D, et al. libigl: A simple C++ geometry processing library. 2018. Https://libigl.github.io/.

[19] Blanco, JL, Rai, PK. nanoflann: a C++ header-only fork of FLANN, a library for nearest neighbor (NN) with kd-trees. https://github.com/jlblancoc/nanoflann; 2014.

[20] Sharp, N, et al. Polyscope. 2019. Www.polyscope.run.

[21] Lescoat, T, Liu, HTD, Thiery, JM, Jacobson, A, Boubekeur, T, Ovsjanikov, M. Spectral mesh simplification. In: Computer Graphics Forum; vol. 39. Wiley Online Library; 2020, p. 315–324.