remote sensing

MDPI

Article

Nonrigid Point Cloud Registration Using Piecewise Tricubic Polynomials as Transformation Model

Philipp Glira [1,*], Christoph Weidinger [1], Johannes Otepka-Schremmer [2], Camillo Ressl [2], Norbert Pfeifer [2] and Michaela Haberler-Weber [3]

[1] AAS Assistive and Autonomous Systems, AIT Austrian Institute of Technology, 1210 Vienna, Austria; christoph.weidinger@ait.ac.at
[2] Photogrammetry Research Group, Vienna University of Technology, 1040 Vienna, Austria; johannes.otepka@geo.tuwien.ac.at (J.O.-S.); camillo.ressl@geo.tuwien.ac.at (C.R.); norbert.pfeifer@geo.tuwien.ac.at (N.P.)
[3] ÖBB Austrian Federal Railways, 1020 Vienna, Austria
* Correspondence: philipp.glira@ait.ac.at

Abstract: Nonrigid registration presents a significant challenge in the domain of point cloud processing. The general objective is to model complex nonrigid deformations between two or more overlapping point clouds. Applications are diverse and span multiple research fields, including registration of topographic data, scene flow estimation, and dynamic shape reconstruction. To provide context, the first part of the paper gives a general introduction to the topic of point cloud registration, including a categorization of existing methods. Then, a general mathematical formulation for the point cloud registration problem is introduced, which is then extended to address also nonrigid registration methods. A detailed discussion and categorization of existing approaches to nonrigid registration follows. In the second part of the paper, we propose a new method that uses piecewise tricubic polynomials for modeling nonrigid deformations. Our method offers several advantages over existing methods. These advantages include easy control of flexibility through a small number of intuitive tuning parameters, a closed-form optimization solution, and an efficient transformation of huge point clouds. We demonstrate our method through multiple examples that cover a broad range of applications, with a focus on remote sensing applications—namely, the registration of airborne laser scanning (ALS), mobile laser scanning (MLS), and terrestrial laser scanning (TLS) point clouds. The implementation of our algorithms is open source and can be found our public repository.

Keywords: point cloud registration; iterative closest point; transformation; lidar

Citation: Glira, P.; Weidinger, C.; Otepka-Schremmer, J.; Ressl, C.; Pfeifer, N.; Haberler-Weber, M. Nonrigid Point Cloud Registration Using Piecewise Tricubic Polynomials as Transformation Model. *Remote Sens.* **2023**, *15*, 5348. https://doi.org/10.3390/rs15225348

Academic Editor: Guoqing Zhou

Received: 10 October 2023
Revised: 5 November 2023
Accepted: 6 November 2023
Published: 13 November 2023

1. Introduction

The registration of point clouds, i.e., a set of 2D or 3D points in object space, is relevant in many application domains, e.g., remote sensing, computer vision, robotics, autonomous driving, or healthcare. The general objective is to minimize the distances between overlapping point clouds. To achieve this, some kind of geometric transformation $\mathcal{T}$ is estimated and applied individually to each nonfixed point cloud. The transformed point clouds can be regarded as optimally registered if the residual distances are purely random, i.e., if they are nonsystematic. In case a rigid-body transformation is not sufficient to model the discrepancies between the point clouds, a nonrigid transformation is needed—an example is shown in Figure 1.

Most point cloud registration methods are inspired indubitably by the works of Besl and McKay [1] and Chen and Medioni [2], who introduced approximately at the same time the iterative closest point (ICP) algorithm. It is used to improve the alignment of two point clouds by minimizing iteratively the distances within the overlap area of these point clouds. Nowadays, the term ICP does not necessarily refer to the algorithm presented in these original publications, but rather to a group of point cloud registration algorithms that

have in common the following aspects: (I) correspondences are established iteratively; (C) the closest point, or more generally, the corresponding point, is used as correspondence; and (P) correspondences are established on a point basis [3].

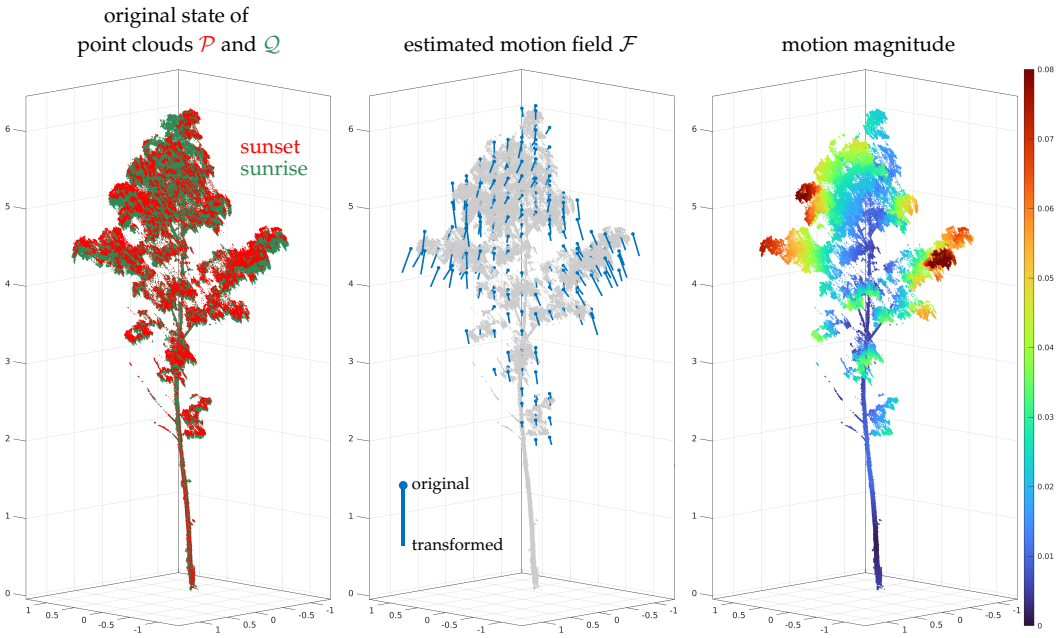

Figure 1. Example for a nonrigid registration between two point clouds. Our method is used here to estimate the motion of a maple tree between sunset and sunrise. The motion field $\mathcal{F}$ is scaled for better visualization. Details can be found in Section 6.4. All units in meter.

A general taxonomy for ICP-based algorithms was introduced by Rusinkiewicz and Levoy [4]—we follow this recommendation throughout this paper. Accordingly, a traditional point cloud registration pipeline can be roughly divided into five stages, cf. Figure 2. For the registration of a fixed point cloud $\mathcal{Q}$ and a loose point cloud $\mathcal{P}$, these stages are

1. **Selection**: A subset of points (instead of using each point) is selected within the overlap area in one point cloud [3]. For this, the fixed point cloud $\mathcal{Q}$ is typically chosen.
2. **Matching**: The points, which correspond to the selected subset, are determined in the other point cloud, typically the loose point cloud $\mathcal{P}$.
3. **Rejection**: False correspondences (outliers) are rejected on the basis of the compatibility of points. The result of these first three stages is a set of correspondences $\mathcal{C}$ with an associated set of weights $\mathcal{W}_\mathcal{C}$.
4. **Optimization**: The transformation $\mathcal{T}$ for the loose point cloud is estimated by minimizing the weighted and squared distances (e.g., the Euclidean distances) between corresponding points.
5. **Transformation**: The estimated transformation $\mathcal{T}$ is applied to the loose point cloud: $\mathcal{T}(\mathcal{P})$.

Finally, a suitable convergence criterion is tested. If it is not met, a new iteration restarts from the matching stage using the transformed loose point cloud $\mathcal{T}(\mathcal{P})$. The iterative nature of the ICP algorithm results from the following basic assumption: in the first iteration, correspondences are often imperfect due to a typically relatively large displacement of the two point clouds. With each transformation of the loose point cloud $\mathcal{P}$, however, the correspondence assignments get better. Thus, this process is repeated until the correspondences become stable, i.e., until the variations become statistically insignificant. In this case, convergence is assumed to be achieved and the algorithm ends.

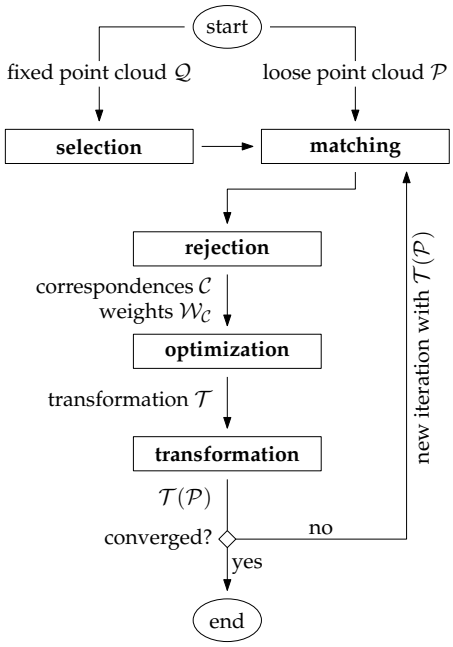

Figure 2. ICP-based point cloud registration pipeline.

1.1. Variants of Point Cloud Registration Algorithms

For each of the five stages, multiple variations have been proposed in the past for many different applications—literature surveys can be found in [5–8]. As a brief review, point cloud registration algorithms can be roughly classified according to the following properties

- **Coarse registration** vs. **fine registration**: Often the initial relative orientation of the point clouds is unknown in advance, e.g., if an object or a scene is scanned from multiple arbitrary view points. The problem of finding an initial transformation between the point clouds in the global parameter space is often denoted as coarse registration. Solutions to this problem are typically heavily based on matching of orientation-invariant point descriptors [9]. The 3DMatch benchmark introduced by [10] evaluates the performance of 2D and 3D descriptors for the coarse registration problem. Once a coarse registration of the point clouds is found that lies in the convergence basin of the global minima, a local optimization, typically some variant of the ICP algorithm, can be applied for the fine registration. It is noted that in case of multisensor setups, the coarse registration is often observed by means of other sensor modalities. For instance, in case of dynamic laser scanning systems, e.g., airborne laser scanning (ALS) or mobile laser scanning (MLS), the coarse registration between overlapping point clouds is directly given through the GNSS/IMU trajectory of the platform—in such cases, only a refinement of the point cloud registration is needed, e.g., by strip adjustment or (visual-)lidar SLAM (see below).
- **Rigid transformation** vs. **nonrigid transformation**: Rigid methods apply a rigid-body transformation to one of the two point clouds to improve their relative alignment. A rigid-body transformation has 3/6 degrees of freedom (DoF) in 2D/3D and is usually parameterized through a 2D/3D translation vector and 1/3 Euler angles. In contrast, nonrigid methods have usually a much higher number of DoF in order to model more complex transformations. Consequently, the estimation of a nonrigid transformation field requires a much larger number of correspondences. Another challenging problem is the choice of a proper representation of the transformation field: on the one hand, it must be flexible enough to model systematic discrepancies between the point clouds, and on the other hand, overfitting and excessive computational costs must be avoided. We will discuss these and other aspects in Sections 1.2 and 3.

- **Traditional** vs. **learning based**: Traditional methods are based entirely on handcrafted, mostly geometric relationships. This may also include the design of handcrafted descriptive point features used in the matching step. Recent advances in the field of point cloud registration, however, have been clearly dominated by deep-learning-based methods—a recent survey is given by [11]. Such methods are especially useful for finding a coarse initial transformation between the point clouds, i.e., to solve the coarse registration problem. In such scenarios, deep-learning-based methods typically lead to a better estimate of the initial transformation by automatically learning more robust and distinct point feature representations. This is particularly useful in the presence of repetitive or symmetric scene elements, weak geometric features, or low-overlap scenarios [6]. Recently, deep-learning-based methods have also been published for the *nonrigid* registration problem, e.g., HPLFlowNet [12] or FlowNet3D [13].
- **Pairwise** vs. **multiview**: The majority of registration algorithms can handle a single pair of point clouds only. In practice, however, objects are typically observed from multiple viewpoints. As a consequence, a single point cloud generally overlaps with >1 other point clouds. In such cases, a global (or joint) optimization of all point clouds is highly recommended. Such an optimization problem is often interpreted as a graph where each node corresponds to an individual point cloud with associated transformation and the edges are either the correspondences themselves (single-step approach, e.g., [14]) or the pairwise transformations estimated individually in a preprocessing step (two-step approach, e.g., [15–17]).
- **Full overlap** vs. **partial overlap** Many algorithms (particularly also in the context of nonrigid transformations, e.g., [18,19]) assume that the two point clouds are fully overlapping. However, in practice, a single point cloud often corresponds only to a small portion of the observed scene, e.g., when scanning an object from multiple viewpoints. It is particularly difficult to find valid correspondences (under the assumption that the point clouds are *not* roughly aligned) in low-overlap scenarios, e.g., point clouds with an overlap below 30%. This challenge is addressed by [7] and the therein introduced *3DLoMatch* benchmark, where the algorithm by [20] currently leads to the best results.
- **Approximative** vs. **rigorous**: Most registration algorithms are approximative in the sense that they use the 2D or 3D point coordinates as inputs only and try to minimize discrepancies across overlapping point clouds by applying a rather simple and general (rigid or nonrigid) transformation model. Ref. [21] describes this group of algorithms as rubber-sheeting coregistration solutions. In contrast, rigorous solutions try to model the point cloud generation process as accurately as possible by going a step backwards and using the sensor's raw measurements. The main advantage of such methods is that point cloud discrepancies are corrected at their source, e.g., by sensor self-calibration of a miscalibrated lidar sensor [22]. Rigorous solutions are especially important in case of point clouds captured from moving platforms, e.g., robots, vehicles, drones, airplanes, helicopters, or satellites. In a minimal configuration, such methods simultaneously register overlapping point clouds and estimate the trajectory of the platform. More sophisticated methods additionally estimate intrinsic and extrinsic sensor calibration parameters and/or consider ground truth data, e.g., ground control points (GCPs), to improve the georeference of the point clouds. If point clouds need to be generated online, e.g., in robotics, this type of problem is addressed by SLAM (simultaneous localization and mapping), and especially lidar SLAM [23] and visual-lidar SLAM [24] methods. For offline point cloud generation, however, methods are often summarized under the term (rigorous) strip adjustment, as the continuous platform's trajectory is often divided into individual strips for easier data handling—an overview can be found in [21,25].
- **2D** or **3D**: Finally, it should be noted that many early highly cited algorithms, especially for the *nonrigid* registration problem, have originally been introduced for 2D point

clouds only, e.g., [18,26]. However, it is usually rather straightforward to extend these methods to the third dimension.

Classification of our method: The features of our method are: fine registration, nonrigid transformation, traditional, pairwise, partial and full overlap, approximative, 2D or 3D. However, it is emphasized that the core of this contribution is the nonrigid transformation framework. Within the entire point cloud registration pipeline, subcomponents can be relatively easily replaced at different ICP stages, e.g., usage of learning-based correspondences instead of using simply nearest neighbor correspondences or an extension from pairwise to multiview alignment.

1.2. Motivation for Nonrigid Transformations

There are many cases where nonrigid transformation models can be helpful. Typical use cases are dynamic shape reconstruction [27], registration of medical images or surfaces [19,28,29], estimation of scene flow [12,13], or registration of lidar point clouds of dynamic environments, e.g., for change detection [6]. In the remainder of this subsection, we would like to describe in more detail an important use case in the field of remote sensing, namely, the *registration of historical ALS data*. However, we want to stress that due to the general character of our method, it is applicable in many other areas, both 2D and 3D, cf. Sections 6.2–6.6.

Many public and private archives containing historical ALS data exist. A quality control procedure often reveals large discrepancies between the point clouds of overlapping strips, observable, e.g., as large height differences [30]. Such discrepancies can, e.g., lead to sudden jumps along the borders of the strips in a thereof derived digital terrain model (DTM) [17]. These strip discrepancies are typically minimized by means of *strip adjustment* [21]. Ideally, a *rigorous* strip adjustment is performed (see our previous works: [3,14,31,32]). However, the rigorous approach requires the ALS raw data as input, i.e., the original polar measurements of the lidar sensor and the GNSS/IMU trajectory of the platform. In practice, however, often only the already georeferenced strips (or, equivalently, tiled point clouds with strip ID as point attribute) are available. Consequently, only an *approximative* strip adjustment, i.e., a strip adjustment without raw data, can be performed.

Before discussing some prior work on the topic of approximative strip adjustments, we would like to give a brief review of the major error sources in dynamic lidar systems, e.g., ALS or MLS—an extensive discussion can be found, e.g., in [33] or [34]. Dynamic lidar systems consist at least of a GNSS receiver, an IMU, and the lidar sensor itself. To generate georeferenced point clouds, three data inputs must be combined (*direct georeferencing*): (1.) the polar measurements of the lidar sensor, (2.) the GNSS/IMU trajectory, and (3.) the mounting calibration of the lidar sensor, which defines the 6 DoF relative orientation of the sensor to the trajectory. Each of these three inputs can be affected by systematic errors, which in turn cause irregular displacement vectors of the lidar points. This raises the question about the pattern of these point displacements and what could be an appropriate transformation model to correct them, especially in the case of an approximate strip adjustment (i.e., without trajectory information). For this, we consider as an example the following scenario, cf. Figure 3: A lidar strip of 100 m length is acquired from a flying platform at a height of 50 m above ground level (AGL). Figure 3a shows the error-free points with trajectory (top) and a single lidar scan line (bottom). Figure 3b shows exemplarily the effect of an erroneous mounting calibration, specifically for a slight misalignment of the lidar sensor and the IMU around a single axis. Figure 3c shows an often observed effect of a miscalibrated lidar sensor, namely, the effect of a constant range offset, which leads to a bending of the strips across the flight direction. Figure 3d shows the effect of a trajectory error—here, it is important to stress that we found in [31] that trajectory errors (drifts) are typically time dependent and continuous. Finally, Figure 3e shows the point displacement caused by the sum of all errors from Figure 3b–d. The aim of a strip adjustment is to correct for these errors. Looking at Figure 3e, one can observe that the error pattern is smooth and continuous, and the magnitude is depending on the location.

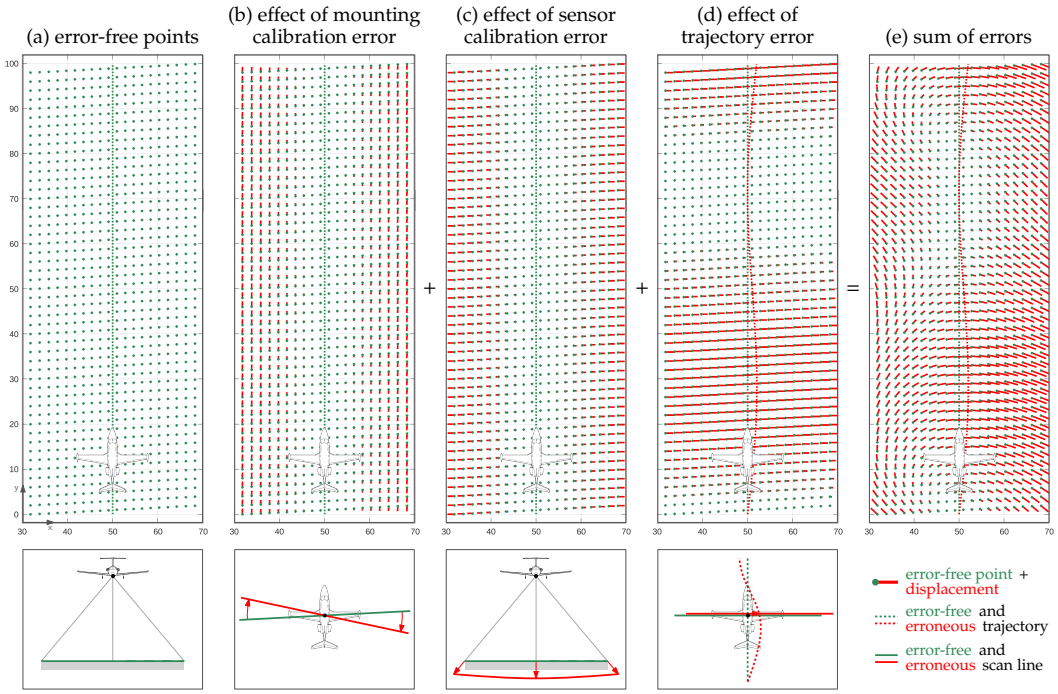

Figure 3. Typical systematic georeferencing errors (**b**–**d**) of an airborne laser scanning (ALS) point cloud (**a**). The superposition of all errors corresponds to (**e**). Without access to the trajectory and lidar data, such an error pattern can only be corrected by means of a nonrigid transformation.

Now, let us briefly summarize which transformation models have been proposed to correct such an error pattern in prior works. Typically, an individual transformation $\mathcal{T}$ is estimated and applied to each strip. Thereby, the number of independent transformation parameters varies considerably. For example, Ref. [35] estimates a stripwise height translation only (1 DoF); Ref. [36] estimates a stripwise 3D translation (3 DoF); Ref. [37] estimates a stripwise 3D translation, a roll angle, and an affine yaw parameter (5 DoF); Ref. [38] estimates a stripwise similarity transformation (7 DoF); Ref. [39] estimates a 3D translation, a spatial rotation, and a differential rotation change (9 DoF); and [17] estimates a stripwise 3D affine transformation (12 DoF), which, by the way, is the first-order approximation of any nonrigid 3D transformation. In our view, all these methods are limited in two ways: (a) they correct only a small portion of the systematic errors, namely, the linear part, and (b) a fixed number of parameters are used for each strip, irrespective of whether a strip has a length of 100 km or 1 km. To recover a larger portion of these errors, we propose in this work a nonrigid transformation with a uniform resolution, i.e., a resolution that does not depend on the strip length. We continue the discussion of the scenario in Figure 3 in Section 6.1.

1.3. Main Contributions

This paper offers several key contributions to the field of point cloud registration. Besides the already given general introduction to the registration problem, the paper also introduces a general mathematical formulation for point cloud registration, extending it to nonrigid registration. A novel method specifically for nonrigid registration of point clouds is proposed, followed by a comprehensive evaluation across various applications, scales, and domains. The method is made available to the community as open source.

1.4. Structure of the Paper

The remainder of the paper is organized as follows: Section 2 presents a general mathematical formulation of the point cloud registration problem. This section is essential for providing in Section 3 a more structured discussion of related works in the context of nonrigid registration. Section 4 introduces our proposed method, and Section 5 provides

some details about its implementation in Matlab/C++. Section 6 presents experimental results, featuring seven use cases. Section 7 concludes the paper and offers an outlook on future work.

2. The Point Cloud Registration Problem

We introduced in Section 1 the five main stages of a point cloud registration framework (cf. Figure 2). In the following, a formal description of the problem is given.

It is assumed that two sets of points are given in the Euclidean space $\mathbb{R}^3$: the loose point cloud $\mathcal{P} = \{\mathbf{p}_1, \ldots, \mathbf{p}_{n_p}\}$ and the fixed point cloud $\mathcal{Q} = \{\mathbf{q}_1, \ldots, \mathbf{q}_{n_q}\}$. Generally, the aim of point cloud registration is to obtain a transformed point cloud $\mathcal{P}'$ by applying a geometric transformation $\mathcal{T}$ to the original point cloud $\mathcal{P}$:

$$\mathcal{P}' = \mathcal{T}(\mathcal{P}). \tag{1}$$

The transformation $\mathcal{T}$ is thereby obtained by minimizing an alignment error E_{align} between the two point clouds:

$$\mathcal{T} = \text{argmin}\{E_{\text{align}}\}. \tag{2}$$

The alignment error E_{align} is typically defined as the sum of squared distances between corresponding points of the two point clouds. For this, let

$$\mathcal{C} = \text{match}(\mathcal{P}, \mathcal{Q}) = \{(\mathbf{p}, \mathbf{q}) : \mathbf{p} \in \mathcal{P}, \mathbf{q} \in \mathcal{Q}\} \tag{3}$$

be the set of corresponding points between $\mathcal{P}$ and $\mathcal{Q}$. In case of fine registration problems, $\mathbf{p}$ is usually defined as the nearest neighbor of $\mathbf{q}$. The alignment error can now be written as

$$E_{\text{align}} = \sum_{(\mathbf{p},\mathbf{q})\in\mathcal{C}} (\text{dist}(\mathcal{T}(\mathbf{p}), \mathbf{q}))^2. \tag{4}$$

Here, one can immediately see the least squares form of the optimization problem. Often, an additional set of weights is associated with the correspondences:

$$\mathcal{W}_\mathcal{C} = \{w_\mathcal{C}(\mathbf{p}, \mathbf{q}) : \forall (\mathbf{p}, \mathbf{q}) \in \mathcal{C}\}. \tag{5}$$

By multiplying the squared distances with these weights, the influence of individual correspondences on the alignment error can be increased or decreased:

$$E_{\text{align}} = \sum_{(\mathbf{p},\mathbf{q})\in\mathcal{C}} w_\mathcal{C}(\mathbf{p}, \mathbf{q}) \, (\text{dist}(\mathcal{T}(\mathbf{p}), \mathbf{q}))^2. \tag{6}$$

This was proven to be useful in many cases, e.g., to reduce the influence of outliers (*reweighted least squares* ([40], Chapter 4.7.4.1)) or to increase the influence of correspondences in regions of high interest.

The two most commonly used distance functions (*error metric*) are (a) the point-to-point distance and (b) the point-to-plane distance. The point-to-point distance corresponds to the Euclidean distance between corresponding points and is defined as

$$\text{dist}(\mathcal{T}(\mathbf{p}), \mathbf{q}) = ||\mathcal{T}(\mathbf{p}) - \mathbf{q}||. \tag{7}$$

The point-to-plane distance corresponds to the perpendicular (signed) distance of one point to the tangent plane of the other point and is defined as

$$\text{dist}(\mathcal{T}(\mathbf{p}), \mathbf{q}) = (\mathcal{T}(\mathbf{p}) - \mathbf{q})^\top \cdot \mathbf{n}, \tag{8}$$

where $\mathbf{n}$ is the normal vector of $\mathbf{q}$. It was shown in [4] that the registration problem converges faster when using the point-to-plane distance function—the main reason is that

flat regions can slide along each other without costs, i.e., without increasing the value of the alignment error E_{align}, cf. Equation (2) [3]. Consequently, it is the standard in both rigid and nonrigid registration pipelines [41].

Extension to Nonrigid Transformations

In this section, a short formal introduction to nonrigid transformations is given. We start with the transformation $\mathcal{T}$ of a single point $\mathbf{p}$, which is part of the loose point cloud $\mathcal{P}$. The transformation can be written according to Equation (1) as

$$\mathbf{p}' = \mathcal{T}(\mathbf{p}) = \mathbf{p} + \mathbf{\Delta p}, \qquad \mathcal{T} : \mathbb{R}^3 \to \mathbb{R}^3, \tag{9}$$

where $\mathbf{\Delta p}$ is the translation vector to be added to the original point $\mathbf{p}$ in order to obtain the transformed point $\mathbf{p}'$. Thereby, the translation vector at the position $\mathbf{p}$ is defined by a transformation field $\mathcal{F}$ (an example is visualized in Figure 4), sometimes also denoted as *deformation*, *distortion*, or *warp field*:

$$\mathbf{\Delta p} = \mathcal{F}(\mathbf{p}), \qquad \mathcal{F} : \mathbb{R}^3 \to \mathbb{R}^3. \tag{10}$$

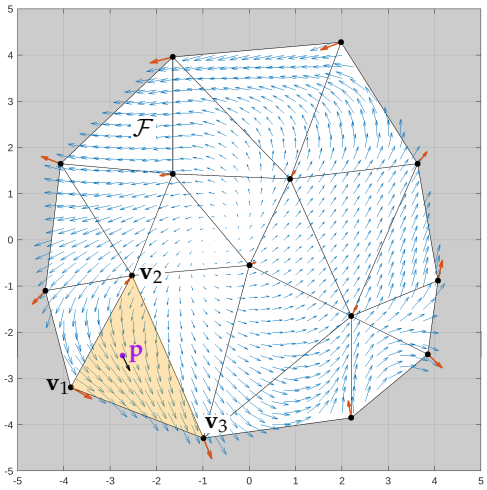

Figure 4. Example of a 2D transformation field $\mathcal{F}$ with a graph-based control structure, linear interpolation as a continuity model, and rigid-body transformation as a local transformation model. The white area corresponds to the domain of $\mathcal{F}$.

We can infer from the literature that such a transformation field $\mathcal{F}$ must fulfill in general three basic requirements: (a) it must be continuous, (b) it must be smooth (i.e., differentiable), and (c) its numerical solution must be relatively stable (ideally, the optimization problem has a closed-form solution). Additionally, it is often desirable that local shapes are preserved (*local rigidity* or *local conformity*) , e.g., to prevent strong local distortions of surfaces. These requirements are either enforced by the transformation model itself or by introducing an additional regularization term in the optimization, cf. Equation (15) below.

In order to better categorize previously published models, we define $\mathcal{F}$ as the composition of two individual functions f and g:

$$\mathcal{F}(\mathbf{p}) = g(f(\mathbf{p})), \qquad f : \mathbb{R}^3 \to \mathbb{R}^n, g : \mathbb{R}^n \to \mathbb{R}^3, \tag{11}$$

where n is the number of independent transformation parameters. We denote the functions f and g as *continuity model* and *local transformation model*, respectively.

The functions f and g can best be explained by means of a simple two-dimensional example, which is visualized in Figure 4. Here, we choose exemplarily the *linear interpolation* as *continuity model* f and the *rigid-body transformation* as *local transformation model* g.

Additionally, we assume a *graph-based control structure* for $\mathcal{F}$. In the following, we will explain these terms in more detail.

The *control structure* defines the data points of $\mathcal{F}$ and the relations between them (topology)—in this example, a Delaunay triangulation consisting of n_v vertices (nodes) and edges is used as a graph-based control structure. The domain of $\mathcal{F}$ corresponds to the convex hull of the triangulation. Consequently, one should be aware that the transformation is undefined for points outside this domain (gray area).

Each vertex $\mathbf{v}_i$ of $\mathcal{F}$ ($i \in 1, \ldots, n_v$) has an associated individual set of transformation parameters $\mathbf{u}_i$. The model to be used is thereby defined by the *local transformation model* g, in this example, the *rigid-body transformation*. In the two-dimensional Euclidean space ($d = 2$), the rigid-body transformation is defined by a rotation angle α_i and a translation $\mathbf{t}_i$ ($\mathbf{t}_i \in \mathbb{R}^2$)—consequently, the number of independent parameters n equals 3 and $\mathbf{u}_i \in \mathbb{R}^3$. With this, we can write the translation vector $\mathbf{\Delta p}_i$ at a specific vertex position $\mathbf{v}_i$ as

$$\mathbf{\Delta p}_i = g(\mathbf{u}_i) = \mathsf{R}(\alpha_i)\,\mathbf{v}_i + \mathbf{t}_i, \qquad \mathbf{u}_i = [\alpha_i, \mathbf{t}_i^\top], \tag{12}$$

where R is the rotation matrix defined by α_i. However, in general, a point $\mathbf{p}$ does not coincide with the vertices of the control structure. The *continuity model* f defines how the values of the transformation parameters $\mathbf{u}$ change between the data points of $\mathcal{F}$, i.e., between the vertices $\mathbf{v}_i$. In our example, we chose the *linear interpolation* as a continuity model. Considering that our control structure is a triangulation, the values of the parameter vector $\mathbf{u}$ at a general position $\mathbf{p}$ is given by

$$\mathbf{u} = f(\mathbf{p}) = \mathrm{TBLI}(\mathbf{p}, \mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3, \mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3), \tag{13}$$

where TBLI denotes a *triangulation-based linear interpolation*, which considers the vertices $\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3$ and the associated parameter vectors $\mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3$ of the triangle in which $\mathbf{p}$ lies, cf. Figure 4. Given $\mathbf{u}$, the translation vector $\mathbf{\Delta p}$ can now be computed with

$$\mathbf{\Delta p} = g(\mathbf{u}) = \mathsf{R}(\alpha)\,\mathbf{p} + \mathbf{t}, \qquad \mathbf{u} = [\alpha, \mathbf{t}^\top] \tag{14}$$

and $\mathbf{p}$ can finally be transformed to $\mathbf{p}'$ by Equation (9).

For the estimation of $\mathcal{F}$, the alignment error E_{align} (2) is usually combined with an additional error term E_{reg}:

$$\mathcal{F} = \operatorname{argmin}\{E_{\text{align}} + E_{\text{reg}}\}. \tag{15}$$

E_{reg} is a regularization term that can serve multiple purposes. However, it is mostly used to control the smoothness of $\mathcal{F}$, to avoid the overfitting of $\mathcal{F}$, and to ensure the estimability of $\mathcal{F}$ (e.g., in case of data gaps, i.e., areas without correspondences). This is typically accomplished by adding penalty terms for the unknown parameters.

3. Related Work in the Context of Nonrigid Point Cloud Registration

Over the last few decades, hundreds of different nonrigid transformation models have been proposed in multiple research fields, especially computer vision, computer graphics, medical imaging, and robotics. This huge number of different models can be explained by the fact that the real physical model that led to the distortions to be compensated is mostly unknown. Consequently, an alternative, approximative transformation model must be chosen, a choice that in general can be considered somewhat arbitrary. Two comprehensive surveys on nonrigid registration methods for 3D point clouds have been published in [42,43]—the latter also covers learning-based methods. A review of spatial transformation models for nonrigid 2D image registrations can be found in [44].

We discuss in the following some prior works with respect to the *continuity model* f, the *local transformation model* g, and the *control structure* of $\mathcal{F}$. We cite for each category a few works that are highly relevant for the aspects under discussion.

3.1. Continuity Model

The continuity model f defines the progression of the transformation parameter values within the domain of $\mathcal{F}$. Suitable models ensure that the transformation parameters change smoothly so that neighboring points have similar transformations. Continuity models can be grouped according to their theoretical basis [44]:

- **Physically based models**: These models use some kind of physical analogy to model nonrigid distortions. They are typically defined by partial differential equations of continuum mechanics. Specifically, they are mostly based on the theory of *linear elasticity* (e.g., [45]), the theory of *motion coherence* (e.g., [18,27]), the theory of *fluid flow* (e.g., [46]), or similarly, the theory of *optical flow* (e.g., [47]).
- **Models based on interpolation and approximation theory**: These models are purely data driven and typically use basis function expansion to model the transformation field $\mathcal{F}$. For this, some sort of piecewise polynomial functions with a degree ≤ 3 are widely used, e.g., *radial basis functions*, *thin-plate splines* (e.g., [26]), *B-splines* (e.g., [48]), or *wavelets*. Other methods use simply a weighted mean interpolation (e.g., [41,49–51]), penalize changes of the parameter vector $\mathbf{u}$ (e.g., [52]) or the translation vector $\mathbf{\Delta p}$ (e.g., [53–55]) with increasing distance, or try to preserve the length of neighboring points (e.g., [56]).

3.2. Local Transformation Model

The local transformation g model defines which type of deformation is applied locally [49]. This concept is mainly used to enforce local shape preservation, most often *local rigidity*. We briefly review the three most frequent approaches:

- **Local translation** ($n = 3$; linear model): This is the simplest and most intuitive model: the transformation is defined at each position $\mathbf{p}$ by an individual translation vector $\mathbf{t}$ ($\mathbf{t} \in \mathbb{R}^3$). Accordingly, Equation (14) simplifies to the trivial form

$$\mathbf{\Delta p} = g(\mathbf{u}) = \mathbf{t}, \qquad \mathbf{u} = \mathbf{t} \tag{16}$$

  and the transformation parameters $\mathbf{u}$ directly correspond to $\mathbf{\Delta p}$.
  An example is the *coherent point drift* (CPD) algorithm, a rather popular solution introduced by [18]. It is available in several programs, e.g., *PDAL* (http://pdal.io, accessed on 10 October 2023) or Matlab (function *pcregistercpd*). The transformation model is based on the *motion coherence theory* [57]. Accordingly, the translations applied to the loose point cloud are modeled as a temporal motion process. The displacement field is thereby estimated as a continuous velocity field, whereby a motion coherence constraint ensures that points close to one another tend to move coherently. A modern interpretation of the CPD algorithm with several enhancements was recently published by [27]. Another widely used algorithm in this category was published by [26]. The transformation model is thereby based on the above-mentioned *thin-plate splines* (TPS), a mechanical analogy referring to the bending of thin sheets of metal. In our context of point cloud registration, the authors interpret the bending as the displacement of the transformed points w.r.t. to their original position. The TPS transformation model ensures the continuity of the transformation values. Large local oscillations of these values are avoided by minimizing the bending energy, i.e., by penalizing the second derivatives of the transformation surface (in 2D) or volume (in 3D).
  The local translation model offers the highest level of flexibility as it does not couple the transformation to any kind of geometrical constraint. However, this flexibility comes also with the risk of un-natural local shape deformations due to overfitting, especially in cases where the transformation field $\mathcal{F}$ has a very flexible control structure.
- **Local rigid-body transformation** ($n = 6$; nonlinear model): The transformation at each point $\mathbf{p}$ is defined by an individual set of rigid-body transformation parameters

$\mathbf{u}$. In the 3D case, $\mathbf{u}$ is composed of three rotation angles, α_1, α_2, and α_3, and the translation vector $\mathbf{t}$ ($\mathbf{t} \in \mathbb{R}^3$), and the translation $\mathbf{\Delta p}$ becomes

$$\mathbf{\Delta p} = g(\mathbf{u}) = \mathsf{R}(\alpha_1, \alpha_2, \alpha_3)\,\mathbf{p} + \mathbf{t}, \qquad \mathbf{u} = [\alpha_1, \alpha_2, \alpha_3, \mathbf{t}^\top]. \tag{17}$$

The open-source solution by [41] uses a graph-based transformation field, where each node has an associated individual rigid-body transformation; the transformation values between these nodes are determined by interpolation. A similar graph-based approach used for motion reconstruction is described in [45,55]. The authors of [50] first segment the point cloud into rigid clusters and then map an individual rigid-body transformation to each of these segments.

Generally, the advantage of a rigid-body transformation field—especially in comparison with the less restricted translation field—is that it implicitly guarantees local shape preservation and needs less correspondences due to geometrical constraints implicitly added by the transformation model. The main disadvantages, however, are the nonlinearity of the model due to the involved rotations and the larger number of unknown parameters in the optimization.

- **Local affine transformation** ($n = 12$; linear model): This is the most commonly used model in the literature. The transformation at each point $\mathbf{p}$ is defined by an individual set of affine transformation parameters $\mathbf{u}$:

$$\mathbf{\Delta p} = g(\mathbf{u}) = \mathsf{A}\mathbf{p} + \mathbf{t}, \qquad \mathbf{u} = [\mathbf{a}^\top, \mathbf{t}^\top], \tag{18}$$

where $\mathbf{u}$ is composed of the vectors $\mathbf{a}$ (holding the elements of the affine matrix A) and the translation vector $\mathbf{t}$.

A popular early example of an affine-based transformation field is presented by [52]. Ref. [49] proposes a graph-based transformation field, where each node corresponds to an individual affine transformation. To avoid un-natural local shearing, they use additional regularization terms, which ensure that the transformation is locally "as-rigid-as-possible". Specifically, additional condition equations are added to the optimization so that the matrix A is "as-orthogonal-as-possible", i.e., so that it is very close to an orthogonal rotation matrix. Ref. [53] additionally allows a local scaling of the point cloud by constraining the local affine transformation to a similarity transformation in an "as-conformal-as-possible" approach.

In terms of flexibility, the affine transformation lies between the local translation model (more flexible) and the rigid-body transformation model (less flexible). An important advantage compared with the rigid-body transformation is the linearity of the model. However, the linearity often gets lost by the introduction of additional nonlinear equations, e.g., for local rigidity or local conformity. This model leads in comparison with the ones discussed above to the highest number of unknown parameters in the optimization.

3.3. Control Structure

The control structure defines the data points of $\mathcal{F}$ together with their topology—by that, it also defines the domain of $\mathcal{F}$. The proper choice of a structure often involves a trade-off between the flexibility (expressiveness) of $\mathcal{F}$ and computational costs. Additionally, it must be considered that a higher flexibility, on the one hand, leads to a better alignment of the point clouds, but, on the other hand, also increases the risk of overfitting, a problem that can typically be recognized in the form of undesirable large local deformations of the transformed point cloud [58]. The following control structures have been used predominately in the past, cf. Figure 5:

- **Graph-based**: This is the most commonly used control structure. The graph for a transformation field is typically constructed by selecting a subset of the observed points as nodes, e.g., by using a random or uniform sampling approach [3]. Consequently, the nodes lie directly on the scanned objects. Nodes are typically connected

by undirected edges, which indicate local object connectivities. The flexibility of the transformation field can be adjusted by the density of the nodes.
In the context of nonrigid deformation of moving characters, a widely used and highly efficient subsampling algorithm was introduced in [59]—it was also used in [49] to obtain evenly distributed nodes over the entire object. Ref. [55] extended the concept of graph-based structures to a double-layer graph, where the inner layer was used to model the human skeleton and the outer layer was used to model the deformations of the observed surface regions. Ref. [41] defined the nodes by subsampling the point cloud with a voxel-based uniform sampling method.
Considering that graph-based control structures are tightly bound to the observed objects (e.g., humans or animals), they can be regarded as best suited in cases where transformations should model the movement (deformation) of these objects. On the downside, this concept is difficult to adopt to large scenes that include multiple heterogeneous objects and complex geometries (e.g., vegetation). For example, in lidar-based remote sensing, point clouds of relatively large areas (of up to hundreds of square kilometers) that include many very different objects (buildings, vegetation, cars, persons, etc.) are acquired. In such cases, the proper definition of a graph-based control structure is rather difficult.

- **Segment based**: Such methods split the point clouds in multiple segments and estimate an individual transformation (often a rigid-body transformation under the assumption of local rigidity) for each segment. A frequent application is the matching of human scans where individual segments correspond to, e.g., upper arms, forearms, upper legs, and shanks.
Ref. [60] determines such segments under the assumption of an isometric (distance-preserving) deformation and predetermined correspondences using the RANSAC framework. Ref. [56] additionally blends the transformations between two adjacent segments in the overlapping region to preserve the consistency of the shape. A similar approach was presented in [50]; however, global consistency is achieved here by defining the final transformation of a point as a weighted sum of the individual segment transformations, whereby the weights decrease with growing segment distances.
An advantage of this type of methods is a relatively low DoF, which lowers the risk of overfitting and processing time. A major limitation, however, is that the point clouds to be registered must be divisible into multiple rigid segments. In this sense, their usability is also not very versatile.
- **Grid based**: These methods use regularly or irregularly spaced grids as a control structure of $\mathcal{F}$. The flexibility of the control structure can be easily influenced by the grid spacing.
An early work using a hierarchical grid-based control structure that is based on an octree is described in [48]—deformations are thereby modeled by volumetric B-splines. Ref. [51] discretizes the object space in a regular 3D grid, i.e., a voxel grid. A local rigid-body transformation is associated with each grid point. Transformation parameter values between the grid points are obtained via trilinear interpolation. Ref. [54] also used a voxel grid in combination with local rigid-body transformations; however, the transformation values at the voxel resolution are obtained by interpolating transformations of an underlying sparse graph-based structure—this way, the number of unknown parameters can drastically be reduced, which in turn allows for an efficient estimation of the transformation field.
A regularly spaced grid-based control structure is typically object independent; i.e., the grid structure is not influenced by the type of objects that are in the scene. In this sense, it is a much more general choice compared with the two control structures discussed above, which are mostly tailored to specific use cases or specific measurement setups. Consequently, a grid-based structure seems also to be a natural choice for large, complex, multiobject scenes, e.g., for large lidar point clouds. Another relating

advantage is that it is easier to control the domain of $\mathcal{F}$—for example, the domain can be easily set to a precisely defined 3D bounding box of the observed scene.

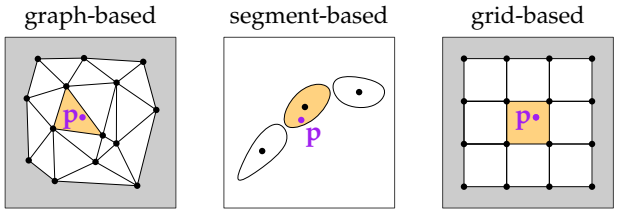

Figure 5. Types of control structures for transformation fields.

Classification of our method: Our method uses a *local translation* as a transformation model g and models the transformation field $\mathcal{F}$ as a *grid-based displacement field*, whereby the mathematics are based on *interpolation theory*.

4. Method

The registration problem of point clouds and its solution is already presented in Section 2. In this section, we focus primarily on the main contribution of this paper, namely, a new model for the nonrigid transformation of point clouds. The upcoming Section 4.1 describes the definition and advantages of the transformation model, Section 4.2 its regularization, and Section 4.3 deepens the understanding through a simple 2D example.

4.1. The Nonrigid Transformation Model

A nonrigid transformation model can be described by a *continuity model* f and a *local transformation model* g, c.f. Equation (11). We propose in this work the usage of **piecewise tricubic polynomials** (PTCP) as the *continuity model* f

$$\mathbf{u} = f(\mathbf{p}) = \mathrm{PTCP}(\mathbf{p}), \tag{19}$$

and the **local translation** as the *local transformation model* g

$$\mathbf{\Delta p} = g(\mathbf{u}) = \mathbf{t}, \qquad \mathbf{u} = \mathbf{t} = [t_x \;\; t_y \;\; t_z]^\top. \tag{20}$$

In the following, a formal description of this transformation model will be given. Afterwards, we will motivate in detail the choice of this specific model.

The idea of using PTCP to model the transformation parameters is borrowed from the tricubic interpolation (TCI)—the underlying mathematics were described in detail by Lekien and Marsden [61]. TCI is the extension of the popular and highly efficient bicubic interpolation (used, e.g., for image resampling) to the third dimension. It is a three-dimensional interpolation method that uses piecewise polynomials to derive smooth values from a given set of sparse and irregularly spaced data points. The core idea of this paper is to use TCI to model the space-varying values of the transformation parameters. TCI uses a grid-based 3D *control structure*, i.e., a voxel structure. The voxel size is the main parameter to adjust the resolution of the transformation field $\mathcal{F}$. The interpolated values change continuously (C^0 continuity) and smoothly (C^1 continuity) across the entire voxel structure, i.e., not only within a single voxel, but also across the voxel faces. The overall model is composed of PTCP, and accordingly, the values in each voxel are defined by a cubic polynomial with an individual set of 64 coefficients.

We use PTCP to model the space-varying values of the transformation parameters. More specifically, the values of each transformation parameter are represented by an individual scalar field, cf. Figure 6. Accordingly, three scalar fields are used to model the components of $\mathbf{t}$, namely, for t_x, t_y, and t_z. The transformation field $\mathcal{F}$, i.e., the translations $\mathbf{\Delta p}$ in form of a vector field (cf. Equation (10)), is obtained by combining these three individual scalar fields.

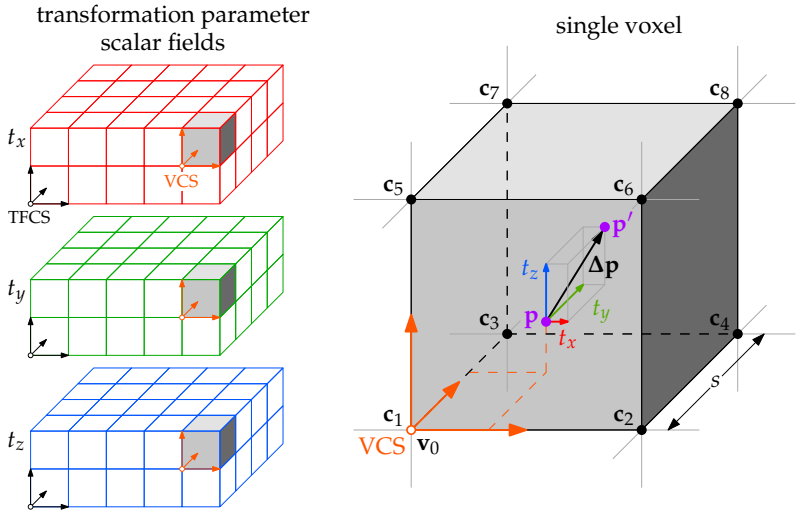

Figure 6. The nonrigid transformation is modeled by piecewise tricubic polynomials in a voxel structure. The original point $\mathbf{p}$ is translated by the vector $\Delta\mathbf{p}$ to the transformed point $\mathbf{p}'$. The components of the translation vector $\Delta\mathbf{p}$ are t_x, t_y, and t_z. TFCS = transformation field coordinate system (e.g., UTM), VCS = voxel coordinate system.

In the following, we describe the definition of a single scalar field, namely, $t_x(\mathbf{p})$; the scalar fields $t_y(\mathbf{p})$ and $t_z(\mathbf{p})$ are defined analogously. The translation t_x is defined at the position $\mathbf{p}$ by the cubic polynomial:

$$t_x(\mathbf{p}) = \sum_{i=0}^{3}\sum_{j=0}^{3}\sum_{k=0}^{3} a_{ijk}\bar{x}^i\bar{y}^j\bar{z}^k \tag{21}$$

where a_{ijk} are the 64 coefficients corresponding to the voxel in which the point $\mathbf{p}$ is located and $\bar{x}$, $\bar{y}$, and $\bar{z}$ are the reduced and normalized point coordinates of $\mathbf{p}$. These coordinates are defined in a local voxel coordinate system (VCS) by

$$\bar{\mathbf{p}} = [\bar{x}\ \ \bar{y}\ \ \bar{z}]^\top = \frac{\mathbf{p} - \mathbf{v}_0}{s}, \tag{22}$$

where $\mathbf{v}_0$ ($\mathbf{v}_0 \in \mathbb{R}^3$) is the local origin of the voxel and s is the voxel size.

In order to achieve global C^0 and C^1 continuity, the coefficients a_{ijk} of each voxel cannot be estimated independently. Instead, one must ensure that the values and its derivatives are continuous at the contact faces of neighboring voxels. Lekien and Marsden [61] presented an elegant and efficient solution to this problem by relating the coefficients of a voxel to the values and its derivatives at the 8 corners ($\mathbf{c}_1, \ldots, \mathbf{c}_8$) of this voxel. For this, first, the tricubic polynomial (21) is expressed as the scalar product

$$t_x(\mathbf{p}) = \mathbf{b}^\top\mathbf{a}. \tag{23}$$

Thereby, the column vector $\mathbf{a}$ ($\mathbf{a} \in \mathbb{R}^{64}$) contains the 64 coefficients a_{ijk} of the tricubic polynomial (21)—the elements are defined as

$$a_{1+i+4j+16k} = a_{ijk}, \qquad \forall i,j,k \in \{0,1,2,3\}. \tag{24}$$

Similarly, the column vector $\mathbf{b}$ ($\mathbf{b} \in \mathbb{R}^{64}$) contains the products of the exponentiated coordinates $\bar{x}$, $\bar{y}$, and $\bar{z}$—the elements are defined as

$$b_{1+i+4j+16k} = \bar{x}^i\bar{y}^j\bar{z}^k, \qquad \forall i,j,k \in \{0,1,2,3\}. \tag{25}$$

Now, a new column vector $\mathbf{f}$ ($\mathbf{f} \in \mathbb{R}^{64}$) is introduced, which is composed of the values and the first, second, and third derivatives of the scalar field t_x at the 8 corners of the voxel:

$$f_i = \begin{cases} t_x(\mathbf{c}_i) & \forall i \in \{1,\ldots,8\} \\ \frac{\partial t_x}{\partial x}(\mathbf{c}_{i-8}) & \forall i \in \{9,\ldots,16\} \\ \frac{\partial t_x}{\partial y}(\mathbf{c}_{i-16}) & \forall i \in \{17,\ldots,24\} \\ \frac{\partial t_x}{\partial z}(\mathbf{c}_{i-24}) & \forall i \in \{25,\ldots,32\} \\ \frac{\partial t_x}{\partial x \partial y}(\mathbf{c}_{i-32}) & \forall i \in \{33,\ldots,40\} \\ \frac{\partial t_x}{\partial x \partial z}(\mathbf{c}_{i-40}) & \forall i \in \{41,\ldots,48\} \\ \frac{\partial t_x}{\partial y \partial z}(\mathbf{c}_{i-48}) & \forall i \in \{49,\ldots,56\} \\ \frac{\partial t_x}{\partial x \partial y \partial z}(\mathbf{c}_{i-56}) & \forall i \in \{57,\ldots,64\}. \end{cases} \tag{26}$$

The relationship between $\mathbf{a}$ and $\mathbf{f}$ can now be formulated using a matrix M ($\mathsf{M} \in \mathbb{R}^{64\times64}$) by

$$\mathbf{f} = \mathsf{M}\mathbf{a}, \tag{27}$$

where the elements of M are defined by

$$\mathsf{M}_{ij} = \frac{\partial f_i}{\partial a_j}, \qquad \forall i,j \in \{1,\ldots,64\}. \tag{28}$$

The matrix M is rather sparse (46.9% sparsity), and its elements are integer numbers. These numbers do not depend on the actual values of the coefficients $\mathbf{a}$. Consequently, M is a constant matrix whose elements are known in advance. The determinant of M equals 1, and as a consequence, M is invertible. We provide the matrices M and M^{-1} in our public repository (https://github.com/AIT-Assistive-Autonomous-Systems/3D_nonrigid_tricubic_pointcloud_registration, accessed on 10 October 2023). The inverse matrix can be used to compute the coefficients $\mathbf{a}$ from $\mathbf{f}$ with

$$\mathbf{a} = \mathsf{M}^{-1}\mathbf{f}. \tag{29}$$

With this, finally, the tricubic polynomial (23) can be written in the elegant form

$$t_x(\mathbf{p}) = \mathbf{b}^\top \mathsf{M}^{-1}\mathbf{f}. \tag{30}$$

With this form, the scalar field can now be defined through the values of $\mathbf{f}$ (instead of using the coefficients $\mathbf{a}$), which means by 8 parameters at each voxel corner. Accordingly, the elements of $\mathbf{f}$ for the entire voxel structure correspond to the unknown parameters to be estimated in the optimization process (15). Notably, Lekien and Marsden [61] proved that the continuity of $\mathbf{f}$ at the corners of neighboring voxels is sufficient to achieve global C^0 and C^1 continuity of the scalar field. In other words, the continuity of the values and derivatives at the voxel corners is sufficient to achieve also continuity at the contact faces of the voxels.

There are three important advantages of form (30) over form (21). The first advantage is that the scalar field can be defined by a significantly smaller number of parameters—we would like to illustrate this with an example. For this, we assume to have 3 rather small scalar fields with $5 \times 4 \times 2 = 40$ voxels as the ones depicted in Figure 6. With form (21), one would need for each voxel an individual set of 64 coefficients to represent a single scalar field; this leads in sum to $40 \times 64 \times 3 = 7680$ parameters for all three scalar fields. Additionally, one must define 8 continuity constraints (for the values and the derivatives) at the adjoining corners of the voxel structure; this leads in sum to 5520 additional constraints. In contrast, with form (30), a single scalar field is defined through the 8 values and derivatives at the voxel corners; this leads in sum for all three scalar fields to only $6 \times 5 \times 3 \times 8 \times 3 = 2160$ parameters, where $6 \times 5 \times 3$ is the number of the voxel corners.

Additional constraints are not needed. In summary, with form (30), the number of parameters can be reduced by approximately 72% in this case.

The second important advantage of form (30) is the efficient evaluation of the scalar fields for a large amount of points. This is particularly important when applying an estimated transformation to the entire point cloud, which potentially consist of hundreds of millions of points. For this, we assume to have a set of points $\mathcal{P}_v = \{\mathbf{p}_1, \ldots, \mathbf{p}_{n_v}\}$ in a single voxel. The scalar field can then be evaluated efficiently for all points at once with

$$t_x(\mathcal{P}_v) = \mathsf{B}\mathsf{M}^{-1}\mathbf{f}, \tag{31}$$

where the matrix B ($\mathsf{B} \in \mathbb{R}^{n_v \times 64}$) is defined as

$$\mathsf{B} = [\mathbf{b}_1 \ \ldots \ \mathbf{b}_{n_v}]^\top. \tag{32}$$

The evaluation of the scalar field through Equation (31) is particularly advantageous when used in interpreted programming languages like Python or Matlab. This is because performing matrix multiplications (*vectorized operation*) for a large set of points is much more efficient than transforming each point individually, i.e., looping over all points.

Finally, a third advantage of form (30) is that it is much easier and intuitive to manipulate the transformation field $\mathcal{F}$ by manipulating $\mathbf{f}$ instead of $\mathbf{a}$. For example, one can easily adjust the smoothness of the transformation field by directly manipulating the derivatives of $\mathcal{F}$ at the voxel corners, e.g., by defining regularizing observations (see next section), constraints, or upper limits for the corresponding parameters in $\mathbf{f}$ (26). Specifically, such additional observations or constraints can be useful to mitigate large unmotivated oscillations of the transformation values, e.g., in regions with only few correspondences.

In summary, the main motivations for the proposed nonrigid transformation model are

- *Continuity*: The transformation field $\mathcal{F}$ is C^0 and C^1 continuous; i.e., transformation values change smoothly over the entire voxel structure.
- *Flexibility*: The domain of $\mathcal{F}$ corresponds to the extents of the voxel structure. Thus, it can easily be defined by the user, e.g., to match exactly the extents of point cloud tiles. Moreover, the resolution of $\mathcal{F}$ can easily be adjusted through the voxel size.
- *Efficiency*: The transformation field $\mathcal{F}$ can efficiently be estimated for two reasons. First, the number of unknown parameters is relatively low. Second, the transformation is a linear function of the parameters in $\mathbf{f}$. In other words, the parameters of $\mathcal{F}$ can be estimated through a closed-form solution that does not require an iterative solution or initial values for the parameters. Moreover, the transformation of very large point clouds can efficiently be implemented using Equation (31).
- *Intuitivity*: The parameters of the transformation field $\mathcal{F}$ can easily be interpreted as they directly correspond to the translation values and the derivatives. Thus, it is also rather easy to manipulate these parameters by introducing additional parameter observations, constraints, or upper limits to the optimization.

4.2. Regularization

Regularization [40] (p. 82) is often used when estimating nonrigid transformations—we discussed this briefly at the end of Section 2 and introduced thereby an additional error term E_{reg} in Equation (15). In our context, regularization serves two purposes:

1. To solve an *ill-posed* or *ill-conditioned* problem. Our problem becomes ill posed (underdetermined) if the domain of $\mathcal{F}$, i.e., the voxel structure, contains areas with too few or even no correspondences. As a consequence, a subset of the unknown parameters cannot be estimated. Relatedly, the problem can be ill conditioned (indicated by a high condition number C of the equation system) if the correspondences have *locally* an unfavorable geometrical constellation; for example, the scalar fields t_x and t_y can hardly be estimated when matching two nearly horizontal planes. By regularization,

an ill-posed or ill-conditioned problem can be transformed into a well-posed and well-conditioned problem.

2. To control the smoothness of the transformation field $\mathcal{F}$ and thereby also prevent overfitting. The smoothness of $\mathcal{F}$ is controlled by directly manipulating the unknown parameters, i.e., the function values and the derivatives at the voxel corners. Simultaneously, overfitting can also be avoided, i.e., the suppression of excessively fluctuating values of the scalar fields t_x, t_y, and t_z.

Specifically, we use a *Tikhonov regularization*, also known as *ridge regression* [62]. It can be interpreted as the regularizing direct observation of $\mathbf{u}$, i.e., of all unknown parameters describing the transformation field $\mathcal{F}$. Accordingly, the error term E_{reg} can be written—again exemplary for the translation t_x— as

$$\begin{aligned} E_{\text{reg}} = \sum_{i=1}^{n_r} \Bigg(& w_{d_0} \left(\left(t_x(\mathbf{c}_i) \right)^2 \right) + \\ & w_{d_1} \left(\left(\frac{\partial t_x}{\partial x}(\mathbf{c}_i) \right)^2 + \left(\frac{\partial t_x}{\partial y}(\mathbf{c}_i) \right)^2 + \left(\frac{\partial t_x}{\partial z}(\mathbf{c}_i) \right)^2 \right) + \\ & w_{d_2} \left(\left(\frac{\partial t_x}{\partial x \partial y}(\mathbf{c}_i) \right)^2 + \left(\frac{\partial t_x}{\partial x \partial z}(\mathbf{c}_i) \right)^2 + \left(\frac{\partial t_x}{\partial y \partial z}(\mathbf{c}_i) \right)^2 \right) + \\ & w_{d_3} \left(\left(\frac{\partial t_x}{\partial x \partial y \partial z}(\mathbf{c}_i) \right)^2 \right) \Bigg), \end{aligned} \tag{33}$$

where $\mathbf{c}_i$ ($i \in 1, \ldots, n_r$) corresponds to the corners of the entire voxel structure of $\mathcal{F}$ and w_{d_0}, w_{d_1}, w_{d_2}, and w_{d_3} are the weights associated with the regularizing observations of the scalar field values, as well as their first, second, and third derivatives, respectively. In other words, these weights directly influence the values, the slope, the curvature, and the torsion of the three scalar fields t_x, t_y, and t_z.

4.3. A Synthetic 2D Example

In this section, we will discuss various aspects of the proposed nonrigid transformation model on the basis of an example. It is emphasized that the optimization problem is solved according to the solution described in Section 2. Specifically, the transformation field $\mathcal{F}$ is estimated according to the least squares objective function (15), whereby E_{align} and E_{reg} are defined by Equations (4) and (33), respectively.

In order to better visualize scalar and vector fields, the example takes place in the two-dimensional Euclidean space $\mathbb{R}^2$. The main differences to the previously described transformation in $\mathbb{R}^3$ are the bicubic polynomial has only 16 coefficients (instead of the 64 coefficients of the tricubic polynomial), the transformation field $\mathcal{F}$ is obtained by combining the scalar fields t_x and t_y (instead of combining t_x, t_y, t_z), and the control structure is composed of two-dimensional cells (instead of voxels).

The two point clouds to be registered are visualized in the upper left image of Figure 7. The fixed point cloud $\mathcal{Q}$ is synthetically generated and consists of two axially parallel lines, four simple geometric forms, and a dense point raster. The transformed point cloud $\mathcal{P}$ is generated from $\mathcal{Q}$ by applying two consecutive transformations. First, a rigid-body transformation with $t_x = -1.5$, $t_y = 3.0$, and $\alpha = -3.0^\circ$ is applied. Then, an additional sinusoidal translation (amplitude = 2, period = 15) is added in the y direction. The goal of this example is to estimate the combination of these two transformations using the nonrigid transformation model presented in the previous sections.

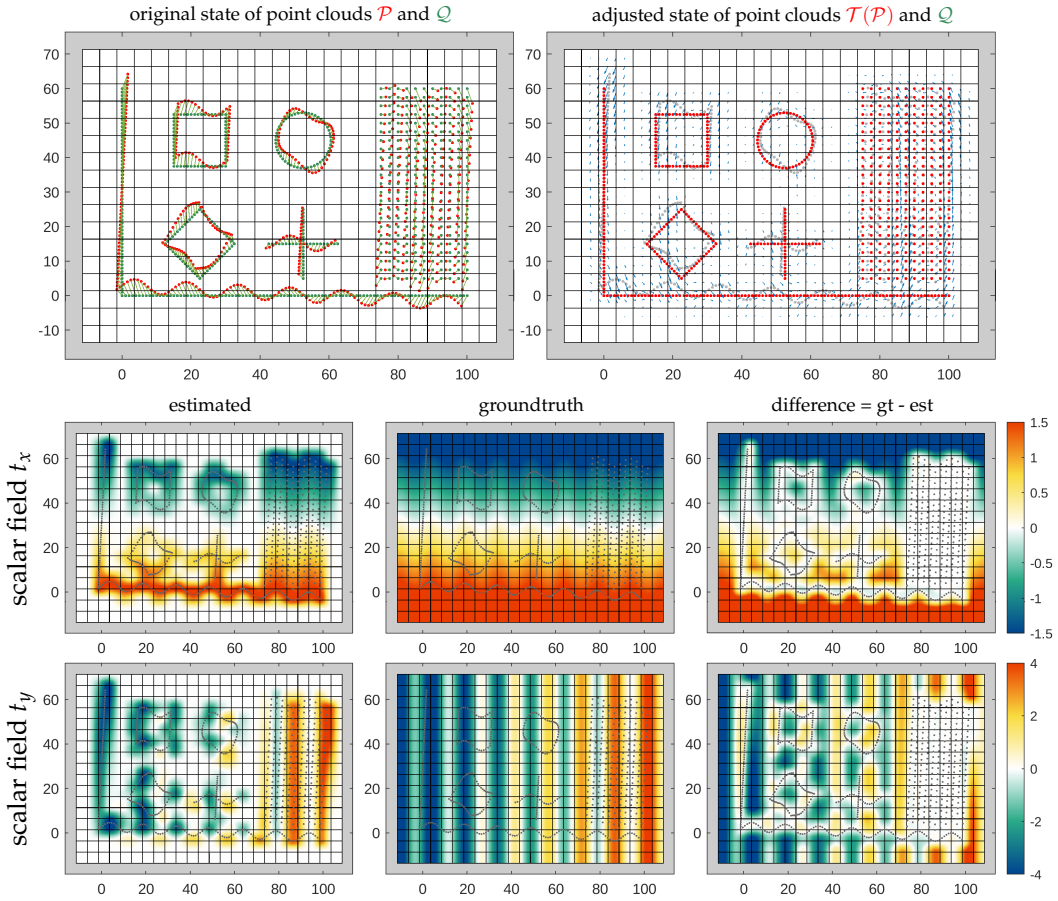

Figure 7. Example of nonrigid registration of two synthetic point clouds. Red: loose point cloud $\mathcal{P}$. Green: fixed point cloud $\mathcal{Q}$. Correspondences between $\mathcal{P}$ and $\mathcal{Q}$ are visualized as lines in the upper-left image. The transformation field $\mathcal{F}$ is visualized as a vector field in the upper-right image.

For this, 632 correspondences (It is noted that these correspondences are error-free in this example as, for each point in $\mathcal{P}$, the original partner in $\mathcal{Q}$ is known. When working with real data, however, correspondences are typically established by matching each point in $\mathcal{Q}$ to the nearest neighbor in $\mathcal{P}$, cf. Section 6) between the point clouds $\mathcal{P}$ and $\mathcal{Q}$ are used. The point-to-point distance (8) is minimized between these correspondences. The *control structure* of $\mathcal{F}$ consists of $17 \times 24 = 408$ cells with a cell size of 5. Considering that $\mathbf{f}$ has 4 elements in $\mathbb{R}^2$ (e.g., for the scalar field t_x: t_x, $\frac{\partial t_x}{\partial x}$, $\frac{\partial t_x}{\partial y}$, $\frac{\partial t_x}{\partial x \partial y}$, c.f. Equation (26)), the number of unknown parameters for both scalar fields t_x and t_y equals $18 \times 25 \times 4 \times 2 = 3600$. These parameters are estimated by solving an overdetermined linear equation system according to the least squares principle. The equation system consists of 4232 condition equations: 632 point-to-point distance observations and 3600 regularizing observations. Consequently, the redundancy of the equation system is 632. The weights of the regularizing observations w_{d_0}, w_{d_1}, and w_{d_2} are set to 0.02, 0.01, and 0.01, respectively.

In the upper-right image of Figure 7, the adjusted state of the point clouds is visualized. One can see that the two point clouds match very well after adding the estimated transformation field $\mathcal{F}$ to $\mathcal{P}$—the mean and standard deviation of the distance residuals are 0.000 ± 0.002. The vector field shows the estimated translations $\mathbf{\Delta p}$ at selected points in scaled form. The lower part of Figure 7 shows a comparison between the estimated scalar fields t_x and t_y and their ground truth values. Additionally, Figures 8 and 9 show the effects on the estimated scalar fields t_x and t_y when varying the weights w_{d_0} and w_{d_1} and the cell size—the main results from Figure 7 are thereby located in the middle of each parameter variation. For each variant, the condition number C of the normal matrix and the goodness of fit (GoF), defined as the sum of squared distance residuals, is specified.

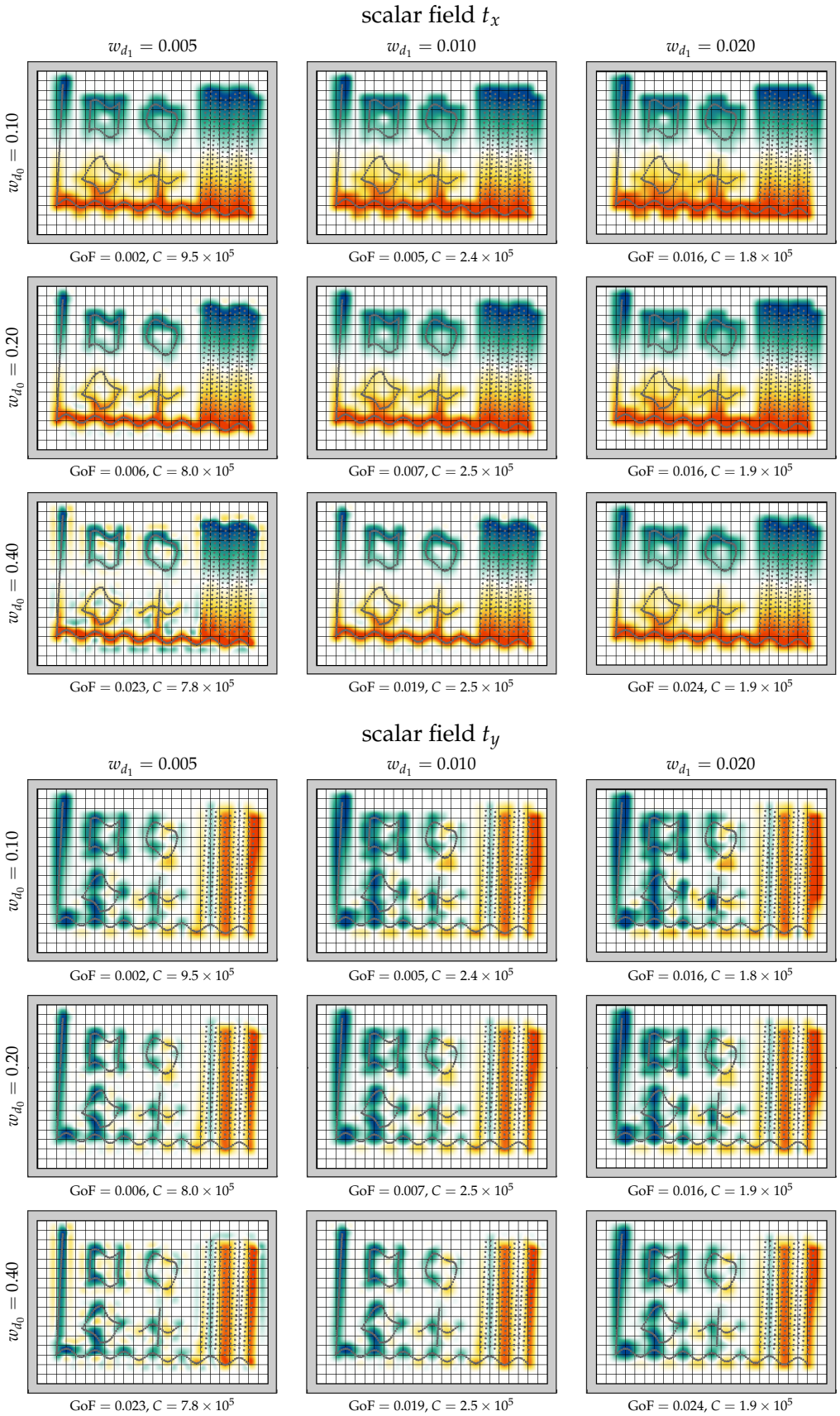

Figure 8. Effect of varying the weights w_{d_0} and w_{d_1} of the regularizing observations on the estimated scalar fields t_x and t_y; GoF = goodness of fit, C = condition number.

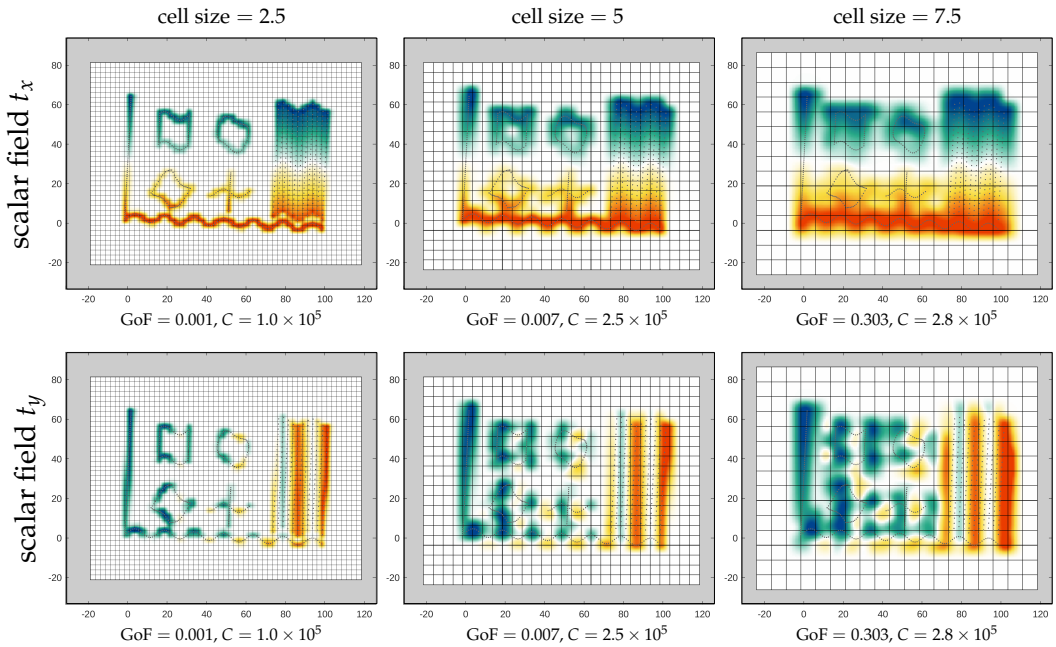

Figure 9. Effect of different cell sizes on the estimated scalar fields t_x and t_y; GoF = goodness of fit, C = condition number.

These results lead us to the subsequent observations:

- In areas with dense correspondences, the transformation can be well estimated; i.e., the differences between the estimated scalar fields and their ground truth fields are nearly zero in these areas. In correspondence-free areas the transformation tends towards zero due to a lack of information.
- The locality of the transformation depends mainly on the cell size. Minor adjustments of the locality can be made by modifying the weight w_{d_1}. The cell size needs to be adjusted to the variability of the transformation to be modeled.
- The scalar fields tend to oscillate if the ratio w_{d_0}/w_{d_1} is large—in such cases, the scalar fields have relatively steep slopes at the cell corners.
- The GoF is better for lower weights and smaller cell sizes. However, in case of correspondences with even small random errors, a small cell size also increases the risk of overfitting.
- The condition number C decreases with higher weights; i.e., the stability and efficiency of the parameter estimation increase.

5. Implementation Details

We have implemented the proposed method in two variants:

- *Matlab* (2D): This is an open-source prototype implementation for two-dimensional point clouds (Figure 10). It can be downloaded here: https://github.com/AIT-Assistive-Autonomous-Systems/2D_nonrigid_tricubic_pointcloud_registration (accessed on 10 October 2023). Parameters can easily be modified through a graphical user interface (GUI). The least squares problem is defined using the *problem-based optimization setup* from the *Optimization Toolbox*; thereby, all matrix and vector operations are *vectorized* for efficiency reasons. The problem is solved using the linear least squares solver *lsqlin*. In addition to the *Optimization Toolbox*, the *Statistics and Machine Learning Toolbox* is required to run the code. As a reference, solving the optimization for the example depicted in Figure 10 takes approximately 0.4 s on a regular PC (CPU Intel Core i7-10850H).
- *C++/Python* (3D): This is a highly efficient implementation of our method for large (e.g., lidar-based) three-dimensional point clouds. It can be downloaded here: https://github.com/AIT-Assistive-Autonomous-Systems/3D_nonrigid_tricubic_pointcloud_

registration (accessed on 10 October 2023). The full processing pipeline is managed by a Python script and consists of three main steps. In the first step, the loose point cloud $\mathcal{P}$ and the fixed point cloud $\mathcal{Q}$ are preprocessed using *PDAL* (https://pdal.io, accessed on 10 October 2023); the preprocessing includes mainly a filtering of the point clouds and the normal vector estimation. In the second step, a C++ implementation of the registration pipeline depicted in Figure 2 is used to estimate the transformation field $\mathcal{F}$ by matching the preprocessed point clouds. Thereby, the main C++ dependencies are *Eigen* (https://eigen.tuxfamily.org, accessed on 10 October 2023) and *nanoflann* (https://github.com/jlblancoc/nanoflann, accessed on 10 October 2023). *Eigen* is used for all linear algebra operations and for setting up and solving the optimization problem. A benchmark has shown that the *biconjugate gradient stabilized solver* (BiCGSTAB) is the most efficient solver for our type of problem. Finally, in the third step, the estimated transformation $\mathcal{F}$ is applied to the original point cloud $\mathcal{P}$. As a reference, the estimation of the transformation field $\mathcal{F}$ for the point clouds in Section 6.4 takes approximately 10 s, again on the regular PC mentioned above.

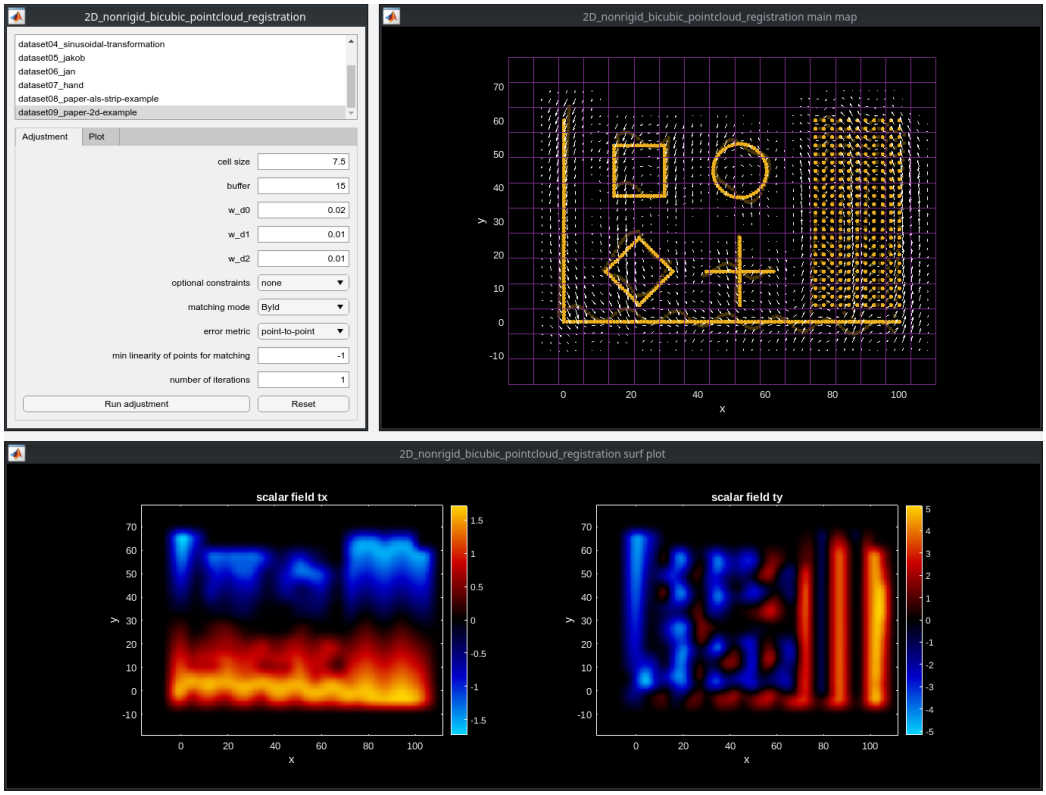

Figure 10. Open-source implementation of our method for two-dimensional point clouds—it can be downloaded here: https://github.com/AIT-Assistive-Autonomous-Systems/2D_nonrigid_tricubic_pointcloud_registration (accessed on 10 October 2023). The depicted example corresponds to the right column in Figure 9.

6. Experimental Results

The method introduced in this study can be used as a versatile and broadly applicable tool for the nonrigid alignment of point clouds. To showcase its flexibility, we perform a series of experiments that span a diverse range of scales and applications. Within the 3D domain, we align point clouds obtained from airborne laser scanning (ALS), mobile laser scanning (MLS), and terrestrial laser scanning (TLS). Within the 2D domain, the method is applied to estimate a dense optical flow in image space and to align two popular 2D nonrigid registration datasets. In contrast with the piecewise *tricubic* polynomials used in 3D, in the 2D domain, the transformation field is composed of piecewise *bicubic* polynomials, cf. Section 4.3. An overview of these experiments is provided in Table 1.

Table 1. Overview of the datasets used in the experiments and the corresponding tuning parameter values (#corresp. = number of correspondences, #it. = number of iterations).

Experimental Results									
Use Case	Sec.	2D/3D	Cell Size	Regularization Weights				#corresp.	#it.
				w_{d_0}	w_{d_1}	w_{d_2}	w_{d_3}		
1 ALS 1	Section 6.1	3D	125.0 m	2.00	2.00	2.00	2.00	20,000	3
2 ALS 2	Section 6.2	3D	100.0 m	1.00	1.00	1.00	0.10	20,000	3
3 MLS	Section 6.3	3D	5.0 m	0.10	0.10	0.10	0.10	10,000	3
4 TLS	Section 6.4	3D	2.0 m	0.01	0.01	0.01	0.01	10,000	5
5 Opt. flow	Section 6.5	2D	15.0 px	0.20	0.10	0.10	–	6713	1
6 Fish	Section 6.6	2D	7.5	0.10	0.10	0.10	–	91	1
7 Hand	Section 6.6	2D	15.0	0.05	0.05	0.10	–	36	1

6.1. Use Case 1: Airborne Laser Scanning (ALS)—Alignment of Historical Data

The city of Vienna, Austria, maintains a public archive of geospatial data. This archive includes digital surface models (DSMs) derived from ALS point clouds of the entire urban area, segmented into tiles. When comparing the DSMs from different years, discrepancies in x, y, and z are observed. These discrepancies are not solely attributable to real changes, such as construction activities, changes in vegetation, or the presence of dynamic objects like cars or persons. One of the main causes for these discrepancies is georeferencing errors of the original lidar point clouds as discussed in Section 1.2. In this use case, we aim to correct these errors using the method proposed here.

Figure 11 displays a single tile of the dimensions 1000 × 1250 m. The two DSMs stem from the years 2007 and 2015, respectively. The height differences between the two original DSMs show significant and systematic discrepancies, in the order of several decimeters. Thereby, only smooth areas (streets, roofs, etc.) and areas where the magnitude of height differences is less than 30 cm were considered (the assumption is that differences above 30 cm are not due to georeferencing errors but are a result of natural changes).

For the nonrigid registration, these two DSMs were converted to the 3D point clouds $\mathcal{P}$ and $\mathcal{Q}$. The more recent DSM from 2015, presumably more accurate in terms of georeference, is thereby considered to be fixed, while the older DSM from 2007 is considered to be loose and thus subject to transformation. The estimated scalar fields of the transformation field $\mathcal{F}$, evaluated at the surface of $\mathcal{P}$, are shown in the right column of Figure 11. The transformation field $\mathcal{F}$ was estimated using a cell size of 125 m and 20,000 corresponding points, cf. Table 1. The point-to-plane error metric was minimized between these correspondences. It is immediately evident that the scalar field t_z largely follows the pattern of the original height differences. The estimated shifts in x and y are relatively small in comparison. This is primarily because the scene mainly consists of horizontal surfaces. Vertical surfaces, such as building facades, are scarcely present due to the origin of the data as 2.5D rasters. However, there are a few isolated instances of sloped roof surfaces that support the estimation of translations in the x and y directions. One such example is found at the coordinates x ≈ 300 and y ≈ 600. Here, the original height differences clearly indicate a shift in the y direction, which is evident from the different signs of the height differences of the two roof surfaces. Consequently, the translation in the y direction can be accurately estimated at this point, as clearly shown at the corresponding location in the scalar field t_y.

The height differences in the adjusted state indicate that systematic discrepancies between the two DSMs can be largely eliminated. Larger residual discrepancies result from imperfect masking, such as the roof extensions between 2007 and 2015 at x ≈ 250 and y ≈ 750. In summary, we have demonstrated in this example how our method can be used to transform on a tile-by-tile basis older historical datasets to the georeference of more recent datasets. This can be particularly useful for the analysis of long-term changes.

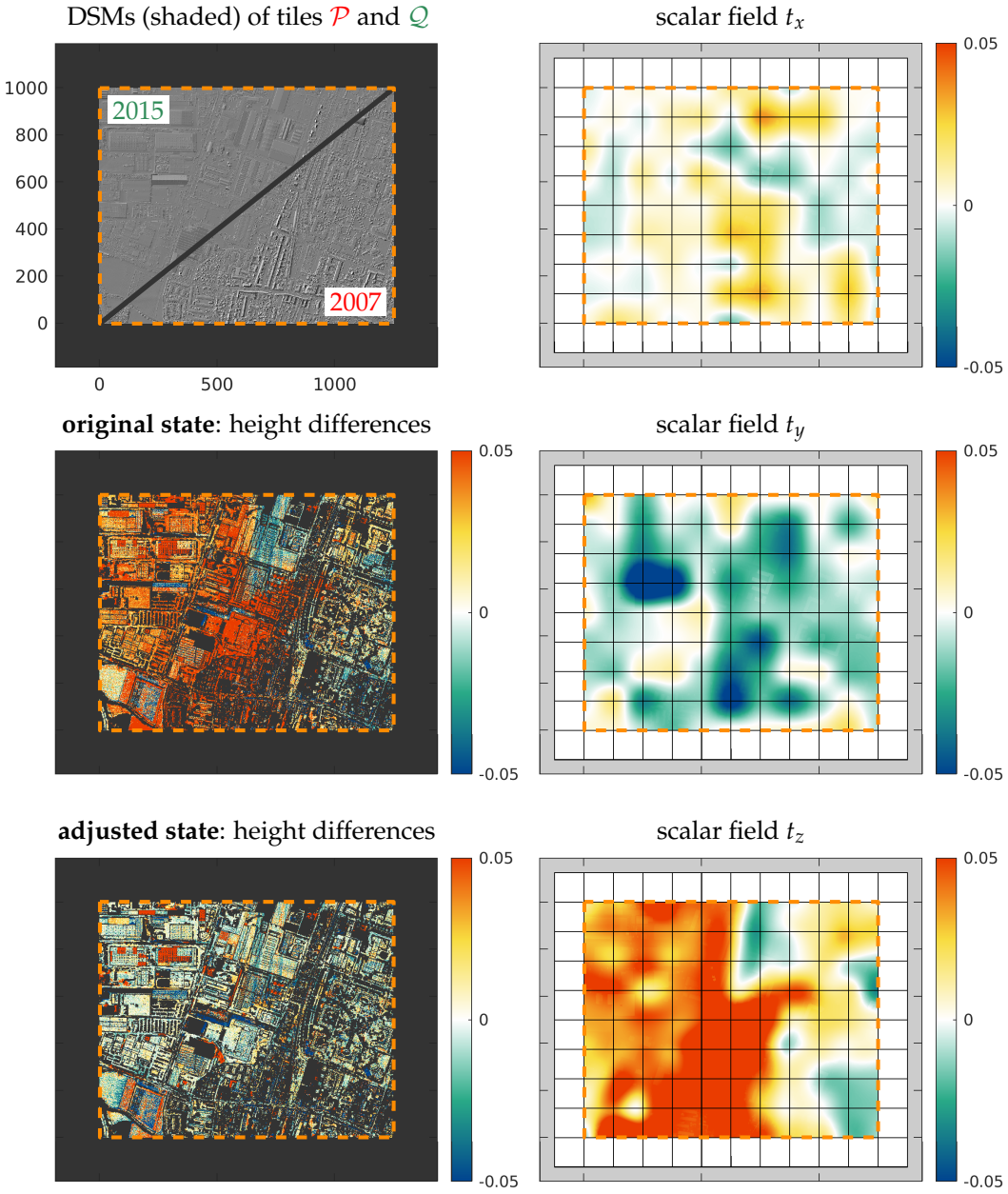

Figure 11. Improvement of the alignment between two DSMs from 2007 and 2015. The height differences show the errors before (original state) and after (adjusted state) the application of our method. The three images on the right show the estimated shifts in the x, y, and z directions for the point cloud $\mathcal{P}$. All units in meter.

6.2. Use Case 2: Airborne Laser Scanning (ALS)—Post-Strip-Adjustment Refinement

In general, registration errors between overlapping strips cannot be completely corrected by ALS strip adjustment [31]. The most common reasons are limitations of the optimizations' geometrical and physical model or the lack of correspondences in some areas. Residual errors can best be identified by means of strip differences [30]. Typically, one can find in these strip differences a few areas where the errors amount to a few centimeters. This might seem a minor issue, but it can lead to major difficulties while postprocessing the lidar data, e.g., in case of very thin structures (powerlines, poles, etc.), which appear duplicated in the fusioned point cloud. With the method proposed in this work, the registration errors within such areas can be further reduced in a post-strip-adjustment refinement step.

This use case is demonstrated on the basis of two ALS strips, cf. Figure 12. The survey area is located to the south of Innsbruck, Austria. The data were acquired from a manned

aircraft equipped with a Riegl VQ-820-G laser scanning system. This system allows for combined topographic and bathymetric surveying [63]. The aircraft's trajectory loosely followed the course of the Sill River. The flight experienced turbulence due to strong winds, causing sudden and severe changes in the roll angle. These changes are evident at the boundaries of the individual flight strips. The aircraft's highly dynamic movements could not be accurately estimated in the trajectory estimation step (Kalman filter), nor was it possible to substantially improve the estimation by strip adjustment. Consequently, several areas with major residual errors can still be identified in the strip difference after strip adjustment, cf. Figure 12 (original state).

By applying our method, these errors can be reduced, as seen in Figure 12 (adjusted state). Especially height differences that are continuous and widespread can be well minimized. Noncontinuous errors, however, such as at x $\approx$ 2100, cannot be completely corrected due to the smoothness of the estimated transformation field $\mathcal{F}$. The improvement of the distributions of the strip differences can be seen in the corresponding histograms: the mean and standard deviation of the strip differences could be improved from 0.000 ± 0.017 to 0.000 ± 0.011 m. The estimated scalar fields in the x, y, and z directions are shown in the lower three images of Figure 12. The cell size of the voxel structure was set to 100 m. For the matching, 20,000 corresponding points and the point-to-plane error metric were used. Since the laser scanner observes the scene from above, the largest magnitudes are estimated in the z direction. We can also observe that corrections can only be estimated within the overlapping area of the two strips. For example, at the right boundary of the strip $\mathcal{P}$, all three scalar fields smoothly decrease to zero.

6.3. Use Case 3: Low-Cost Mobile Laser Scanning (MLS)

In the research field of robotics, sensors are generally more cost-effective compared with those used in surveying. Additionally, data must typically be processed in real time, making it impossible to use computationally intensive methods. As a result, registration errors between point clouds are typically larger than those in the previous examples. In this use case, we demonstrate the applicability of our method to such low-cost sensors.

Figure 13 shows a section of an MLS recording, captured in an urban area in Vienna, Austria. The car's trajectory was estimated exclusively using low-frequency GNSS (1 Hz) and lidar odometry (based on KISS-ICP [64], 10 Hz), i.e., without using any high-frequency IMU data. The lidar sensor on this platform is an *Ouster OS1-64*, and the GNSS data stem from a *u-blox ZED-F9P* module. Within the depicted area, two point clouds captured in opposite directions overlap for a length of approximately 150 m.

In their original state, the point clouds deviate from each other by several decimeters. As a consequence, the fusioned point cloud can hardly be used for further processing. The signed distances between the two point clouds were calculated using the method described in [65]. For this, only smooth surfaces were considered, mainly roads and facades in this scene. An area is considered to be smooth if the points' roughness attribute is smaller than 0.03 m—the roughness attribute was thereby calculated from the points' neighborhood using the method described in (Section 4) of [3].

Using a standard ICP method with 6 degrees of freedom (corresponding to a rigid-body transformation) improves the registration globally, but leaves relatively large local errors due to its limited flexibility. By applying our method to the original state of the point clouds, the distances between the two point clouds can be strongly minimized in the entire overlapping area. For this scene, we have chosen a transformation field $\mathcal{F}$ with a cell size of 5 m and used 10,000 corresponding points (with the point-to-plane error metric) for matching the two point clouds. The histograms of the residual distances clearly show the benefit of our method: the mean and standard deviation of the distances improve from -0.004 ± 0.105 (original state) to 0.015 ± 0.048 (original state transformed by 6DoF-ICP), and finally, to 0.000 ± 0.025 m (original state transformed by our method).

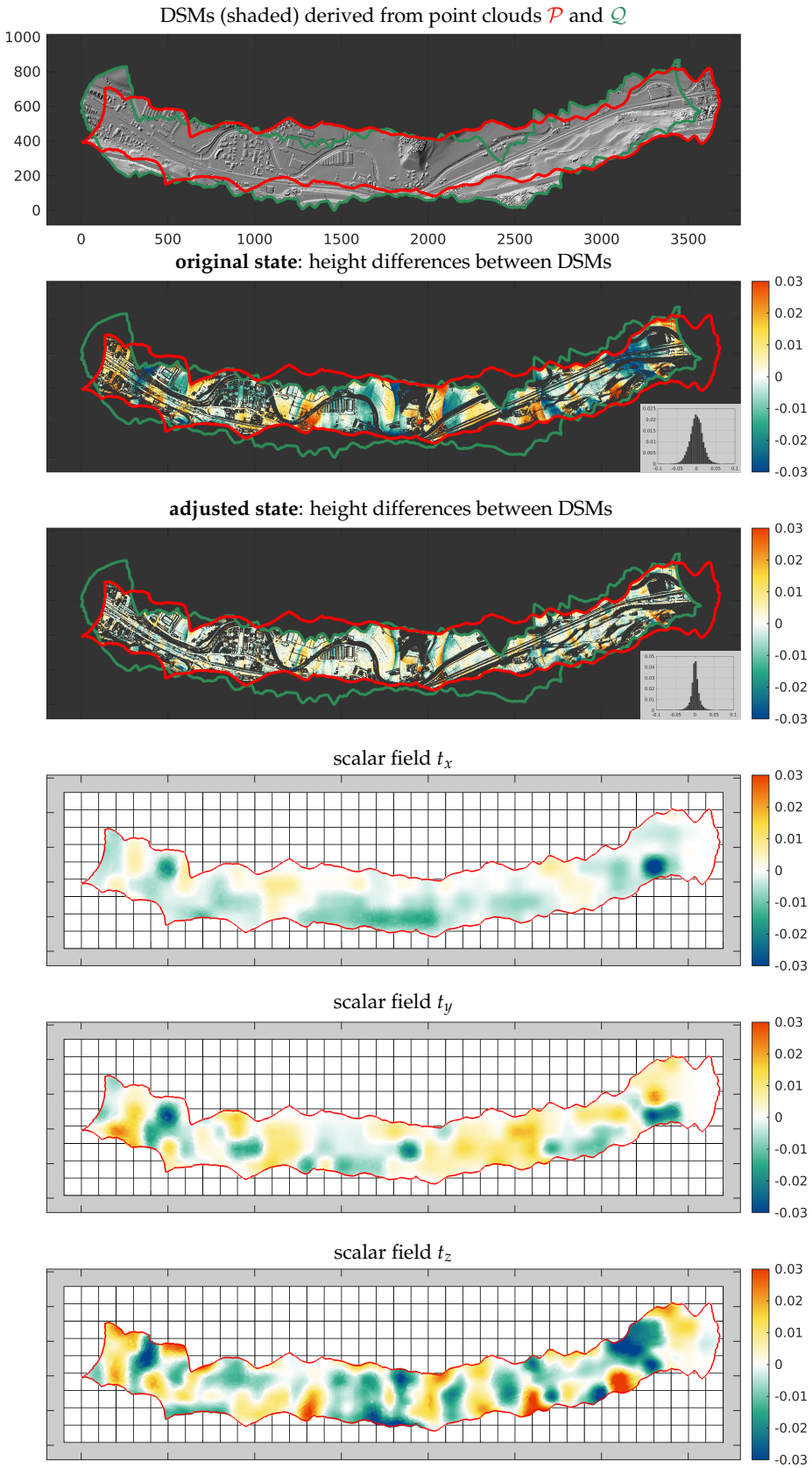

Figure 12. Improvement of the alignment of two ALS strips. As in the previous example, the height differences show the errors before (original state) and after (adjusted state) the application of our method. The lower three images show the estimated shifts in the x, y, and z directions for the point cloud $\mathcal{P}$. All units in meter.

6.4. Use Case 4: Terrestrial Laser Scanning (TLS)

In previous studies [66,67], terrestrial laser scanning was used to investigate the short-term plant structural dynamics of trees, particularly with respect to their circadian rhythm, i.e., their periodic movement with a 24-h cycle. This use case is based on terrestrial lidar point clouds measured from a Norway maple *Acer platanoides* in Finland between the time of sunset and sunrise in August 2016. The data were collected with three separate terrestrial laser scanners. We employed our method to estimate the tree's motion between sunset and sunrise. The resulting motion field $\mathcal{F}$ is depicted in Figure 1; a corresponding video is available here (https://youtube.com/watch?v=JNK9PtjtBlY, accessed on 10 October 2023). Our results suggest a plausible increase in movement as the distance from the trunk grows, with the farthest points having a motion magnitude of approximately 10 cm. Comparable results were also found in [68], where the tool *PlantMove* was used to estimate the motion field of a birch tree over the course of one night.

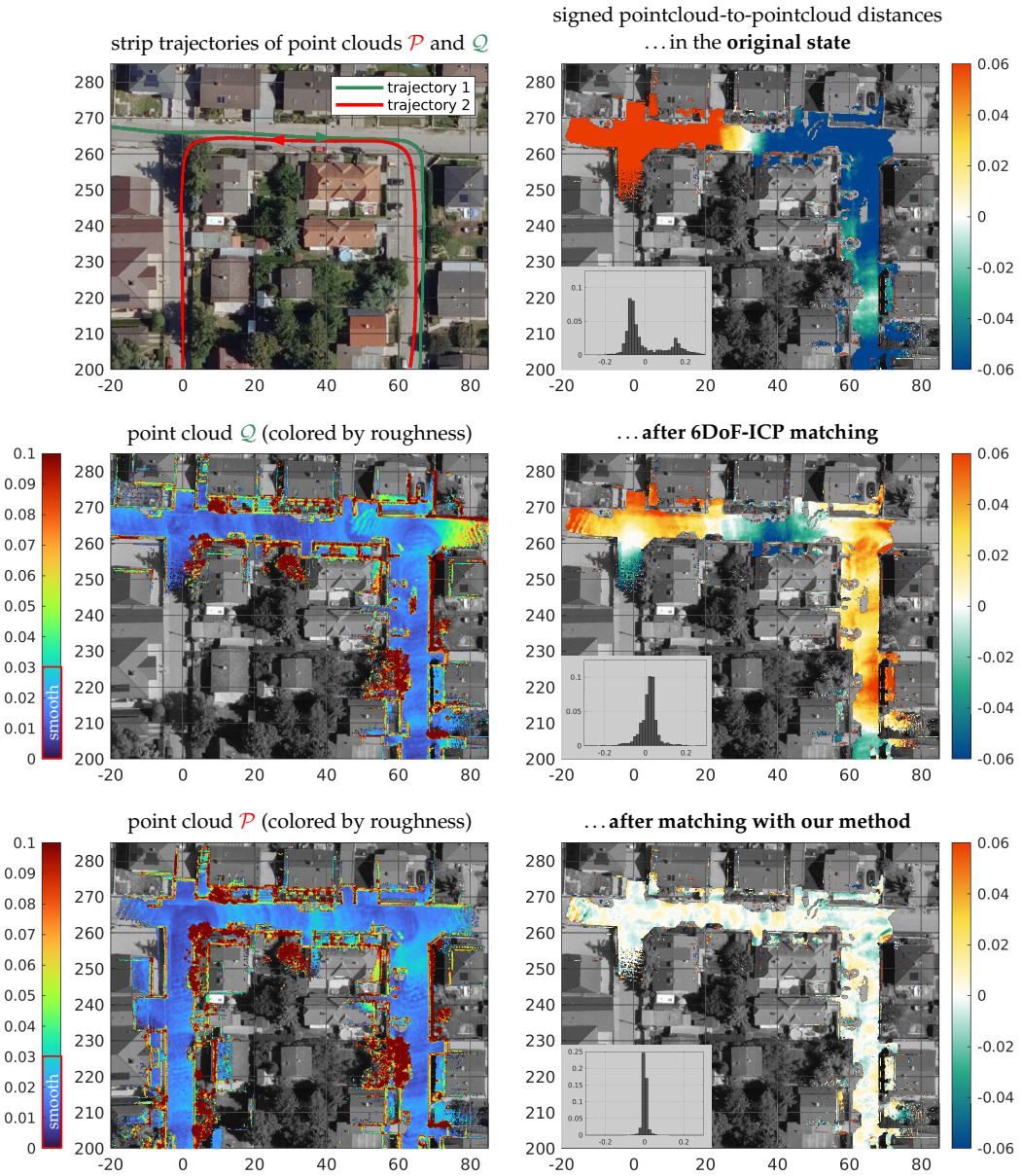

Figure 13. Improvement of alignment of two MLS strips. Only smooth areas like roads and facades are considered during the registration process. The images on the right show the alignment error between the two point clouds in their original state (**top**), after adjustment with a 6DoF-ICP (**middle**) and after our method (**bottom**). All units in meter.

6.5. Use Case 5: Dense Optical Flow

This example demonstrates a possible application of our method in the two-dimensional domain. We estimate the dense optical flow between two images based on given image correspondences. The results are presented in Figure 14. The image correspondences were found using AKAZE point descriptors [69] and brute force matching. The cell size of the estimated optical flow field $\mathcal{F}$ was set to 15 pixel. It is noted that the given correspondences also included some incorrect matchings. However, the results indicate that, due to the continuity and smoothness of $\mathcal{F}$, these have only a minimal impact. A limitation of our method is that discontinuities in the optical flow cannot be modeled, e.g., at occlusion boundaries. Instead, the optical flow is smoothly interpolated across these boundaries.

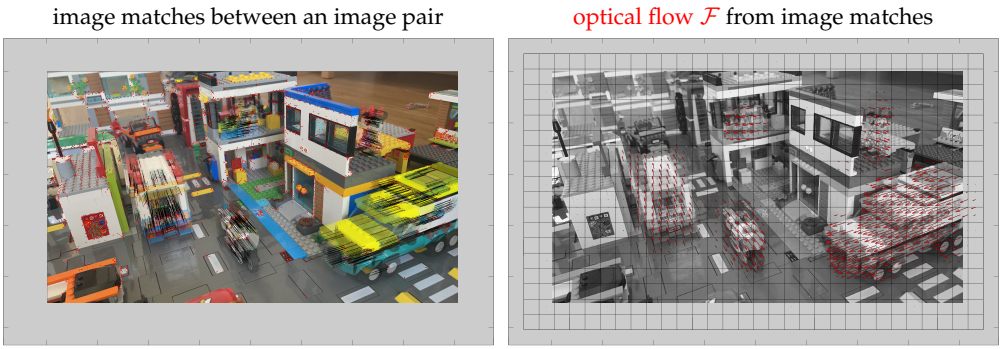

Figure 14. Estimation of a dense optical flow between two images.

6.6. Use Case 6: Popular Datasets

In the interest of completeness, we have also applied our method on two popular datasets commonly used in the literature as benchmarks for nonrigid registration techniques. In both cases, point-to-point correspondences between the two point clouds are given. Accordingly, the point-to-point error metric (8) was minimized in the optimization. The first pair of point clouds depicting two fishes originates from [18]. The results visualized in Figure 15 indicate that our method can accurately estimate the nonrigid deformations between these point clouds. The second dataset consisting of two hand-shaped point clouds stems from *MathWorks* and is presented in Figure 16. In this case as well, our method successfully registers the two point clouds.

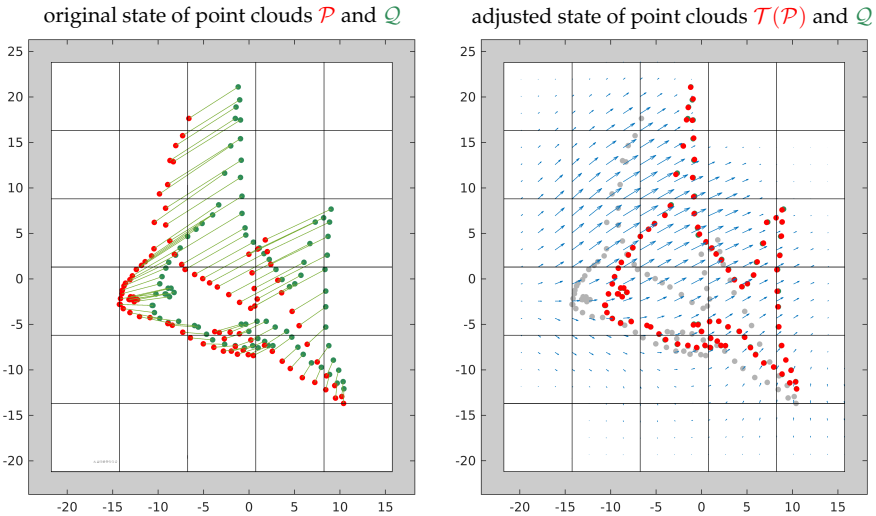

Figure 15. Nonrigid registration of two point clouds from [18]. The values are unitless.

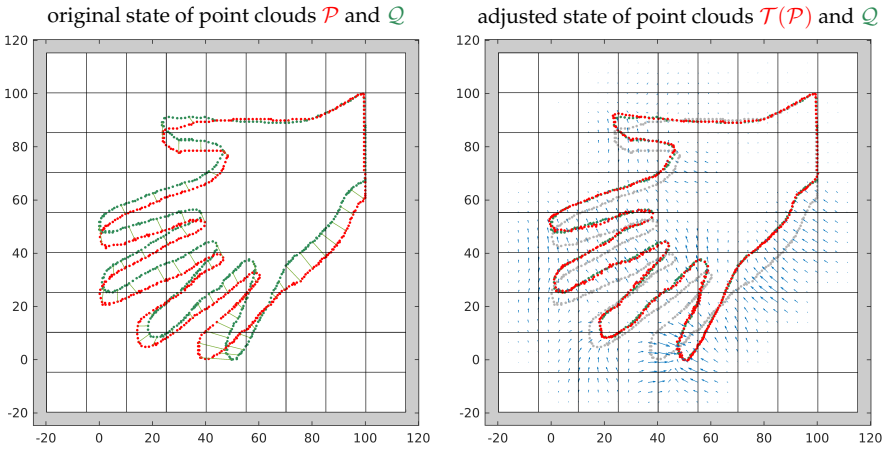

Figure 16. Nonrigid registration of two hand-shaped point clouds. The values are unitless.

7. Conclusions

In this research, we looked at the complex area of point cloud registration, focusing on the special challenges of nonrigid registration. The paper serves multiple functions: it provides a thorough introduction to the point cloud registration problem, categorizes existing methods in the field, and introduces a mathematical framework that extends to the nonrigid registration problem. Most notably, we introduce a new method for nonrigid registration that uses a grid-based transformation model based on piecewise tricubic polynomials.

Our method has several benefits. The flexibility of the transformation model can be adjusted by a small and intuitive set of tuning parameters (cf. Table 1), the optimization has a closed-form solution, and the method can be used to efficiently transform huge point clouds, e.g., airborne laser scanning data. We have validated our method across a wide range of applications and scales, with a particular focus on remote sensing tasks, such as the registration of ALS, MLS, and TLS point clouds. We also open-sourced our work, so others can use it and build on it.

Despite its strengths, our method also has some limitations. Like other nonrigid registration techniques, it faces challenges in modeling discontinuities due to the inherent smoothness and continuity of the transformation field. Additionally, the transformation field can only be reliably estimated when there are densely sampled correspondences within the entire overlapping area of the point clouds.

As for future work, we plan to integrate our method into established point cloud processing frameworks, such as OPALS or PDAL. This will not only make our method more accessible but also offer a platform for ongoing improvements and evaluations. Afterwards, we plan to extend our method to the multiview case, where >2 overlapping point clouds are registered simultaneously.

Author Contributions: Data curation, project administration, supervision, visualization, and writing (original draft): P.G.; conceptualization and methodology: P.G., C.R. and N.P.; formal analysis and validation: P.G. and C.R.; funding acquisition: P.G. and N.P.; investigation: P.G., C.W. and C.R.; resources: P.G. and M.H.-W.; software: P.G., C.W. and J.O.-S.; writing (review and editing): P.G., C.W., J.O.-S., C.R., N.P. and M.H.-W. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by the Austrian Research Promotion Agency (FFG) under the project *OPTRALS* (orientation and processing of terrestrial, railborne, and airborne laser scanning data), part of the *Verkehrsinfrastrukturforschung 2019* (*VIF 2019*) program.

Data Availability Statement: Data used in Section 6 can partly be found in the source code repositories here (https://github.com/AIT-Assistive-Autonomous-Systems/3D_nonrigid_tricubic_pointcloud_registration, accessed on 10 October 2023) and here (https://github.com/AIT-Assistive-Autonomous-Systems/2D_nonrigid_tricubic_pointcloud_registration accessed on 10 October 2023).

Acknowledgments: The source of the data used in Section 6.1 is "Stadt Wien—data.wien.gv.at" (accessed on 10 October 2023). The data used in Section 6.2 were collected within the Austrian Research Promotion Agency (FFG) COMET-K project *Airborne Alpine Hydro Mapping—From Research to Practice (AAHM-R2P)*. The data used in Section 6.4 were originally collected for research and funded by Academy of Finland grant nos. 265949 and 272195. We thank Eetu Puttonen for providing the data.

Conflicts of Interest: The authors declare no conflict of interest.

Nomenclature

Scalars will be denoted in italic font x, vectors in bold font $\mathbf{x}$, and matrices in sans serif font X. All vectors are defined as column vectors.

Notation			
Symbol(s)	**Description**	**Type**	**Dim.**
Point cloud registration			
$\mathcal{P}, \mathcal{Q}$	loose and fixed set of points (point clouds), resp.	set	n_p, n_q
$\mathbf{p}, \mathbf{q}$	individual point of point cloud $\mathcal{P}$ and $\mathcal{Q}$, resp.	vector	$\mathbb{R}^3$
$\mathcal{T}(\mathcal{P}), \mathcal{T}(\mathbf{p})$	transformation of point cloud $\mathcal{P}$ and point $\mathbf{p}$, resp.	func.	$\mathbb{R}^3 \to \mathbb{R}^3$
$\mathcal{P}'$	transformed point cloud $\mathcal{P}$	set	n_p
$\mathbf{p}'$	transformed point $\mathbf{p}$	vector	$\mathbb{R}^3$
$\mathbf{\Delta p}$	translation vector	vector	$\mathbb{R}^3$
$\mathbf{n}$	normal vector	vector	$\mathbb{R}^3$
$\mathcal{C}$	set of correspondences between $\mathcal{P}$ and $\mathcal{Q}$	set	n_c
$\mathcal{W}_\mathcal{C}$	set of weights associated with $\mathcal{C}$	set	n_c
$w_\mathcal{C}$	individual weight of $\mathcal{W}_\mathcal{C}$	scalar	$\mathbb{R}$
$\mathcal{F}$	transformation field	func.	$\mathbb{R}^3 \to \mathbb{R}^3$
f	continuity model	func.	$\mathbb{R}^3 \to \mathbb{R}^n$
g	local transformation model	func.	$\mathbb{R}^n \to \mathbb{R}^3$
$\mathbf{u}$	vector containing transformation parameters	vector	$\mathbb{R}^n$
Optimization			
n_u	overall number of unknown parameters	scalar	$\mathbb{R}$
E	error term of objective function	scalar	$\mathbb{R}$
C	condition number of equation system	scalar	$\mathbb{R}$
Piecewise tricubic polynomials			
$\bar{\mathbf{p}}$	reduced and normalized coordinates of point $\mathbf{p}$	vector	$\mathbb{R}^3$
$\mathbf{a}$	vector containing coefficients of single voxel	vector	$\mathbb{R}^{64}$
$\mathbf{f}$	vector containing function values and derivatives of single voxel	vector	$\mathbb{R}^{64}$
M	matrix for mapping between $\mathbf{a}$ and $\mathbf{f}$	matrix	$\mathbb{R}^{64\times64}$
$\mathbf{b}$	vector containing products of $\bar{\mathbf{p}}$	vector	$\mathbb{R}^{64}$
B	matrix containing products of $\bar{\mathbf{p}}$ for n_v points	matrix	$\mathbb{R}^{n_v\times64}$
$\mathbf{v}_0$	voxel origin	vector	$\mathbb{R}^3$
s	voxel size	scalar	$\mathbb{R}$

References

1. Besl, P.J.; McKay, N.D. Method for registration of 3-D shapes. In *Robotics-DL Tentative*; International Society for Optics and Photonics: Bellingham, WA, USA, 1992; pp. 586–606.
2. Chen, Y.; Medioni, G. Object modelling by registration of multiple range images. *Image Vis. Comput.* **1992**, *10*, 145–155. [CrossRef]
3. Glira, P.; Pfeifer, N.; Briese, C.; Ressl, C. A Correspondence Framework for ALS Strip Adjustments based on Variants of the ICP Algorithm. *PFG Photogramm. Fernerkund. Geoinf.* **2015**, *2015*, 275–289. [CrossRef]
4. Rusinkiewicz, S.; Levoy, M. Efficient variants of the ICP algorithm. In Proceedings of the Third International Conference on 3-D Digital Imaging and Modeling, Quebec City, QC, Canada, 28 May–1 June 2001; pp. 145–152.
5. Pomerleau, F.; Colas, F.; Siegwart, R. A review of point cloud registration algorithms for mobile robotics. *Found. Trends Robot.* **2015**, *4*, 1–104. [CrossRef]
6. Dong, Z.; Liang, F.; Yang, B.; Xu, Y.; Zang, Y.; Li, J.; Wang, Y.; Dai, W.; Fan, H.; Hyyppä, J.; et al. Registration of large-scale terrestrial laser scanner point clouds: A review and benchmark. *ISPRS J. Photogramm. Remote. Sens.* **2020**, *163*, 327–342. [CrossRef]

7. Huang, S.; Gojcic, Z.; Usvyatsov, M.; Wieser, A.; Schindler, K. Predator: Registration of 3D point clouds with low overlap. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Nashville, TN, USA, 20–25 June 2021; pp. 4267–4276.
8. Li, L.; Wang, R.; Zhang, X. A tutorial review on point cloud registrations: Principle, classification, comparison, and technology challenges. *Math. Probl. Eng.* **2021**, *2021*, 9953910. [CrossRef]
9. Yang, J.; Li, H.; Campbell, D.; Jia, Y. Go-ICP: A globally optimal solution to 3D ICP point-set registration. *IEEE Trans. Pattern Anal. Mach. Intell.* **2015**, *38*, 2241–2254. [CrossRef] [PubMed]
10. Zeng, A.; Song, S.; Nießner, M.; Fisher, M.; Xiao, J.; Funkhouser, T. 3DMatch: Learning Local Geometric Descriptors from RGB-D Reconstructions. In Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017.
11. Zhang, Z.; Dai, Y.; Sun, J. Deep learning based point cloud registration: An overview. *Virtual Real. Intell. Hardw.* **2020**, *2*, 222–246. [CrossRef]
12. Gu, X.; Wang, Y.; Wu, C.; Lee, Y.J.; Wang, P. HPLFlowNet: Hierarchical Permutohedral Lattice FlowNet for Scene Flow Estimation on Large-scale Point Clouds. In Proceedings of the 2019 IEEE International Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 15–20 June 2019.
13. Liu, X.; Qi, C.R.; Guibas, L.J. FlowNet3D: Learning Scene Flow in 3D Point Clouds. In Proceedings of the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 15–20 June 2019; pp. 529–537. [CrossRef]
14. Glira, P.; Pfeifer, N.; Briese, C.; Ressl, C. Rigorous Strip Adjustment of Airborne Laserscanning Data Based on the ICP Algorithm. *ISPRS Ann. Photogramm. Remote. Sens. Spat. Inf. Sci.* **2015**, *II-3/W5*, 73–80. [CrossRef]
15. Theiler, P.W.; Wegner, J.D.; Schindler, K. Globally consistent registration of terrestrial laser scans via graph optimization. *ISPRS J. Photogramm. Remote. Sens.* **2015**, *109*, 126–138. [CrossRef]
16. Brown, B.; Rusinkiewicz, S. Global Non-Rigid Alignment of 3-D Scans. *ACM Trans. Graph.* **2007**, *26*, 1–9. [CrossRef]
17. Ressl, C.; Pfeifer, N.; Mandlburger, G. Applying 3D affine transformation and least squares matching for airborne laser scanning strips adjustment without GNSS/IMU trajectory data. In Proceedings of the ISPRS Workshop Laser Scanning 2011, Calgary, Canada, 29–31 August 2011.
18. Myronenko, A.; Song, X.; Carreira-Perpinan, M. Non-rigid point set registration: Coherent point drift. *Adv. Neural Inf. Process. Syst.* **2006**, *19*, 1009–1016.
19. Liang, L.; Wei, M.; Szymczak, A.; Petrella, A.; Xie, H.; Qin, J.; Wang, J.; Wang, F.L. Nonrigid iterative closest points for registration of 3D biomedical surfaces. *Opt. Lasers Eng.* **2018**, *100*, 141–154. [CrossRef]
20. Qin, Z.; Yu, H.; Wang, C.; Guo, Y.; Peng, Y.; Xu, K. Geometric Transformer for Fast and Robust Point Cloud Registration. *arXiv* **2022**. [CrossRef]
21. Toth, C.K. Strip Adjustment and Registration. In *Topographic Laser Ranging and Scanning-Principles and Processing*; Shan, J., Toth, C.K., Eds.; CRC Press: Boca Raton, FL, USA, 2009; pp. 235–268.
22. Lichti, D.D. Error modelling, calibration and analysis of an AM–CW terrestrial laser scanner system. *ISPRS J. Photogramm. Remote. Sens.* **2007**, *61*, 307–324. [CrossRef]
23. Hess, W.; Kohler, D.; Rapp, H.; Andor, D. Real-time loop closure in 2D LIDAR SLAM. In Proceedings of the 2016 IEEE International Conference on Robotics and Automation (ICRA), Stockholm, Sweden, 16–21 May 2016; pp. 1271–1278.
24. Zhang, J.; Singh, S. Visual-lidar odometry and mapping: Low-drift, robust, and fast. In Proceedings of the 2015 IEEE International Conference on Robotics and Automation (ICRA), Seattle, WA, USA, 26–30 May 2015; pp. 2174–2181.
25. Glira, P. Hybrid Orientation of LiDAR Point Clouds and Aerial Images. PhD Thesis, TU Wien, Vienna, Austria, 2018.
26. Chui, H.; Rangarajan, A. A new point matching algorithm for non-rigid registration. *Comput. Vis. Image Underst.* **2003**, *89*, 114–141. [CrossRef]
27. Fan, A.; Ma, J.; Tian, X.; Mei, X.; Liu, W. Coherent Point Drift Revisited for Non-Rigid Shape Matching and Registration. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), New Orleans, LA, USA, 18–24 June 2022; pp. 1424–1434.
28. Keszei, A.P.; Berkels, B.; Deserno, T.M. Survey of non-rigid registration tools in medicine. *J. Digit. Imaging* **2017**, *30*, 102–116. [CrossRef] [PubMed]
29. Dai, M.; Xiao, G.; Fiondella, L.; Shao, M.; Zhang, Y.S. Deep Learning-Enabled Resolution-Enhancement in Mini- and Regular Microscopy for Biomedical Imaging. *Sens. Actuators A Phys.* **2021**, *331*, 112928. [CrossRef] [PubMed]
30. Ressl, C.; Kager, H.; Mandlburger, G. Quality Checking of ALS Projects using Statistics of Strip Differences. In Proceedings of the International Society for Photogrammetry and Remote Sensing 21st Congress, Beijing, China, 3–7 July 2008; Volume XXXVII. Part B3b, pp. 253–260.
31. Glira, P.; Pfeifer, N.; Mandlburger, G. Rigorous Strip adjustment of UAV-based laserscanning data including time-dependent correction of trajectory errors. *Photogramm. Eng. Remote. Sens.* **2016**, *82*, 945–954. [CrossRef]
32. Glira, P.; Pfeifer, N.; Mandlburger, G. Hybrid Orientation of Airborne Lidar Point Clouds and Aerial Images. *ISPRS Ann. Photogramm. Remote. Sens. Spat. Inf. Sci.* **2019**, *4*, 567–574. [CrossRef]
33. Glennie, C. Rigorous 3D error analysis of kinematic scanning LIDAR systems. *J. Appl. Geod.* **2007**, *1*, 147–157. [CrossRef]
34. Habib, A.; Rens, J. Quality assurance and quality control of Lidar systems and derived data. In Proceedings of the Advanced Lidar Workshop, University of Northern Iowa, Cedar Falls, IA, USA, 7–8 August 2007.

35. Kager, H. Discrepancies between overlapping laser scanner strips–simultaneous fitting of aerial laser scanner strips. *Int. Arch. Photogramm. Remote. Sens. Spat. Inf. Sci.* **2004**, *35*, 555–560.
36. Filin, S.; Vosselman, G. Adjustment of airborne laser altimetry strips. *ISPRS Arch. Photogramm. Remote. Sens. Spat. Inf. Sci.* **2004**, *XXXV, B3*.
37. Ressl, C.; Mandlburger, G.; Pfeifer, N. Investigating adjustment of airborne laser scanning strips without usage of GNSS/IMU trajectory data. *Int. Arch. Photogramm. Remote. Sens. Spat. Inf. Sci.* **2009**, *38*, 195–200.
38. Csanyi, N.; Toth, C.K. Improvement of lidar data accuracy using lidar-specific ground targets. *Photogramm. Eng. Remote. Sens.* **2007**, *73*, 385–396. [CrossRef]
39. Vosselman, G.; Maas, H.G. Adjustment and Filtering of Raw Laser Altimetry Data. In Proceedings of the OEEPE Workshop on Airborne Laserscanning and Interferometric SAR for Detailed Digital Terrain Models, Stockholm, Sweden, 1–3 March 2001.
40. Förstner, W.; Wrobel, B. *Photogrammetric Computer Vision—Statistics, Geometry, Orientation and Reconstruction*; Springer: Berlin/Heidelberg, Germany, 2016. [CrossRef]
41. Zampogiannis, K.; Fermüller, C.; Aloimonos, Y. Topology-Aware Non-Rigid Point Cloud Registration. *IEEE Trans. Pattern Anal. Mach. Intell.* **2021**, *43*, 1056–1069. [CrossRef] [PubMed]
42. Tam, G.K.; Cheng, Z.Q.; Lai, Y.K.; Langbein, F.C.; Liu, Y.; Marshall, D.; Martin, R.R.; Sun, X.F.; Rosin, P.L. Registration of 3D Point Clouds and Meshes: A Survey from Rigid to Nonrigid. *IEEE Trans. Vis. Comput. Graph.* **2013**, *19*, 1199–1217. [CrossRef] [PubMed]
43. Deng, B.; Yao, Y.; Dyke, R.M.; Zhang, J. A Survey of Non-Rigid 3D Registration. *Comput. Graph. Forum* **2022**, *41*, 559–589. [CrossRef]
44. Holden, M. A review of geometric transformations for nonrigid body registration. *IEEE Trans. Med. Imaging* **2007**, *27*, 111–128. [CrossRef]
45. Li, W.; Zhao, S.; Xiao, X.; Hahn, J.K. Robust Template-Based Non-Rigid Motion Tracking Using Local Coordinate Regularization. In Proceedings of the 2020 IEEE Winter Conference on Applications of Computer Vision (WACV), Snowmass, CO, USA, 1–5 March 2020; pp. 390–399.
46. Christensen, G.E.; Rabbitt, R.D.; Miller, M.I. 3D brain mapping using a deformable neuroanatomy. *Phys. Med. Biol.* **1994**, *39*, 609–618. [CrossRef]
47. Thirion, J.P. Image matching as a diffusion process: An analogy with Maxwell's demons. *Med. Image Anal.* **1998**, *2*, 243–260. [CrossRef]
48. Szeliski, R.; Lavallée, S. Matching 3-D anatomical surfaces with non-rigid deformations using octree-splines. *Int. J. Comput. Vis.* **1994**, *18*, 171–186. [CrossRef]
49. Sumner, R.W.; Schmid, J.; Pauly, M. Embedded deformation for shape manipulation. *ACM Trans. Graph.* **2007**, *26*, 80–88. [CrossRef]
50. Huang, Q.X.; Adams, B.; Wicke, M.; Guibas, L.J. Non-Rigid Registration Under Isometric Deformations. *Comput. Graph. Forum* **2008**, *27*, 1449–1457. [CrossRef]
51. Innmann, M.; Zollhöfer, M.; Nießner, M.; Theobalt, C.; Stamminger, M. VolumeDeform: Real-Time Volumetric Non-rigid Reconstruction. *arXiv* **2016**, arXiv:1603.08161. https://doi.org/10.48550/arXiv.1603.08161.
52. Allen, B.; Curless, B.; Popovic, Z. The space of human body shapes: Reconstruction and parameterization from range scans. *ACM Trans. Graph.* **2003**, *22*, 587–594. [CrossRef]
53. Yoshiyasu, Y.; Ma, W.C.; Yoshida, E.; Kanehiro, F. As-Conformal-As-Possible Surface Registration. *Comput. Graph. Forum* **2014**, *33*, 1–11. [CrossRef]
54. Newcombe, R.A.; Fox, D.; Seitz, S.M. DynamicFusion: Reconstruction and tracking of non-rigid scenes in real-time. In Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, USA, 7–12 June 2015; pp. 343–352.
55. Yu, T.; Zheng, Z.; Guo, K.; Zhao, J.; Dai, Q.; Li, H.; Pons-Moll, G.; Liu, Y. DoubleFusion: Real-Time Capture of Human Performances with Inner Body Shapes from a Single Depth Sensor. In Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 7287–7296.
56. Chang, W.; Zwicker, M. Automatic Registration for Articulated Shapes. *Comput. Graph. Forum* **2008**, *27*, 1459–1468. [CrossRef]
57. Yuille, A.; Grzywacz, N. The Motion Coherence Theory. In Proceedings of the 1988 Second International Conference on Computer Vision, Tampa, FL, USA, 5–8 December 1988; pp. 344–353. [CrossRef]
58. Yamazaki, S.; Kagami, S.; Mochimaru, M. Non-rigid Shape Registration Using Similarity-Invariant Differential Coordinates. In Proceedings of the 2013 International Conference on 3D Vision, Seattle, WA, USA, 29 June–1 July, 2013; pp. 191–198.
59. Mohr, A.; Gleicher, M. Building efficient, accurate character skins from examples. *ACM Trans. Graph.* **2003**, *22*, 562–568. [CrossRef]
60. Ge, X. Non-rigid registration of 3D point clouds under isometric deformation. *ISPRS J. Photogramm. Remote. Sens.* **2016**, *121*, 192–202. [CrossRef]
61. Lekien, F.; Marsden, J. Tricubic interpolation in three dimensions. *Int. J. Numer. Methods Eng.* **2005**, *63*, 455–471. [CrossRef]
62. Calvetti, D.; Reichel, L. Tikhonov regularization of large linear problems. *BIT Numer. Math.* **2003**, *43*, 263–283. [CrossRef]
63. Mandlburger, G.; Hauer, C.; Wieser, M.; Pfeifer, N. Topo-Bathymetric LiDAR for Monitoring River Morphodynamics and Instream Habitats—A Case Study at the Pielach River. *Remote. Sens.* **2015**, *7*, 6160–6195. [CrossRef]
64. Vizzo, I.; Guadagnino, T.; Mersch, B.; Wiesmann, L.; Behley, J.; Stachniss, C. KISS-ICP: In Defense of Point-to-Point ICP—Simple, Accurate, and Robust Registration If Done the Right Way. *IEEE Robot. Autom. Lett.* **2023**, *8*, 1029–1036. [CrossRef]

65. Lague, D.; Brodu, N.; Leroux, J. Accurate 3D comparison of complex topography with terrestrial laser scanner: Application to the Rangitikei canyon (N-Z). *ISPRS J. Photogramm. Remote. Sens.* **2013**, *82*, 10–26. [CrossRef]
66. Puttonen, E.; Lehtomäki, M.; Litkey, P.; Näsi, R.; Feng, Z.; Liang, X.; Wittke, S.; Pandžić, M.; Hakala, T.; Karjalainen, M.; et al. A Clustering Framework for Monitoring Circadian Rhythm in Structural Dynamics in Plants From Terrestrial Laser Scanning Time Series. *Front. Plant Sci.* **2019**, *10*, 486. [CrossRef] [PubMed]
67. Zlinszky, A.; Molnár, B.; Barfod, A.S. Not All Trees Sleep the Same—High Temporal Resolution Terrestrial Laser Scanning Shows Differences in Nocturnal Plant Movement. *Front. Plant Sci.* **2017**, *8*, 1814. [CrossRef]
68. Wang, D.; Puttonen, E.; Casella, E. PlantMove: A tool for quantifying motion fields of plant movements from point cloud time series. *Int. J. Appl. Earth Obs. Geoinf.* **2022**, *110*, 102781. [CrossRef]
69. Alcantarilla, P.F.; Nuevo, J.; Bartoli, A. Fast Explicit Diffusion for Accelerated Features in Nonlinear Scale Spaces. In Proceedings of the British Machine Vision Conference, Bristol, UK, 9–13 September 2013.