Zan Gojcic*, Caifa Zhou, and Andreas Wieser

F2S3: Robustified determination of 3D displacement vector fields using deep learning

https://doi.org/10.1515/jag-2019-0044
Received October 9, 2019; accepted February 4, 2020

Abstract: Areal deformation monitoring based on point clouds can be a very valuable alternative to the established point-based monitoring techniques, especially for deformation monitoring of natural scenes. However, established deformation analysis approaches for point clouds do not necessarily expose the true 3D changes, because the correspondence between points is typically established naïvely. Recently, approaches to establish the correspondences in the feature space by using local feature descriptors that analyze the geometric peculiarities in the neighborhood of the interest points were proposed. However, the resulting correspondences are noisy and contain a large number of outliers. This impairs the direct applicability of these approaches for deformation monitoring. In this work, we propose Feature to Feature Supervoxel-based Spatial Smoothing (F2S3), a new deformation analysis method for point cloud data. In F2S3 we extend the recently proposed feature-based algorithms with a neural network based outlier detection, capable of classifying the putative pointwise correspondences into inliers and outliers based on the local context extracted from the supervoxels. We demonstrate the proposed method on two data sets, including a real case data set of a landslide located in the Swiss Alps. We show that while the traditional approaches, in this case, greatly underestimate the magnitude of the displacements, our method can correctly estimate the true 3D displacement vectors.

Keywords: Deformation monitoring, point clouds, neural networks, local feature descriptors, outlier detection, displacement vectors, RANSAC

***Corresponding author: Zan Gojcic,** ETH Zürich, Institut for Geodesy and Photogrammetry, Stefano-Franscini-Platz 5, CH-8093 Zürich, Switzerland, e-mail: zan.gojcic@geod.baug.ethz.ch
Caifa Zhou, Andreas Wieser, ETH Zürich, Institut for Geodesy and Photogrammetry, Stefano-Franscini-Platz 5, CH-8093 Zürich, Switzerland, e-mails: caifa.zhou@geod.baug.ethz.ch, andreas.wieser@geod.baug.ethz.ch

1 Introduction

Despite the increasing use of point clouds to detect and quantify changes of man-made and natural structures, several challenges remain unresolved regarding the point cloud-based deformation analysis [10]. In particular, these include the estimation of 3D displacement vector fields, parameterization of deformations and quantification of error probabilities such as false alarm rate and probability of missed detection. These challenges are particularly demanding for point clouds of natural environments. Due to the lack of regular structures and smooth objects that could be represented with geometric primitives or free-form shapes, the deformation analysis of natural scenes is predominantly based on point cloud-based and surface-based deformation models [18, 22].

Point cloud-based deformation models, traditionally represented by the cloud-to-cloud (C2C) and multiscale model-to-model cloud (M3C2) [14] methods, can be used to compare the point clouds directly. Conversely, surface-based models such as cloud-to-mesh (C2M) and mesh-to-mesh (M2M), require that either one (C2M) or both (M2M) point clouds are first triangulated and then the resulting meshes/point clouds are compared. The quantification of the displacement magnitudes or vectors in these models differs mostly in how correspondences among the points are determined. With C2C, correspondences are established by simply selecting the nearest point from the other epoch. Other approaches incorporate some local geometric information by constraining the search for corresponding points along the direction of the normal vector of either the triangulated surface (C2M and M2M) or a plane fitted to the neighboring points (M3C2). These naïve ways of establishing the correspondences typically result in underestimation of the displacement magnitudes in parts of the point clouds that have changed (see Section 3). Furthermore, the point cloud-based and surface-based models are incapable of correctly detecting and estimating the in-plane deformations and rigid body motion [11]. A detailed explanation of point cloud-based and surface-based deformation models is available in [11].

Recently, [21] and [4] proposed to establish pointwise correspondences for point cloud-based deformation monitoring using the local feature descriptors. However, in [21]

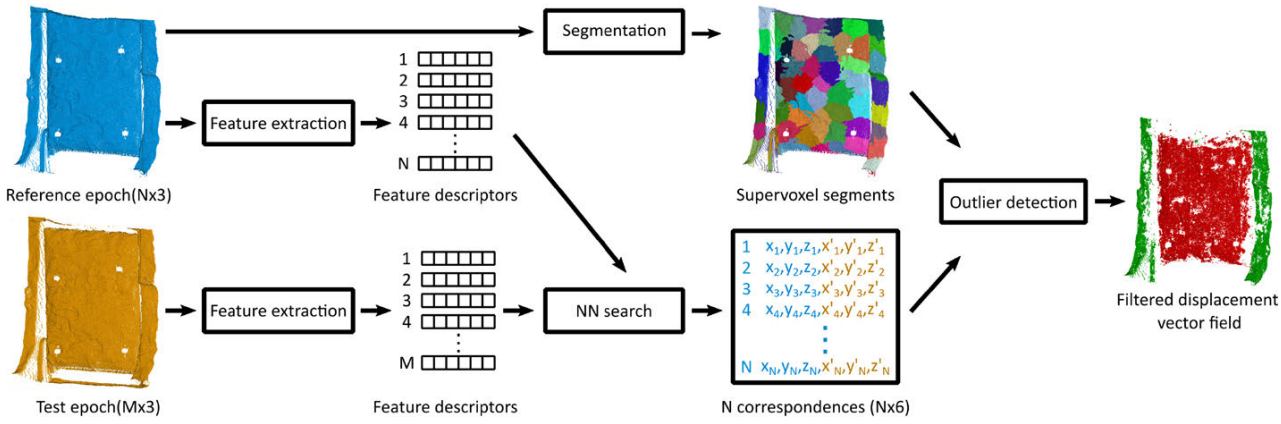

Figure 1: Schematic representation of the F2S3 workflow. In the first step, feature descriptors for each point in both point clouds are extracted and used to establish the putative set of correspondences. Independently, the reference epoch is segmented into supervoxels (Section 2.4). The correspondences are then grouped according to the supervoxels and filtered using an outlier detection algorithm (Section 2.3). The output of F2S3 is a robustly estimated displacement vector field. In this figure the points are colored according to a magnitude based classification (green stable, red moved) but the algorithm outputs the real 3D displacement vectors.

RGB data is required together with the point clouds and the correspondences are actually established in the RGB-D space. Furthermore, despite the large progress that was recently achieved in the performance of the local feature descriptors [2, 5] the established correspondences are still very noisy and contain a large amount of outliers (see Section 2 and 3). These false correspondences originate from various sources including, but not limited to, point cloud resolution, noise of the point clouds, limited descriptiveness of the feature descriptors, and repetitive structures. While theoretically several of the causes could be mitigated, it is practically infeasible that the amount of outliers will intrinsically be reduced to a level that would not hinder the applicability of the local feature descriptors for point cloud-based deformation monitoring.

In this work we therefore complement the idea of establishing the correspondences in feature space [21, 4] with a neural network (NN)-based algorithm that takes a set of putative correspondences as input and performs a binary classification into inliers and outliers. The inliers and outliers denote the correct and false correspondences, respectively. Hereinafter, these terms are used interchangeably. The proposed outlier detection step is based on the local context and can be understood as local smoothing that is spatially constrained within individual boundary persevering supervoxels (see Section 2.4). We denote this novel deformation analysis method for point cloud data as Feature to Feature Supervoxel-based Spatial Smoothing (F2S3).

We evaluate the applicability of F2S3 in an extensive empirical investigation, comparing it to the traditional methods as well as to the raw feature-based correspondences. First, we compare the performance of the methods in a controlled environment on a data set of a rockfall simulator acquired using a terrestrial laser scanner (Section 3.1). Second, we compare the efficiency and performance of the proposed outlier detection step with RANSAC on two data sets (Section 3.1.5 and 3.2.3). Finally, we show on a real case data set that the proposed method, which relies solely on the geometric information intrinsically available in point clouds, is able to correctly estimate 3D displacement vectors, while the traditional deformation models greatly underestimate the magnitudes and are incapable of determining the direction of the displacement vectors (Section 3.2.2).

Even though F2S3 performs favorably in the empirical investigation performed herein, it should not be treated as a replacement to the traditional methods. It should rather be seen as a complementary method, which performs especially favorable on point clouds of scenes with adequate spatial features and large displacements. A complete, rigorous comparison of the available deformation analysis methods, which would also show the limitations of F2S3, would only be possible in a controlled environment covering all the possible scenarios, regarding e. g., type and magnitude of the displacements, point cloud resolution, and type and size of the objects in the scene. This is out of the scope of this paper and is left for future work.

2 Method

This section describes the proposed method F2S3 for estimating a dense displacement vector field from point cloud data (Figure 1). F2S3 consists of three main modules (i) a local feature descriptor (3DSmoothNet) used to infer the feature vectors of all points in the point clouds of both epochs (ii) a novel NN based outlier detection algorithm used to robustify the initial set of correspondences established in the feature space and (iii) a supervoxel segmentation algorithm that provides the boundaries for the spatial smoothing. It is important to note, that the workflow of F2S3 is completely modular and the algorithms used to perform the aforementioned steps, can—and with the progress of the field probably will—be replaced in the future.

2.1 Neural networks – a brief description

The methods developed herein are based on deep learning and more specifically feed-forward and convolutional NNs. In order to make the paper self-contained we provide a brief description thereof. Further information are available in [1, 8].

Feed-forward NNs or multilayer perceptrons (MLPs) are the quintessential deep learning models that allow approximating some (non-linear) function $f : \mathbf{x} \mapsto \mathbf{y}$ based on the training data $\mathcal{T} = \{(\mathbf{x}_i, \mathbf{y}_i)\}$, i. e. data with corresponding ground truth labels [8]. A feed-forward network defines the mapping of the data $\mathbf{x}_i \in \mathbb{R}^D$ to a discrete (classification) or a continuous (regression) output $\hat{\mathbf{y}} \in \mathbb{R}^K$ as $\hat{\mathbf{y}}_i = f(\mathbf{x}_i; \boldsymbol{\theta})$, where f is a composition of differentiable non-linear functions (layers) and $\boldsymbol{\theta}$ are the parameters of the network. More formally, a single layer feed-forward NN can be written as

$$\hat{\mathbf{y}}_k = h(\mathbf{w}_k^T \mathbf{x} + b_k) \tag{1}$$

where $\mathbf{w}_k \in \mathbb{R}^D$ are the weights, b_k is the bias and $\mathbf{y}_k$ is the output with $k = 1, \cdots, K$ denoting the dimension. The non-linearity of NNs is achieved through the non-linear activation functions $h(\cdot)$, which are selected in the process of designing the network architecture. In recent years, the rectified linear unit (ReLU) [17] activation function defined as $h(\cdot) = \max(0, \cdot)$ is the default choice for the intermediate layers, whereas the identity function and sigmoid function (c. f. Eq. 7) are used in the last layer for regression and (binary) classification tasks, respectively [1, 8].

NNs are learning algorithms, which means that the network has to be trained before it can be applied to a specific task. During training, the parameters $\boldsymbol{\theta}$ of the network are optimized by minimizing a predefined loss function (error function) on the training data. Due to the non-linearity of the layers, most loss functions become non-convex. Therefore, NNs are typically trained by using iterative gradient-based optimizers e. g. [13], which aim at driving the loss to a low value [8]. Mainly due to the memory restrictions, the gradient descent is typically performed using only a randomly selected subset (mini-batch) of the data in each iteration. This can be regarded as a stochastic approximation of the gradient descent. Therefore, the iterative minimization of the loss function using randomly selected mini-batches is denoted as stochastic gradient descent (SGD). At each training iteration, the values of the parameters $\boldsymbol{\theta}$ are updated relative to the gradient of the loss function with respect to the parameters. This process is commonly referred to as backpropagation.

Convolutional neural networks (CNNs) are a special kind of feed-forward NNs, which use convolution instead of general matrix multiplication in at least one of the layers [8]. CNNs became especially popular in computer vision, where a grid-like topology—necessary for convolutions—is naturally given e. g. pixels in images. Compared to the traditional fully connected layers, convolutional ones offer many advantages [1, 8]. Not only do they reduce the number of learnable parameters due to the parameter sharing, but they are also equivariant to the translation of the input. More information about CNNs is available in [8].

2.2 3DSmoothNet—3D local feature descriptor

We have recently proposed 3DSmoothNet [5], a deep learning based 3D local feature descriptor, which has low output dimension, high descriptiveness, and is fully rotation invariant. In 3DSmoothNet, we combine the traditional components of handcrafted descriptors, such as the estimation of the local reference frame for achieving rotation invariance, with the novel smoothed density value voxelization that is amenable to fully convolutional layers of standard deep learning libraries. 3DSmoothNet is a non-linear, learned function that maps the input, i. e. a voxelized spherical neighborhood of the point, to a low dimensional feature vector, by solely exploiting the local geometrical properties of the data.

Machine learning and especially deep learning algorithms typically need a large amount of annotated data. Specifically, to train 3DSmoothNet a large data set, consisting of overlapping point clouds with ground truth correspondences is required. Gathering training data of out-

door scenes including the ground truth correspondences for a typical geomonitoring scenario would have been infeasible. Therefore, we resorted to an indoor benchmark data set denoted as 3DMatch [24], which is an RGB-D data set consisting of 62 real-world indoor scenes ranging from offices and hotel rooms to tabletops and restrooms. In [5] we show that the 3DSmoothNet trained only using these indoor RGB-D data can generalize to outdoor point clouds acquired using a laser scanner, without any fine-tuning.

Several other 3D learned local feature descriptors, e. g. [12, 23, 2] were proposed concurrently to 3DSmoothNet and could be used to infer the putative correspondences for the approach presented herein. However, as 3DSmoothNet significantly outperforms the state-of-the-art on indoor as well as outdoor data [5], we use it hereinafter to infer pointwise local descriptors and putative correspondences. A more detailed description of 3DSmoothNet is beyond the scope of this paper and the interested reader is referred to [5] for more information.

2.3 NN-based filtering algorithm

In order to obtain a dense displacement vector field, we do not compute feature descriptors only for some selected points, i. e. keypoints, but rather for each point in the point clouds of both epochs. Under the assumption that for each point from the reference epoch, there is a corresponding point in the test epoch and that correspondence is properly measured by the closeness in the feature space, the displacement vector field can be established by a nearest neighbor search. However, due to the sampling process yielding the point clouds, 6-DOF motion and occlusions, this assumption does not (always) hold. In particular, we also expect that some parts of the area may be deformed too much and may not be recognizable anymore. This, in conjunction with false correspondences due to repeating structures, changes in point cloud density and noise of the point clouds, results in an initial set of correspondences, which is very noisy and typically contains outliers.

We therefore propose a binary classification algorithm to identify the putative correspondences as inliers or outliers. The proposed algorithm is based on the local consistency assumption, i. e. locally the point clouds are assumed to (approximately) represent a rigid body rather than a significantly deformed surface. However this assumption does not hold on the discontinuities of the displacement vector field and edges of the rigid bodies. We therefore impose this constraint only inside the boundary aware supervoxels (see Section 2.4). The local consistency assumption is also crucial for the good performance of the local feature descriptors. More formally, consider two point clouds $\mathbf{P} \in \mathbb{R}^{N\times 3}$ and $\mathbf{Q} \in \mathbb{R}^{M\times 3}$, representing the test and reference epoch respectively. Let $(\mathbf{P})_i = \mathbf{p}_i = [x_i, y_i, z_i]$ and $(\mathbf{Q})_j = \mathbf{q}_j = [x'_j, y'_j, z'_j]$ represent the coordinate vectors of individual points of the point clouds. If each point of the point cloud $\mathbf{P}$ is matched to its nearest neighbor in the point cloud $\mathbf{Q}$ based on the descriptor distance, a group of N^c correspondences $\mathbf{c}_i$ is obtained and can be written in matrix form as

$$\begin{aligned} \mathbf{X}' &= [\mathbf{c}_1; \cdots; \mathbf{c}_{N^c}] \\ \mathbf{c}_i &= [x_i, y_i, z_i, x'_i, y'_i, z'_i] \end{aligned} \tag{2}$$

In order to avoid problems due to large baselines for rotation, we scale all the coordinates in $\mathbf{X}'$ to the interval $[-1, 1]$ as $\mathbf{X} := \mathbf{X}' \oslash (\mathbf{J} \odot \max(|\mathbf{X}'|)) \in \mathbb{R}^{N^c\times 6}$, where $\mathbf{J} \in \mathbb{R}^{N^c\times 6}$ is a matrix of ones and $\odot$ and $\oslash$ denote the Hadamard (elementwise) multiplication and division, respectively. $\mathbf{X}$ containing only the scaled coordinates of the putative correspondences, represents the input to the filtering algorithm $f_{\boldsymbol{\theta}}$, which maps $\mathbf{X}$ to a vector of scores

$$\mathbf{s} = [s_1; \cdots; s_{N^c}] \tag{3}$$

where $s_i \in [0, 1]$, with $s_i = 0$ indicating (strongly) that the correspondence $\mathbf{c}_i$ is an outlier and $s_i = 1$ that it is an inlier. We approximate $f_{\boldsymbol{\theta}}$ using a feed-forward NN.

2.3.1 Network architecture

The proposed network architecture is based on a deep residual learning framework [9]. Instead of learning the desired mapping $\mathcal{H}(\mathbf{O})$ the residual network (ResNet) layers (Figure 2) aim at learning the residual mapping $\mathcal{F}(\mathbf{O}) := \mathcal{H}(\mathbf{O}) - \mathbf{O}$, where $\mathbf{O}$ denotes the output of the previous layer [9]. Thereby, the original mapping is recast into $\mathcal{H}(\mathbf{O}) = \mathcal{F}(\mathbf{O}) + \mathbf{O}$, which can be realized by the feedforward NNs using shortcut connections [19, 9]. These shortcut connections skip one or more layers of the network and in the case of ResNet layers just perform the identity mapping (Figure 2). Residual learning framework eases the optimization of the parameters and improves the performance of the traditional networks [9].

Our architecture (Figure 3), inspired by [16], consists of a 128-dimensional weight-sharing perceptron layer, followed by 12 ResNet layers (Figure 2), which are succeeded by another weight-sharing perceptron layer that reduces the dimension of each branch (a single putative correspondence) to one. The output of the last layer is passed through tanh and ReLU activation functions, which map the inferred score to the $[0, 1]$ range.

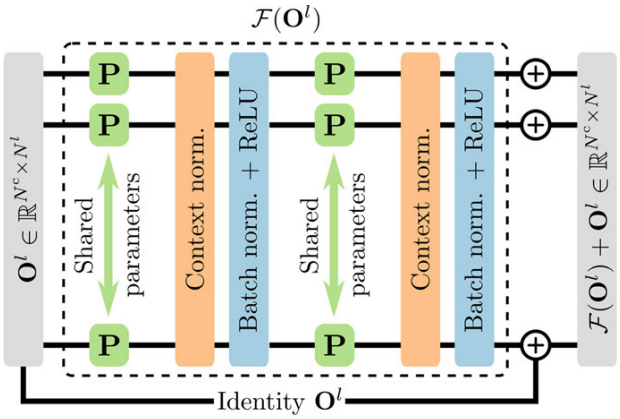

Figure 2: A single ResNet layer combines two blocks consisting of a 128 dimensional ($N^l = 128$) weight-sharing perceptron (weights copied for each correspondence) followed by a context normalization layer, batch normalization layer and ReLU activation function. Context normalization explicitly aggregates the information of the neighboring points within the point cloud pair.

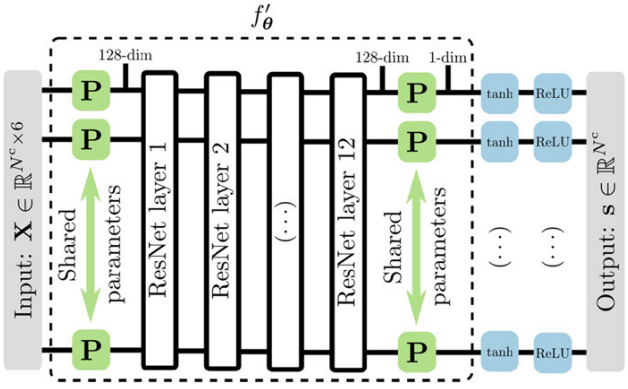

Figure 3: The proposed network architecture, adopted from [16], consists of 12 ResNet layers (see Figure 2), which are preceded and followed by a single layer of weight-sharing perceptrons that operate on each correspondence independently. The last weight-sharing perceptron is followed by the ReLU and tanh activation functions, which map the score to the [0, 1] range. For improved readability, we depict only 3 ResNet layers.

Because of the unordered nature of the point clouds, the order of the correspondences is arbitrary and permuting the rows of the input $\mathbf{X}$ should result in the equivalent permutation of the scores $\mathbf{s}$. To achieve the invariance to input permutation the weight-sharing perceptrons operate independently on each individual correspondence.[1] Because of this, the individual branches of the network do not obtain the information about the neighboring points explicitly and the local context is implicitly established using context normalization (CN) layers (Figure 2) defined as [16]

$$CN(\mathbf{O}^l) = (\mathbf{O}^l - \mathbf{1} \otimes \boldsymbol{\mu}^l) \oslash (\mathbf{1} \otimes \boldsymbol{\sigma}^l) \tag{4}$$

where $\mathbf{O}^l \in \mathbb{R}^{N^c \times N^l}$ is the output and N^l denotes the number of neurons of the layer l. $\boldsymbol{\mu}^l \in \mathbb{R}^{N_l}$ and $\boldsymbol{\sigma}^l \in \mathbb{R}^{N_l}$ are the row-wise mean value and standard deviation of $\mathbf{O}^l$. $\mathbf{1} \in \mathbb{R}^{N^c}$ is a vector of ones and $\otimes$ denotes the Kronecker product. Contrary to other normalization techniques used to improve the performance and convergence of NNs (e. g. batch normalization), CN operates across all correspondences but independently for each point cloud pair (see Section 2.4) [16]. Because CN layers are a composition of symmetric functions (mean value and standard deviation) the permutation invariance of the network is preserved.

Loss function

In order to optimize the parameters of the NN, we minimize the hybrid loss function

$$\mathcal{L}(\boldsymbol{\theta}, \mathbf{X}) = \sum_{n=1}^{B} \alpha \mathcal{L}^{\mathrm{b}}(\boldsymbol{\theta}, \mathbf{X}_n) + \beta \mathcal{L}^{\mathrm{t}}(\boldsymbol{\theta}, \mathbf{X}_n) \tag{5}$$

where $\mathcal{L}^{\mathrm{b}}$ is the binary classification loss and $\mathcal{L}^{\mathrm{t}}$ is the transformation loss. $\boldsymbol{\theta}$ denotes the parameters of the network and $\mathbf{X}_n$ represents a set of putative correspondences for point cloud pair n in a mini-batch consisting of B training examples.[2] The contribution of both loss functions is controlled using the hyperparameters α and β. The binary classification loss penalizes both types of error, the false positives as well as false negatives. Given N_n^c putative correspondences $\mathbf{c}_i$ with the corresponding ground truth labels y_i it is defined as a binary cross entropy function

$$\mathcal{L}^{\mathrm{b}} = \frac{1}{N_n^c} \sum_{i=1}^{N_n^c} -y_i \log(h(\mathbf{X}_n)_i) - (1 - y_i) \log(1 - h(\mathbf{X}_n)_i) \tag{6}$$

where $h(\cdot)$ is a sigmoid function

$$h(\mathbf{X}_n) = \frac{1}{1 + \exp(-f'_{\boldsymbol{\theta}}(\mathbf{X}_n))} \tag{7}$$

and $f'_{\boldsymbol{\theta}}(\mathbf{X}_n)$ denotes the output of the last weight-sharing perceptron layer before the tanh and ReLU activation functions (Figure 3).

1 This is achieved by copying the weights of the perceptrons N^c-times on the fly and thus making the width of the network variable.

2 Herein, a point cloud pair is represented by all correspondences belonging to a single supervoxel (Section 2.4) and a mini-batch denotes a group of supervoxels that fit into the memory of a GPU.

Minimizing the classification loss proves robust but can still let some outliers undetected [16]. Therefore, additional supervision—supported by the rigid body assumption—is introduced by penalizing the deviations from the ground truth rotation matrix $\mathbf{R}$ as

$$\mathcal{L}^{\mathrm{t}} = ||\mathbf{R} - g(\mathbf{X}_n, \mathbf{s})||_F^2 \tag{8}$$

where $g(\cdot)$ is a function, which estimates the rotation matrix based on the singular value decomposition of the weighted covariance matrix $\mathbf{\Sigma}_{\mathbf{X}_n} = \mathbf{X}_{n,1}^T \mathbf{S} \mathbf{X}_{n,2}$, with $\mathbf{X}_{n,1} := (\mathbf{X}_{ij})_{1\leq i\leq N, 1\leq j\leq 3}$, $\mathbf{X}_{n,2} := (\mathbf{X}_{ij})_{1\leq i\leq N, 4\leq j\leq 6}$, and $\mathbf{S} := \mathrm{diag}(\mathbf{s})$ [20]. $||\cdot||_F$ is the Frobenius norm.

2.3.2 Optimization

We optimize the parameters of the network using a variant of the stochastic gradient descent [13] in combination with the same benchmark data set that is used for the training of 3DSmoothNet, i. e. the 3DMatch data set [24]. The weights of the network are initialized by sampling from the truncated zero mean normal distribution and biases are set to zero. We have empirically discovered that adding the transformation loss from the start can actually harm the convergence, because the network is still incapable of correctly filtering out the outliers, which influence the estimation of the rotation matrix. Therefore, we start training by setting $\alpha = 1$ and $\beta = 0$. In a later stage of training, when the network can already correctly classify the majority of the correspondences we change β to 0.1, which enables additional improvement of the performance. We train the network using the batch-size of 16 and learning rate of 0.01. The batch-size is selected such that the data fit to the memory of the GPU and the learning rate was determined experimentally using a validation data set. For the empirical (Section 3) investigations we use a model corresponding to the iteration with the highest performance on the validation data set.

2.4 Oversegmentation using the supervoxel approach

In typical geomonitoring applications, parts of the repeatedly scanned scene may be stable over time, while others change due to the flow of earth and debris. Individual objects may be large enough to be detected as translated and rotated rigid bodies [4]. This causes discontinuities in the ground-truth displacement vector field. In order not to violate the local consistency assumption, the point clouds can therefore not be analyzed as one object, but rather have to be segmented into parts that do not cross these discontinuities. The discontinuities of the displacement vector field are not know beforehand, but they predominantly appear at the boundaries of larger objects. We therefore use a segmentation algorithm with boundary preservation [15] to segment the point cloud of the reference epoch into segments, denoted as supervoxels. These segments are deliberately allowed to be smaller than the expected actual objects. This over-segmentation is preferred to object segmentation because it allows the apparent geometrical changes within an object to be larger than the changes between objects. With supervoxels, only small, similar segments are clustered together thus enabling better boundary preservation. The supervoxels are obtained by minimizing the energy function

$$E(\mathbf{Z}) = \sum_{i=1}^{N}\sum_{j=1}^{N} z_{ij} d(\mathbf{p}_i, \mathbf{p}_j) + \lambda |C(\mathbf{Z}) - N_s| \tag{9}$$

where $z_{ij} \in \{0, 1\}$ with $z_{ij} = 1$ if the point $\mathbf{p}_i$ is a representative point of a supervoxel and the point $\mathbf{p}_j$ belongs to that supervoxel. λ is the adaptive weighting factor and $C(\mathbf{Z})$ is a function that counts the current number of supervoxels. The boundary preservation is achieved by incorporating the cosine similarity along with the traditional Euclidean distance in the similarity measure

$$d(\mathbf{p}_i, \mathbf{p}_j) = 1 - |\mathbf{n}_{\mathbf{p}_i} \cdot \mathbf{n}_{\mathbf{p}_j}| + 0.4 \frac{||\mathbf{p}_i - \mathbf{p}_j||_2}{r} \tag{10}$$

where $\mathbf{n}_{\mathbf{p}_i}$ and $\mathbf{n}_{\mathbf{p}_j}$ are the normal vectors of the points $\mathbf{p}_i$ and $\mathbf{p}_j$ respectively and r denotes the approximate size of the supervoxels in terms of a radius, which indirectly sets the approximate number of segments N_s.

Herein, the over-segmentation is performed only for the reference epoch and the putative correspondences are established independently from the segmentation using the whole point clouds. Following the segmentation, each supervoxel is fed to the NN-based filtering algorithm as an individual example, thus satisfying the local consistency assumption. Finally, the putative correspondences are filtered by rejecting all the correspondences with s_i strictly smaller than the classification threshold $\tau^{\mathrm{c}} = 0.5$ and the remaining correspondences are concatenated to form a single point cloud.[3] Such a robustly estimated dense displacement vector field can be used as the final output or

3 Empirically, for more than 95 % of the putative correspondences, the $s_i < 0.1$ or $s_i > 0.9$ in the rockfall simulator example. Therefore, we do not optimize the classification threshold and use $\tau^{\mathrm{c}} = 0.5$, if not explicitly specified differently.

as an input to an additional smoothing/interpolation algorithm such as the one given in [6].

3 Experiments

In this section we analyze the performance of F2S3, based on experiments conducted on two data sets. We start with a detailed analysis on a data set of a rockfall simulator, acquired in a controlled environment with per point ground truth, before evaluating the generalization capacity on a real case data set of a landslide located in the Alps. As baseline algorithms, we use the established C2C, C2M and M3C2 algorithms implemented in the open source software CloudCompare[4] as well as the raw correspondences established in the features space using 3DSmoothNet [5]. Additionally, we compare the results of the proposed outlier detection module to the results of RANSAC-based [3] filtering.

3.1 Rockfall simulator

The rockfall simulator (Figure 4) is a computer controlled piece of hardware composed of a rigid frame and a part that can rigidly translate vertically and rotate about a horizontal axis. The surfaces have a similar texture as rocks. The simulator allows mimicking a rockfall and was originally built for educational purposes. The moving part of the simulator (see Figure 4 right) is equipped with four mini prisms that can be used to establish the ground truth. Herein, we consider a scenario in which the moving part is displaced vertically by about 3 cm. The rockfall simulator was scanned from a distance of about five meters in two epochs using a Leica MS50. The mean resolution of the point clouds is 3 mm. The ground truth transformation parameters for the moving part of the simulator were estimated from high precision polar coordinate measurements to the four mini prisms using the same Leica MS50.

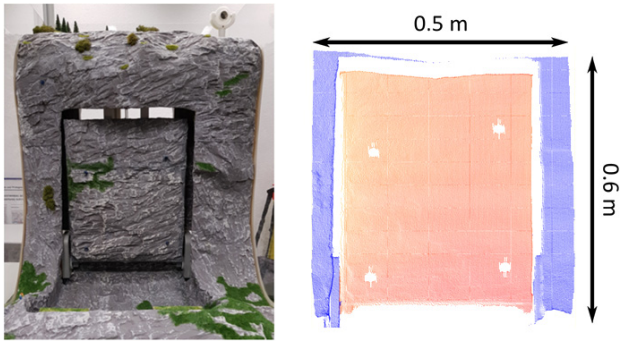

Figure 4: Rockfall simulator (left) and a point cloud acquired using a Leica MS50 (right). Red color shows the moving part and the blue color the stable part of the simulator. Holes in the red part of the point cloud show the location of the four mini prisms.

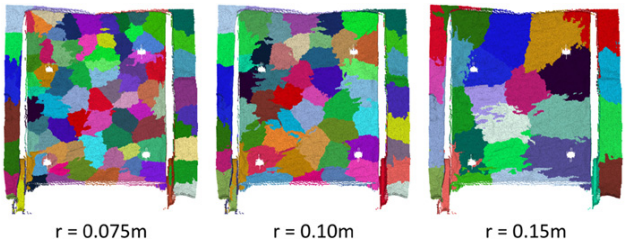

Figure 5: Results of the supervoxel segmentation of the rockfall simulator point clouds in relation to r. When r is too big the supervoxels cross the object boundaries (see for example the top right corner of the right figure). Colors of the supervoxels are selected randomly.

3.1.1 Supervoxel segmentation

The size of the supervoxels, and indirectly also their approximate number, is controlled with the hyperparameter r of Eq. 10. According to [15], the object boundaries can be preserved even if the distances between boundaries are smaller than the selected value of r. We evaluate the supervoxel segmentation algorithm on the reference epoch point cloud of the rockfall simulator (Figure 5). This data set is challenging for the supervoxel segmentation algorithm because individual parts of the simulator are almost parallel (stable and moving parts) and contribute little to the cosine similarity of the normal vectors in the dissimilarity measure (Eq. 10). Therefore, the boundaries can only be partially preserved, when r gets too big (see Figure 5 right). Based on this qualitative result, we use $r = 0.1$ m for the rockfall simulator experiments presented herein.

3.1.2 Filtering false correspondences

The results of the proposed NN-based outlier detection algorithm are depicted in Figure 6. The displacement vector field established in the 3DSmoothNet feature space contains a lot of outliers (Figure 6 left), which can be successfully removed using the proposed F2S3 (Figure 6 right). By directly using the model trained on the indoor scenes, more than 94 % of the points can be classified correctly as outliers and inliers, while less than 5 % are false negatives and about 1 % false positives. Here, a false negative

4 https://www.danielgm.net/cc/

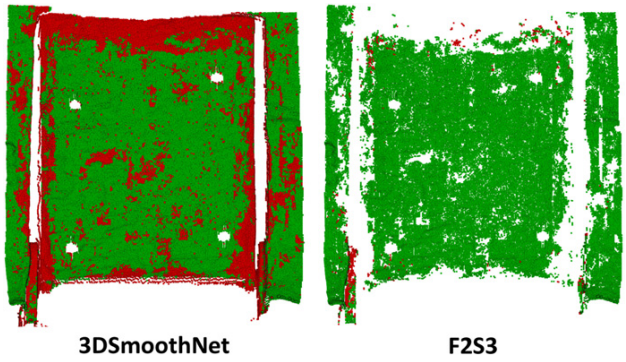

Figure 6: Results of the correspondence search in the feature space, before (left) and after (right) filtering. Red color denotes the outlier and the green color the inlier correspondences.

means that the ground truth label indicates an inlier i. e., the correspondence established in 3DSmoothNet feature space is actually correct, but the algorithm classifies the correspondence as wrong. Conversely, a false positive denotes that the correspondence established in 3DSmoothNet feature space is wrong but the algorithm classifies the correspondence as correct. The ground truth labels for the correspondences were established by considering the deviations between the putative and the ground truth corresponding point. If this deviation is less than 7.5 mm (2.5 times the mean resolution), the correspondence is labeled as an inlier and otherwise as an outlier.

3.1.3 Pointwise classification into stable and moved parts

We continue the comparison to the baseline algorithms by evaluating the capability of the algorithms to classify the point clouds into moved and stable parts. The ground truth labels are defined manually and can be seen in Figure 4 (right). For the C2C, C2M and M3C2 analysis we compute the displacements using the implementation available in CloudCompare and infer the labels based on the magnitude of the displacements. Specifically, all the points for which the estimated displacement is larger than 7.5 mm (as above) are classified as moved and the rest are classified as stable. We compare these results to the displacement vectors established using 3DSmoothNet and the proposed F2S3. After the outlier detection step of F2S3, only a subset of points (inliers) are remaining and can be classified as stable or moved based on the aforementioned threshold. To ensure a fair comparison, we infer the labels for the remaining points (outliers) based on the majority voting inside individual supervoxels.

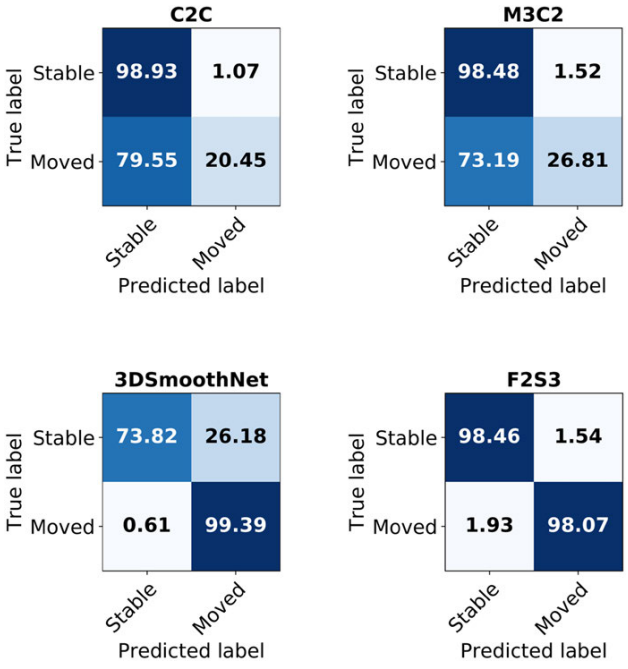

Figure 7: Confusion matrix. We use the algorithms to classify the points of the rockfall simulator into two classes: moved and stable. While the traditional algorithms perform better in stable areas and fail in the moved ones, the performance of 3DSmoothNet is better in the moved areas. The F2S3 achieves a consistent performance irrespective of the ground truth class. The results are shown in percentage.

The results are depicted in Figure 7.[5] Whereas all the traditional algorithms achieve high accuracy for the ground truth class "stable", they fail in classifying the moved areas correctly. Indeed, they classify more than 70 % of the moved points as stable. On the other hand, when using 3DSmoothNet, more than 26 % of the stable points are falsely classified as moved. This is caused by using the magnitude of the displacement vectors as the classification rule, which results in classifying a majority of all false correspondences as moved. Finally, F2S3 achieves more than 98 % accuracy for both classes and its performance is equal for stable and moving parts.

3.1.4 Quantitative analysis of the displacements

The goal of the point cloud based deformation analysis is not only detecting which parts of the scenery have moved/changed and which have remained stable, but also the quantification of the displacements. Because some of

5 We omit the results of C2M, which are very similar to the results of M3C2, in Figure 7.

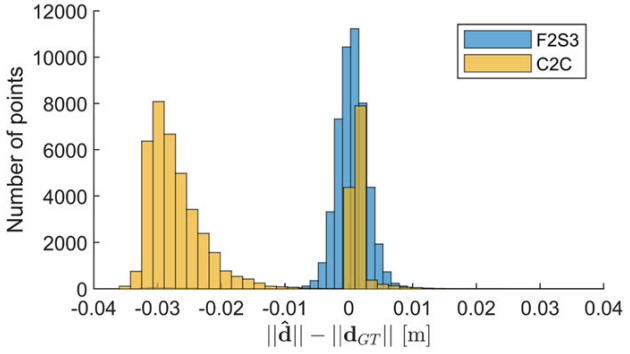

Figure 8: Deviations from the ground truth. For clarity, we only show the results of the F2S3 and C2C.

Table 1: Precision and recall of the estimated displacement magnitudes by different methods for the rockfall simulator point clouds.

	Precision [%]	Recall [%]
C2C	26.2	26.2
C2M	25.8	25.8
M3C2	26.5	25.8
3DSmoothNet	73.3	73.3
RANSAC-based	98.0	69.9
F2S3	98.8	66.7
F2S3 (vector distance)	98.4	66.5

the baseline algorithms output only the magnitudes of the displacements and not full 3D displacement vectors, we use the residuals between the estimated and the ground truth displacement magnitude as the performance metric. For F2S3, only the correspondences that were classified as inliers (67.5 % of all points) are considered in this analysis. Figure 8 and Table 1 show the results of the quantitative analysis of the estimated displacements using the rockfall simulator point cloud data. The residuals yielded by F2S3 are normally distributed with a mean value of 0 mm and a standard deviation of 3 mm. On the other hand, the residuals of C2C show a multimodal distribution with one peak centered at 0 mm and one at approximately 3 cm, which corresponds to the actual displacement of the moving part, indicating that this displacement is not detected by the C2C analysis. For improved readability we do not show the results of C2M, M3C2 and 3DSmoothNet in Figure 8, but the C2M and M3C2 deviations also show a multimodal distribution similar to the one of C2C, whereas the distribution of 3DSmoothNet resembles the one from F2S3 but with a very long tail (outliers). Table 1 summarizes the results for all the methods, where the precision denotes the ratio between the number of correct correspondences (i. e. the residuals smaller than 7.5 mm) and the number of all correspondences. Similarly, recall denotes the ratio between the number of correct correspondences and the number of all points in the reference epoch. For the traditional methods the precision equals approximately 26 %, which corresponds to the ratio of the stable points within the entire scene. In fact, as also indicated by the results denoted in Figure 7, the majority of the scenery is identified as stable using these standard algorithms, irrespective of the ground truth motion.

Our method instead performs equally well in stable and non-stable areas and achieves a much higher precision and recall, indicating that more than 98 % of the identified correspondences are correct. However, this is only possible because our approach rejects putative correspondences identified as outliers. Therefore, the recall cannot achieve 100 % but rather indicates the percentage of the point cloud with sufficiently unique features. Moreover, the difference between the recall of F2S3 and 3DSmoothNet denotes the percentage of the correct correspondences established by 3DSmoothNet that were classified as outliers by the proposed outlier detection step.

Because our method outputs real 3D displacement vectors, we also include the analysis based on the distances between the estimated and the ground truth displacement vectors. Table 1 shows that not only our method reaches a significantly higher precision and recall, the small drop in performance between "F2S3" and "F2S3 (vector distance)" highlights that it can also efficiently estimate the real 3D displacement vectors.

3.1.5 Comparison to RANSAC-based filtering

During training of the NN-based filtering algorithm we provide supervision through the ground truth transformation parameters of the congruence transformation (Eq. 8). Thereby, we make a (soft) assumption that the motion of each individual point pair or supervoxel can be explained by a single set of transformation parameters. Considering this assumption the filtering of the correspondences can also be formulated in the RANSAC framework.

Specifically, we consider each supervoxel independently and use the 3D congruent transformation model. Based on the selected probability $p := 0.99$ to sample at least three correct correspondences the required number of RANSAC iterations equals 20000. The relation

$$N_i^{\mathrm{R}} = \frac{log(1-p)}{log(1-\gamma_i^3)} \tag{11}$$

which gives a number of iterations N_i^{R} needed by RANSAC to sample at least three correct correspondences with

Table 2: Time complexity of the RANSAC-based and the proposed outlier detection step of F2S3 for the Rockfall simulator point clouds. We report the mean value and standard deviation of 20 runs. Initialization denotes the loading of the weights for the NN-based algorithm in the memory, which only has to be performed a single time independent of the number of point cloud pairs that are analyzed.

	Initialization [s]		Classification [s]	
	Mean	STD	Mean	STD
RANSAC-based	/	/	17.1	4.0
F2S3	8.6	0.1	2.5	0.1

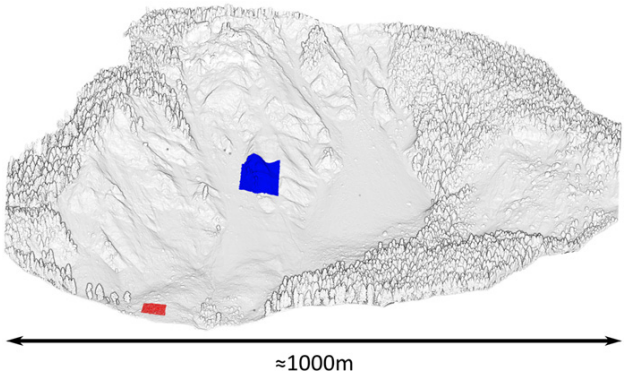

Figure 9: Reference point cloud of the landslide located in the Swiss Alps acquired using a drone based LiDAR system. We perform the deformation analysis for the areas marked with the red (Area 1) and blue (Area 2) color.

probability p, can be used to perform a dynamic check after each iteration i. γ_i denotes the inlier ratio and is in our case computed as a ratio between the cardinality of the largest set of inliers obtained up to the iteration i and the number of all putative correspondences. In order to avoid excessive iterations the RANSAC process is stopped if $N_i^{\mathrm{R}} < i$. All putative correspondences are classified as inliers if their Euclidean distance, after applying the estimated transformation parameters, is smaller than the RANSAC classification threshold τ^{R}, which is set to 7.5 mm for the rockfall simulator point clouds.

The results of the RANSAC-based filtering for Rockfall simulator point clouds are presented in Table 1 and the comparison of time complexity is given in Table 2. RANSAC-based filtering algorithm achieves a comparable precision and recall as the NN-based filtering algorithm proposed herein. Table 2 shows that even when analyzing a point cloud pair with few points and high percentage of inlier correspondences, the RANSAC-based filtering algorithm is approximately 2 times slower than the algorithm proposed herein.

3.2 Real case landslide in the Alps

The second data set used for demonstration herein consists of point clouds of a real landslide located in the Swiss Alps (Figure 9). The point clouds were acquired using a drone based LiDAR system (Riegl RiCOPTER) in two epochs about 2.5 months apart (July and September 2018). The point clouds of both epochs were georeferenced and the registration is hereinafter considered as error free. Based on the GPS and total station measurements, which are available for selected points on the edges of the landslide, displacements in the range of a couple of centimeters (bottom part) to a couple of decimeters (top part) were expected. Due to the large amount of data—the reference epoch has more than one billion points—we focus the analysis on the two selected areas for which also a ground truth is available through the high precision total station or GPS measurements (Area 1/blue and Area2/red of Figure 9). Each of these areas contains approximately one million points per epoch.

3.2.1 Quantitative analysis of the displacements

We follow the same evaluation procedure as in the rockfall simulator case. Specifically, we compare our method to C2C, C2M and M3C2. For improving the readability, we refrain from showing the results of 3DSmoothNet, which yields a heavily tailed distribution of the residuals, also in this example. Additionally, we compare the results to the ground truth, which is available in form of a displacement magnitude for a single point in the middle of Area 1 and a single point in the vicinity of Area 2. As we analyze relatively small areas, we assume that the displacement magnitude of these ground truth points is representative for all the points within the respective area. For the supervoxel segmentation we use $r = 1.5$ m for Area 1 and $r = 2.0$ m for Area 2, which is approximately 30 times the resolution of the point clouds in the respective area and corresponds to the ratio used in the analysis of the rockfall simulator.

The results are depicted in Figure 10 and summarized in Table 3. When compared to the ground truth, traditional methods significantly underestimate the magnitude of the displacements. The results of C2M and M3C2 are very similar, which is to be expected as both methods search for the corresponding points in the direction of the normal vector of either the underlying triangulated surface of plane fitted to the neighborhood of the point. Indeed, M3C2 could

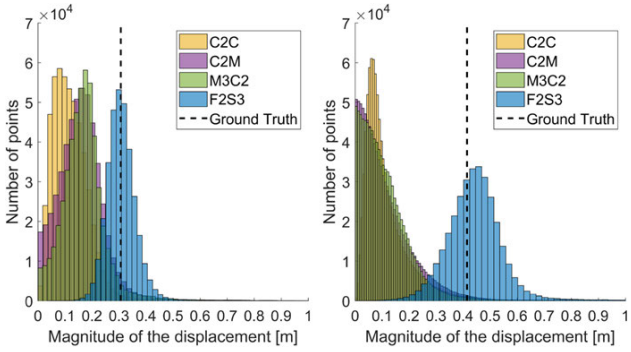

Figure 10: Histogram of the displacement magnitudes for Area 1 (left) and Area 2 (right). Traditional methods greatly underestimate the magnitude of the displacements.

Table 3: Median displacement magnitude. The results of our method ($\tau^c = 0.5$) for Area 2 are slightly biased due to the long tail of the displacement magnitude distribution.

Method	Area 1 Displ. [m]	Area 2 Displ. [m]
C2C	0.119	0.118
C2M	0.144	0.117
M3C2	0.160	0.113
F2S3 ($\tau^c = 0.5$)	0.307	0.464
F2S3 ($\tau^c = 1$)	0.304	0.439
Ground truth	0.306	0.414

be understood as a robuster version of C2M, as the plane fitting smooths out the noise of the point clouds. Furthermore, Table 3 shows that while the ground truth displacement for Area 2 is larger than the one for Area 1, all traditional methods estimate lower displacement magnitudes for Area 2. This hints, that the traditional methods are not only dependent on the magnitude of the displacements, but rather also on the type of the motion, point cloud resolution and possibly surface roughness. Further analysis of these dependencies is left for future work. On the other hand, our method can accurately estimate the displacement magnitudes with an error smaller than (Area 1) or close to (Area 2) 5 % of the actual motion. This error corresponds to less than half of the mean resolution of the point clouds.

3.2.2 Outlier detection analysis

The lower resolution and higher noise of the point clouds, combined with the larger motion, presumably have a negative effect on the correspondence search and on the outlier detection algorithm. This results in a long tail of the distribution of the displacement magnitudes, especially for Area 2, which is to a large extent covered with small gravel with no distinct features. We therefore perform an additional analysis in which we assume that the ground truth displacement magnitude $||\mathbf{d}_{GT}|| = 0.414$ m is representative for all the points in the Area 2. Assuming the standard deviation of the estimated displacement magnitudes $\sigma_{||\hat{\mathbf{d}}||} \approx 0.1$ m for all the methods in Table 3, we label all correspondences $\mathbf{c}_i$ for which

$$||\mathbf{d}_{GT}|| - 3\sigma_{||\hat{\mathbf{d}}||} \leq ||\hat{\mathbf{d}}_{\mathbf{c}_i}|| \leq ||\mathbf{d}_{GT}|| + 3\sigma_{||\hat{\mathbf{d}}||} \tag{12}$$

as correct and the rest as wrong. Considering these labels, 65 % of the correspondences established using 3DSmoothNet without outlier detection are wrong for Area 2. After the outlier detection with the approach proposed herein (using $\tau^c = 0.5$), more than 81 % of the inliers are actually correct correspondences and less than 19 % are wrong, with a recall of 33 %. As shown in Figure 11 (left), most of the false positives left after the outlier detection lie in the areas covered with gravel (left and right sides of the ridge). Due to the relative large number of false positives, we analyze the effect of increasing the threshold τ^c to 1, i. e. accepting only the correspondences with very high confidence.[6] With this, the percentage of correct correspondences can be increased to 98.5 % with a recall of 27.7 % (Figure 11). This result indicates that a scene specific threshold might be beneficial; the appropriate choice is a topic for future research.

3.2.3 Comparison to RANSAC-based filtering

Finally, we compare the results of the proposed approach to the RANSAC-based filtering as described in Section 3.1.5. We set the RANSAC classification threshold τ^R to 20 cm, which corresponds to two sigma standard deviation of the estimated displacement magnitudes (see Section 3.2.2). The comparison of the time performance for both Area 1 and Area 2 is shown in Table 4 and the results for Area 2 are graphically depicted in Figure 11. While for Area 2, RANSAC-based filtering achieves a higher recall of 31.1 % compared to 27.7 %, it also yields a larger percent of false positives 4.8 % compared to 1.39 % of our method.

Furthermore, the time complexity of the RANSAC-based filtering algorithm is strongly dependent on the percentage of inliers and, when compared to our method, the

6 To avoid the quantization errors we use $1 - 1 \cdot 10^{-5}$ in our code.

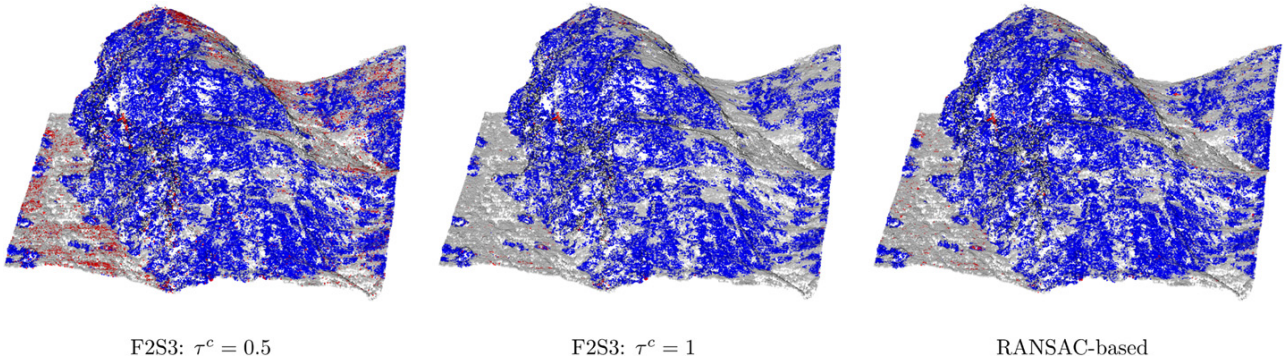

Figure 11: Point cloud of the Area 2 after the outlier detection algorithm using the algorithm proposed herein with $\tau^c = 0.5$ (left) and $\tau^c = 1$ (middle) and with RANSAC-based outlier detection algorithm (right). Blue color denotes the true positives, red color the false positives and grey color the negatives. While yielding a slightly higher recall, RANSAC-based outlier detection also yields more false positives. Note how the false positives mostly lie on the flat areas covered by gravel and small cobbles with no distinctive features. Figure best seen in color.

filtering can be up to 300 times slower (see Area 2 in Table 4) if only approximately 30 % of the putative correspondences are actual inliers. On the other hand, our method is independent of the inlier precentage and requires no iterations (inference is performed in a single forward pass).

This high time complexity strongly limits the applicability of the RANSAC-based outlier detection for the near real-time deformation monitoring of large natural scenes. For example, the whole point clouds of the landslide (Figure 9) were split into more than 400 areas for the analysis. If one would replace our method with the RANSAC-based filtering, the computation time of the outlier detection, would increase to 20 days while the NN-based approach presented herein only takes 2 hours, assuming the mean classification time of Area 1 and Area 2 shown in Table 2.

Table 4: Time complexity of the RANSAC-based and the proposed outlier detection step of F2S3 for the point clouds of the real landslide in the Swiss Alps. Initialization denotes the loading of the weights for the NN-based algorithm in the memory, which only has to be performed a single time independent of the number of point cloud pairs that are analyzed.

	Initialization [s]		Classification [s]	
	Area 1	Area 2	Area 1	Area 2
RANSAC-based	/	/	1159.8	7573.1
F2S3	8.3	8.5	13.2	25.4

4 Conclusion

In this work, we have proposed F2S3 a new method for deformation analysis based on point cloud data. We complement the recent idea of establishing the correspondences in the feature space with an outlier detection algorithm, which classifies the putative correspondences into inliers and outliers. Using two different data sets, we highlight the shortcomings of the traditional approaches, while showing that our approach correctly estimates 3D displacement vectors and its outlier detection step is very time efficient.

In the future work we will perform an extensive sensitivity analysis of the available deformation analysis methods for point cloud data, regarding the point cloud resolution, magnitude and type of the motion, as well as the available geometric and radiometric features of the surface in a simulated environment. The results of this analysis will provide a better understanding about the limitations of individual methods and could serve as a guideline, hinting at which method to use for certain application cases.

Acknowledgment: The point cloud data of the landslide in the Swiss Alps used in this paper were collected and georeferenced by ALTAMETRIS. We thank Nicolas Ackermann (Swiss Federal Railways) for providing these data. The Amt für Wald und Naturgefahren of Kanton Graubünden (Andreas Huwiler) has provided the ground truth data of the landslide in the Swiss Alps.

References

[1] Christopher M Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[2] Haowen Deng, Tolga Birdal, and Slobodan Ilic. Ppf-foldnet: Unsupervised learning of rotation invariant 3d local descriptors. In *European conference on computer vision (ECCV)*, 2018.

[3] Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[4] Z Gojcic, C Zhou, and A Wieser. Learned compact local feature descriptor for tls-based geodetic monitoring of natural outdoor scenes. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, pages 113–120, 2018.

[5] Zan Gojcic, Caifa Zhou, Jan D Wegner, and Andreas Wieser. The perfect match: 3d point cloud matching with smoothed densities. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5545–5554, 2019.

[6] Zan Gojcic, Caifa Zhou, and Andreas Wieser. Feature-based approaches for geomonitoring using terrestrial laser scanning point clouds. In *MLEG2019*, 2019.

[7] Zan Gojcic, Caifa Zhou, and Andreas Wieser. Robust pointwise correspondences for point cloud based deformation monitoring of natural scenes. In *Joint International Symposium on Deformation Monitoring (JISDM)*, 2019.

[8] Ian J. Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, Cambridge, MA, USA, 2016. http://www.deeplearningbook.org.

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE conference on computer vision and pattern recognition (CVPR)*, pages 770–778, 2016.

[10] Christoph Holst and Heiner Kuhlmann. Challenges and present fields of action at laser scanner based deformation analyses. *Journal of applied geodesy*, 10(1):17–25, 2016.

[11] Christoph Holst, Berit Schmitz, Achim Schraven, and Heiner Kuhlmann. Eignen sich in standardsoftware implementierte punktwolkenvergleiche zur flächenhaften deformationsanalyse von bauwerken? eine fallstudie anhand von laserscans einer holzplatte und einer staumauer. *ZfV-Zeitschrift für Geodäsie, Geoinformation und Landmanagement*, (zfv 2/2017), 2017.

[12] Marc Khoury, Qian-Yi Zhou, and Vladlen Koltun. Learning compact geometric features. In *IEEE International Conference on Computer Vision (ICCV)*, 2017.

[13] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, pages 5545–5554, 2019.

[14] Dimitri Lague, Nicolas Brodu, and Jérôme Leroux. Accurate 3d comparison of complex topography with terrestrial laser scanner: Application to the rangitikei canyon (nz). *ISPRS journal of photogrammetry and remote sensing*, 82:10–26, 2013.

[15] Yangbin Lin, Cheng Wang, Dawei Zhai, Wei Li, and Jonathan Li. Toward better boundary preserved supervoxel segmentation for 3d point clouds. *ISPRS journal of photogrammetry and remote sensing*, 143:39–47, 2018.

[16] Kwang Moo Yi, Eduard Trulls, Yuki Ono, Vincent Lepetit, Mathieu Salzmann, and Pascal Fua. Learning to find good correspondences. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2666–2674, 2018.

[17] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted Boltzmann machines. In *International conference on machine learning (ICML)*, pages 807–814, 2010.

[18] H Neuner, C Holst, and H Kuhlmann. Overview on current modelling strategies of point clouds for deformation analysis. *Allg. Vermess. Nachr.(AVN)*, 123:328–339, 2016.

[19] Brian D Ripley and NL Hjort. *Pattern recognition and neural networks*. Cambridge University Press, 1996.

[20] Olga Sorkine-Hornung and Michael Rabinovich. Least-squares rigid motion using svd. *Computing*, 1(1), 2017.

[21] Andreas Wagner, Wolfgang Wiedemann, and Thomas Wunderlich. Fusion of laser-scan and image data for deformation monitoring—concept and perspective. In *International Conference on Engineering Surveying (INGEO)*, pages 157–164, 2017.

[22] Th Wunderlich, W Niemeier, D Wujanz, C Holst, F Neitzel, and H Kuhlmann. Areal deformation from tls point clouds—the challenge. *Allg. Vermess. Nachr.(AVN)*, 123(12), 2016.

[23] Zi Jian Yew and Gim Hee Lee. 3dfeat-net: Weakly supervised local 3d features for point cloud registration. In *European Conference on Computer Vision (ECCV)*, 2018.

[24] Andy Zeng, Shuran Song, Matthias Nießner, Matthew Fisher, Jianxiong Xiao, and Thomas Funkhouser. 3dmatch: Learning local geometric descriptors from rgb-d reconstructions. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1802–1811, 2017.

Article note: This paper is a significantly extended version of the paper originally published at JISDM 2019 [7].