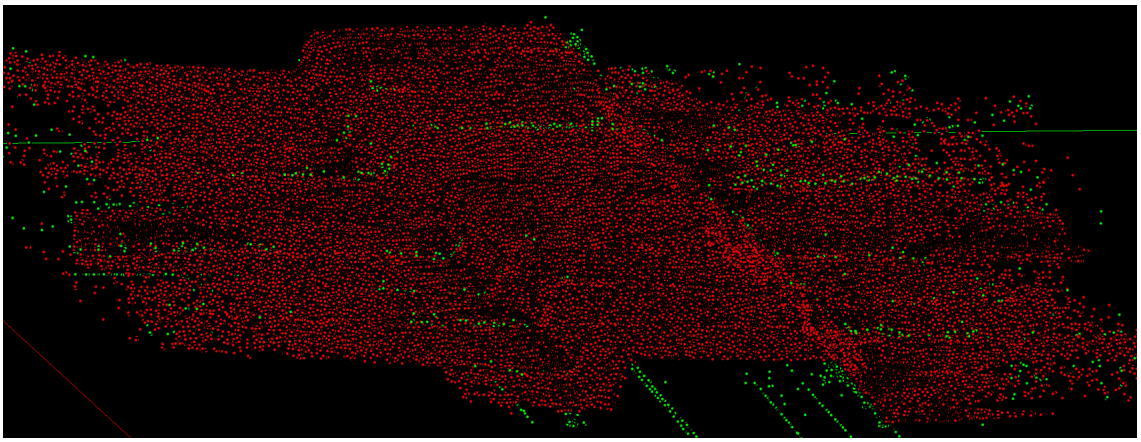
RnD Project Report

Outlier Detection in Sonar Point Cloud

Name	Email
Rasmus Ellerbæk Ørndrup	201371042@post.au.dk
Thomas Hallager Pedersen	201371002@post.au.dk



2018

Aarhus University

Contents

Glossary

ALM Augmented Lagrange Multiplier.

GPU Graphics Processing Unit.

KNN K-Nearest Neighbors.

NB Naive-Bayes.

PCA Principal Component Analysis.
PCP Principal Component Pursuit.

RPCA Robust Principal Component Analysis.

SVM Support Vector Machine.

Introduction 1

Point clouds are sets of data points existing in a 3-dimensional space. They are often used to efficiently represent 3D shapes and surfaces, which make them useful in a wide range of applications such as 3D CAD modeling and surveying. This project is concerned with point clouds created from seafloor surveys using sonar, where the point clouds model the surface structure of the surveyed sea floor. The sensing processes used to create a point cloud introduce noise and erroneous measurements, and the accuracy of the surface estimation is dependent on filtering out these noisy and erroneous samples. Determining a reliable process for noise filtering is therefore an important step in utilizing a point cloud.

1.1 Project Description

The goal of the project is to find an efficient and effective method of filtering noisy samples from a point cloud generated from sonar seafloor surveys. The project is done in cooperation with EIVA[3], who has posed the problem and supplied the experimental datasets.

The project will analyze the characteristics of the experimental data and explore different solutions to the problem to determine an efficient, effective and generalized method of filtering noisy samples. The methods are evaluated in terms of accuracy w.r.t. the supplied labels of the dataset, but also in terms of a visual inspection of the resulting surface estimation.

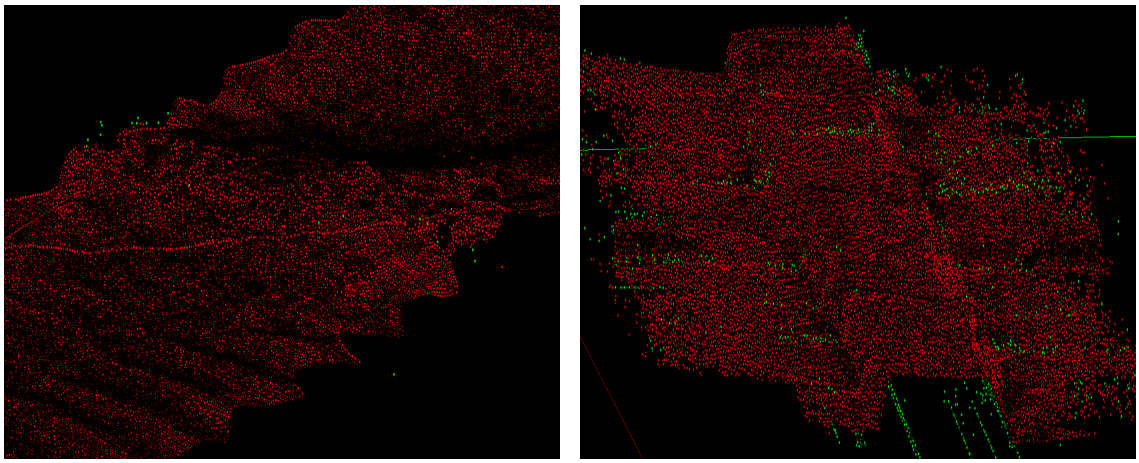
1.2 Related Work

The work in this project has been inspired by [1], which utilizes Robust Principal Component Analysis (RPCA) to perform outlier detection on a point cloud dataset constructed from a depth camera. RPCA is an algorithm for decomposing a data matrix into a low-rank and a sparse matrix, which represent the data in a manner that simplifies outlier detection. [2] describes a method of computing RPCA.

Data Description 2

The experimental data is based measurements from sonar surveys. Initially, only one dataset of approximately 530.000 samples was supplied for the project, but this dataset proved insufficient for proper analysis due to a lack of noisy samples (0.048%). An additional dataset was therefore supplied, which is both larger with approximately 1.810.000 samples and contains a higher ratio of noise samples (2.2%). This secondary dataset became the primary focus of analysis for the project. The two datasets are illustrated in fig. 2.1. Both datasets describe an elongated stretch of surface, but while the first dataset is a fairly flat surface with a single depression, the second dataset has a distinct *trench* along the entire surface.

The measured surface is less well-defined at the outer edges of the dataset, which is apparent in fig. 2.1(b). The surface at the outer edges of the dataset is less densely populated by data points, and also contains significant gaps. For this reason, the correct samples appear very similar to noise, and may be difficult to distinguish. The performance is therefore expected to be worse at the outer edges of the dataset.



(a) (b)

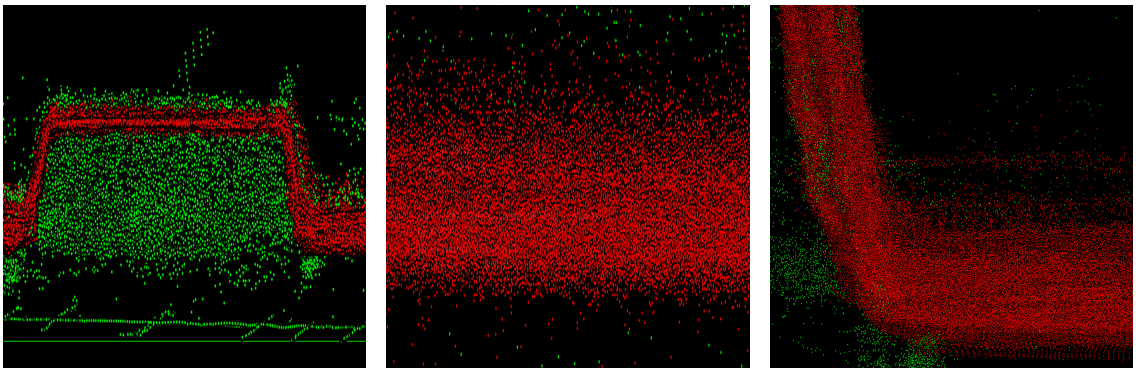
Figure 2.1: (a) illustrates the first dataset, and (b) illustrates the second dataset. Red data points are labeled as correct samples and green data points are labeled as errors.

2.1 Noise Characteristics

When analyzing the point cloud dataset, two categories of noisy samples are present in the data. The first and most prominent category is what appears to be measurement errors, which are data points located far from the actual surface. fig. 2.2(a) illustrates a subset

of the data, in which there are large amounts of measurement errors. These appear to be correlated with rapid vertical changes in the surface - as is apparent from the dense cloud of noise samples inside the trench. This may stem from interference between the sonar sound waves reflecting off the corners and more vertically aligned surfaces at the sides of the trench. Additionally, there are also artifacts present in the bottom of the image. These are present as straight lines far from the actual surface.

The second and more ambiguous category of noise is Gaussian noise around the actual surface measurement. Figure 2.2(b) and fig. 2.2(c) illustrates this at two distinct parts of the dataset - one in an area where the surface is almost flat, and one in an area where the surface contains rapid changes in verticality. It is apparent from the figures that these seemingly noisy samples are not labeled as such, which is to be expected since it would require an excessive amount of manual labelling. We consider these samples as noise, and attempt to filter them as such.



(a) (b) (c)

Figure 2.2: (a) illustrates a subset of the data with large amounts of apparent measurement errors, (b) and (c) illustrates Gaussian noise around the actual surface. Red data points are labeled as correct samples and green data points are labeled as errors.

Problem Analysis 3

There are different approaches to solving the problem of filtering noisy outlier samples from a point cloud dataset. This section provides a general analysis of the problem and potential solutions, to determine what solution(s) are the best candidates for further research - given the problem and the available data. Initial tests were done to test the efficacy of the proposed solutions, and to gain a better understanding of their particular limitations.

The raw point cloud dataset consists of number of points, represented in 3 dimensions - x, y and z. Extrapolating on the noise characteristics described in section 2.1, and putting them in the context of the problem, it would seem that there is little correlation between the x and y components of the data points and the *noisiness* of a data point. The primary indicator of the noisiness of a data point is therefore its z component - and in particular, its z component in relation to its local neighborhood of samples. This fact is leveraged by both of the solution proposals.

The performance analysis of the methods has to be discussed, because it is not completely straightforward. The simplest metric is the classification accuracy in comparison to the supplied labels of the dataset. However, as shown in section 2.1, many samples are ambiguous in nature - and their label dependent on the granularity at which the problem is analyzed. In this context, strictly looking at the classification accuracy is therefore not necessarily a sufficient metric for the performance of a method. Knowing this, the goal then becomes to detect as many noise-labeled samples as possible, while only detecting the non-noise-labeled samples that are ambiguous in nature. We therefore tolerate some classifications that would be considered false positives in relation to the labels. To accomplish this, we have to also rely on visual inspection of the results to ensure that the detected noise samples are coherent with the goal. This is not as well-defined a metric, as could be desired, but it was deemed out of scope of the project to manually label the ambiguous samples - thus the need for working within the confines of what was available.

3.1 Feature-Based Classification

The first solution proposal is to treat the problem of noise filtering as a classification problem and solve using supervised machine learning methodologies. Specifically, we looked at traditional machine learning classification methods such as K-Nearest Neighbors (KNN), Support Vector Machine (SVM) and Naive-Bayes (NB).

A classification approach is predicated on the existence of a model that can adequately

discriminate noise across different datasets. Creating such a model requires a representation of the data points that is independent of the scale of x, y and z components, while preserving the relative differences between samples. This is necessary to generalize a model to different datasets with different scales. Normalizing the data achieves this, but as it has been established that there is little correlation between the x and y components of a data point, and its characteristic as noise, only the z components is left as a feature to use for classification. This is not adequate to discriminate noise samples, so additional features are required. These features should describe the properties of a data point, which best separates noise from correct samples. Determining what features accomplishes this is the primary focus of research for this approach, in addition to determining what classification algorithm to utilize.

3.1.1 Feature Candidates

The supplied datasets contain a set of pre-computed features, which serve as a basis for evaluation. Two instances of these pre-computed features are supplied - one with a search radius of $0.8m$ and $K = 8$, the other with a search radius of $1.4m$ and $K = 14$. These are shown in table 3.1.

Table 3.1: Existing features

ID	**Feature description**	**Parameters**
F1	Distance to neighbor	-
F2	Distance to average surface	search radius
F3	Neighbors in sphere	search radius
F4	Z-summation in circle	search radius
F5	Mean z-distance to K-nearest neighbors	K neighbors
F6	Mean distance to K-nearest neighbors	K neighbors

In addition to the pre-computed features, we also considered other potential features. These are shown in table 3.2.

Table 3.2: New features

ID	**Feature description**	**Parameters**
F7	Squared, centered (within radius) z-value	search radius
F8	Max distance to K-nearest neighbors	K neighbors
F9	Number of neighbors within sphere	search radius
F10	Neighbors in sphere	search radius
F11	Centered z-summation within sphere	search radius
F12	Centered, mean z-value within sphere	search radius

Common for all features is that they describe the local neighborhood of data points. The features are based on the notion that the surface spans the entire point cloud dataset, and that the density of data points within the surface is higher than for the surrounding noise. Thus, the noise samples are the ones that differ significantly from the mean, local z-value and those that lie within a less densely populated neighborhood of the dataset.

3.1.2 Preliminary Tests and Limitations

The features were evaluated individually and collectively to asses their efficacy in conjunction with the proposed classification algorithms. Results of the proposed features in table 3.2 has not been listed as the performance had a similar trend to the results of the existing features in table 3.1.

Tables 3.3 to 3.4 shows the performance of the KNN using the existing features in table 3.1. These tables shows that the KNN algorithm does not generalize the problem of classifying the noise samples the true samples. It is seen that a KNN using $k = 2^0 = 1$ performs the best and increasing the number of voting neighbors lowers the performance as the algorithm tends to classify all samples as *not noise / true samples*. In the tables the labels *S*, *N*, *CS* and *CN* are used. *S* and *N* are abbreviations of (annotated) true samples and noise respectively, whereas CS and CN are abbreviations of ***classified*** true samples and ***classified*** noise respectively. This accuracy metric is used to better understand the classification as the usual accuracy metric would only show the percentage of true sample classified correctly. As the number of noise samples is very small related to the total number of samples the usual overall accuracy metric would indicate good results (97.8%) if no noise samples were found. We see that this is the case for the KNN algorithm both for the features individually and collectively. Almost all samples are classified as true samples and thus the KNN algorithm has a poor performance.

Table 3.3: kNN using a single feature

kNN		**F1**		**F2**		**F3**		**F4**		**F5**		**F6**	
		S	*N*	*S*	*N*	*S*	*N*	*S*	*N*	*S*	*N*	*S*	*N*
$k = 2^0$	*CS*	0.978	0.022	0.979	0.021	0.980	0.020	0.978	0.022	0.978	0.022	0.978	0.022
	CN	0.972	0.028	0.977	0.023	0.981	0.019	0.974	0.026	0.976	0.024	0.972	0.028
$k = 2^1$	*CS*	0.999	0.001	1	0	1	0	0.999	0.001	1	0	0.999	0.001
	CN	0.999	0.001	0.999	0.001	1	0	0.999	0.001	1	0	0.999	0.001
$k = 2^2$	*CS*	1	0	1	0	1	0	1	0	1	0	1	0
	CN	1	0	0.999	0.001	1	0	0.999	0.001	1	0	1	0
$k = 2^3$	*CS*	1	0	1	0	1	0	1	0	1	0	1	0
	CN	1	0	1	0	1	0	1	0	1	0	1	0
$k = 2^4$	*CS*	1	0	1	0	1	0	1	0	1	0	1	0
	CN	1	0	1	0	1	0	1	0	1	0	1	0
$k = 2^5$	*CS*	1	0	1	0	1	0	1	0	1	0	1	0
	CN	1	0	1	0	1	0	1	0	1	0	1	0
$k = 2^6$	*CS*	1	0	1	0	1	0	1	0	1	0	1	0
	CN	1	0	1	0	1	0	1	0	1	0	1	0
$k = 2^7$	*CS*	1	0	1	0	1	0	1	0	1	0	1	0
	CN	1	0	1	0	1	0	1	0	1	0	1	0
$k = 2^8$	*CS*	1	0	1	0	1	0	1	0	1	0	1	0
	CN	1	0	1	0	1	0	1	0	1	0	1	0

Table 3.4: kNN using all 6 existing features

	$k = 2^0$		$k = 2^1$		$k = 2^2$		$k = 2^3$		$k = 2^4$		$k = 2^5$		$k = 2^7$		$k = 2^8$	
kNN	*S*	*N*	*S*	*N*	*S*	*N*	*S*	*N*	*S*	*N*	*S*	*N*	*S*	*N*	*S*	*N*
CS	0.978	0.022	0.999	0.001	1	0	1	0	1	0	1	0	1	0	1	0
CN	0.972	0.028	0.998	0.002	0.999	0.001	1	0	1	0	1	0	1	0	1	0

The performance of the NB and SVM algorithms are shown in table 3.5 and table 3.6

respectively. Table 3.5 lists the performance of the NB algorithm for individual and collective features. Table 3.6 lists the configurations for the SVM algorithm. It is apparent that neither algorithm performs well in these initial tests, since the classifications appear to be skewed heavily towards either reaching a high CS accuracy or a high CN accuracy - never achieving a balance between the two.

The parameters were chosen by a coarse grid search. The listed performance may therefore not represent the highest-achievable performance using these features. However, it is apparent that more work is required to achieve satisfactory performance - either w.r.t. parameter optimization, feature generation or both. We were concerned by the poor initial performance, which made us consider whether it would be possible to reach a satisfactory performance at all. Non of the features indicates to make the good clusters or linearly separable metrics as the kernel SVM has the ability to somewhat classify both noise and true samples. However the SVM didn't show good performance results w.r.t the the grid search made. Further thorough investigating of the the features could be done by plotting them individually or in groups of 2 or 3 (x, y, z -axis'). In the preliminary investigations similar plots were created however we did not see any clear clustering or possibility of separation of the data. We therefore also looked at entirely different approaches to solving the noise detection problem. The most promising of these approaches is described in the next section.

Table 3.5: Naive Bayes

	F1		**F2**		**F3**		**F4**		**F5**		**F6**		**F1 to F6**	
NB	*S*	*N*	*S*	*N*	*S*	*N*	*S*	*N*	*S*	*N*	*S*	*N*	*S*	*N*
S	0.995	0.005	0.991	0.009	1	0	0991	0.009	0.998	0.002	0.994	0.06	0.973	0.027
N	0.991	0.009	0.990	0.010	1	0	0.989	0.011	0.995	0.005	0.990	0.010	0.965	0.035

Table 3.6: Kernel (RBF) SVM

Kernel SVM		*S*	*N*
F6: $\gamma = 2^{-4}; C = 2^{-5}, tol = 1e^{-5}$	*S*	0.864	0.136
	N	0.821	0.179
F6: $\gamma = 2^{-7}; C = 2^{-5}, tol = 1e^{-3}$	*S*	0.383	0.617
	N	0.397	0.603
F6: $\gamma = 2^{-10}; C = 2^{-5}, tol = 1e^{-1}$	*S*	0.155	0.845
	N	0.152	0.848
F1 to F6: $\gamma = 2^{-1}; C = 2^{15}, tol = 1e^{-3}$	*S*	0.395	0.605
	N	0.335	0.665
F1 to F6: $\gamma = 2^{-1}; C = 2^{15}, tol = 1e^{-1}$	*S*	0.605	0.395
	N	0.335	0.665

3.2 Statistical Outlier Detection

Another approach to solving the problem, is to think of the noise samples as statistical *outliers*, and treating the problem as a statistical outlier detection problem. This can be done in an unsupervised manner, thus removing the dependency on manual labeling and the limitations of models imposed by mislabeling. This is a considerable advantage, since manually labeling a point cloud dataset is an expensive task, and even more so to reach the granularity required to classify the most ambiguous noise samples.

3.2.1 Principal Component Analysis

Principal Component Analysis (PCA) is a statistical procedure that applies an orthogonal transformation to a set of data points, transforming the possibly correlated dimensional variables of the feature space into linearly uncorrelated variables called *principal components*. These components will represent the orthogonal axes along which the set of data points have the highest degree of variance. This can be a useful tool for transforming the point cloud, in its original 3-dimensional (x, y, z) feature space, into a feature space where it is easier to discriminate outliers from *inliers*.

With the assumption that the measured surface spans the entire dataset, we can separate the dataset into regions small enough to ensure that the surface is roughly planar within each region. Within each of these regions the outlier detection is then a matter of finding those data points that lie sufficiently far from the plane. However, this plane is not explicitly defined and has to be estimated, which is where PCA is a useful tool. Applying PCA to a small region of the dataset is illustrated in fig. 3.1, where the first and second principal components span the plane and the third component is the normal to the plane. In this *ideal* scenario, the outliers are separable from the inliers as the samples that have a non-zero third component. In scenarios where the plane is not exactly planar, the discrimination requires a threshold along the third component, which will define the aggressiveness of the filtering.

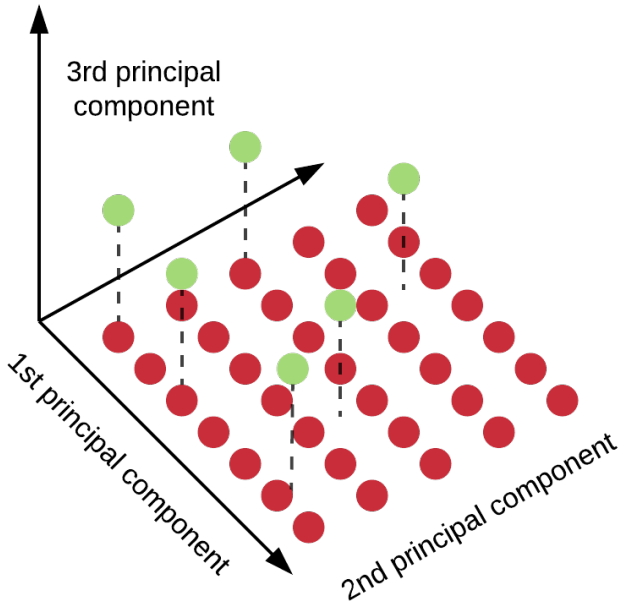


Figure 3.1: Example of how PCA may be used for surface estimation. The first and second principal components span the plane, while the third component is the normal to the plane.

3.2.2 Robust PCA

PCA works under the assumption that inliers vastly outnumber the outliers, as the procedure utilizes the covariance matrix of the data, which is sensitive to outliers. In scenarios where the data is grossly affected by outliers, PCA will not produce the desired result. RPCA is a modification to PCA that allows it to work in the presence of grossly corrupted data. RPCA is based on the assumption that a set of measurements can be decomposed into two components - a low-rank matrix describing the linearly dependent aspects of the measurements and a sparse matrix describing the remainder. There are several ways of solving this decomposition problem, one of which is Principal Component Pursuit (PCP) as described in [2].

In the context of this problem, where the surface is analyzed in small roughly planar regions, the surface is expected to be highly correlated and linearly dependent w.r.t. the plane spanned by the x and y dimensions with small deviations along the z-axis. The inliers will therefore be described primarily by the low-rank matrix, and be represented sparsely (0's along most dimensions) in the sparse matrix. Conversely, the outliers are uncorrelated w.r.t. all dimensions, and will be primarily described by the sparse matrix. An example of the sparse matrix found by applying RPCA is shown in fig. 3.2. It is apparent that the bulk of inliers are located near the origin, and that all inliers are sparse or close to sparse in the z-dimension. It is also apparent that the z-value of most outliers enables discrimination from inliers by a threshold along the z-axis.

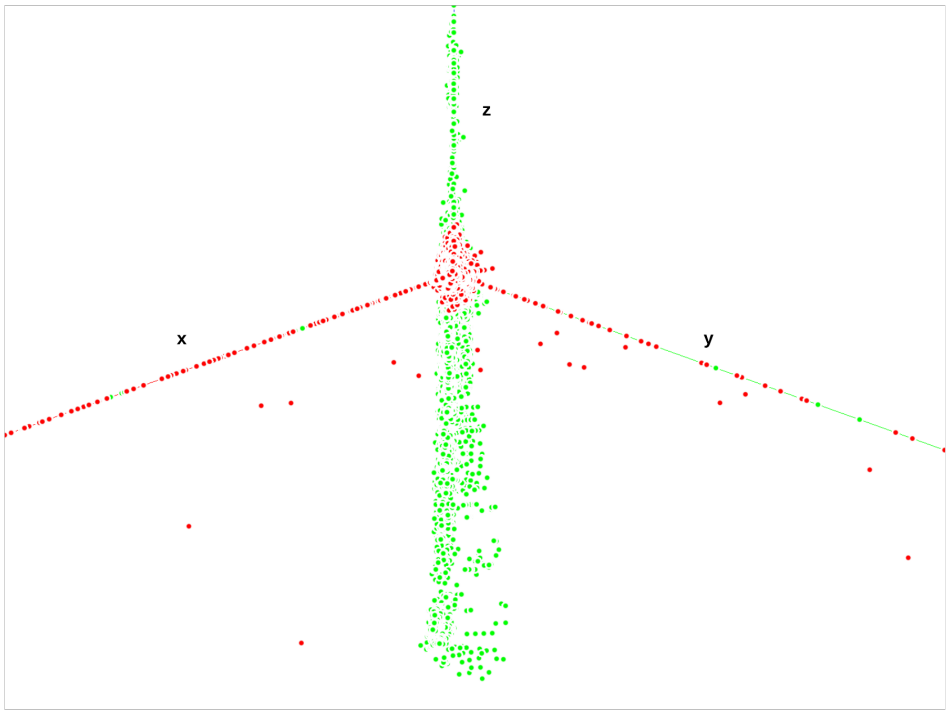


Figure 3.2: Aggregation of the sparse matrices from RPCA applied to 101×101 regions of the dataset.

Methodology 4

From initial analysis and tests, using RPCA for statistical outlier detection seemed feasible and promising. We therefore decided to focus on this method moving forward. The algorithmic approach used to detect and filter outliers using RPCA is described in this section.

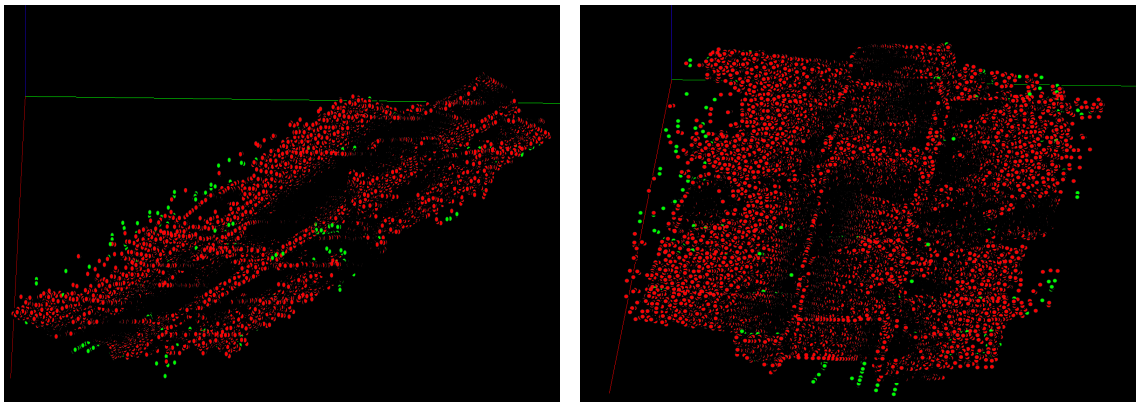
4.1 Preprocessing

4.1.1 Scaling and Axis Alignment

The elements of the data matrix $\mathbf{M}$ are scaled to be within the internal $[0; 1]$ by subtracting the minimum value along each dimension and dividing by the maximum value along each dimension. This provides numeric stability and improves the convergence of the RPCA procedure.

$$\begin{aligned} \mathbf{M_{shift}} &= \mathbf{M} - min(\mathbf{M}) \\ \mathbf{M_{norm}} &= \frac{\mathbf{M_{shift}}}{max(\mathbf{M_{shift}})} \end{aligned} \tag{4.1}$$

The scaled data is then rotated so that it is aligned with the axes of the coordiate system. This is done by finding the three principal components of the entire dataset using PCA and projecting the dataset onto these principal components. Aligning the data simplifies the process of separating data into small regions, increases the speed of the algorithm and was found to improve performance. The result is shown in fig. 4.1.

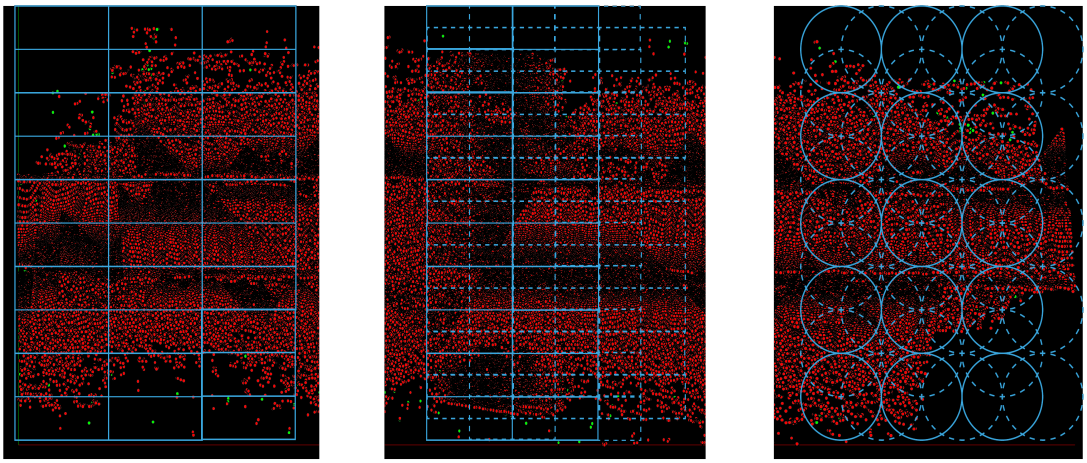


(a) (b)

Figure 4.1: (a) shows the non-aligned data, and (b) shows the aligned data.

4.1.2 Grid Creation

The dataset is separated into a set of smaller regions, within which the surface can be considered roughly planar. These regions are defined by overlaying a uniform grid on top of the data. The grid consists of cells uniformly spaced along the x and y dimensions, with each cell spanning the entire depth of the z-dimension. We choose to utilize a 2D-grid and have cells span the entire depth of the data as opposed to a 3D-grid, since the surface is only present at one level of the z-dimension for any given point in the space spanned by x and y. The grid may be constructed using different window shapes as shown in fig. 4.2. For this project we have looked at rectangular windows and ellipse-shaped windows with and without overlap. In the case of an ellipse grid, an overlap of 50% is required to ensure that all points are seen at least once. The details of how to handle overlapping windows w.r.t. outlier detection are discussed in section 4.3.



(a) (b) (c)

Figure 4.2: (a) shows a rectangular grid with no overlap, (b) shows a rectangular grid with 50% overlap and (c) shows an ellipse grid with 50% overlap.

4.2 RPCA

The RPCA algorithm estimates a low rank matrix, $\mathbf{L}$ and a sparse matrix $\mathbf{S}$ from the point cloud matrix $\mathbf{M}$. The point cloud data consists of n data points where each data point $\mathbf{v}$ is described by 3D coordinates x, y, z such that $\mathbf{v} \in \mathbb{R}^{1\times 3}$ and $\mathbf{M} \in \mathbb{R}^{n\times 3}$. The point cloud may be decomposed as described in eq. (4.2)

$$\mathbf{M} = \mathbf{L} + \mathbf{S} \tag{4.2}$$

The low rank matrix and the sparse matrix is found by estimating the PCP [2] shown as the minimization problem in eq. (4.3).

$$\begin{aligned} &\text{minimize} && \|\mathbf{L}\|_* + \lambda \|\mathbf{S}\|_1 \\ &\text{subject to} && \mathbf{L} + \mathbf{S} = \mathbf{M} \end{aligned} \tag{4.3}$$

This optimization problem can be solved using an Augmented Lagrange Multiplier (ALM) algorithm, where the operator is denoted $l()$. Following the approach by [5] the matrices $\mathbf{L}$

and $\mathbf{S}$ are found by optimizing eq. (4.4) and eq. (4.5) in an iterative approach.

$$\underset{\mathbf{S}}{\arg\min} \quad l(\mathbf{L}, \mathbf{S}, \mathbf{Y}) = \mathcal{Q}_{\lambda\mu^{-1}}(\mathbf{M} - \mathbf{L} + \mu^{-1}\mathbf{Y}) \tag{4.4}$$

$$\underset{\mathbf{L}}{\arg\min} \quad l(\mathbf{L}, \mathbf{S}, \mathbf{Y}) = \mathcal{D}_{\mu^{-1}}(\mathbf{M} - \mathbf{S} + \mu^{-1}\mathbf{Y}) \tag{4.5}$$

In the above equations two operators, $\mathcal{D}(\cdot)$ and $\mathcal{Q}(\cdot)$, have been introduced. Firstly $\mathcal{Q}(\cdot)$ is the shrinkage operator $\mathcal{Q}_\tau(\cdot) = \text{sgn}(\cdot)\max(|\cdot| - \tau, 0)$ and secondly $\mathcal{D}(\cdot)$ which denotes the singular value threshold operator $\mathcal{D}_\tau(\cdot) = S\mathcal{Q}_\tau(\sum)V^T$. In an iterative fashion the augmented Lagrangian l is firstly optimized with respect to $\mathbf{L}$ fixing $\mathbf{S}$ followed by l optimized with respect to $\mathbf{S}$ fixing $\mathbf{L}$.

Algorithm 1 Iterative estimation of the low rank / sparse decomposition

1: **begin**
2: Initialize *matrices* $\mathbf{L}_0 = \mathbf{S}_0 = \mathbf{Y}_0 = 0$
3: **do** optimize augmented Lagrangian
4: $\mathbf{L}_{i+1} = \mathcal{D}_{\mu^{-1}}(\mathbf{M} - \mathbf{S}_i + \mu^{-1}\mathbf{Y}_i)$
5: $\mathbf{S}_{i+1} = \mathcal{Q}_{\lambda\mu^{-1}}(\mathbf{M} - \mathbf{L}_i + \mu^{-1}\mathbf{Y}_i)$
6: $\mathbf{Y}_{i+k} = \mathbf{Y}_i + \mu(\mathbf{M} - \mathbf{L}_{i+1} - \mathbf{S}_{i+1})$
7: **until** error below threshold or max number of iterations
8: **return** $\mathbf{L}, \mathbf{S}$
9: **end**

In line 6 of algorithm 1 the sign of $\mathbf{S}_{i+1}$ has been corrected compared to [5] as there is a notation error. The choice of μ and the error estimation for the stopping criterion follows the proposed implementation in [5] where $\mu = n^2/4\,\|M\|_1$ and the stopping criterion is $\|\mathbf{M} - \mathbf{L} - \mathbf{S}\|_F \leq \delta\mathbf{M}_F$ with $\delta = 10^{-7}$.

4.3 Outlier Detection

The result of the RPCA procedure applied to subsets of the data is a set of sparse matrices representing distinct regions of the data. Outliers are detected and filtered within each sparse subset individually by using an *adaptive* threshold on the z-axis based on an estimation of the surface of each subset. Filtering each subset individually means that a threshold has to be determined for each subset, which is obviously more computationally expensive than processing the entire dataset at once. However, processing subsets individually was found to improve performance significantly.

4.3.1 Surface Estimation as a Gaussian Distribution

The threshold is determined by assuming that the real surface is planar, and that *inliers* follow a Gaussian normal distribution along the z-axis where the mean represents the plane. With this assumption, *outliers* can be detected in a probabilistic manner by their deviation from the mean z-value in comparison to the standard deviation of the Gaussian distribution. Looking at fig. 3.2, this assumption seems fair.

Estimating the Gaussian distribution of inliers within each sparse subset S_i of the data is therefore the first step of detecting outliers. The estimated distribution should only

represent the inliers, and thus has to disregard the outliers. The challenge is that the subset contains both inliers and outliers, and as the goal is to use an unsupervised approach, the labels of samples are not known. The simplest method is to assume that the inliers vastly outnumber the outliers so that the estimation error can be disregarded when estimating the Gaussian distribution for the entire subset. This method works well for large portions of the dataset, but encounters issues in the presence of the dense cloud of noise within the *trench*. An improvement to this method is an iterative approach, where outliers are gradually removed from the estimation by iteratively calculating the mean and standard deviation, removing all samples outside a specified confidence interval, which for the example below is chosen as 95% confidence (two standard deviations: $\mu \pm 2\sigma$), and repeating until no outliers are present in the estimation. The iterative procedure is described by algorithm 2 and is illustrated in fig. 4.3.

Algorithm 2 Iterative estimation of Gaussian distribution for sparse subset S_i with 95% confidence

begin
    initialize *inliers* $I_0 = S_i$
    compute $\mu_0 = \mu(I_0)$, $\sigma_0 = \sigma(I_0)$
    do remove samples outside 95% confidence interval
        $\mu_j - 2\sigma_j <= I_j <= \mu_j + 2\sigma_j$
        recompute $\mu_j = \mu(I_j)$, $\sigma_j = \sigma(I_j)$
    until no change in I_j
    return μ_j, σ_j
end

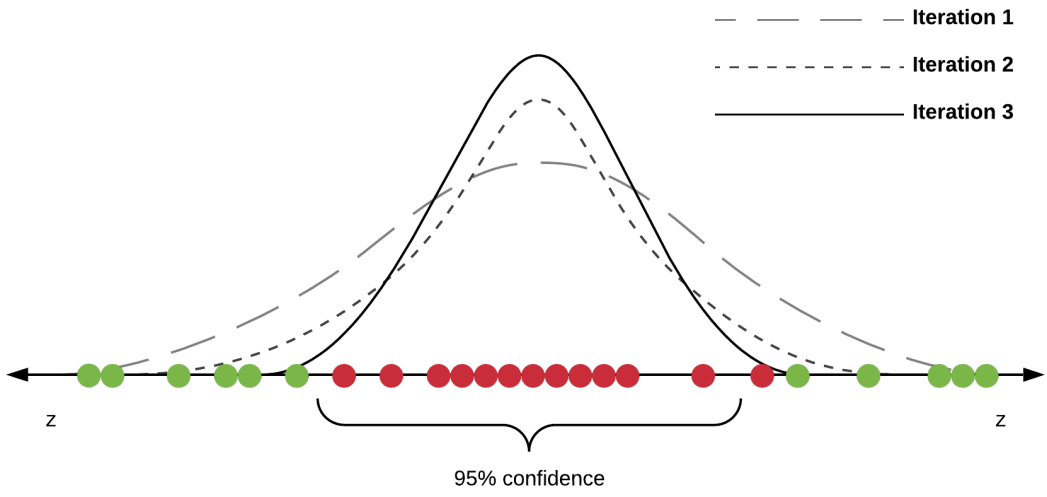


Figure 4.3: Illustration of the iterative Gaussian estimation with 95% confidence

The resulting Gaussian distribution is then used to detect outliers by labeling all samples outside the confidence interval as noise. The confidence interval is considered a parameter of the algorithm that allows for more lenient or aggressive filtering to fit a given dataset.

4.3.2 Handling Overlapping Subsets

In section 4.1 it was mentioned that the regions may overlap - or have to, as is the case for the ellipse-shaped regions. The reasoning is that overlap may improve performance by introducing redundancy in the detection of outliers. This is achieved by introducing a *voting* system, where each region subset votes on a label for each data point within the subset. After iterating all subsets, the votes are aggregated and a decision for each data point is made based on the aggregated votes. For this project we have given equal weight to each of the possible labels, so that one vote for *inlier* carries the same weight as one vote for *outlier*. Points are assigned whichever label has more votes. In the scenario of equal votes, points are assigned *outlier* labels, since they must be ambiguous. We speculate that performance may be improved by assigning different weights to labels based on the prominence of noise in the dataset, and the desired aggressiveness of the filtering. In the extreme case, even one vote for *outlier* may be considered evidence of the sample being ambiguous, and therefore should be labeled as an *outlier* - no matter what the remaining votes are.

4.4 Algorithm Parameters

The overall algorithm is summarized in algorithm 3. The parameters of the overall algorithm control the coarseness of the detection, and the aggresiveness of the filtering:

- **Region size**: height and width ($w_r \times h_r$) in the case of rectangular regions, or radius (r_r) in the case of ellipse regions
- **Region overlap**: Percentage overlap of regions
- **Outlier detection threshold**: confidence interval used for surface estimation and detection

The smaller **region size** enables the algorithm to perform better in scenarios where the measured surface is uneven, at the cost of increased computational effort. The ideal region size will depend on the structure of the measured surface, and will have to be found empirically. The performance gain from increasing the number of regions appears to be exponentially decaying, meaning it quickly reaches a point where the gain is small compared to the additional computations required. Starting with a large region size, and incrementally decreasing it by a factor of two until a satisfactory performance is achieved is therefore a good way to find the optimal region sizes.

The **region overlap** adds a degree of redundancy to the outlier detection. This was found to provide a small improvement in performance at *corners* where the surface has rapid changes in verticality.

The **outlier detection threshold** determines the tolerance of deviations from the ideal surface. A lower threshold will tolerate a smaller deviation from the ideal surface, before classifying a sample as an outlier. This means that more *outliers* will be detected, but potentially at the cost of also detecting some *inliers* as *outliers*.

Algorithm 3 Outlier Detection Algorithm

Preprocess
    Scale elements measurement matrix M to be within $[0; 1]$
    Align dimensions of M with coordinate axes using PCA
    Initialize subset regions by height and width $w_r \times h_r$ or radius r_r and overlap
Apply RPCA to region subsets
    for i in M
        Compute low-rank and sparse decomposition $M_i = L_i + S_i$ using RPCA
Detect outliers
    for j in S
        Estimate Gaussian distribution of surface within S_j
        Vote for samples outside confidence interval as *outliers*, and remainder as *inliers*
    for k in M
        label $l_k = 0$ if votes for *inliers* are fewer than votes for *outliers*, otherwise $l_k = 1$
Filter outliers
    Remove all samples with the label 1 from the dataset M

Implementation 5

In previous sections we have described the theoretical algorithms and the analyzed arguments for using RPCA for outlier detection. Implementing the algorithms and setting up a testing environment is critical for encapsulating proper results. Throughout this chapter we will describe specific implementation decisions. This will allow for understanding the produced code and for further development on the project.

5.1 Coding Environment

The environment which is used for implementing the algorithms and processing the data is python. For implementation, multiple tools have been utilized to process the data. Many libraries come ready to use and allow for plotting of data, matrix arithmetic, and using optimized computer vision and machine learning algorithms. Python is therefore used as it conveniently allows for rapid prototyping.

Data plots have been created by use of the library VisPy. As described by the designers of the library; VisPy is a python library for interactive scientific visualization that is designed to be fast, scalable, and easy to use. This visualization tool is Graphics Processing Unit (GPU) accelerated, which greatly speeds up data interaction when plotting the roughly 1.8 million samples.

Mathematics and matrix arithmetic has been enabled through the library tools from SciPy. NumPy, which is a SciPy tool, is a fast and effective library for matrix manipulation.

The scikit-learn package[4] has been used as it facilitates implementations of machine learning algorithms. The SVM, KNN and Naive Bayes algorithms was used for noise classifications through supervised learning.

RPCA is implemented using an external repository[5]. This repository implements the functionality required to compute RPCA, and is well-suited for a proof-of-concept. It should be noted that the repository does not contain a license, so the commercial rights are unspecified.

5.2 Scikit-learn Architecture

The general classifier structure from scikit-learn has been adopted to the implementation of the RPCA algorithm. In general every scikit-learn classifier implements a set of inherited super-class methods. That is, each classifier class has a CONSTRUCTOR method for setting up the behavior of the algorithm. A FIT method is used to train the classifier taking data to fit on as arguments. A PREDICT method is used for classifying data based on the fitted classifier. As the RPCA algorithm is unsupervised the is no need in splitting the data set in a training and testing set and therefor only the FIT will take arguments.

Experiments and Results 6

The RPCA algorithm was implemented and tested as described in the previous chapters. The results are listed in the same manner as previously in Section 3.1.2, meaning tables 6.1 to 6.6 contains confusion matrices. These allow for an analysis of the false positives and true negatives, which are the primary indicators of performance. In the tables, the labels *S*, *N*, *CS* and *CN* are used. *S* and *N* are abbreviations of (annotated) true samples and noise respectively, whereas CS and CN are abbreviations of ***classified*** true samples and ***classified*** noise respectively. This accuracy metric is used to better understand the classification as the usual accuracy metric would only show the percentage of correctly classified samples.

The testing was performed as a grid search of the algorithm parameters described in Section 4.4 and the type of window used. Firstly the outlier detection threshold / confidence interval has been tested for three values $\sigma = \{1.5, 2.0, 2.5\}$. This parameter and thus these values are expected to show that the algorithm will tend to classify more true samples correctly for high values (i.e. $\sigma = 2.5$) at a cost of fewer noise samples detected and conversely tend to detect more noise samples correctly for low values (i.e. $\sigma = 1.5$) at a cost of fewer true samples detected. $\sigma = 2.0$ is expected to be in between and thus balance misclassified true samples and noise samples.

Secondly, the parameter region size will be tested in set of regions $A = \{[25 \times 25], [75 \times 75], [125 \times 125], [175 \times 175], [225 \times 225]\}$ with the third parameter being the overlap between regions. The choice and combination of these two parameters are expected to be important w.r.t the performance that can be achieved in a specific data set. The overlaps used are dependent on the type of window, where the two window types are *rectangles* and *ellipses*. The values of overlap between the rectangular regions are $ol_{rect} = \{0\%, 25\%, 50\%, 75\%\}$ and between the ellipse regions the overlaps tested are $ol_{rect} = \{50\%, 60\%, 70\%, 80\%\}$.

After noise detection some of the results with the best parameter configuration will be plotted for visual inspection. We wish to show that a rough filtering configuration where most noise is filtered at the cost of some true samples being miss-classified as noise is not detrimental, as the surface can be reconstructed with interpolation of the remaining data.

6.1 Results

The results have been split in 6 tables where each of the tables shows an experiment using one of the mentioned confidence intervals and region types. Table 6.1 and table 6.2 shows the experiments using $\sigma = 1.5$ as a confidence threshold value with a rectangular grid or ellipse grid respectively. The results of both tables shows that even though a rough

threshold value is chosen many true samples are classified correctly clearly outperforming the machine learning preliminary tests from section 3.1.2. Among the best results using $\sigma = 1.5$ are the configuration using a rectangular grid with region size 225×225 and 25% overlap or using an ellipse grid with region size 225×225 and 50% overlap. However the rectangular result shows to perform the best with a true sample accuracy of 80.5% and noise sample accuracy of 94.3%.

Table 6.1: RPCA with rectangular grid and 1.5σ confidence

RPCA		25×25		75×75		125×125		175×175		225×225	
% overlap		S	N	S	N	S	N	S	N	S	N
0%	CS	0.332	0.668	0.508	0.492	0.658	0.342	0.735	0.265	0.775	0.225
	CN	0.079	0.921	0.020	0.980	0.034	0.966	0.049	0.951	0.084	0.916
25%	CS	0.631	0.369	0.534	0.466	0.672	0.328	0.758	0.242	0.805	0.195
	CN	0.201	0.799	0.022	0.978	0.035	0.965	0.046	0.954	0.057	0.943
50%	CS	0.700	0.300	0.422	0.578	0.557	0.443	0.666	0.334	0.740	0.260
	CN	0.241	0.759	0.034	0.966	0.013	0.987	0.025	0.975	0.040	0.960
75%	CS	0.684	0.316	0.299	0.701	0.436	0.564	0.536	0.464	0.622	0.378
	CN	0.239	0.761	0.045	0.955	0.010	0.990	0.011	0.989	0.019	0.981

Table 6.2: RPCA with ellipse grid and 1.5σ confidence

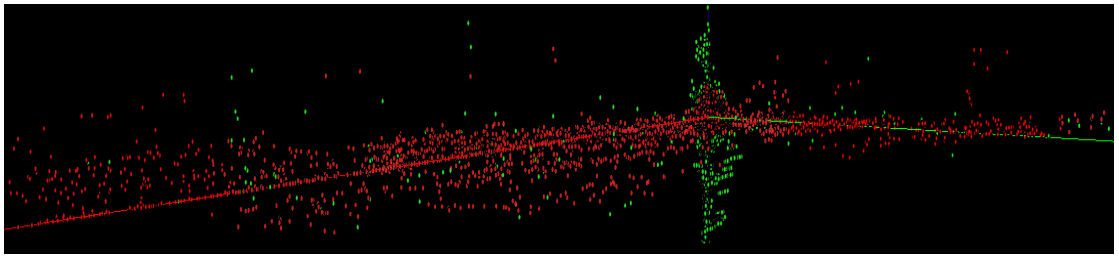
RPCA		25×25		75×75		125×125		175×175		225×225	
% overlap		S	N	S	N	S	N	S	N	S	N
50%	CS	0.604	0.396	0.415	0.585	0.558	0.442	0.665	0.335	0.735	0.265
	CN	0.196	0.804	0.019	0.981	0.019	0.981	0.025	0.975	0.035	0.965
60%	CS	0.661	0.339	0.401	0.599	0.535	0.465	0.649	0.351	0.725	0.275
	CN	0.220	0.780	0.025	0.975	0.012	0.988	0.022	0.978	0.033	0.967
70%	CS	0.643	0.357	0.349	0.651	0.485	0.515	0.603	0.397	0.683	0.317
	CN	0.200	0.800	0.029	0.971	0.008	0.992	0.015	0.985	0.027	0.973
80%	CS	0.675	0.325	0.318	0.682	0.463	0.537	0.574	0.426	0.654	0.346
	CN	0.224	0.776	0.033	0.967	0.006	0.994	0.012	0.988	0.019	0.981

The tendencies of the two tables shows that a change in region size will weight the distribution of correctly classified true samples and noise samples. It would seem that the general tendency is that an increase in grid size will increase the true sample accuracy at a cost of a decrease in noise sample accuracy. However when the grid size is small (25×25) the tendency seems irregular and the performance may be better or worse depending on the chosen window type and overlap. The chosen overlap seems to have a similar effect as the chosen region size, where an increase in the amount of overlap results in an increase in noise sample accuracy at a cost of a decrease in true sample accuracy. Again when the amount of overlap is small (rectangular grid with 0% overlap) the tendency is irregular.

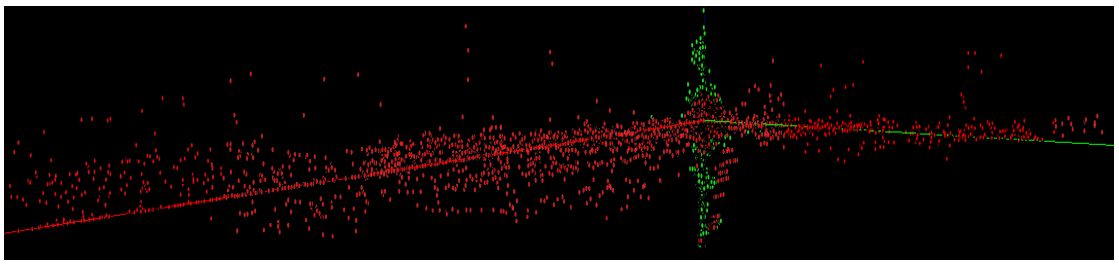
These observations indicate that an optimum performance is found by choosing a good combination of the parameters. Visual inspection combined with a smaller grid search can be used as a tool to optimize the parameters when an annotated data set is not available.

The described best result from table 6.1 with region size 225×225 and 25% overlap have been plotted for visual inspection. Figure 6.1 shows the S matrix with true sample labels

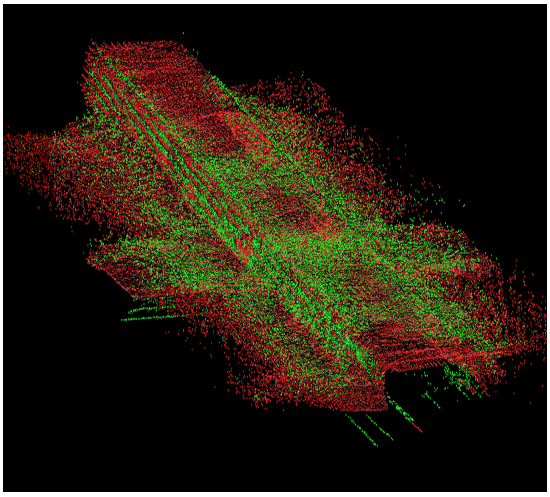
and classified sample labels, the data set with sample classifications and the filtered data set. These plots indicate that the filtering is quite rough. This means that almost all of the noise are removed but also many of the true samples are removed as-well. The fig. 6.1(c) and fig. 6.1(d) indicates that many of the removed true samples are removed from areas with almost vertical edges.



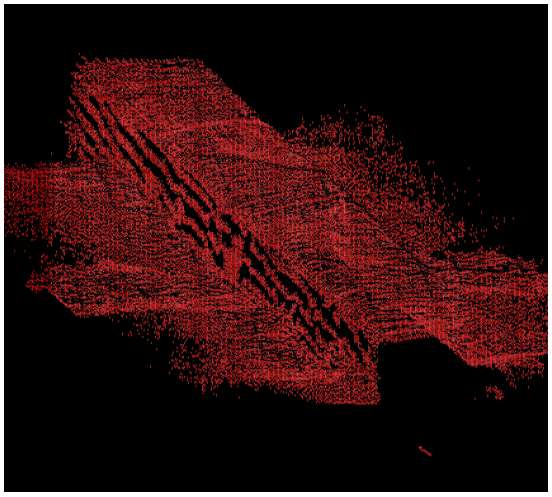
(a)



(b)



(c)



(d)

Figure 6.1: (a) shows the RPCA sparse matrix S combined from the S subsets with the true labels, (b) shows the RPCA sparse matrix S combined from the S subsets with the classified labels, (c) shows the classified samples (red) and classified noise (green) (d) shows the filtered data set. ((a), (b), (c) and (d) all have the configurations: rectangular grid; region size 225×225; 25% overlap)

Table 6.3 and table 6.4 shows the experiments using $\sigma = 2$ as a confidence threshold value with a rectangular grid or ellipse grid respectively. Relating the results tables with the previously discussed tables with $\sigma = 1.5$ as a confidence threshold it can be observed that increasing the confidence interval increases the amount of correctly classified true samples, however at a cost of decreased noise classification. Among the best results using $\sigma = 2$ are the configuration using a rectangular grid with region size 125×125 and 25% overlap or using an ellipse grid with region size 225×225 and 80% overlap. However the ellipse result shows to perform the best with a true sample accuracy of 87.9% and noise sample accuracy of 77.8%.

Table 6.3: RPCA with rectangular grid and 2σ confidence

RPCA		25×25		75×75		125×125		175×175		225×225	
% overlap		S	N	S	N	S	N	S	N	S	N
0%	CS	0.647	0.353	0.721	0.279	0.787	0.213	0.826	0.174	0.863	0.137
	CN	0.181	0.819	0.188	0.812	0.227	0.773	0.279	0.721	0.300	0.700
25%	CS	0.807	0.193	0.879	0.121	0.885	0.115	0.898	0.102	0.908	0.092
	CN	0.249	0.751	0.261	0.739	0.244	0.756	0.289	0.711	0.329	0.671
50%	CS	0.816	0.184	0.893	0.107	0.885	0.115	0.895	0.105	0.901	0.099
	CN	0.299	0.701	0.285	0.715	0.265	0.735	0.261	0.739	0.278	0.722
75%	CS	0.792	0.208	0.810	0.190	0.858	0.142	0.859	0.141	0.876	0.124
	CN	0.268	0.732	0.209	0.791	0.233	0.767	0.235	0.765	0.234	0.766

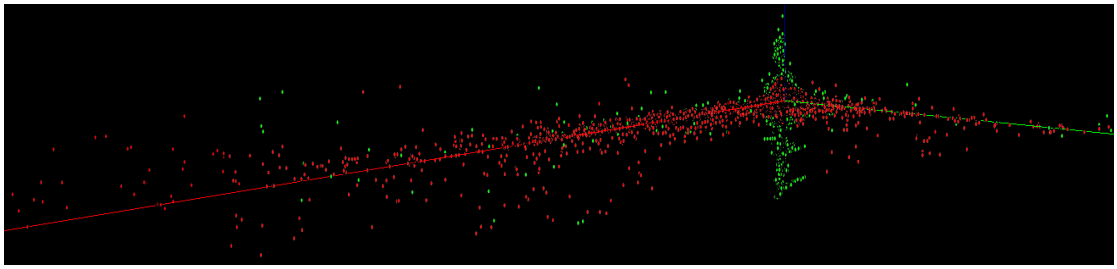
Table 6.4: RPCA with ellipse grid and 2σ confidence

RPCA		25×25		75×75		125×125		175×175		225×225	
% overlap		S	N	S	N	S	N	S	N	S	N
50%	CS	0.762	0.238	0.856	0.144	0.842	0.158	0.865	0.135	0.877	0.123
	CN	0.229	0.771	0.245	0.755	0.228	0.772	0.231	0.769	0.286	0.714
60%	CS	0.785	0.215	0.875	0.125	0.860	0.140	0.880	0.120	0.888	0.112
	CN	0.270	0.730	0.251	0.749	0.241	0.759	0.235	0.765	0.281	0.719
70%	CS	0.751	0.249	0.852	0.148	0.849	0.151	0.869	0.131	0.879	0.121
	CN	0.232	0.768	0.228	0.772	0.228	0.772	0.227	0.773	0.238	0.763
80%	CS	0.781	0.220	0.841	0.159	0.859	0.141	0.868	0.132	0.879	0.121
	CN	0.251	0.749	0.209	0.791	0.222	0.778	0.219	0.781	0.222	0.778

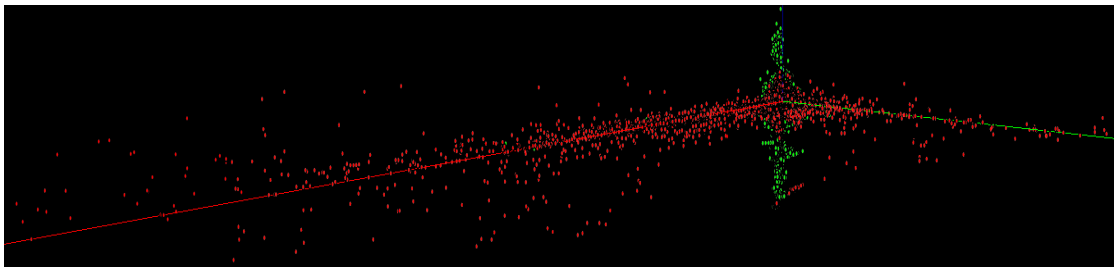
The two tables shows the same tendencies as for table 6.1 and table 6.2 that an increase in grid size will increase the true sample accuracy at a cost of a decrease in noise sample accuracy. The chosen overlap seems to maximize the true sample accuracy around 50% overlap, however due to the rough granularity of overlap the tendency could be otherwise. Again these observations indicate that an optimum performance is found by choosing a good combination of the parameters.

The described best result from table 6.4 with region size 225×225 and 80% overlap have been plotted for visual inspection. Figure 6.2 shows the S matrix with true sample labels and classified sample labels, the data set with sample classifications and the filtered data set. In contrast to fig. 6.1 the areas with almost vertical edges is not affected by the filtering as roughly and thus more true samples are correctly classified. Due to the increase in confidence threshold value some of the noise samples are not identified which

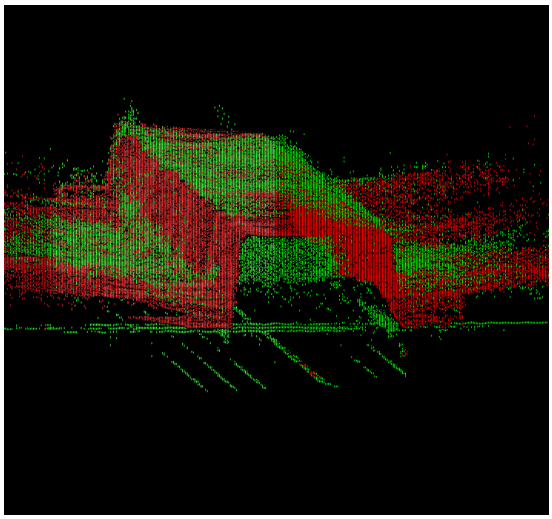
is seen from fig. 6.2(c) and fig. 6.2(d). The surface however seem clearly defined without indications of noise samples in the vicinity of the surface.



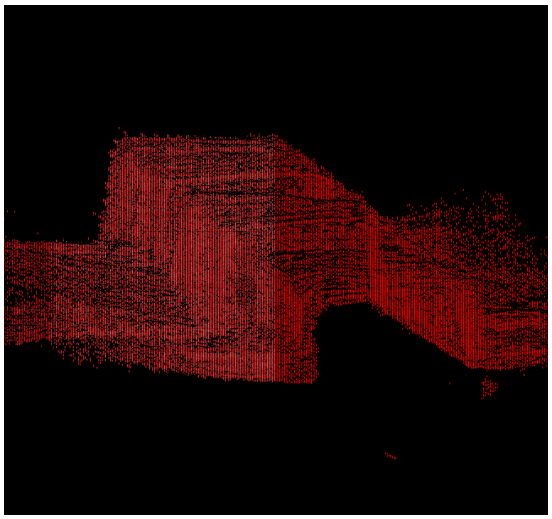
(a)



(b)



(c)



(d)

Figure 6.2: (a) shows the RPCA sparse matrix S combined from the S subsets with the true labels, (b) shows the RPCA sparse matrix S combined from the S subsets with the classified labels, (c) shows the classified samples (red) and classified noise (green) (d) shows the filtered data set. ((a), (b), (c) and (d) all have the configurations: ellipse grid; region size 225×225; 80% overlap)

Table 6.5 and table 6.6 shows the experiments using $\sigma = 2.5$ as a confidence threshold value with a rectangular grid or ellipse grid respectively. Relating the results tables with the previously discussed tables with $\sigma = 1.5$ and $\sigma = 2$ as confidence thresholds the trend again shows that an increase in the confidence interval increases the amount of correctly classified true samples at a cost of decreased noise classification. Among the best results using $\sigma = 2.5$ are the configuration using a rectangular grid with region size 125×125 and 75% overlap or using an ellipse grid with region size 75×75 and 50% overlap. However the rectangular result shows to perform the best with a true sample accuracy of 97.6% and noise sample accuracy of 68.0%.

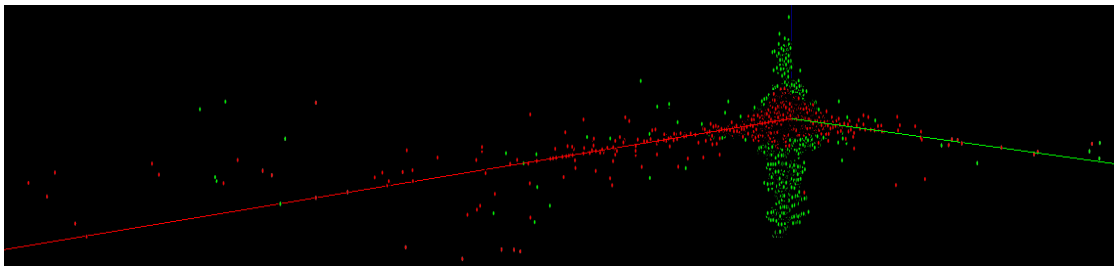
Table 6.5: RPCA with rectangular grid and 2.5σ confidence

RPCA		25×25		75×75		125×125		175×175		225×225	
% overlap		S	N	S	N	S	N	S	N	S	N
0%	CS	0.884	0.116	0.913	0.087	0.909	0.091	0.910	0.090	0.918	0.082
	CN	0.263	0.737	0.322	0.678	0.399	0.601	0.409	0.591	0.417	0.583
25%	CS	0.935	0.065	0.977	0.023	0.966	0.034	0.960	0.040	0.956	0.044
	CN	0.338	0.662	0.331	0.669	0.404	0.596	0.443	0.557	0.457	0.543
50%	CS	0.933	0.067	0.984	0.016	0.974	0.026	0.966	0.034	0.961	0.039
	CN	0.366	0.634	0.359	0.641	0.347	0.653	0.413	0.587	0.440	0.560
75%	CS	0.882	0.118	0.979	0.021	0.976	0.024	0.969	0.031	0.961	0.039
	CN	0.327	0.673	0.346	0.654	0.320	0.680	0.325	0.675	0.403	0.597

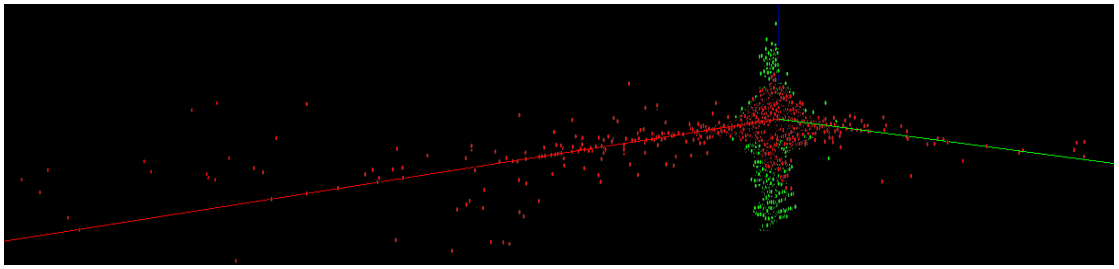
Table 6.6: RPCA with ellipse grid and 2.5σ confidence

RPCA		25×25		75×75		125×125		175×175		225×225	
% overlap		S	N	S	N	S	N	S	N	S	N
50%	CS	0.910	0.090	0.977	0.023	0.965	0.035	0.955	0.045	0.952	0.048
	CN	0.334	0.666	0.317	0.683	0.337	0.663	0.422	0.578	0.428	0.572
60%	CS	0.927	0.073	0.981	0.019	0.971	0.029	0.962	0.038	0.956	0.044
	CN	0.350	0.650	0.335	0.665	0.324	0.676	0.415	0.585	0.426	0.574
70%	CS	0.893	0.108	0.979	0.021	0.971	0.029	0.963	0.037	0.956	0.044
	CN	0.319	0.681	0.332	0.668	0.323	0.677	0.365	0.635	0.419	0.581
80%	CS	0.899	0.101	0.982	0.018	0.974	0.320	0.965	0.035	0.958	0.042
	CN	0.334	0.666	0.335	0.665	0.320	0.680	0.355	0.645	0.412	0.588

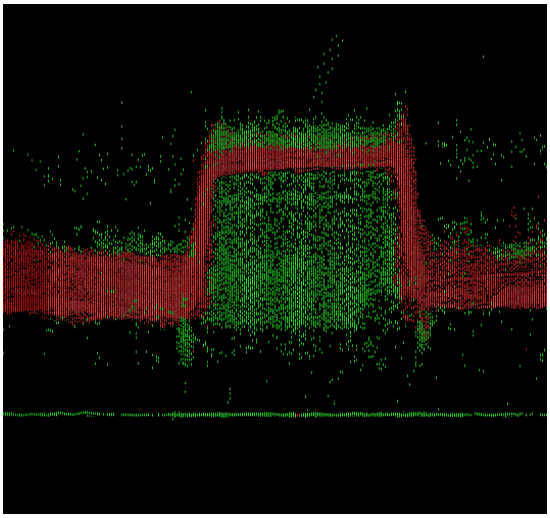
The described best result from table 6.5 with region size 125×125 and 75% overlap have been plotted for visual inspection. Figure 6.2 shows the S matrix with true sample labels and classified sample labels, the data set with sample classifications and the filtered data set. From fig. 6.2(c) and fig. 6.2(d) it can be seen that the algorithm now struggles to identify the noise samples along corners and close to the surface inside enclosed areas near vertical edges. Although fewer noise samples have been filtered the surface still appears clearly defined.



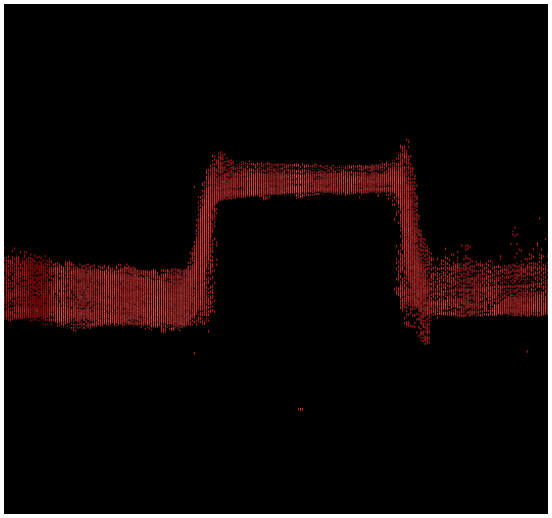
(a)



(b)



(c)



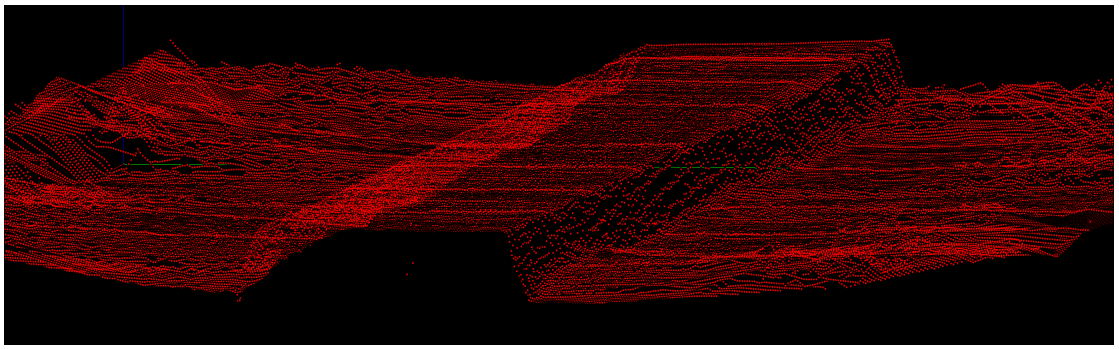
(d)

Figure 6.3: (a) shows the RPCA sparse matrix S combined from the S subsets with the true labels, (b) shows the RPCA sparse matrix S combined from the S subsets with the classified labels, (c) shows the classified samples (red) and classified noise (green) (d) shows the filtered data set. ((a), (b), (c) and (d) all have the configurations: rectangular grid; region size 125×125; 75% overlap)

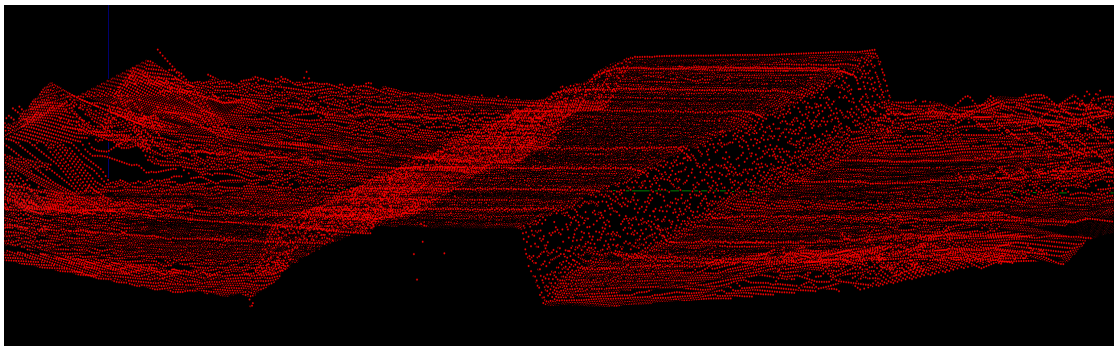
6.2 Surface Reconstruction

The previous results shows that the algorithm indeed can classify both true samples and noise samples. The configuration of the RPCA algorithm will determine whether more true samples or noise samples are detected. Visual inspection of a surface reconstruction might aid in deciding how to choose the correct parameters. By *linear interpolation* of the filtered data points, the surface can be reconstructed. A well defined amount of samples are used to reconstruct the surface where the samples are evenly spaced with a grid resolution of 500×200 samples. Figure 6.4 shows the reconstructed surfaces of fig. 6.1(d), fig. 6.2(d) and fig. 6.3(d). From the reconstructions it can be seen that the even though fig. 6.1(d) has far less correctly classified samples - with detection problems along vertical surfaces - the reconstruction is still quite good.

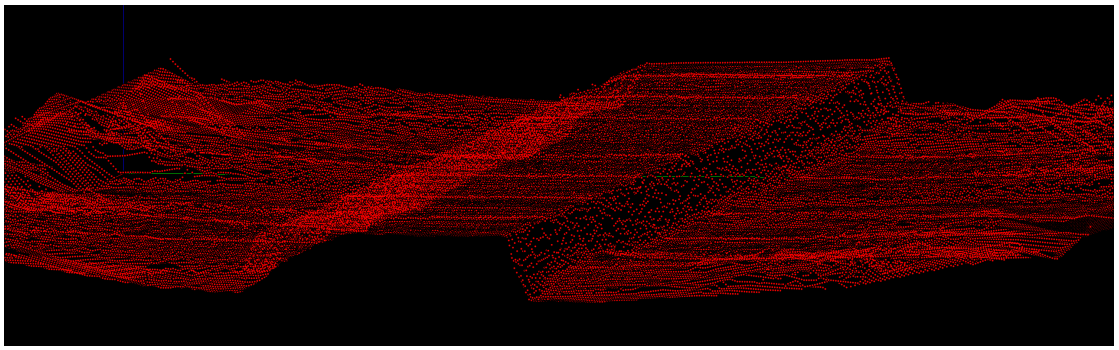
In fact as it visually is hard to differentiate between the three configuration of fig. 6.4 we can conclude that it is beneficial to weight the removal of noise higher than the correct classification of true samples. This is due to the fact that the surfaces can still be reconstructed in clusters of samples where true samples are miss-classified (example: the vertical surfaces in fig. 6.1(d)). It should be noted that the reconstruction of the surface is not accurate at the boundaries of the data set. As previously discussed, this is not unexpected, since the surface is not well-defined along the boundaries.



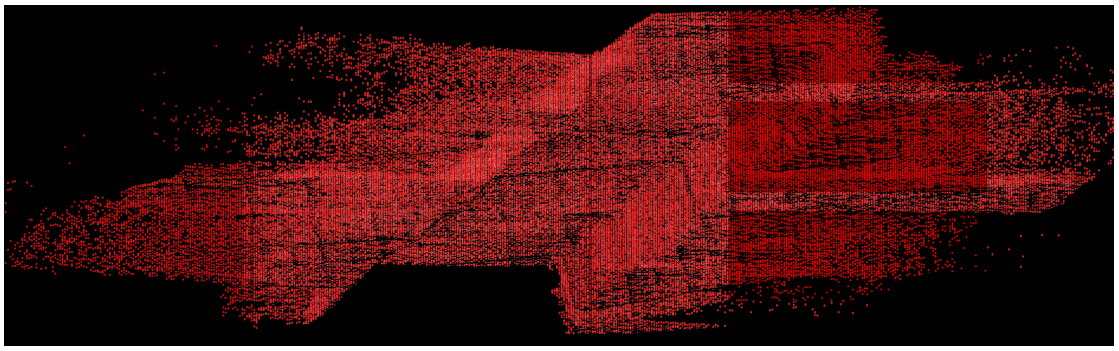
(a)



(b)



(c)



(d)

Figure 6.4: (a) shows the reconstructed surface of fig. 6.1(d), (b) shows the reconstructed surface of fig. 6.2(d), (c) shows the reconstructed surface of fig. 6.3(d), (d) shows the true data set samples

6.3 Generalization

The results (in section 6.1) are all based on the second data set in fig. 2.1(b). As the proposed RPCA algorithm is supposed to be a generic filtering approach of depth 3D imagery this section will provide test results on the first data set in fig. 2.1(a) with the best RPCA configurations from section 6.1. Each of the three confidence interval best performance configurations are tested and the results listed in tables 6.7 to 6.9. Similar results as the second data set is seen with the first data set. An increase in confidence interval threshold cause in increase in true sample classification accuracy and a decrease in noise sample classification. The classification accuracy seems to be somewhat balanced with a confidence interval of $\sigma = 2$. These observations indicates that the algorithm generalizes well between data sets as the configurations achieved from searching good parameters (in section 6.1) shows similar performance characteristics on a new data set.

As previously mentioned, the first data set has significantly fewer noise samples than the second data set. These noise samples are located around a single depression in the otherwise quite flat surface. The algorithm performs even better when the amount of noise samples are low compared to the amount of true samples. In the case of the configuration with $\sigma = 2$ the performance on the second data set was a true sample accuracy of 80.5% and noise sample accuracy of 94.3%. In contrast the performance on the first data set the performance is a true sample accuracy of 95.7% and noise sample accuracy of 92.2%. This indicates that the accuracy drops slightly with the classification difficulty of the data set. Thus, if this tendency is general; the algorithm will provide even better results with improvements in the sampling equipment.

Figure 6.5 is a visualization of the filtering capabilities of the three RPCA configurations. The filtered data is plotted with true labels where the true samples are marked green and noise samples are marked red. Along with the filtered data the complete data set is shown to allow for a visual inspection of classification characteristics. That is, what kind of noise samples could not be classified and therefor not be filtered by the algorithm?

Table 6.7: RPCA with configurations: rectangular grid; region size 225×225; 50% overlap; $\sigma = 1.5$

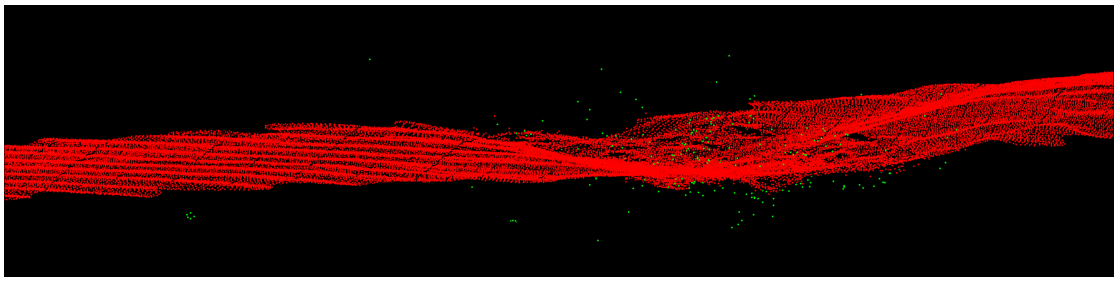
	S	N
CS	0.885	0.115
CN	0.047	0.953

Table 6.8: RPCA with configurations: ellipse grid; region size 225×225; 80% overlap; $\sigma = 2$

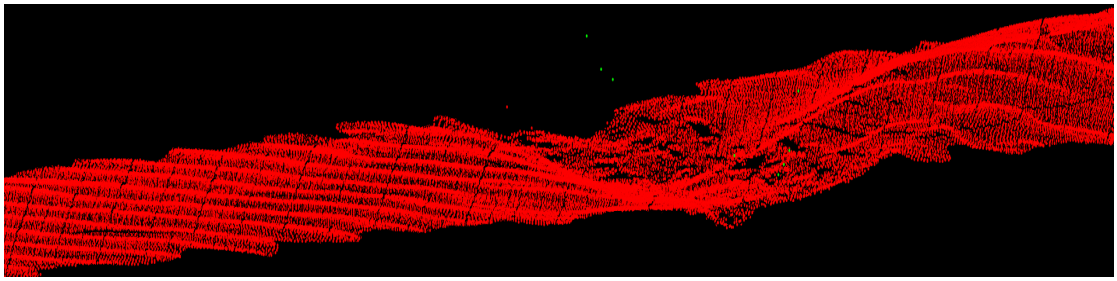
	S	N
CS	0.957	0.043
CN	0.078	0.922

Table 6.9: RPCA with configurations: rectangular grid; region size 125×125; 75% overlap; $\sigma = 2.5$

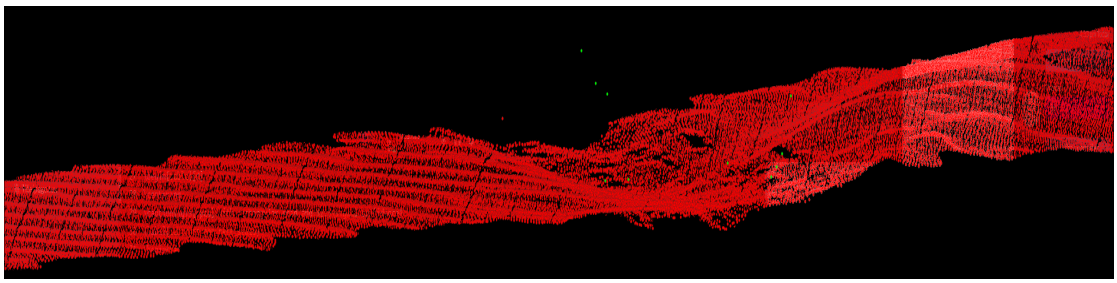
	S	N
CS	0.988	0.012
CN	0.266	0.734



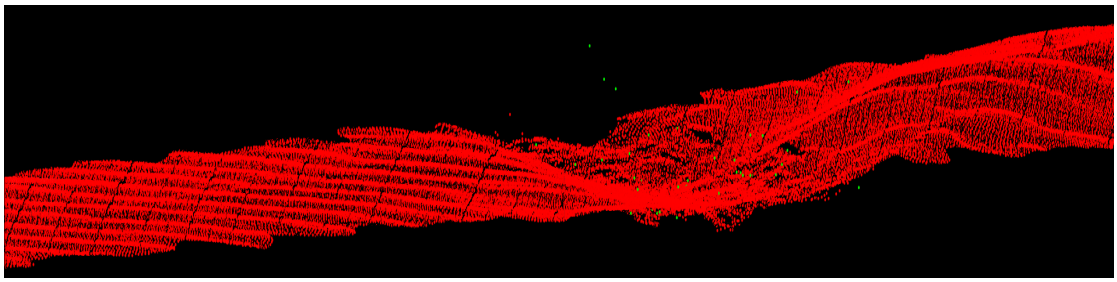
(a)



(b)



(c)



(d)

Figure 6.5: (a) shows the first data set as in fig. 2.1(a), (b) shows the filtered data with true sample labels (configuration: rectangular grid; region size 225×225; 50% overlap; $\sigma = 1.5$) , (b) shows the filtered data with true sample labels (configuration: ellipse grid; region size 225×225; 80% overlap; $\sigma = 2$), (b) shows the filtered data with true sample labels (configuration: rectangular grid; region size 125×125; 75% overlap; $\sigma = 2.5$)

Discussion and Future Work 7

The results in chapter 6 describe the performance of the devised RPCA algorithm and shows that the algorithm can filter noise in an unsupervised manner and with a high degree of accuracy. The results also show that an increase in accuracy of outlier detection also corresponds to an increase in *annotated* inliers being detected outliers. Chapter 2 describes how some of the samples annotated as inliers are considered ambiguous and may be outliers depending on the desired granularity of the filtering process. However, the visual results show that a good portion of the falsely positive inliers are actual inliers, and that filtering leaves small gaps in the surface of the filtered data. For the dataset analyzed in this project, most of the issues occur at the vertical edges of the *trench*, which would indicate that the algorithm has the most issues in areas that contain a large degree of variability from the surrounding area. The results also show that if the parameters of the algorithm are chosen appropriately, a high accuracy of outlier detection can be achieved, while also keeping the number of falsely detected inliers low enough to accurately interpolate the resulting gaps and recreate the surface. Tuning the parameters of the algorithm must therefore be considered an exercise in balancing the false and true positives so that surface reconstruction is possible.

The use of RPCA is based on the assumption that the density of the surface is always high enough to ensure that the plane is always well-defined. Thus, if the density is too low or the window becomes too small, the assumed plane may not be well-defined enough to properly detect outliers along the z-axis. Additionally, if the density of the surface is not sufficiently higher than the density of the noise, the same problem may arise, and the algorithm may not perform as well as expected.

The algorithm described in this report may be considered a *proof-of-concept*, and therefore allows plenty of room for improvement. Firstly, RPCA appears to be most effective at removing the Gaussian noise close to the surface, while not being as effective when extending it to non-Gaussian outliers. Performance may therefore be slightly improved if the obvious and significant outliers were removed prior to applying RPCA. Secondly, we detect outliers by thresholding S matrices only along the z-axis value. Although this methodology is very effective, it is a simplistic approach which may be improved. We have looked into classifying the sparse points based on all three components - that is the x, y and z dimensions - to determine whether this might provide a performance increase. Specifically, we looked at filtering samples based on their sparsity w.r.t. to all three dimensions, so that inliers should be sparse along the z-axis and either x, y or both (hence a minimum of two sparse dimensions). This was based on an analysis of the RPCA illustration in fig. 3.2. Initial tests showed promising results, but the implementation was not stable enough to include in this project or make any definite conclusions.

An entirely different approach, that may provide improved performance, would be to revert to treating the problem as a *classification* problem, but instead of computing features in the original (x, y, z) feature space, they are computed in the sparse feature space of S. Figure 3.2 shows why this may be advantageous, since the outlier samples are located in a manner that, at first glance, appear to simplify the discrimination compared to the original feature space shown in fig. 2.1. Another advantage of the sparse domain is that it provides an abstraction from the spatial dimensions of the actual surface, so that more general features may be defined. We consider this an advantage since our initial tests indicated that the ideal radius of the search-region and the ideal number of neighbors used for computation of features are dependent on the location and density of noise in the dataset. We have only had access to two different datasets, but these differed significantly in terms of noise characteristics, so we have to assume that this is also the case for other datasets.

Summarizing, the primary advantage of the proposed methodology based on RPCA, is that it provides a good performance using an unsupervised method of outlier detection. This is significant, because we suspect that a supervised classification model based on traditional machine learning and features would require a diverse set of datasets to become generalized. These datasets must be labeled and as the process of manually labeling samples in large point clouds is very labor-intensive, a supervised model incurs a significant labor-overhead. An unsupervised algorithm, such as the one proposed, is therefore clearly preferable if the performance is satisfactory.

Conclusion 8

This report shows that noise filtering of a point cloud dataset may be treated as an outlier detection problem, and hence solved with an unsupervised approach using statistical methodologies. Specifically, RPCA may be used to decompose a matrix of 3-dimensional measurements into a low-rank and sparse matrix, where the sparse matrix enables outlier detection through a threshold along the z-dimension. Assuming that inliers of the measured surface may be represented by a Gaussian distribution, a threshold is found by estimating the distribution and selecting a confidence interval - outside which all samples are considered outliers.

The algorithm has been tested with different parameter configurations to determine the optimal performance for the given dataset. The optimal performance was evaluated as a combination of classification accuracy w.r.t. the annotated labels and visual inspection since many of the annotated inliers are ambiguous and may be considered outliers. The results showed that the parameters of the algorithm may be configured to achieve a noise filtering that achieves a high degree of outlier detection accuracy, though at the cost of a potential decrease in inlier detection accuracy. The algorithm was able to achieve accuracies of 80.5%/94.3%, 87.9%/77.8% and 97.6%/68.0% (inlier acc. / outlier acc.) with different configurations, which highlight the flexibility of the algorithm. Using these configurations, the algorithm was also able to generalize well and have similar performance on a different dataset. The results also show that the falsely detected inliers create small gaps in the filtered surface, but that the surface may still be reconstructed, to a fair degree of accuracy, using linear interpolation. The results are all significant improvements to the initial tests of feature-based classification using traditional machine learning models. This is not to say that our conclusion is to disregard the classification approach completely, but simply that we consider an unsupervised approach preferable - particularly in the case where performance is similar or better. For this reason, we would recommend further investigation of unsupervised outlier detection methodologies.

Project code A

The project code has been made available through GitHub at:

`https://github.com/Woobs8/AU_RnD_Project`.

In this repository all algorithms, prepossessing functions and utilities have been provided, however without the project data sets as these are confidential and provided by EIVA [3].

New data sets should be saved as a `.txt` file and be formatted as shown in listing A.1. The first line is a column wise naming of the data. This line should be commented with a `#`. The first three values should be x, y, z as this is expected as default by most of the created tools. Data set features can be placed in between the x, y, z values and the noise label. Noise is indicated by a `true` value or a `1` and true samples indicated by a `false` or a `0`. Values are separated by a space `" "`.

```
#   x,   y,    z,  Feature1, Feature2, , FeatureEnd, NoiseLabel
    1.56 0.59 0.9 703.91867 0.000       62.0600000 1
    2.22 0.59 4.0 232.19269 0.000       0.00000000 1
    15.6 0.58 6.1 88.881544 3.199       0.00000000 0
```

Listing A.1: Data Format to the provided code

Bibliography

[1] Gerasimos Arvanitis et al. “Real-Time Removing of Outliers and Noise in 3D Point Clouds Applied in Robotic Applications”. In: *Second International Conference, Interactive Collaborative Robotics (ICR) 2017*. Aug. 2017, pp. 11–19. ISBN: 978-3-319-66470-5.

[2] Emmanuel J. Candès et al. “Robust Principal Component Analysis?” In: *J. ACM* 58.3 (June 2011), 11:1–11:37. ISSN: 0004-5411. DOI: 10.1145/1970392.1970395. URL: http://doi.acm.org/10.1145/1970392.1970395.

[3] *EIVA*. https://www.eiva.com.

[4] F. Pedregosa et al. “Scikit-learn: Machine Learning in Python”. In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[5] *robust-pca*. https://github.com/dganguli/robust-pca.