POLISH MARITIME RESEARCH 2 (90) 2016 Vol. 23; pp. 91-98
10.1515/pomr-2016-0025

DEPTH IMAGES FILTERING IN DISTRIBUTED STREAMING

Tomasz Dziubich, Assoc. Prof.
Julian Szymański, Assoc. Prof.
Adam Brzeski, M. Sc.
Jan Cychnerski, M. Sc.
Waldemar Korłub, M. Sc.
Gdańsk University of Technology, Poland

ABSTRACT

In this paper, we propose a distributed system for point cloud processing and transferring them via computer network regarding to effectiveness-related requirements. We discuss the comparison of point cloud filters focusing on their usage for streaming optimization. For the filtering step of the stream pipeline processing we evaluate four filters: Voxel Grid, Radial Outliner Remover, Statistical Outlier Removal and Pass Through. For each of the filters we perform a series of tests for evaluating the impact on the point cloud size and transmitting frequency (analysed for various fps ratio). We present results of the optimization process used for point cloud consolidation in a distributed environment. We describe the processing of the point clouds before and after the transmission. Pre- and post-processing allow the user to send the cloud via network without any delays. The proposed pre-processing compression of the cloud and the post-processing reconstruction of it are focused on assuring that the end-user application obtains the cloud with a given precision.

Keywords: point cloud processing, distributed system, parallel processing, depth image filtering

INTRODUCTION

Underwater (UW) imagery presents several challenging problems for automated remote target recognition. One of them is the development of real-time data processing methods. We can divide these methods into two groups. The first of them includes methods running in a local execution environment (located on an autonomous device/object/vehicle) and the second group include techniques that operate in a remote processing centre on locally acquired data (so called client-server model). In both cases, the researchers focus on providing real-time services for underwater facilities products and delivering an efficient and high-performance parallel computation platform.

Nowadays LIDAR sensors in autonomous inspection of UW activities are becoming increasingly popular. This technique often supports acoustic signals analysis which is still a dominant technique in the UW solutions. The combination of both methods is called a multi-modal approach. It assures better measurement precision, reduction in risks, economic benefits and superior data products compared to conventional means.

An example of such a solution is an underwater camera system with a laser line source to measure seafloor features at a millimetre scale [1]. The quality of underwater photography is limited by the visibility of the water column. In real underwater environments there are always floating particles that scatter the light. As a result, photographic images taken under such conditions tend to be blurred. Consequently, series of image filters and transformation need to be applied in order to achieve better quality. Measurements show, that the error rate of this solution is less than 1.5 mm when the target is scanned from a distance of 1 m. One of the disadvantages of the described method is low computation efficiency (ca. 5 fps). System performance decreases if we apply additional image filters.

Another example is the DP2TM 3D LiDAR built-in the Marlin Autonomous Underwater Vehicle for detection and localization of structural changes vs. reference model [8]. This device has better parameters than the solution mentioned above (performance of 3D imaging ca.10 fps, higher resolution > 0.040 Mpx, and different range > 3Km). It can also operate in degraded visual environments and requires dual scans for 100% data validation. This solution was deployed in water depths of 50 - 3,000 meters with measurements distances varying from 6 - 90 meters.

3D LiDAR can not only distinguish between shapes and objects, but detects moving objects as well. This means that it is possible to acquire data of moving seafloor hydrothermal plumes or oil spills. 10fps real-time detection enables even measurement of the space between a number of objects and their speed. Another characteristic of the 3D measurement is the ability to see data in a 3D environment, from a head-on perspective, the bird's eye view or any other.

In [11] authors present a methodology that utilizes visual cues in multi-modal optical and sonar images, namely, the occluding contours of various scene objects that can be detected and matched more robustly than point features. Unfortunately, the use of this method in AUV is possible in very limited range due to computing power.

Data obtained from LIDARs can be stored using data structures referred to as Point Clouds, which allows storing points along with additional information. It can also store colour or size, depending on the implementation of the structure. The points stored in the Point Clouds can be very often considered as not connected voxels (three-dimensional pixels) and they can be used for objects visualization.

Rusu and Cousins [10] have called point cloud a high quality representation of the world. Generally, this statement referred to clouds obtained from LIDAR scanners, but it can be used to describe any point cloud received from a depth sensor, which is able to acquire data of the observed world. Obviously, the scale of the represented view of the world may differ. One of listed in [10] cloud advantages is the fact, that space robots will "see" the world in 3D in the future. Theoretically, it is already possible with PCL. In [3, 14] authors managed to represent an entire city using a single point cloud. For example, an aerial scan with a spatial resolution of 7 cm of the city of Munich, roughly containing 200.000 individual buildings and being spread over about 300 km^2. In this case the total raw data size consists of 61*109 points (approx. 180 TB) [14]. In the case of point-cloud shape detection for city-modelling, the large dataset needs to be processed in parallel. It can be assumed, that each building consists of only 6 individual faces with a total surface area of approximately 1000 m^2, resulting in about 32000 points per primitive shape. One of the problems of the merging process for providing the point cloud data (PCD) is to assure high performance of the system, especially in parallel and distributed environments.

To process huge amount of data we need to ensure high computing power with a stable level of energy consumption. We can achieve this aim using a pipeline model in a parallel computer system. In real-time 3D data acquisition systems with motion detection and recognition feature an efficient processing pipeline has to be tailored to used hardware. Currently, using a multi-core and many-core computer architectures as NVidia CUDA and Intel Xeon Phi real-time data processing is most common. However software for these architecture has to be very sophisticated.

In this paper we propose a distributed pipeline used for point cloud processing and transferring them using computer network that fulfil requirements related to their effectiveness.

PCD DATA FLOW

Point cloud processing pipeline can be divided into the following steps shown in Fig. 1:

- Cloud acquisition (depth image obtaining, point cloud building).
- Pre-processing (filtering – cutting off, reducing the noise and the point cloud size, compression).
- Network transmission.

After that, the second node also has to process the frame. The second part of the pipeline is the same as on the first (input) machine, but the acquisition is different:

- Depth image obtaining – getting a depth image from local sensor and from remote sensor via network,
- Point cloud building – depth image to point cloud conversion (optional).

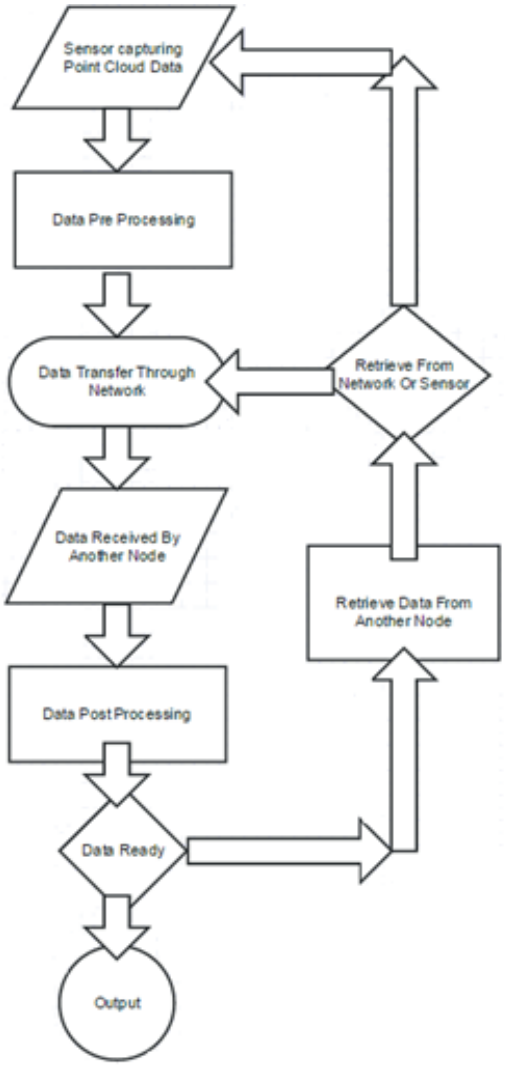

Fig. 1. Dataflow in parallel point clouds processing

Existing systems often focus on a part of the actual problem, instead of all of the sub-problems related to the processing. In this paper we divide those problems into several parts. These groups can be considered as potential stages mapped from processing pipeline: the cloud is grabbed and then pre-processed – filtered so that only interesting points can be found in the cloud. Then, such cloud should be optionally compressed, if the first stage is insufficient. Next the cloud is sent via network.

The last stage – the reconstruction is the opposite stage to the filtering and compression. The networking stage should be transparent for the end user, the cloud should be visually (almost) identical with the data received from the sensor (it does not apply to e.g. background subtraction filters).

Hence, if during pre-processing any significant or necessary points were removed, reconstruction is required. Otherwise, this step may be skipped.

Generally, filtering is needed almost in all applications that employ depth cameras. In our experiments we used the Kinect sensor as a depth camera. The error of information provided by Kinect increases with an object distance from the sensor. What is more, the factor of this growth is square [13].

For some projects the accuracy of depth image data may be crucial. For the expected solution described in this paper the accuracy is not so important, since it can be assumed, that the end-user (e.g. application that gets point clouds provided by the system) may use their own filters in order to increase accuracy. A comparison of depth sensors were provided e.g. in [4].

Thumbunpeng et al. [12] were trying to use depth camera to measure the proportion between burn area of human body and the body surface. They wanted to provide an alternative method for eye estimation, since the quality of the estimation highly influences the treatment efficiency (patients need to get the proper doses of water depending on calculated area). They decided to use spatial filters in order to reduce camera error and distortions, and proved, that the proposed solution increases the quality of cloud surface significantly.

In literature, some authors also design dedicated filtering and compression algorithms. E.g. in [2], authors proposed Enhanced Vector Quantization (EVQ) algorithm, an enhanced version of standard EQ algorithm. EVQ reduces disadvantages of similar approaches and is easy to use, requiring only to set the compression level. The goal of this work was to create an appropriate model for building a mesh.

In [9] authors focus on point cloud filtering as a pre-processing stage in robotics learning. They decided to use Growing Neural Gas network, and to prove that this method can be more effective than e.g. Voxel Grid filter.

Wenming et al. proposed their own algorithm for point cloud processing [15], dedicated for cloud de-noising. As in many algorithms, they proposed to divide the cloud into a grid and then de-noise it using neighbourhood distance calculations. As they report this algorithm is very simple and easy to implement. In comparison to bilateral mesh de-noising algorithm, the proposed one seems to be much more promising.

In [5] authors propose an algorithm for filtering LiDAR point clouds. Their main goal was to remove any objects like buildings, leaving territory malformations intact. Interestingly, the algorithm works also on mountainous territories – all buildings are removed while the territory remains intact. That would allow to create e.g. physical maps. Despite this algorithm was invented for LiDAR point clouds to filter territory surfaces, that would be used also in other fields – e.g. in de-noising of general surfaces of observed objects, but this idea would require further research.

In [7] a different approach to surface de-noising has been shown. Since the goal of a large number of algorithms is to make surfaces smooth, the authors attempt to preserve features of the cloud. This may be important for creating e.g. point clouds of some historical objects with carved details, like inscriptions. In order to achieve this, they create smooth surfaces using standard algorithms, additionally preserving extra data like high vectors – distance between old and smoothed position. Then, vectors of neighbouring points are compared and new positions are calculated. Despite some limitations of the algorithm, the authors achieved the expected result. Nevertheless, this approach may not be relevant in the presented work, since probably the smallest details will be lost in order to perform optimization.

FILTERS IN POINT CLOUD PROCESSING

There exists a wide range of filters. The most popular are listed below:

- Pass Through – enables cutting off point cloud parts,
- Voxel Grid – allows to replace a set points with a mean point,
- Points projecting – allows to project points onto e.g. plane,
- Indices extracting – uses segmentation algorithm to extract inliers,
- Conditional removal – removes points which does not meet given conditions,
- Statistical Outlier Removal – deletes outliers,
- Radius Outlier Removal – deletes outliers,
- Spatial filter – performs a cloud smoothing,
- Growing Neural Gas network (GNG) – allows to down sample the cloud (similarly to Voxel Grid),
- Enhanced Vector Quantization (EVQ) – allows to down sample the cloud.

Technically, the filters that do not perform down sampling or removing points should be rejected, since they do not resolve the problem addressed in the paper. Moreover, filters should be widely available and allow to process clouds in real-time. The first condition makes indices extracting and spatial filters not useful. GNG and EVQ algorithms seem to be promising, but they are not widely available, since they are novel procedures. It makes them hard to use and to implement in an optimized version. What is more, the results they produce are very similar to Voxel Grid filter. Considering the fact that they may need some kind of initialization [5] (which depreciates their use in changing environment) and that the better results may influence the computational time [9], only Voxel Grid algorithm will be considered from this group. Another group of filters are Outlier Removal filters. They can increase the quality of the cloud reducing their size, so they have to be taken into consideration. What's more, they are well known and easy to use, so they set up good baseline for evaluation. They do not require any kind of initialization (but may need one additional iteration for calculating e.g. some mean values). The last group enables removal points under given conditions, and here Conditional Removal filter seems be very promising. However, Conditional Removal is a general concept and typically it is based on a Pass Through filter (which may be considered a special case of Conditional Removal). Thus we select from this group the Pass Through filter for testing. Some of the filters are not useful for optimization purposes, while alternative ones have other disadvantages.

Our review allows us to select four most important, popular and promising filters that have been chosen for evaluation, namely: Pass Through, Voxel Grid, Statistical and Radial Outlier Removal.

Voxel, which is a group of cube units distributed in the centre of the orthogonal grid, can be understood as the extension of two-dimensional pixel into three-dimensional space. The point cloud data generated by a computer vision method is usually density-uneven. It samples by the voxel grid method and creates 3D voxel grid for the input point cloud data, with centroid of all the points in voxel to approximate the other points, all of which can not only reduce the point cloud data, but also maintain the shape characteristics of point cloud and more accurate approximation of the surface. All points in the voxel are expressed with a centroid, then:

$$\bar{x} = \frac{1}{S}\sum_{x,y,z\in V} x, \quad \bar{y} = \frac{1}{S}\sum_{x,y,z\in V} y, \quad \bar{z} = \frac{1}{S}\sum_{x,y,z\in V} z$$

where S is the total number of discrete points in voxel V, x, y, z are dimensions of voxel (referred to as leaf).

The PassThrough filter removes points that lie outside a given range for the specified user-given dimension. For example, if all points laying farther than 3 m away have to be discarded, the filter would have to be run on the Z coordinate with a range of [0; 3 m]. This filter can be useful to discard unneeded objects from the cloud, but a different reference frame may have to be adapted if the default one (relative to the sensor) is inappropriate. For example, filtering on the Y value to remove all points not laying on a given surface will yield unwanted results if the camera is at an odd angle. So we define visibility range as R =<0; z_{max}>, where z_{max} is the maximal depth of voxel from the camera plan.

Outliers are single points that are spread through the cloud. They are the product of the sensor's inaccuracy, which inappropriately registers measurements from empty space. Outliers are considered undesired noise, because they may introduce calculation errors, e.g. in normal estimation. Hence, removing the points from the cloud will not only make the computations faster, but also more precise. The radius-based outlier removal is the simplest method of this type. First, search radius r must be specified as well as the minimum number of neighbours K that a point should have to avoid being labelled as outlier. The algorithm will then iterate through all of the points (which can be extremely slow in if the cloud is big) and perform the check: if less than that the given number of the points are found within the radius, the point is removed.

The statistical outlier removal process is a more advanced method. First, for every point the mean distance to its K neighbours is computed. Then, assuming that the result is a Gaussian distribution with a mean μ and a standard deviation σ, all points with mean distances falling out of the global mean plus deviation are removed. It preforms statistical analysis of the distances between neighbouring points, and trims all which are not considered "normal" (which is a parameter of the algorithm). In further part of paper we assign symbol K to the number of neighbours to analyse for each point and m to the standard deviation multiplier.

TESTS

The test system was implemented in a way that the results could be kept as accurate as possible, relative, and insensitive to environment changes.

The application takes as input the data from Kinect cameras (up to 8) - RGB video stream with a monochrome depth video stream [6]. The stream specification is as follows:

- each point is represented on 9 bytes, 6 which describe coordinates in 3D space and 3 represent RGB colour. Every frame contains 640x480 points (the VGA normal resolution), resulting in 2 764 800 bytes of data. Multiplied by 30 frames per second, the result is 82 944 000 bytes/sec, which is equal to 79.1 megabytes per second,
- frame output of the camera is 640x480 (VGA),
- frame output can be imposed, in our research 30 frames per second was assumed.

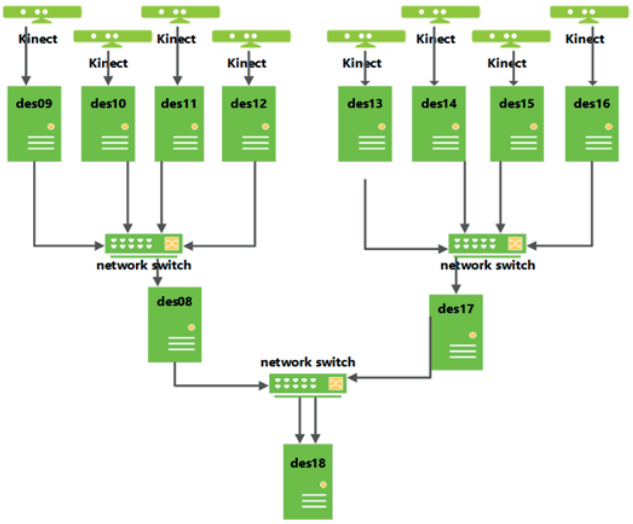

Fig. 2. The schematic of the test environment

The output of the application is a merged point cloud. It is released only after the input from every camera was fully delivered to the final node.

The tests were run for different numbers of cameras (2-8), for one point cloud resolution (see Fig. 2). We were able to observe the delay that was caused both by cloud processing and sending it to next node. The limitation of the system is implied by the largest number of cameras that can be handled by the network. It allows to choose the best configuration for a concrete system, and to decide which configuration is most scalable. The perfect configuration would enable adding as many cameras as possible, without any visible impact on the system delay. Since it is not possible to provide such configuration, we aim at finding the best possible solution.

Obviously, the tests results should not depend on the changes in the input frames, assuming that all of them are similar. For that reason, for each test tens of thousands of

point clouds were sent. For each point on the x-axis (see charts below) there were sent up to three thousands of point clouds.

The test results are given as percentages – we assume, that 100% of frequency is an ideal frequency, equal to the frequency of capturing data from the sensor (30 Hz). The 100% of point cloud size is the size of unfiltered cloud. Because of the possible occurrence of changes in the environment (and consequently the relative results inaccuracy), the unfiltered clouds are sent during the test along the filtered ones. For each single test (for each single x-axis point) the same number of filtered and unfiltered cloud sequences were sent. Between these sequences also delay periods were introduced in order to avoid possible overlays. Finally, for each single test arithmetic means were computed. Tests were performed for each of the algorithms described above. All filters were tested in terms of changes of the point cloud size and the transmitting frequency.

All presented charts are two-dimensional, where the axes represent respectively:

- Changes of the considered parameter of filter – this can be understood as test range of different values of attributes,
- Values – point cloud size, differential growth or frequency.

The measures are described in more detail below.

For each of the filters two charts are presented:

1. Mean size – size of the cloud, which should be reduced by the filter. Note, that the cloud cannot be too small, because it would became useless for the user. On the other hand, if the cloud is too big, then the filter is not helpful at all, and superfluously wastes resources.
2. Frequency – the rate of the cloud processing on the server. By default, the cloud is captured 30 times per second. If the filtering is too slow, the frequency is smaller than 30 frames per second, and this should be avoided, i.e. if frequency is no longer effective, then the last effective point was probably the most effective configuration. This means, that probably the best parameters were just passed, i.e. this is the smallest cloud which can be effectively achieved using this filter. Although, if the system had no requirement of real-time operation, then this issue would not be as important as in the considered system.

RESULTS

VOXEL GRID FILTER

The Voxel Grid test was performed by changing the leaf size. Generally, the leaf size is composed of three dimensional variables (X, Y and Z). All of them were manipulated in the same way and the same time. Manipulating the components independently is probably useful only in specific applications. Generally, the Voxel Grid "boxes" shape is a cube, ensuring that the point dispersion is balanced. Note, that changing the box size for each dimension simultaneously causes rapid change of cube volume. Pre-processing using filters allows to modify the image: extract particular features and hide others. To give general impression about capabilities of the filtering in Fig. 3 we present the effect of pre-processing using a Voxel Grid applied for a three single frames.

Fig 3. Effect of pre-processing using a vortex grid on a single frame (leaf= 0.01)

The results shown in the Fig. 4a show, that there is a rapid change of point cloud size in the first test, but the size very quickly reaches to 0 value. This means, obviously, that the filter is very effective, but sending nearly empty point cloud is useless. When observing point cloud size changes, only the first values of the leaf size should be considered as useful. The frequency chart (Fig. 4b) shows, that the frames per second factor is very stable (with some fluctuations). Fortunately, the frequency is stable at 100% and changes to about 80% when the size of leaf is close to 0.50. 80% is not an acceptable value in terms of real-time system requirements, but for some systems it would be considered as acceptable too, even for real-time in specific cases. The last chart shows, that as a results of very high frequency rates, only the first chart (size) should be considered in the majority of cases, as it has higher impact on the effects of the working system. The differential changes are not very useful for this test, because the size of the cloud very rapidly reaches values close to zero.

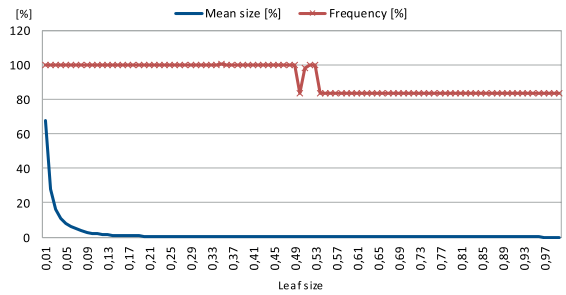

Fig. 4. (a) Changes of the point cloud size and (b) frequency of cloud transmitting

PASS THROUGH FILTER

The Pass Through filter is very characteristic because of its irregularity and difficulties in estimation of usability. This is because the efficiency of the filter strictly depends on the shape of the cloud. What is more, it is possible, that the filter can remove some important point concentration when used improperly. Generally, the charts may help to properly rate the filter, but its configuration (attribute choice) should depend also on the visual observation.

The test was performed in such a way, that the range of filtered area changes in one dimension. However, in practise, it would be worth performing it for each axis independently. The range was changed bilaterally. The changes of the size (Fig. 5a) are irregular, as expected, but the size never grows. The zero value at the end is also expected, because the range of the filter was configured to finally filter out the entire cloud. Looking at the chart, the size reduction is useful for range attribute value lower than 5. However, as it was mentioned before, it is important to make sure, that significant points were not deleted, because the filter may give different results for different clouds (i.e. acquired in different environments).

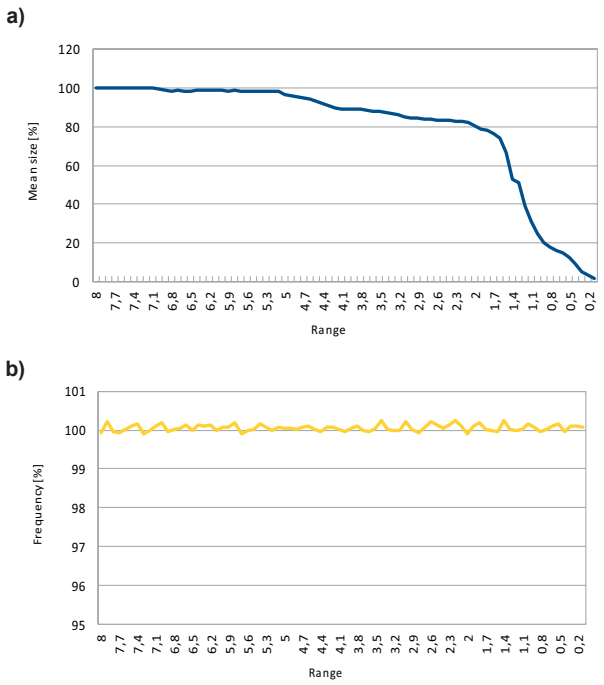

Fig. 5. (a) Changes of the point cloud size and (b) frequency of cloud transmitting

The frequency chart (Fig. 5b) shows, that Pass Through filter works very fast for any values of the range attribute. Again, this means, that the filter should be rated using size changes only, including the differential chart.

FILTERING WITH STATISTICAL OUTLIER REMOVAL

For Statistical Outlier Removal filter we performed two different tests because there are two different factors of the filter (K and m - number of neighbours and multiplier quantity). It turned out, that the number of neighbours was not very significant, so we skipped them. The multiplier tests were performed using number 10 for the neighbours.

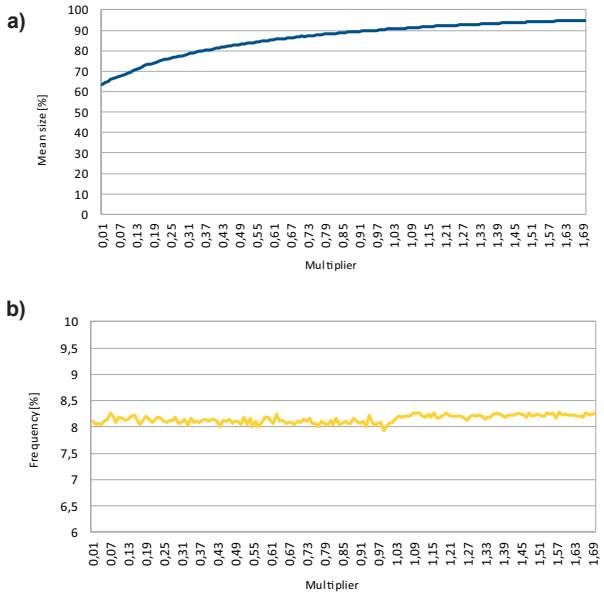

Fig. 6. (a) Size of cloud filtered (b) frequency of cloud transmitting using Statistical Outlier Removal filter (K=10)

The point cloud size (Fig. 6a) dropped along with the value of the multiplier. High values of the multiplier result in the higher number of points accepted by the filter. Because the minimal size of the cloud for this filter corresponds to most cases, the tests for smaller values were not necessary. Changing the multiplier is mostly useful for small values, as no significant change was observed for high values. Unfortunately, the frequency chart (Fig. 6b) shows, that the filter does not apply for real-time systems. What is more, the frequency seems to be the biggest for larger values of the multiplier.

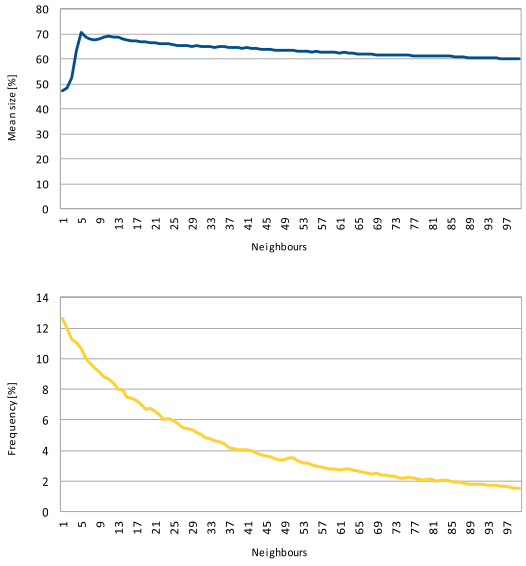

Fig. 7. (a) Size of the cloud filtered (b) frequency of cloud transmitting using Statistical Outlier Removal filter (m=0.10)

The size chart (Fig. 7) might indicate, that additional tests for bigger number of neighbours should be performed, but the frequency chart shows, that the frequency of cloud transmitting would be too low. The filter definitely does not meet the requirements, As the performed tests exclude the filter from real-time use. However, the filter may still be useful for some calculations that can be performed after the process of grabbing the clouds, e.g. if all clouds are stored at the server, then such filter could be used to reduce the size of clouds, discarding the useless points.

FILTERING WITH RADIAL OUTLIER REMOVAL

The Radius Outlier Removal test, similarly to Statistical Outlier Removal, was performed for two factors separately: neighbours K and radius r (instead of multiplier for Statistical Outlier Removal). Note, that the neighbours test was performed twice because the results were different for two different values of radius (r=0.1 and 0.05). The radius test was performed using the number of neighbours K=25.

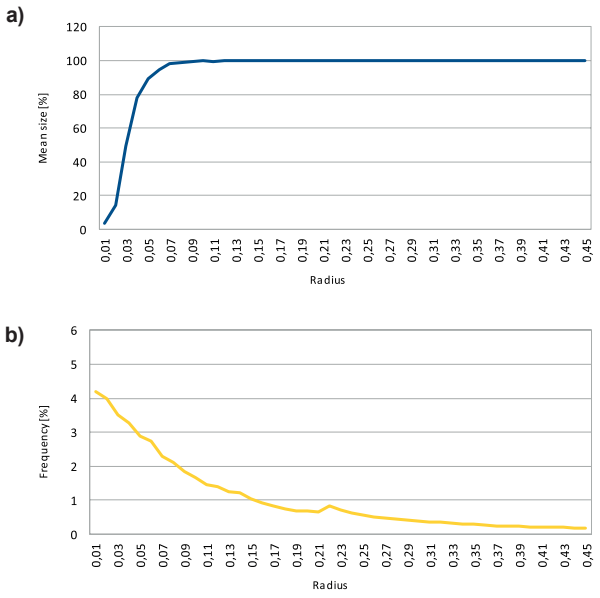

Fig. 8. (a) Size of the cloud filtered (b) frequency of cloud transmitting using Radial Outlier Removal filter (K=25)

In the radius test, the size (Fig. 8a) changes for only small radius values. For high radius values, more and more points are accepted so the cloud is actually not filtered. This means, that the filter is useful only for values of radius up to 0.07 assuming, that the number of neighbours is 25. Note, that the result may be slightly different for different kinds of point cloud. For stable cloud size equal to 100% of the original (not filtered) cloud, the differential value is zero as well. The differential chart shows the points, where the size grows in a higher or lower degree. Interestingly, it can be used to find the point of inflection (the extreme of the differential). This information can be useful for the deep analysis of the filter efficiency. Similarly to Statistical Outlier Removal, the analysed filter is not appropriate for real-time systems. The frequency rate (Fig. 8b) is too small. Of course, it may be useful for some processing performed on the last node, which does not require real-time operation.

The tests of Radius Outlier Removal with changing neighbours number were performed twice, because of two different results for two similar radius values. Obviously, the higher value of neighbours' condition, the smaller the point cloud (Fig. 9a). This is because a higher number of neighbours is required for a point in order to not be deleted. Interestingly however, the change of the size is not rapid. The further experiments were not performed (for higher numbers of neighbours), because of the frequency results (see Fig. 9b), and because of much more satisfactory results acquired in the tests performed with radius 0.05.

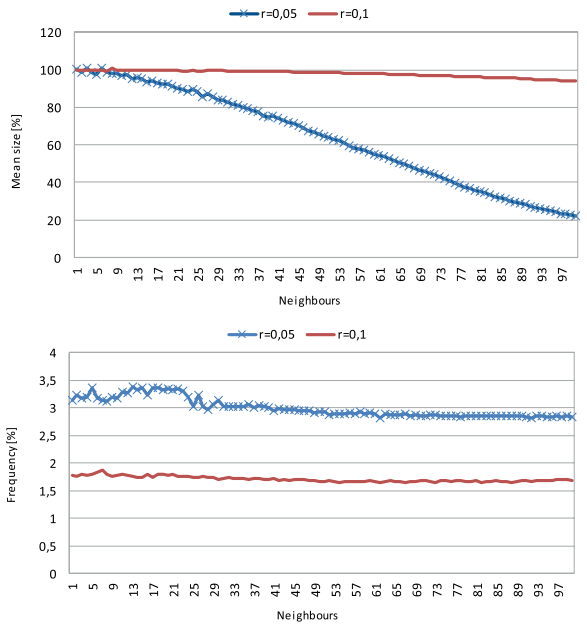

Fig. 9. (a) Size of the cloud filtered (b) frequency of cloud transmitting using Radial Outlier Removal filter (r=0.1 and 0,05)

Fig. 9b shows again, that the Outlier Removal filters are not useful for real-time systems. It shows that very small change of radius parameter (from 0.1 to 0.05) causes dynamic changes in frequency. For the same number of neighbours, the cloud size changes much faster and makes the filter much more useful. Interestingly, not only the size results were better, but also frequency.

SUMMARY AND FUTURE WORK

The paper compares four filters with various parameters choice. We performed time-consuming and precise tests to get the best and most accurate results. The results were averaged and collected in diversified, coherent and legible charts. Tests prove, that both Voxel Grid and Pass Through filters can be used mostly in any appropriate parameters configuration. Of course they should be used with caution, to be able to perform cloud reconstruction and not to lose important or relevant points, since all of them carry some information.

The Outlier Removal filters do not meet real-time requirements, so it is not recommended to use them in such

systems. However, they can be used in non-real-time systems. What is more, they can be possibly applied in real-time systems with less strict requirements. As optimization is a large area to investigate, the research can be continued in many ways. As mentioned in the beginning of the paper, there are many approaches for optimising point cloud transmission and consolidation. What is more, Point Cloud processing, even if performed in distributed environment, can be optimized in more than one way and does not have to be focused only on networking. Each of the steps of PCD data flow (described in Section 2) can be more or less optimized. This paper was mostly focused on the pre-processing strictly connected with networking. However, other parts were also partially covered. A sort of optimization was also tried from physical layer perspective. In General, all of these stages can be optimised. First of all, the main topic of the paper – optimization of pre-processing can be performed in other ways. There are some other filters which can be tested instead of the presented ones. Moreover, there are also other ways of point cloud reduction (i.e. compression). Also networking can be optimised (e.g. the choice of the protocol). Additionally, post-processing is a good area to optimize. It is also worth to find a more real-time adjusted solution. Filters should be used with caution, in order to keep the possibility of performing cloud reconstruction or not to lose important or relevant points.

Voxel Grid and Pass Through filters turned to be applicable in any appropriate configuration. Outlier Removal do not meet real-time requirements, and are not recommended for use in such systems. They can be still used in non-real-time systems.

REFERENCE

1. W. Chau-Chang, C. Min-Shine: Nonmetric Camera Calibration for Underwater Laser Scanning System. IEEE Journal of Oceanic Engineering, vol. 05, 32(2), (2007), 383-399.

2. S. Ferrari, I Frosio, V. Piuri, N.A Borghese: Enhanced vector quantization for data reduction and filtering. Proceedings of 2nd International Symposium on 3D Data Processing, Visualization and Transmission, 3DPVT, (2004), 470–477.

3. S. Gernhardt, X. Cong, M. Eineder, S. Hinz, R. Bamler: Geometrical fusion of multitrack ps point clouds, IEEE Geoscience and Remote Sensing Letters, vol. 9(1), (2012), 38–42.

4. H. Haggag, M. Hossny, D. Filippidis, D. Creighton, S. Nahavandi, V. Puri: Measuring depth accuracy in rgbd cameras. 7th International Conference on Signal Processing and Communication Systems (ICSPCS), (2013), 1–7.

5. L. Hong Xie, Z. Zhao: A new method of cylinder reconstruction based on unorganized point cloud, 18th International Conference on Geoinformatics, (2010), 1–5.

6. P. Kiljański, Optimization of PCD consolidation process in distributed system, Master Thesis, Gdansk University of Technology, 2014

7. P. Li, H. Wang, Z. Liu: A morphological LIDAR point cloud filtering method based on fake scan lines, International Conference on Electronics, Communications and Control (ICECC), (2011), 1228–1231.

8. D. McLeod, J. Jacobson, M. Hardy, C. Embry: Autonomous inspection using an underwater 3D LiDAR. 2013 OCEANS ,San Diego, (2013), 1-8.

9. S. Orts-Escolano, V. Morell, J. Garcia-Rodriguez, M. Cazorla: Point cloud data filtering and downsampling using growing neural gas. International Joint Conference on Neural Networks(IJCNN), (2013), 1–8.

10. R. Rusu, S. Cousins: 3D is here: Point cloud library (PCL). Proc. of International Conference in Robotics and Automation (ICRA), (2011), 1-4.

11. K. Santilli, K. Bemis, D. Silver, J. Dastur, P. Rona: Generating realistic images from hydrothermal plume data. Visualization, 2004. IEEE, (2004), 91-98

12. P. Thumbunpeng, M. Ruchanurucks, A Khongm: Surface area calculation using Kinect's filtered point cloud with an application of burn care. International Conference on Robotics and Biomimetics (ROBIO), (2013), 2166–2169.

13. Y. Wan, Z. Miao, Z. Tang: Reconstruction of dense point cloud from uncalibrated widebaseline images. IEEE International Conference on Acoustics Speech and Signal Processing (ICASSP), (2010), 1230–1233.

14. Y. Wang, X. Xiang Zhu, R. Bamler, S. Gernhardt: Towards terrasarx street view: Creating city point cloud from multi-aspect data stacks. Proc. of Joint Urban Remote Sensing Event (JURSE), (2013), 198–201.

15. H. Wenming, L. Yuanwang, W. Peizhi, W. Xiaojun: Algorithm for 3d point cloud denoising. 3rd International Conference on Genetic and Evolutionary Computing WGEC '09, (2009), 574–577

CONTACT WITH THE AUTHOR

Tomasz Dziubich

Gdańsk University of Technology
G. Narutowicza 11/12 street
80-233 Gdańsk
Poland,

e-mail: dziubich@eti.pg.gda.pl