Advances in Science, Technology and Engineering Systems Journal Vol. 2, No. 1, 76-83 (2017)

www.astesj.com

Special Issue on Computer Systems, Information Technology, Electrical and Electronics Engineering

ASTESJ

ISSN: 2415-6698

Frame Filtering and Skipping for Point Cloud Data Video Transmission

Carlos Moreno, Ming Li

Department of Computer Science, California State University, Fresno, 93740, USA

ARTICLE INFO

Article history:
Received: 18 December, 2016
Accepted: 19 January, 2017
Online: 28 January, 2017

Keywords:
Filtering
Frame Skipping
Point Clouds

ABSTRACT

Sensors for collecting 3D spatial data from the real world are becoming more important. They are a prime research area topic and have applications in consumer markets, such as medical, entertainment, and robotics. However, a primary concern with collecting this data is the vast amount of information being generated, and thus, needing to be processed before being transmitted. To address the issue, we propose the use of filtering methods and frame skipping. To collect the 3D spatial data, called point clouds, we used the Microsoft Kinect sensor. In addition, we utilized the Point Cloud Library to process and filter the data being generated by the Kinect. Two different computers were used: a client which collects, filters, and transmits the point clouds; and a server that receives and visualizes the point clouds. The client is also checking for similarity in consecutive frames, skipping those that reach a similarity threshold. In order to compare the filtering methods and test the effectiveness of the frame skipping technique, quality of service (QoS) metrics such as frame rate and percentage of filter were introduced. These metrics indicate how well a certain combination of filtering method and frame skipping accomplishes the goal of transmitting point clouds from one location to another. We found that the pass through filter in conjunction with frame skipping provides the best relative QoS. However, results also show that there is still too much data for a satisfactory QoS. For a real-time system to provide reasonable end-to-end quality, dynamic compression and progressive transmission need to be utilized.

1. Introduction

This paper is an extension of work originally presented in the International Conference on Internet and Multimedia Technologies 2016 as part of the World Congress on Engineering & Computer Science 2016 [1]. We extend our work by implementing an additional technique to achieve a better quality of service (QoS). Originally, we relied on filtering methods to lower the cost of network transmission for point cloud data (PCD). In this work, we explore the use of a frame skipping technique in addition to filtering to improve the frame rate and other QoS metrics.

The collection of 3D spatial data from the real world for video streaming is a method of communication that has experienced growth in recent years. This area provides research challenges and contains applications in consumer markets, such as medical, entertainment, and robotics [2]. Providing good efficiency and high quality for video streaming requires:

(i) a hardware sensor that is able to capture spatial information in all three dimensions;

(ii) encoding and decoding algorithms; and

(iii) data quality assurance mechanisms that meet application needs.

Hardware sensors have been in development for the past decade to allow for 3D image acquisition from a real world environment. A variety of techniques can be used to develop these sensors, such as time-of-light (TOF), stereo, lasers, infrared light, and structured light [3]. Due to their different approaches in acquiring depth data, there is a varying cost and size for these sensors. For example, the Velodyne spinning LiDAR system is expensive, costing thousands of dollars, making it, and sensors similar to it, impractical for many projects [4]. In contrast, there has been a rise in low cost solutions to collect RGB-D data, such as Microsoft's Kinect sensor [5]. For this study, we used the Kinect sensor to collect the 3D spatial data and the Point Cloud Library (PCL) to process them.

*Corresponding Author: Carlos Moreno, Department of Computer Science, California State University, Fresno, 93740, USA
Email: mmxzbnl@mail.fresnostate.edu

The primary concern with these hardware sensors is the rate and volume of information that is being generated. Depending on the quality of the sensor and the physical environment being sensed, a single capture can range from thousands to millions of points. The Kinect sensor, specifically, generates 307,200 points per frame (ppf), which leads to a data rate of about 45 megabytes per second (MBps) [6]. Such a data rate is impractical to achieve over a network for real-time video streaming, especially when juxtaposed with the average American household bandwidth of 12.6 megabits per second (Mbps) [7].

Therefore, in order to overcome the network bandwidth bottleneck, the use of filtering methods is considered a viable solution. Filtering allows us to modify or remove data from a dataset according to certain criteria. Numerous methods for filtering data have been developed over the past years, originating from different services and application needs. Given the variation in filtering algorithms, a comparative study is necessary that is able to compare and contrast different filters for use in PCD video streaming.

While filtering the PCD does significantly reduce the data rate for video transmission, it is still impractical for any real world application. Therefore, in addition to studying the effects of different filtering methods, we require the use of a frame skipping technique. While traditional frame skipping is used when buffer overflow occurs, we propose a different use of the scheme [8]. Rather than transmitting every frame that the hardware sensor generates, we check for how similar the frame is to the previously transmitted one. Frames that are deemed too similar are skipped. Using such a technique allows the application to send only the PCD that is significantly different visually for the user, thereby saving transmission costs. Furthermore, as explored in [9], the use of frame skipping techniques is beneficial from the end-user's perspective.

In this paper, we have implemented and conducted experiments to evaluate four filters: pass through filter, voxel grid filter, approximate voxel grid filter, and bilateral filter. In addition, we tested the frame skipping technique both in isolation and with the use of filtering. Results show that, overall, the pass through filter with frame skipping provides the best QoS relative to the other options. Yet, despite the reduction in the number of points being transmitted, the data rate is still too high. We conclude that, in conjunction with filtering methods, a PCD video streaming service will require further techniques such as dynamic compression and progressive transmission.

The rest of this paper is organized as follows. Section 2 discusses the works related to this paper, including a discussion on other uses of the Microsoft Kinect, PCL, and frame skipping. Section 3 reviews the architecture behind the Kinect and PCL, as well as introducing the octree data structure. Section 4 explores the different filtering methods used in the paper. Section 5 discusses the frame skipping technique in the context of our work. Section 6 explains the experiment setup while section 7 details the QoS metrics developed for evaluation purposes. Section 8 provides an analysis of the experiment results. Lastly, sections 9 and 10 summarize the work in this paper and offer a conclusion.

2. Related Works

Microsoft's Kinect was originally meant for entertainment purposes, but since its inception, it has been integrated in many other fields. For example, [10] has integrated the Kinect with Simulink to allow for real-time object tracking. Reference [11] shows that the hardware sensor is applicable to the medical field through its use in a virtual rehabilitation system to help stroke victims regain balance. The Kinect's inherit issues, as described in [12], can be transformed into useful information for use in creating automatic foreground segmentation. Lastly, [13] shows a unique example of utilizing the Kinect in the music field.

PCL has also experienced multiple uses. Reference [14] uses it to develop efficient facial registration processes, encompassed into the Digital Face-Inspection (DFI) system, to help in the dental field. Reference [15] explains how indoor robots also benefit from PCL by allowing for real-time, general object recognition. An important data structure used in PCL is the octree, and it too has implications in other fields, as shown in [16] and [17].

Frame skipping as a technique has seen considerable use in video encoders. For example, [8] proposes a dynamic frame skipping scheme coupled with an adaptive sliding window that is able to reduce the frame rate to better match the available bandwidth in live video streaming. The work in [18] takes this idea further by calculating the optimum frame allocation for the sliding window. A different approach is proposed in [19] by utilizing motion information, buffer state information and changes in the video scene. Lastly, [20] tackles the problem of real-time mobile application in a stereo video setting, and proposes the use of frame skipping as a hybrid approach.

3. System Architecture

Microsoft's Kinect is a peripheral hardware sensor originally developed for the interactive use of Microsoft's Xbox 360 video game console. The RGB-D sensor was developed with three main functions in mind: 3D image detection, human skeleton tracing, and audio processing. This functionality, in addition to its relatively inexpensive price, has drawn researchers and developers to utilize the Kinect for other purposes, such as robot vision and healthcare systems. The sensor itself consists of an RGB camera, an infrared (IR) emitter, an IR camera, and an array of microphones. The Kinect can achieve a maximum frame rate of 30 frames per second (fps) with a resolution of 640 x 480. Each color channel uses 8 bits while the depth data is represented in 16 bits [21]. Therefore, for a single frame consisting of 307,200 points, this Kinect raw data is represented in about 1.46 MB.

The Kinect raw data can be converted to point clouds. To process these point clouds, we require the use of an external

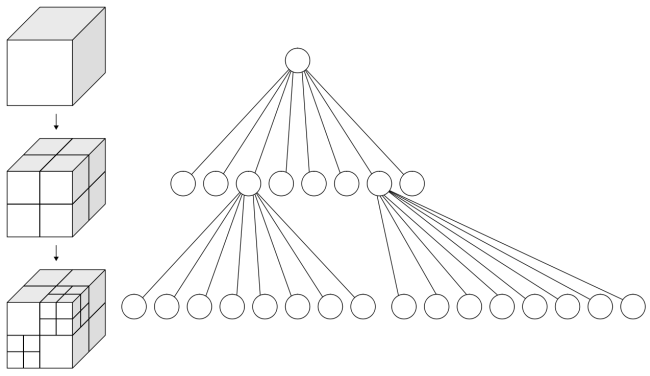

Fig. 1. A visual representation of a voxel (left) and its corresponding octree (right).

library. One such library is PCL, an open-source, fully templated C++ library. It incorporates many algorithms for point clouds and 3D geometry, such as filtering, feature estimation, visualization, segmentation, and more [22]. The PCD for the Kinect is about 10 MB in size; much larger than the raw RGB-D data. If we aimed for a target frame rate of 30 fps, this would result in a data rate of 300 MBps for PCL, compared to the 45 MBps of raw Kinect data. Processing and transmitting at either data rate is impractical.

In PCL, one of the fundamental data structures used to represent a point cloud is the octree. An octree recursively divides a point cloud into eight smaller sections, called voxels (see figure 1). These voxels are 3D cubes that encapsulate a subset of points from the point cloud. The resolution of the frame is determined by the depth level of the octree, i.e. a higher resolution translates to more voxels and a deeper octree.

Hardware and software aside, what makes reliable video streaming particularly challenging is finding the balance between the data rate and the network bandwidth. Having a high-quality hardware sensor and efficient software for processing is wasted if the network bandwidth is unable to deliver the data in real-time. Therefore, for this paper, we assume that the network is in such a state: the bandwidth is much less than the data rate. Moreover, we are using the following configurations for two computers: (i) a wired desktop PC with an Intel i7-6700 processor, GTX 745 GPU, and 16GB DDR3 RAM; and (ii) a wireless laptop with an Intel i7-2630QM processor and 8GB DDR3 RAM.

4. Filtering Methods

The filters used for this comparative study come from PCL. There is an extensive list of available filters; however, not all of these are practical or suited for data transmission in real-time. Narrowing down this list, we determined that the following four filters seem most applicable to real-time video streaming: pass through, voxel grid, approximate voxel grid, and bilateral. These filters each take a point cloud as input and will output a new, filtered point cloud.

4.1. Pass Through

The pass through filter passes the input points through constraints based on a particular field. It iterates through the entire point cloud once, performing two different operations. First, it removes non-finite points. Second, it removes any points that lie outside the specified interval for the specified field. For example, a programmer is able to set the field as the z-dimension (depth) and set the limit so that the filter removes any points that are half a meter away from the sensor.

4.2. Voxel Grid

The voxel grid filter assembles a 3D voxel grid over the entire input point cloud. Visually, this can be represented as a set of cubes being placed over the entire point cloud. For each individual voxel, the points that lie within it are down-sampled with respect to their centroid. This approach has a few drawbacks: (i) it requires a slightly longer processing time as opposed to using the voxel center; (ii) it is sensitive to noisy input spaces; and (iii) it does not represent the underlying surface accurately [23].

4.3. Approximate Voxel Grid

The approximate voxel grid filter attempts to achieve the same output as the voxel grid filter. However, rather than using the method described above, it sacrifices accuracy for speed. Rather than carefully determining the centroid and down-sampling the points, this filter makes a quick approximation of the centroid through the use of a hashing function.

4.4. Bilateral

The bilateral filter preserves the edges in a frame and reduces the noise of a point cloud. This is performed by replacing the intensity value for each point in the frame by the weighted average of intensity values from nearby points, based on a Gaussian distribution. These weights depend on the Euclidean distance and the differences in ranges (such as color and depth). For further information and a detailed explanation of the bilateral filter, see [24].

5. Frame Skipping

In general, frame skipping refers to the process of selecting a key reference frame to interpolate other frames. Rather than transmitting all video frames, the technique is able to skip certain frames that share a large number of similar data. By selecting a reference frame that is transmitted, the decoder is able to interpolate and fill in the skipped frames. The aim for frame skipping algorithms is to minimally affect the video quality and to significantly reduce the transmission size [25].

Frame skipping is typically caused by buffer overflow. This phenomenon frequently occurs after the encoding of an I-frame. As discussed in [26], this is due to the relatively large number of bits entering the buffer in contrast to the bits moving out. Interestingly, as noted in [25], frame skipped videos tend to be nearly as tolerable to packet loss as their raw video stream counterparts.

In this paper, we use frame skipping to refer to a simpler process. Rather than using interpolation, or some other method of estimation, our experiment simply displays the same frame once more. This keeps our system relatively complex-free and allows us to focus on the way the frame skipping algorithm selects which frames to skip. It does so by comparing the octrees of the two point cloud frames. Using PCL's octree change detector data structure, we are able to compare the voxels of the two octrees recursively. This will return a vector containing the indices of the voxels that differ. To calculate a similarity percentage (SP), we use:

$$SP = \frac{\#\ of\ points - \#\ of\ different\ points}{\#\ of\ points} * 100 \quad (1)$$

Frames that have a SP exceeding our similarity threshold are deemed too similar and therefore skipped.

6. Experiment Setup

In order to conduct a survey of different filtering algorithms, we needed to setup an experiment that is able to measure certain characteristics and compare these measurements. Because this is in the context of video streaming, we developed a threaded client/server application (see figure 2) that tests each filter and the possible benefits of frame skipping.

The client program is designed for generating, skipping, filtering, and transmitting the point clouds. It is connected to the Kinect hardware sensor to capture the RGB-D data from the real world environment. The responsibilities of the client program are divided into four threads:

Table 1 (Summary of QoS Metrics)

Filter	Filter Percentage	Branch Similarity	Point Similarity	Color Similarity
No Filter	0.00%	100.00%	100.00%	100.00%
Pass Through	88.07%	13.26%	99.77%	98.81%
Voxel Grid	79.82%	62.46%	78.20%	99.42%
Approximate Voxel Grid	73.27%	74.96%	75.33%	99.19%

Table 2 (Summary of Frame Similarity metric)

Filter	Without frame skipping	With frame skipping
No Filter	98.85%	92.01%
Pass Through	96.32%	87.74%
Voxel Grid	97.92%	92.26%
Approximate Voxel Grid	97.15%	90.93%

(i) convert the raw RGB-D data from the Kinect sensor into PCD using PCL and store this into buffer A;

(ii) access the PCD from buffer A, calculate the similarity percentage against the most recently transmitted PCD, skip the frame if the percentage exceeds the threshold, otherwise store the PCD in buffer B;

(iii) access the PCD from buffer B, filter it, and store the filtered PCD in buffer C; and

(iv) access the PCD from buffer C, fragmentize it, and transmit the datagrams to the server program using UDP.

The server program is designed for receiving the point clouds from the client and visualizing them to the monitor for display. Similar to the client, the server disperses its responsibilities to threads:

(i) receive datagrams containing PCD from the client program, defragmentize, and store the PCD in a buffer; and

(ii) access the PCD from the buffer and visualize it using PCL's visualization functionality.

We ran a total of ten tests: a base case which used no filter nor frame skipping, each of the filters, frame skipping, and each filter with frame skipping. These tests were all conducted in a wired-to using a wired connection to the Internet, while the server is receiving data through a wireless connection.

To test the relative effectiveness of each filter and the possible benefits of using a frame skipping technique in this experiment, we had to develop ways of measuring the QoS. We therefore created a set of QoS metrics that captured information for each test. This allows us to compare them in a quantitative fashion.

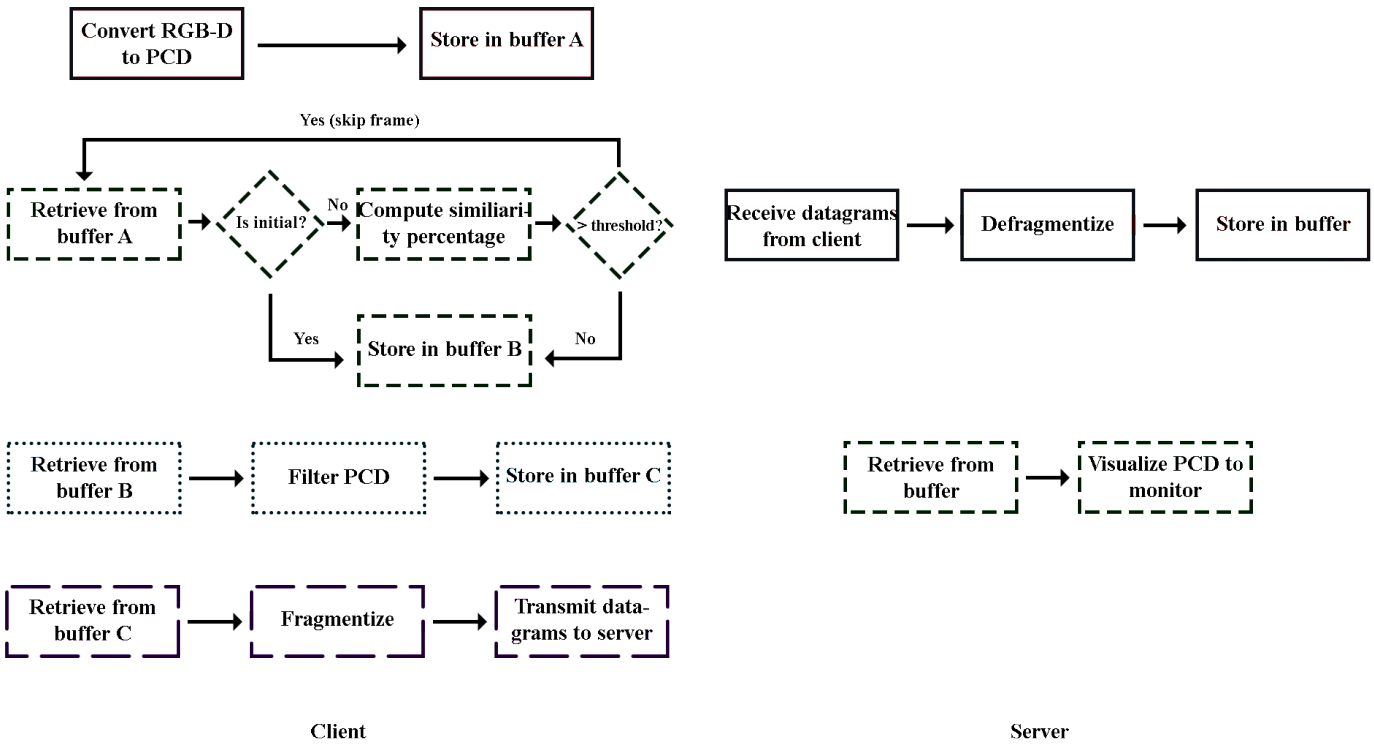

Fig. 2. The experiment flowchart of the threaded client/server applications. The client program (left) consists of four threads. The server program (right) consists of two threads.

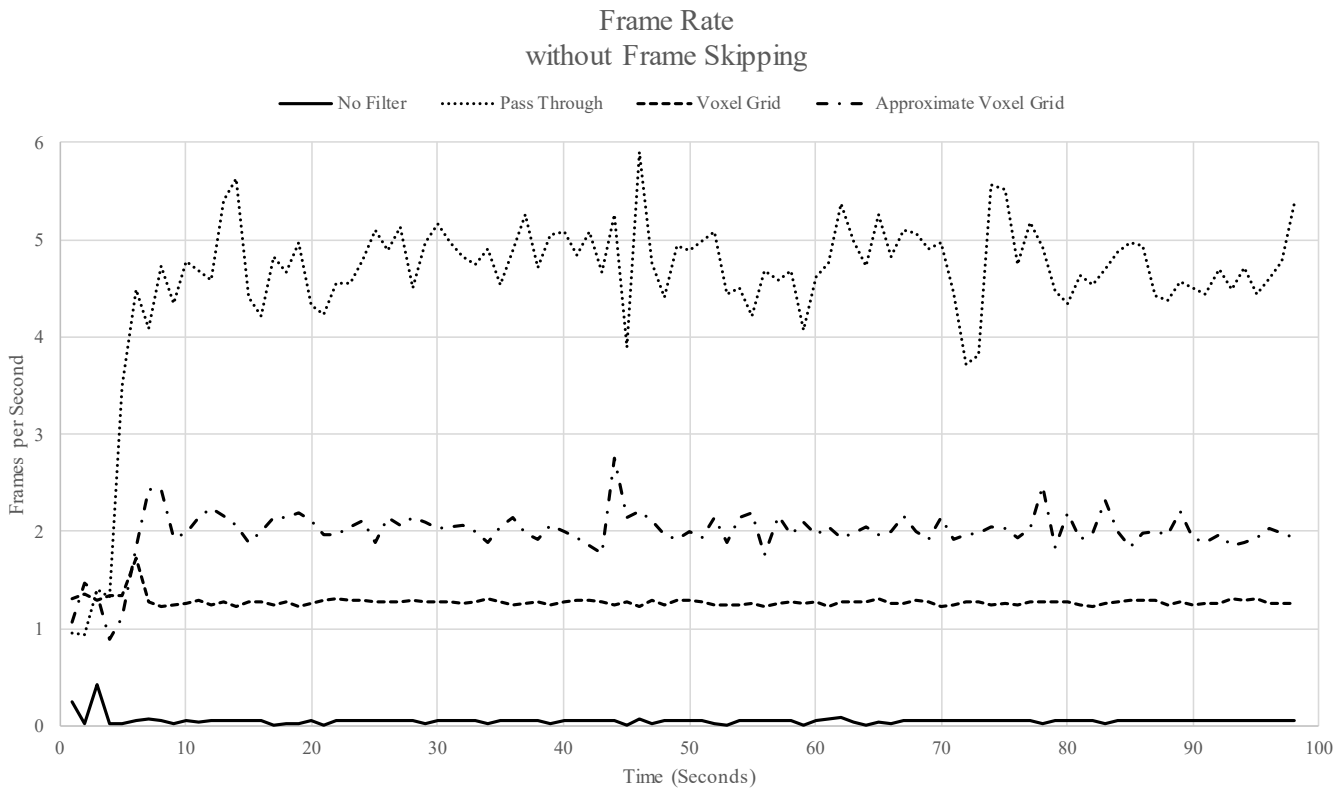

Fig. 3. A scatter plot that compares the frame rate for different filters in a wired-to-wireless network environment. The pass through filter maintains the highest fps of the four cases. This data was collected over a period of 500 frames.

wireless network environment, meaning that the client program is QoS Metrics

QoS is critical for the success of live video streaming. The quality of a video stream is, in part, determined by its *fps*. Video streams with a high fps usually result in a smooth visual experience for the users; low fps, on the other hand, tends to result in a "choppy" stream and poor user experience.

Another important characteristic of video streaming is the overall processing time required for each frame. To measure the QoS for this aspect, we used a *processing time metric*, which is the overall summation of the individual phases. This means, it adds up the time it takes for:

(i) determining if the point cloud should be skipped;

(ii) point cloud filtering;

(iii) point cloud transmission;

(iv) receiving point clouds; and

(v) point cloud visualization.

While the above two metrics work well in determining the QoS for video streaming in general, they do not take into account how well the filters are performing. Therefore, we also developed a *filter percentage metric*, meaning how many of the points in the original point cloud were filtered out. To calculate this, we found the difference in the number of points in the filtered point cloud from the original point cloud, and divided that by the original number of points.

Despite these three metrics, there is still a lack of measurement of how well the video stream performs visually. For that, we developed a set of three additional metrics:

(i) *branch similarity*, which compares the two branch structures of the octrees against each other;

(ii) *point similarity*, which measures how well two point values match up; and

(iii) *color similarity*, which calculates the similarity in the color values.

These visual QoS metrics are calculated by comparing the filtered PCD against the raw PCD.

In addition to the above three, we also required an additional visual QoS metric:

(iv) *frame similarity*, which compares how similar the current frame appears against the previous frame.

This allows us to quantitatively measure the video stream in terms of its visual differences.

7. Results

The tests that used the bilateral filter were disregarded due to the large processing time required. A single point cloud required over half an hour to filter, making it impractical for any live video streaming application.

Using the pass through filter provided the highest frame rate compared to the other filters (see figure 3). Moreover, it also achieved the lowest processing time. These can be explained due

to the pass through filter filtering the most points relative to the others (see table 1). With fewer points, the filtered point clouds from the pass through filter are smaller in data size, which allows for a higher frame rate and lower overall processing time.

In terms of the visual QoS metrics, the pass through filter appears to have a relatively low branch similarity. However, this is due to the nature of the filter. It effectively removes a large portion of the original point cloud, which drastically changes the underlying octree data structure. While the octree might be different, the pass through filter still maintains the highest similarity for the points (excluding the control case which uses no filter). See figure 5 for a visual depiction of the filter effects.

In regards to frame skipping, the results show that using a frame skipping technique provides a better end-to-end QoS. Comparing the five tests that did not use this scheme against the five that did shows that the benefits of the using the frame skipping technique. In terms of the fps, we found that the addition of frame skipping increased the frame rate for all filters (see figure 4). Specifically, the best combination arose from the pass through filter and frame skipping technique, which also achieved the lowest processing time. Moreover, by utilizing this technique, we found that the resulting PCD video stream was more responsive to changes (see table 2).

8. Summary

Among the four filtering methods allowed by PCL, the pass through filter results in the best scores for the QoS metrics. It removes the unnecessary background data, which reduces the point cloud data size and allows for a better experience in PCD video streaming. If the whole frame is required, however, the best filter is the approximate voxel grid, which outperforms the (normal) voxel grid filter in all QoS metrics. The addition of a frame skipping scheme had positive benefits overall. The frame rate increased for all filter options, although at the cost of a slight increase in processing time.

Although the use of filters and frame skipping reduces the original PCD size, the highest average frame rate that was achieved is merely 6.41 fps. Such a low frame rate cannot be considered to provide excellent end-to-end QoS. Therefore, while filtering and frame skipping improve the QoS additional techniques will be required.

For that purpose, we propose three additional techniques. First, compression, which will allow the data size to become even smaller, translating to a higher frame rate. Second, because of the network behavior that causes bandwidth fluctuation, a static compression ratio might work at certain bandwidth rates, but not all; instead, we need a dynamic compression algorithm that adjusts the compression ratio as a response to the bandwidth. Third, a progressive transmission scheme allows us to transmit the PCD layer-by-layer, in which each additional layer provides more details for the frame; the number of layers sent depends on the bandwidth and dynamically adjusts as the network changes.

9. Conclusion

Collecting 3D spatial data for PCD video streaming provides research challenges due to the high volume and high velocity data rate from the hardware sensors. Using Microsoft's Kinect sensor to collect the RGB-D data and PCL to process them, we were able

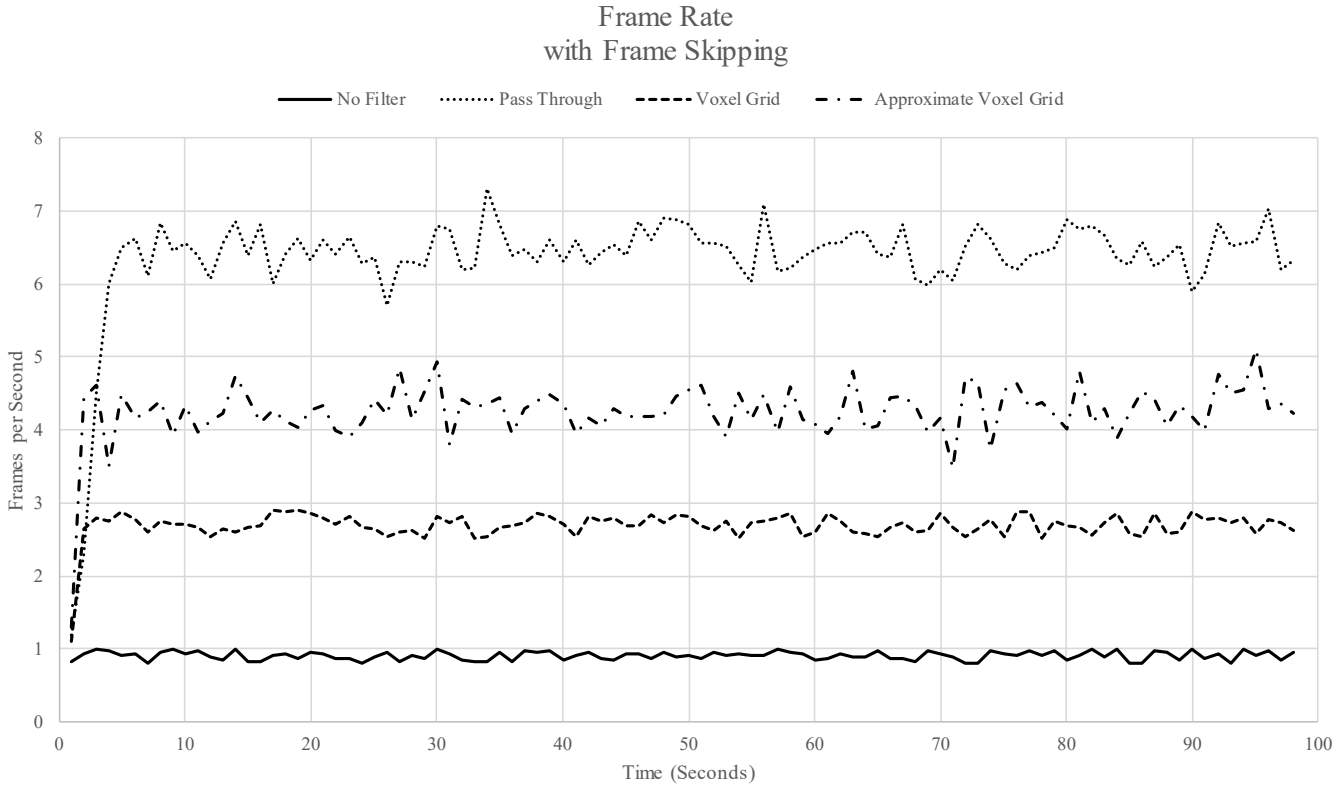

Fig. 4. A scatter plot that compares the frame rate for different filters using the frame skipping technique. The pass through filter maintains the highest fps of the four cases. This data was collected over a period of 500 frames.

(a) no filter

(b) pass through filter

(c) voxel grid filter

(d) approximate voxel grid filter

Fig. 5. A visual comparison of the four filter cases used in the experiment.

to compare and contrast different filtering methods to be used with the PCD. Filtering is a requirement due to the high data rate compared to the low bandwidth (300 MBps vs. 12.6 Mbps). In addition, we explored the use of a frame skipping technique that provides a better QoS experience for the end-user.

Using a threaded client/server application, we were able to survey the different filtering algorithms and the use of the frame skipping scheme by measuring a set of QoS metrics. Our results show that, in a live network environment, the pass through filter in conjunction with frame skipping achieves the highest scores in these metrics. Yet, utilizing only filters will not achieve a desirable video streaming experience. To do so, we propose the use of three additional techniques: octree compression, dynamic compression, and a progressive transmission scheme. Using these techniques provide further improvements by:

- compressing the octree data structure on the client-side to reduce the data rate
- dynamically adjusting the compression ratio in response to network bandwidth to support a reliable end-to-end QoS
- selectively transmitting high importance layers of the octree to lower the data rate

Conflict of Interest

The authors declare no conflict of interest.

References

[1] C. Moreno, M. Li, "A comparative study of filtering methods for point clouds in real-time video streaming," Proceedings of the World Congress on Engineering and Computer Science, San Francisco, CA, 2016.

[2] M. Miknis, R. Davies, P. Plassmann, A. Ware, "Near real-time point cloud processing using the PCL," 2015 International Conference on Systems, Signals and Image Processing, London, 2015.

[3] J. Fu, D. Miao, W. Yu, S. Wang, Y. Lu, S. Li, "Kinect-Like Depth Data Compression," in IEEE Transactions on Multimedia, **15**(6), 1340-1352, 2013.

[4] J. Kammerl, N. Blodow, R. B. Rusu, S. Gedikli, M. Beetz, E. Steinbach, "Real-time compression of point cloud streams," 2012 IEEE International Conference on Robotics and Automation, Saint Paul, MN, 2012.

[5] R. B. Rusu, Z. Marton, N. Blodow, M. Dolha, M. Beetz, "Towards 3D point cloud based object maps for household environments," Robotics and Autonomous Systems, **56**(11), 927-941, 2008.

[6] F. Nenci, L. Spinello, C. Stachniss, "Effective compression of range data streams for remote robot operations using H.264," 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems, Chicago, IL, 2014.

[7] D. Belson, J. Thompson, J. Sun, R. Möller, M. Sintorn, G. Huston, "The state of the Internet," Akamai, Cambridge, MA, Tech. Rep., 2015.

[8] Z. Zhang, H. Shi, S. Wan, "Dynamic frame-skipping scheme for live video encoders," 2010 International Conference on Multimedia Technology, Ningbo, 2010.

[9] Y. Qi, M. Dai, "The effect of frame freezing and frame skipping on video quality," 2006 International Conference on Intelligent Information Hiding and Multimedia, Pasadena, CA, 2006.

[10] J. Fabian, T. Young, J. C. P. Jones, G. M. Clayton, "Integrating the Microsoft Kinect with Simulink: real-time object tracking example," in IEEE/ASME Transactions on Mechatronics, **19**(1), 249-257, 2014.

[11] C. L. Lai, Y. L. Huang, T. K. Liao, C. M. Tseng, Y. F. Chen, D. Erdenetsogt, "A Microsoft Kinect-based virtual rehabilitation system to train balance ability for stroke patients," 2015 International Conference on Cyberworlds, Visby, 2015.

[12] T. Deng, H. Li, J. Cai, T. J. Cham, H. Fuchs, "Kinect shadow detection and classification," 2013 IEEE International Conference on Computer Vision Workshops, Sydney, NSW, 2013.

[13] M. F. Lu, J. S. Chiang, T. K. Shih, S. Wu, "3D sphere virtual instrument with Kinect and MIDI," 2015 8th International Conference on Ubi-Media Computing, Colombo, 2015.

[14] C. T. Hsieh, "An efficient development of 3D surface registration by Point Cloud Library (PCL)," 2012 International Symposium on Intelligent Signal Processing and Communications Systems, New Taipei, 2012.

[15] Q. Zhang, L. Kong, J. Zhao, "Real-time general object recognition for indoor robot based on PCL," 2013 IEEE International Conference on Robotics and Biomimetics, Shenzhen, 2013.

[16] F. Ouyan, T. Zhang, "Octree-based spherical hierarchical model for collision detection," 2012 10th World Congress on Intelligent Control and Automation, Beijing, 2012.

[17] J. He, M. Zhu, C. Gu, "3D sound rendering for virtual environments with octree," IET International Conference on Smart and Sustainable City 2013, Shanghai, 2013.

[18] Y. Zhou, H. Ma, Y. Chen, "A frame skipping transcoding method based on optimum frame allocation in sliding window," 2010 2nd International Conference on Signal Processing Systems, Dalian, 2010.

[19] S. Bhattacharyya, E. Piccinelli, "A novel frame skipping method in transcoder, with motion information, buffer fullness and scene change consideration," 2009 17th European Signal Processing Conference, Glasgow, 2009.

[20] I. L. Jung, T. Chung, K. Song, C. S. Kim, "Efficient stereo video coding based on frame skipping for real-time mobile applications," IEEE Transactions on Consumer Electronics, **54**(3), 1259-1266, 2008.

[21] A. Jana, Kinect for Windows SDK Programming Guide, Packt, 2012.

[22] R. B. Rusu, S. Cousins, "3D is here: Point Cloud Library (PCL)," 2011 IEEE International Conference on Robotics and Automation, Shanghai, 2011.

[23] S. Orts-Escolano, V. Morell, J. García-Rodríguez, M. Cazorla, "Point cloud data filtering and downsampling using growing neural gas," The 2013 International Joint Conference on Neural Networks, Dallas, TX, 2013.

[24] C. Tomasi, R. Manduchi, "Bilateral filtering for gray and color images," 6th International Conference on Computer Vision 1998, Bombay, 1998.

[25] K. W. Lim, J. Ha, P. Bae, J. Ko, Y. B. Ko, "Adaptive frame skipping with screen dynamics for mobile screen sharing applications," IEEE Systems Journal, **PP**(99), 1-12, 2016.

[26] P. Feng, Z. G. Li, L. Keng Pang, G. N. Feng, "Reducing frame skipping in MPEG-4 rate control scheme," 2002 IEEE International Conference on Acoustics, Speech, and Signal Processing, Orlando, FL, 2002.