computers

MDPI

Article

Color Reduction in an Authenticate Live 3D Point Cloud Video Streaming System

Zainab Sultani [1,*], Rana F. Al-Tuma [1] and Sandro Wefel [2]

[1] Computer Science Department, University of Technology, Baghdad 10066, Iraq; 110016@uotechnology.edu.iq

[2] Institute for Computer Science, Martin-Luther-University Halle-Wittenberg, Halle (Saale) 06120, Germany; sandro.wefel@informatik.uni-halle.de

* Correspondence: eng.zainabnamh@yahoo.com; Tel.: +962-792076189

Academic Editor: Laith Al-Jobouri

Received: 12 February 2016; Accepted: 20 July 2016; Published: 26 July 2016

Abstract: In this paper, an authenticate live 3D point cloud video streaming system is presented, using a low cost 3D sensor camera, the Microsoft Kinect. The proposed system is implemented on a client-server network infrastructure. The live 3D video is captured from the Kinect RGB-D sensor, then a 3D point cloud is generated and processed. Filtering and compression are used to handle the spatial and temporal redundancies. A color histogram based conditional filter is designed to reduce the color information for each frame based on the mean and standard deviation. In addition to the designed filter, a statistical outlier removal filter is used. A certificate-based authentication is used where the client will verify the identity of the server during the handshake process. The processed 3D point cloud video is live streamed over a TCP/IP protocol to the client. The system is evaluated in terms of: compression ratio, total bytes per points, peak signal to noise ratio (PSNR), and Structural Similarity (SSIM) index. The experimental results demonstrate that the proposed video streaming system have a best case with SSIM 0.859, PSNR of 26.6 dB and with average compression ratio of 8.42 while the best average compression ratio case is about 15.43 with PSNR 18.5128 dB of and SSIM 0.7936.

Keywords: video streaming; Microsoft Kinect; live 3D point cloud; filtering; compression; authentication; x509 certificate

1. Introduction

Videos are one of the most popular digital content and recently more communication systems are demanding real-time video transmissions [1]. Streaming technology is a cost-effective technique, where it saves significant amount of travel and time [2]. The idea of video conferencing has fascinated researchers in the past years and new interesting devices have been developed to enrich the streaming experience even more. Delivering live events such as conferences or surgical operations from remote locations was once a heavy, complex, and costly process. Today, with new technology, the live streaming becomes less expensive but still demands a lot of resources. Many applications require a full 3D view, which enable viewers near and far to see and feel the actual physical feeling of a scene such as streaming live surgical operations to a conference room [3]. Streaming 3D data has become an interesting popular topic and has obtained special attention in different fields of computer vision and virtual reality. This attention seems to be related and estimated by Japanese wireless industry peers in 2005, where they have expected a roadmap for reality video over wireless network see [4]. Delivering a smooth and steady media is not an easy task, but it is very hard to communicate when the stream is breaking down [5]. Wireless video transmission is commonly used by desktops, laptops and mobile devices. Nevertheless wireless channels introduce new challenges such as low bandwidth, delay

and interference. 3D video requires more transmission bandwidth than 2D video, for example an uncompressed standard 2D video streaming (640 × 480) consumes as much as 211 Mbit for 30 frames, while streaming 3D point cloud video for the same resolution consumes 1055 Mbit for 30 frames [6], and more over, the wireless channels are unstable and fluctuated. Therefore, the current wireless network can not provide enough bandwidth for different users so that end users will only receive the poor 3D perception quality [7]. To address these obstacles, extensive researches have been conducted, such as 3D video compression and 3D video processing [8].

In this paper, a color reduction filter is designed and applied to the captured 3D point cloud frames before being streamed to the client viewer where the reduction of the color values depends on each frame.

Streaming video systems sometimes require some kind of security especially when the application is streaming a highly confidential data; while on the other hand, sometimes the client only needs to verify that the streaming server is authentic. Authentication is based on certificates and a public key infrastructure [9]. The most common form of trusted authentication between parties network communication is the exchange of certificates [9,10]. Transport Layer Security (TLS) and its successor Secure Sockets Layer (SSL) authenticates and secures the data transformation across the network by using certificate-based authentication and public key cryptography such as Diffie-Hellman (DH) Protocol [11].

As mentioned earlier streaming videos are subject to wide variety of distortions starting from video acquisition, processing, filtering, compression, streaming and rendering. Eventually the quality of the streamed 3D video must be considered under these circumstances. An objective video quality metric plays an important role in any media application. It can be used to measure the quality of a video being transmitted in order to adjust the parameters of video streaming and to assist in designing filtering algorithms [12]. Objective quality measurements used in this paper are: peak signal to noise ratio (PSNR) and structural similarity index (SSIM). PSNR is used to measure the quality of reconstruction of lossy compression and it is usually denoted in terms of the logarithmic decibel scale (dB). Acceptable values for wireless transmission quality loss are between 20 and 25 dB [13–15]. Structural similarity (SSIM) index is a method for measuring the similarity of two images [12]. Since compression is used, compression ratio is also measured. Equations (1)–(3) represent the mathematical description of PSNR, SSIM and compression ratio respectively.

$$PSNR = 20 \times \log_{10}\left(\frac{MAX}{\sqrt{MSE}}\right) \tag{1}$$

where MAX is the possible pixel value of the image frame.

$$SSIM = \frac{\left(2\mu_x\mu_y + C_1\right) \times \left(2\sigma_{xy} + C_2\right)}{\left(\mu_x^2 + \mu_y^2 + C_1\right) \times \left(\sigma_x^2 + \sigma_y^2 + C_2\right)} \tag{2}$$

where X and Y are the reference and the decoded images receptively, μ is the mean intensity, σ is the signal contrast (standard deviation), σ_{xy} is the covariance of x and y, and C_1, C_2 are constants.

$$CompressionRatio = \frac{Sizeofuncompressed3Dpointcloudframe}{Sizeofcompressed3Dpointcloudframe} \tag{3}$$

The research space for the proposed live 3D video streaming system can be explained with respect to these four dimensions: (1) 3D video data-representations, (2) efficient 3D video filtering and compression techniques, (3) live capturing and processing and (4) authentication for the 3D video owner. The main problem to be addressed in this research is whether a system can be designed to transmit a live 3D point cloud video stream over a wireless LAN from an authentic server.

The paper presents related work in Section 2. Outlier removal filter and octree compression are presented in Section 3. The framework of the proposed streaming system is presented in Section 4. The experimental results and discussion are demonstrated in Sections 5 and 6 respectively.

2. Related Work

In this research, a 3D point cloud video is generated using 3D depth camera. Important treatments must be applied for the reduction of 3D point cloud data and the elimination of error noise distortion. Filtering 3D point cloud is mainly based on mathematical morphology, such as triangulation, statistical analysis etc.

Kammerl et al. (2012) [16] designed a spatial and temporal lossy compression of a 3D point cloud video stream. They have used Octree to compress point clouds by performing a spatial decomposition. In order to detect and remove temporal redundancy, the differences between consecutive point clouds are considered and compared using XOR operation. They have presented a technique for comparing the Octree data structure of their representation. Only the changes between the frames are encoded and sent to the decoder. This approach shows a strong compression ratio of 40x for controlled scenarios where the different frames have only few changes at coordinate precision of 9 mm while 14 at 1 mm.

A 3D point cloud video stream compression system is designed in [17]. The 3D point cloud is compressed using Octree and the compressed 3D video is streamed over TCP/IP network protocol. Their system have an average compression ratio of nearly 10.

Sultani and Ghani (2015) [18] presented a streaming system of 3D point cloud video live streamed over TCP/IP to a Linux and Android clients. In Linux system, the 3D point cloud video frames are processed using a color histogram-based filter and an octree compression. The color histogram-based filter is designed where the conditional statements are set according to the first frame. They achieved a compression ratio of about 11.

3. Statistical Outlier Filter and Octree Compression

Point clouds are examples of 3D video data. 3D point clouds are a point set representing the surface boundary of a 3D object. Using 3D point clouds reduces time and storage complexity, by avoiding time consuming of surface generation [19]. Kinect camera produces a point cloud of count of 640×480 (307,200 dots) at 30 fps [20]. Each point in a 3D colored kinect has 15 bytes (4 bytes/per vertex axis and 1 byte/ per RGB channel) [6,21]. Usually the data obtained with Kinect cannot be directly fed into the designed computer vision algorithms. The raw depth data contains many noises, and several pixels in the depth image may have no depth value since the multiple reflections, transparent objects or scattering in some areas (such as human hair and tissue). Those inaccurate or missing depth data (holes) should be processed in advanced prior to being used. Therefore, many systems use the Kinect begin with a preprocessing stage, which performs application-specific camera re-calibration and/or depth data filtering [22,23]. Outliers are an example of undesired noise and removing them out of the point cloud will make computations faster and also help to get more accurate values [21]. Noise problems can be removed using statistical methods.

A statistical method is carried out on the 3D point cloud data by measuring the distances between the points in a neighborhood area, and all points which are not considered "normal" will be trimmed out. Outlier removal filter is based on calculating the distribution of point to its neighbor's distances in the input point cloud. The mean is calculated for each point in the 3D point cloud with the corresponding K neighbors. An outlier is considered and removed when all points whose mean distances are outside an interval defined by the global distances mean and standard deviation. Statistical outlier filter is implemented within Point Cloud Library (PCL) [24] for more details see [25,26].

The octree is a hierarchical data structure like kd-tree useful for many applications such as searching, compression or downsampling. Each octree node or voxel has either eight sub-nodes (children) or no children. All points are encapsulated within cubic bounding box that represents the root node. When constructing the octree, if a node is occupied with points it will be set to non-empty node and filled with black color. Each node can be subdivided into 8 octants, until the desired level or size of nodes of the last level are reached. Each node configuration is effectively encoded into single

byte stream (8 bit for 8 octants). The correlation between previously encoded point cloud frame and the current frame is measured by the exploiting the difference between them calculated by using exclusive disjunction operation (XOR). PCL provides point cloud compression functionality, for more details see [17,27].

Octree compression technique depends on the point coordinate precision parameter which it is measured in mm [24]. According to PCL [17], the resolution quality of the compressed point cloud frame depends on the point coordinate precision value; if a medium resolution quality is required then the point precision parameter is set to 5 cubic millimeter resolution, while if a high resolution is required then the parameter is set to 1 cubic millimeter.

4. The Proposed Streaming System

This paper presents a system where a live 3D point cloud video is streamed over a WLAN to an authentic server using TCP/IP protocol, see Figure 1. The streaming action will start only after the server certificate is being authenticated by the client. The authentication process is done through the TLS handshake protocol. Certificate generation is done by creating a private/public key pair for the server and then a certificate signing request is created to be issued and signed by a certificate authority (CA). If the client checks that the server certificate is authentic, the streaming phase will begin. The generated 3D video is represented in 3D point cloud type where filtering and compression are applied to process the generated 3D point cloud frame.

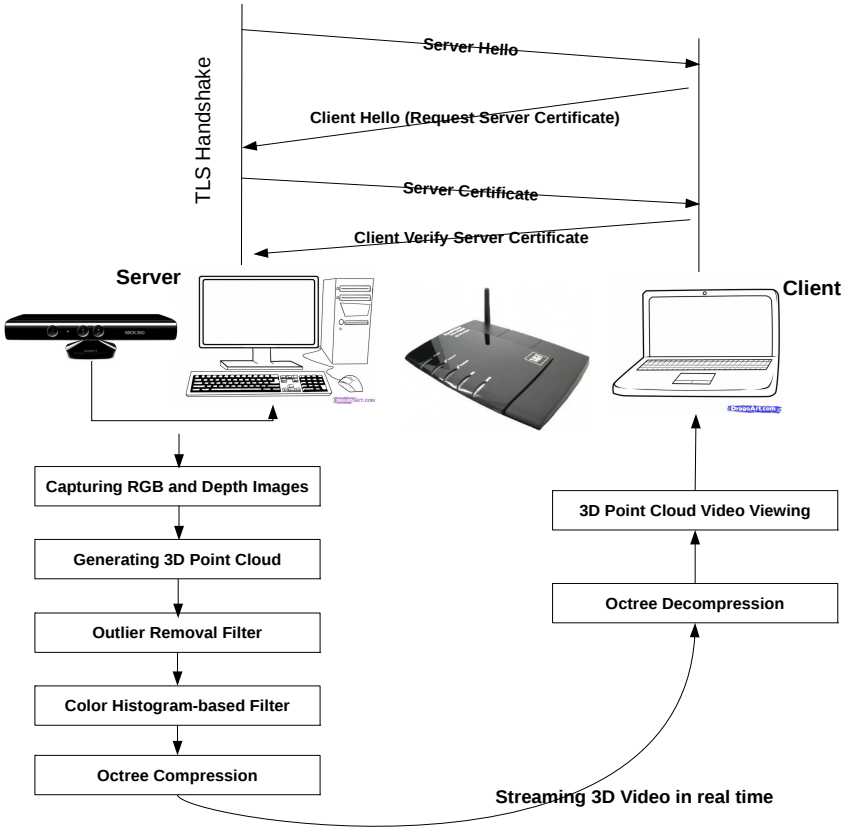


Figure 1. 3D Point Cloud Video Streaming System.

The generated 3D point cloud is processed by applying: (1) statistical outlier removal filter, (2) the designed color histogram-based conditional filter, and (3) performing an Octree compression. The statistical outlier removal filter and an octree compression are designed within PCL as explained

in Section 3. However the color histogram-based conditional filter (CHC) is designed as explained in Section 4.2. For the detailed block diagram, see Figure 2.

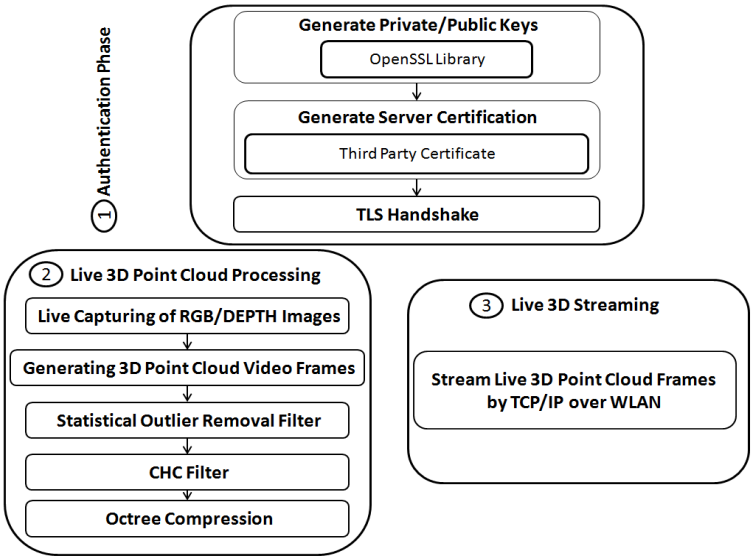


Figure 2. 3D Point Cloud Video Streaming System Block Diagram.

4.1. Authentication Phase—TLS and Server Certificate Generation

In the authentication phase the TLS protocol is used. TLS upper layer is called the Handshake protocol and it is responsible for authentication of the network parties and exchanging the keys. In this layer the DH protocol is used which is considered the best key exchanging protocol [28]. For authentication purposes, an X.509 certificate is used in the TLS handshake protocol to provide firm evidence to a second party (client) that proves the identity of the party (server) that holds the X.509 certificate and the assigned corresponding DH private key [29]. Certificate consists of server DH public key and some additional information inserted into the certificate when it is signed [30]. DH key exchange allows the client and server to share a secret key without explicitly sharing it in the handshake. The server's private key is used to create the certificate signing request (csr) which creates the server X.509 certificate [31].

Using OpenSSL library [30] the private/public key pairs are generated and a shared secret key is exchanged using DH protocol. When the client requests the server certificate and the server presents the certificate, the client will try to validate the server's CA with the client's cache of known and trusted CA certificates.

If the issuing CA is trusted, the client will verify that the server certificate is authentic and has not been changed and the certificate will be accepted as a proof of the server identity [29]. Finally the data streaming will begin after the certification acceptance.

4.2. Live 3D Point Cloud Processing—The Designed Color Histogram-Based Conditional Filter (CHC)

The captured 3D point cloud frames retrieved from the Kinect are processed using the outlier removal filter, then in order to reduce the color redundancy the designed CHC filter is used. CHC filter is designed to minimize redundant color information in each 3D point cloud video frame. Therefore the filter condition values are changed accordingly to each frame being streamed to the client.

A histogram is a graph counting how many pixels are at each color level between (0–255) [32]. Color histograms are three separate histograms, one for each R, G and B channels, and are used because they help to determine the correct exposure in an instant [33]. Using the concept of color

histogram, the color histogram-based conditional filter calculates the color histogram for each video frame to determine the dominating color values for each RGB channel. After the extraction of the dominating color values, the colors in the 3D point cloud frame will be filtered out except the values used in the condition statements which are the range of the dominating colors for each channel. The condition statements are designed based on the mean and the standard deviation of the occurrences, see Algorithm 1.

The CHC filter is based on retrieving the mean and the standard deviation of color occurrences for each color channel for each frame. The color reduction is based on removing the color values which have a very low present. The reduction is based on the following formula:

$$(m - (n \times stddev)) < occurrence < (m + (n \times stddev)) \quad (4)$$

where m is the mean of a color occurrences, $stddev$ is the standard deviation of a color occurrences and n is the standard deviation threshold which should be selected carefully according to a multiple test cases as explained in Section 5.

Algorithm 1 Color Histogram-Based Conditional Algorithm

procedure DEFINE FILTER CONDITIONS
Input: 3D Point Cloud Frame
Output: Filtered 3D Point Cloud Frame
Initialize: $full - vector$, $filtered - color$
    Define the standard deviation multiplier integer n
    for each captured Point Cloud Frame **do**
        Read the Red, Green and Blue colors and Save each one of them in a vector
        for each color vector in RGB **do**
            Calculate the color histogram
            For each color value in the histogram save it with its corresponding occurrence in a new vector $full - vector$
        end for
        Calculate the mean (m) and the standard deviation (stddev) for the occurrences only.
        for each color histogram vector **do**
            for each occurrence in the $full - vector$ **do**
                if $(m - (n \times stddev)) <$ occurrence $< (m + (n \times stddev))$ and occurrence > 1 **then**
                    keep the color of the corresponding occurrence in a new vector $filtered - color$
                end if
            end for
        end for
        for each point in the 3D point cloud **do**
            if The Point's color value has a match in the $filtered - color$ **then**
filtered-color saves the color values where their occurrences pass the condition statement
                Keep the point color
            else
                Set the point color value to zero
            end if
        end for
    end for
end procedure

4.3. Live 3D Video Streaming

In this paper the streaming system is implemented on a client server network infrastructure. The streaming of the 3D point cloud video is done over TCP/IP protocol using sockets (IP address and port number). Algorithms 2 and 3 are used to create server and client streaming system using socket programming.

Nagle algorithm is a mechanism included in Modern TCP implementations, which is used to prevent the transmission of small packets several time. However as stated in [34] quoted:"Nagle is not optimal for streams", therefore it has been disabled.

Algorithm 2 Server Video Streaming

1: **procedure** VIDEO STREAMING - SERVER SIDE
2: Initialize *io_service*
3: Initialize an Input/ Output stream *socketStream*
4: Define the TCP version and Port Number using *endpoint*
5: Set the created service to listen for a new a connection on the defined *endpoint*
6: Create a socket that will represent the connection to the client, and then wait for a connection.
7: Using input/output stream *socketStream*, wait for the connection to be established with the client using *acceptor.accept*
8: Disable Nagle algorithm by setting the *no_delay_option*
9: Copy the compressed point cloud frame into the created *socketstream* where the connection is created on it.
10: **end procedure**

For the client Algorithm 3 the first step is used to set the socket stream to read the connection on the specified port which is defined at the server side. The second and third steps are set to read the compressed 3D point cloud video frame that was received on the stream of the specified port and decode it. Finally, steps 4 and 5 are used to display the decoded 3D point cloud frame on the client viewer.

Algorithm 3 Client Video Streaming

1: **procedure** VIDEO STREAMING - CLIENT SIDE
2: Set the Input/ Output stream to listen to the desired port number specified by the server side
3: Read the stream that came from the specified server port
4: Decode the compressed 3D point cloud frame
5: Display the decoded point cloud frame on the client viewer
6: **end procedure**

5. Experimental Results

In this section, the implementation details of the proposed system in Figure 1 are explained. The 3D point cloud video frames are captured live through the Kinect and directly fed into the system. The server, laptop and Kinect were located in an office within Martin Luther University's Informatik Institute. The system uses two filters: outlier and CHC filters and an octree compression. The coordinate precision for the octree compression is set to 1 mm. Starting with the outlier filter parameters (number of neighborhood to search and the standard deviation multiplier); different values are used to select the best final output in terms of quality. The same is applied with the CHC filter standard deviation multiplier parameter, then using the selected parameters from both filters; test cases are constructed for the whole system. Using the final test cases, the system is evaluated in several terms (compression ratio, total bytes per point, PSNR, and SSIM). Network bandwidth requirement is captured before and after processing. Finally the system is compared to streaming system [17].

5.1. Statistical Outlier Filter Test Cases

Starting with the first filter, the statistical outlier removal filter works by using the mean and the standard deviation threshold parameters. In order to demonstrate the effect of the two parameters, a several test cases were designed to select the best values for this filter.

Figure 3a represents a table in a 3D point cloud where the data points are equal to 460400, this point cloud frame will be used to define the effect of the test cases. The reduction is the process of removing the outliers, which is clearly visible around the legs.

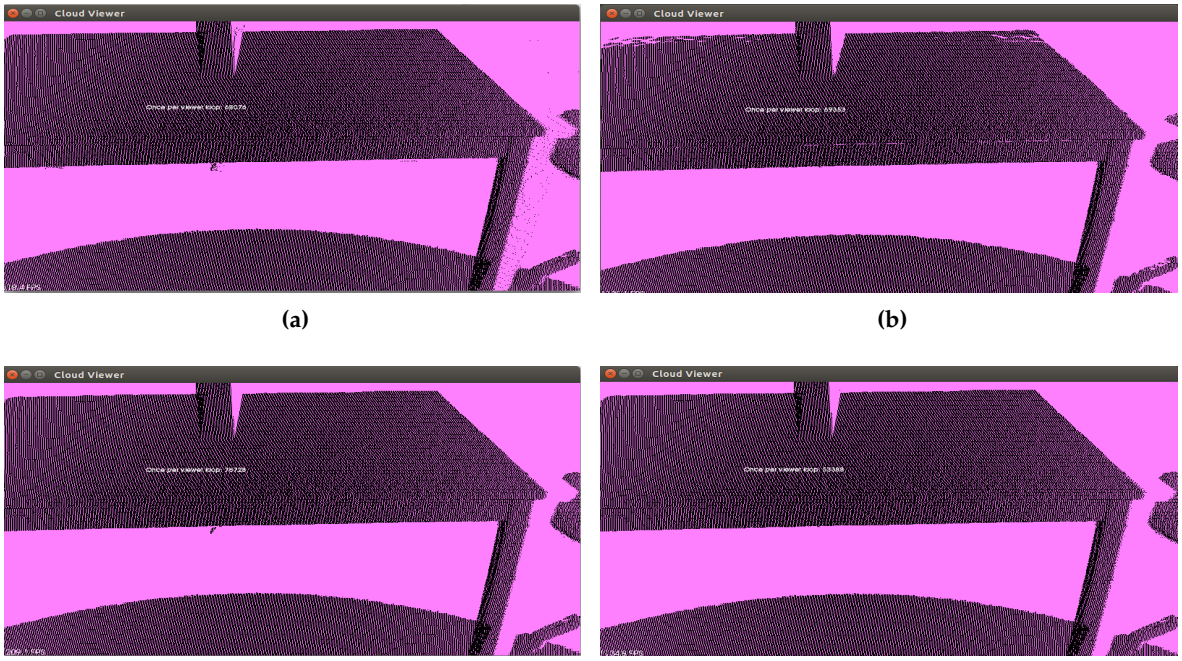


Figure 3. Effect of using Outlier Filter. (**a**) Original Table in 3D Point Cloud; (**b**) Outlier Filter Effect for Case 1; (**c**) Outlier Filter Effect for Case 8; (**d**) Outlier Filter Effect for Case 9.

Table 1 represents several test cases using different numbers of neighborhood search and standard deviation threshold with the corresponding data points reduction.

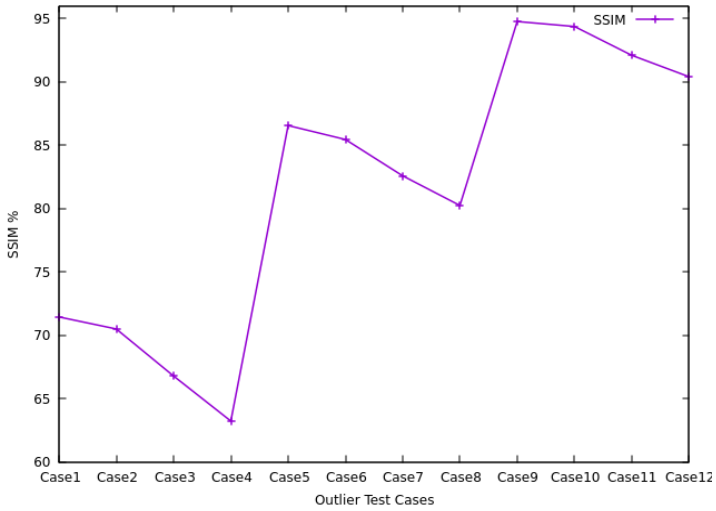
Table 1. Test Cases for the Outlier Filter Parameters.

Cases	No. of Neighbor Search	StdDev Multiplier	Data Points Reduction
1	60	1	450773
2	50	1	451410
3	30	1	452168
4	20	1	452099
5	60	2	456520
6	50	2	456519
7	30	2	456523
8	20	2	456466
9	60	3	457573
10	50	3	457560
11	30	3	457501
12	20	3	457460

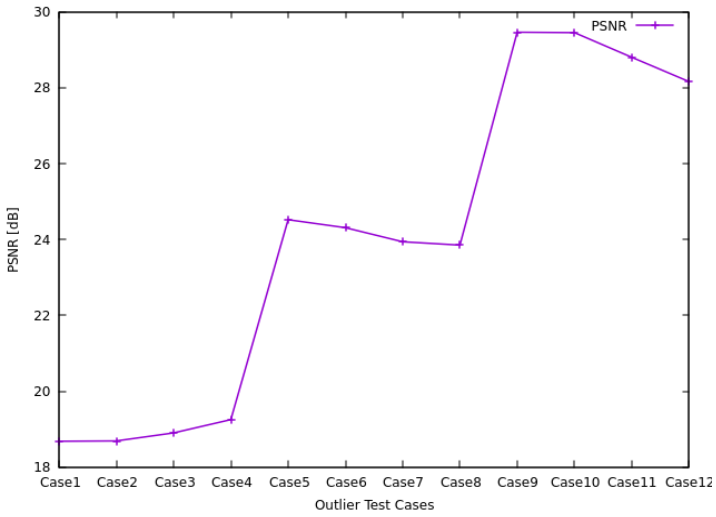
Figure 3b–d are the filter effect results for Cases 1, 8 and 9. As shown, the outlier filter successfully removed the outliers in all cases, but the main difference is that in some cases the filter removed a valid data point in the point cloud, as shown in Case 1 Figure 3b, therefore selecting the best parameters is not a straightforward task.

The selection of the best case depends on both the reduction ratio and the quality of the filtered frame. It is very essential that the best case does not remove legitimate (valid) point in the point cloud, at the same time it should reduce the number of outliers as possible.

Figure 4a is constructed to represent the test cases for the outlier filter and the corresponding SSIM. Figure 4b represents the PSNR values for each test case. The SSIM and the PSNR are calculated between the depth image of the raw point cloud scene and the depth image for each case. The reason behind comparing with depth images relies on the fact that the outlier filter removes the noise according to their depth values. Figure 5a–c represent the depth image for the raw point cloud frame, test Case 9 and test Case 11 respectively.



(a)



(b)

Figure 4. SSIM values and PSNR for Outlier Test Cases. (**a**) SSIM values for Outlier Test Cases; (**b**) PSNR for Outlier Test Cases.

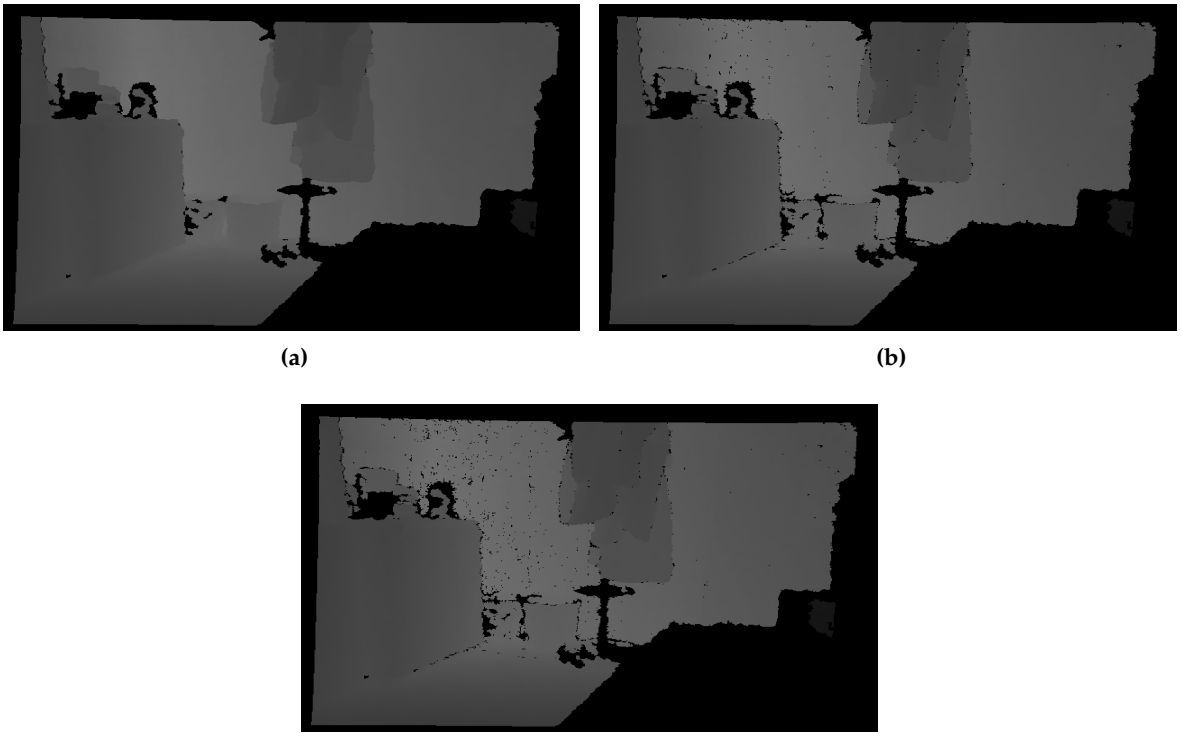


Figure 5. Effect of using Outlier Filter on Depth Images. (**a**) Depth Image for the Raw Point Cloud; (**b**) Depth Image for Test Case 9; (**c**) Depth Image for Test Case 11.

From Figure 4a it can be noticed that the best four Cases are 9,10,11 and 12, where the threshold multiplier of the standard deviation is set to 3. That means that the filter will produce better quality when the condition is not highly restricted as the case when the multiplier was either 1 or 2. Furthermore, the best quality among all cases is Case 9 where the filter is set to search within 60 points.

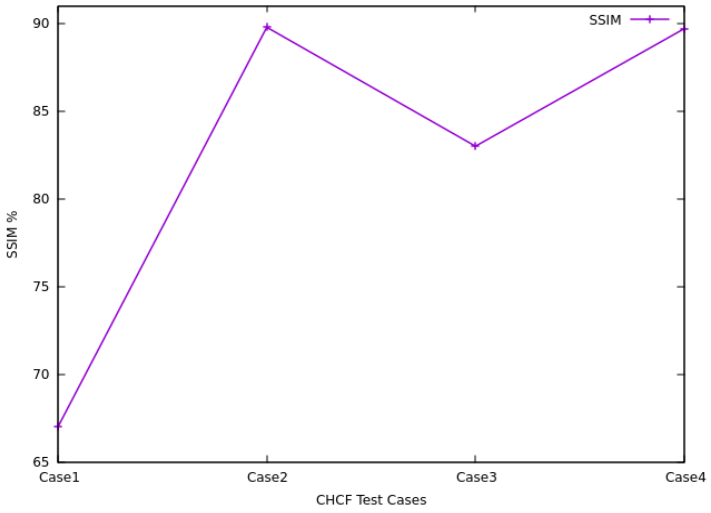
5.2. CHC Filter Test Cases

CHC filter depends on the standard deviation multiplier parameter. Selecting the best parameter value depends on the quality of the reduction of the 3D point cloud. Using this filter, the number of point cloud data is not reduced, though the undesired color will be set to zero. Therefore, the SSIM quality is the required measurement for this filter. Using the raw point cloud frame in Figure 6, Table 2 represents the test cases, and the corresponding SSIM and PSNR are shown in Figure 7a,b for each case.

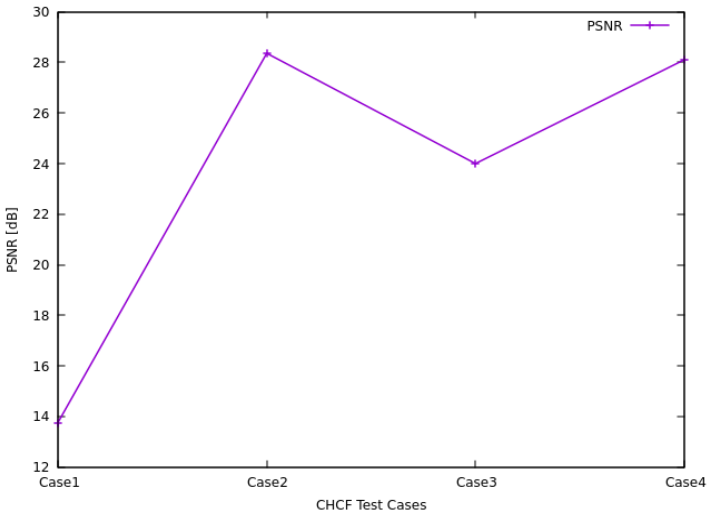
Figure 6. 3D Point Cloud Scene.

Table 2. CHC Filter Test Cases.

Test Case	Standard Deviation Multiplier
Case 1	2
Case 2	5
Case 3	10
Case 4	12



(a)



(b)

Figure 7. SSIM values and PSNR for CHC Test Cases. (**a**) SSIM values for CHC Test Cases; (**b**) PSNR for CHC Test Cases.

According to Figure 7a,b, the best value of the CHC filter parameter is 12 (Case 4), where the SSIM and PSNR are the highest among the other test cases.

5.3. Streaming System Test Cases

Using the selected best parameters from Figures 4a and 7a, Table 3 is constructed with the new test cases. The outlier filter, CHC filter and an octree compression are applied on the streamed live 3D point cloud video.

Table 3. Streaming System Test Cases.

Test Cases	Outlier No. of Neighbor	Outlier Std. Deviation Multiplier	CHC Std. Deviation Multiplier
Case 1	60	3	2
Case 2	60	3	5
Case 3	60	3	10
Case 4	60	3	12

The proposed streaming system was implemented on a server-client network infrastructure, see Table 4 for network type details and the server/client specifications.

Table 4. Network Type and Speed with Server/Client Specifications.

Network Type	WLAN IEEE 802.11g
Network Bandwidth	54 Mbits/s
Server Specification	Desktop Linux OS Intel(R)Core(TM)2 Quad CPU Q9650 3GHz and RAM 7.8 GByte
Client Specification	Laptop Linux Intel(R)Core(TM)i7-4702MQ 2.2GHz CPU and RAM 7.67 GByte

The actual network bandwidth was captured before streaming using IPERF which is a command line tool to measure network link bandwidth and quality [35]. The typical maximum network bandwidth for 802.11g WLAN connection is 54 Mbit/s [36], but as shown in Figure 8 the rate was about 21 Mbit/s, due to the link adaptation mechanism where it is implemented in most 802.11 devices [37].

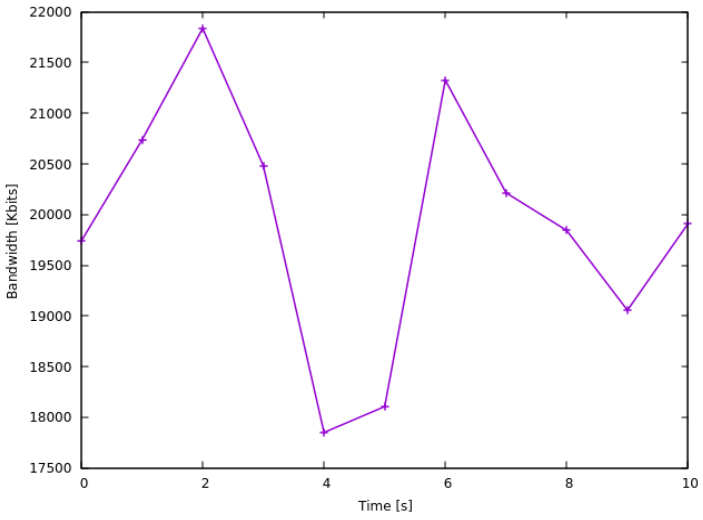


Figure 8. Existed Bandwidth between Client and Server.

Selecting several frames randomly, the compression ratio and total bytes per point were captured for the test cases in Table 3, see Figures 9 and 10.

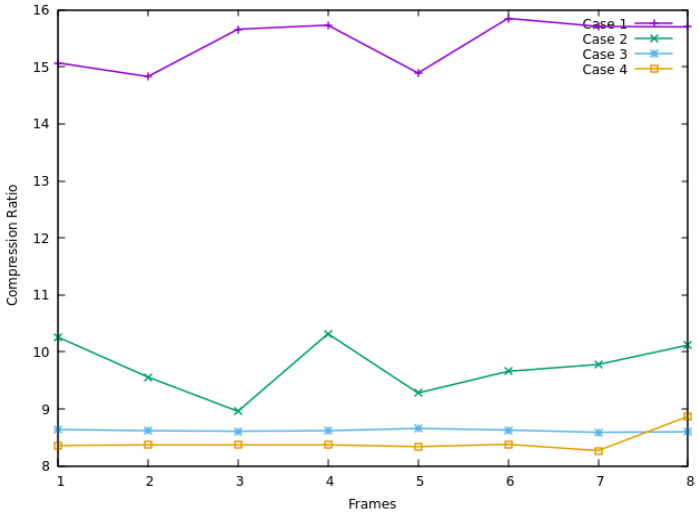


Figure 9. Compression Ratio for Streaming System Test Cases.

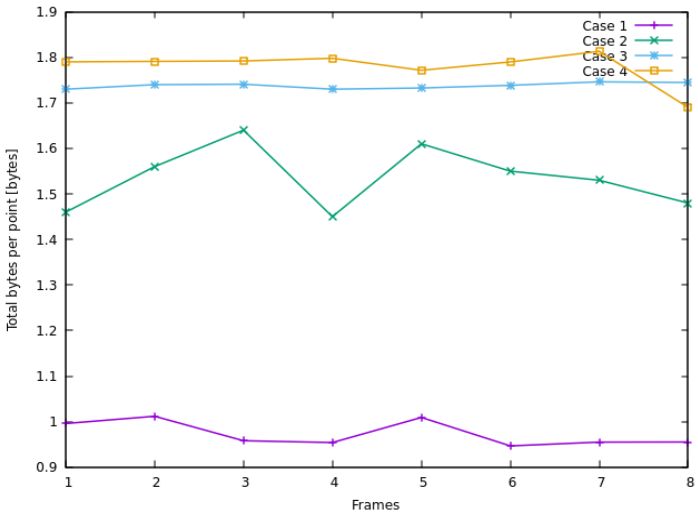


Figure 10. Total Bytes per Point for Streaming System Test Cases.

Case 1 has the best values in terms of compression ratio and total bytes per point. The reduction in Case 1 is obvious, since the CHC parameter is set to one, which is considered a difficult condition statement; only few data points will pass the condition. While in Case 4, the CHC parameter is more flexible and many numbers can meet the statement, though the compression ratio was the lowest.

Since the quality of the streaming video depends on the visual appearance therefore compression ratio and total bytes per point are not adequate for testing that, SSIM and PSNR are used.

From Figures 11 and 12 it can be noticed that Case 1 has the worst video quality in terms of SSIM which is equal to 0.5042 and PSNR of 8.6 dB, while it has the best compression ratio. Case 4 has the best SSIM quality value among all test cases where the SSIM is about 0.8594, however the PSNR value of 26.57 db is ranked the second-best.

Selecting a specific case as the best case depends on the application requirements. If the quality in terms of SSIM and PSNR are the only consideration, then Case 4 is the best case, but if a compromise is required between the reduction (compression ratio and total bytes per points) and the quality (SSIM and PSNR), then Case 2 is the best case. For the proposed streaming system requirements Case 2 is considered the best case.

Figure 13a represents the processed (filtered and compressed) live 3D point cloud video at the server. Figure 13b represents the streamed live 3D point cloud video decoded and rendered at the client. From the visual appearance perspective, both figures are nearly identical on the viewer's eyes, despite that the SSIM value is around 0.86.

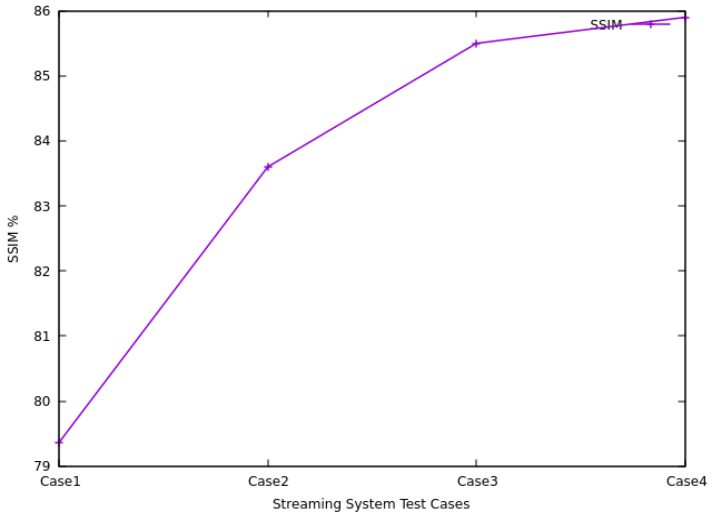


Figure 11. SSIM values for Streaming System Test Cases.

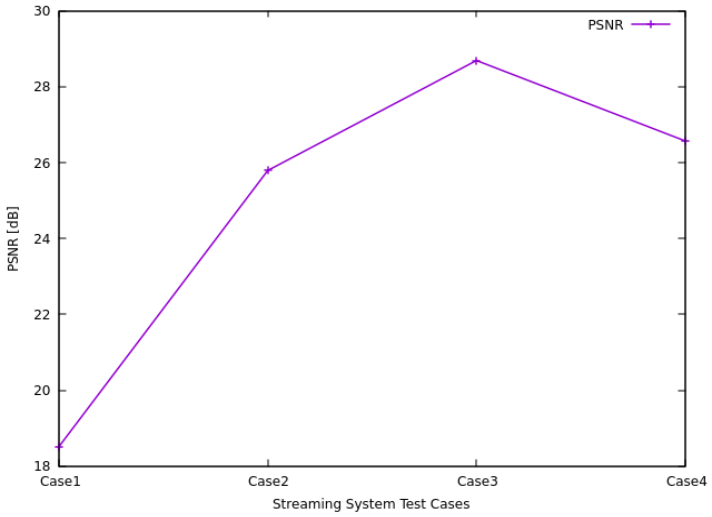


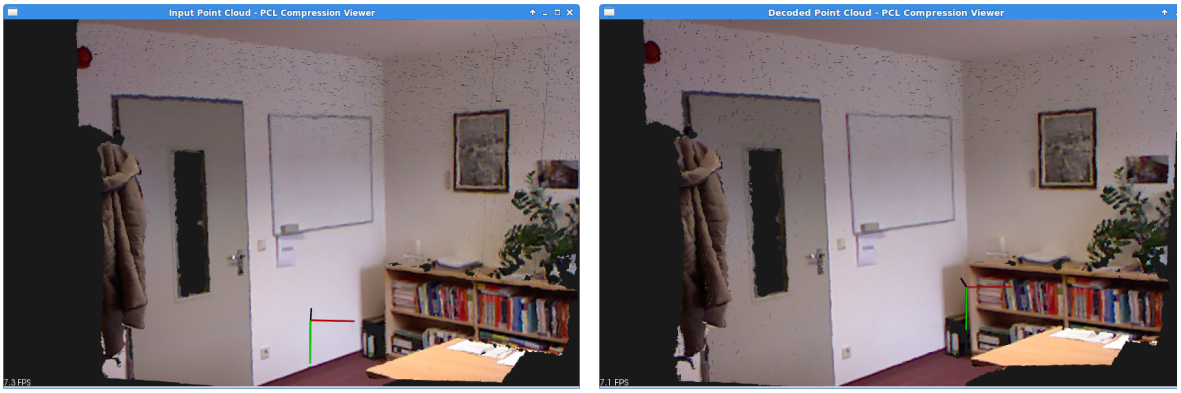
Figure 12. PSNR for Streaming System Test Cases.

Figure 14a,b display the captured RGB frame from the original raw Kinect video and the RGB frame from the decoded video at the receiver.

Capturing the decoded frame at the client shows a few black areas, which represent an unknown depth values. Looking closely and comparing Figure 14a,b, the black area at the ceiling is the room light, and since there are sunlight in the room, the Kinect fails to read the depth values.

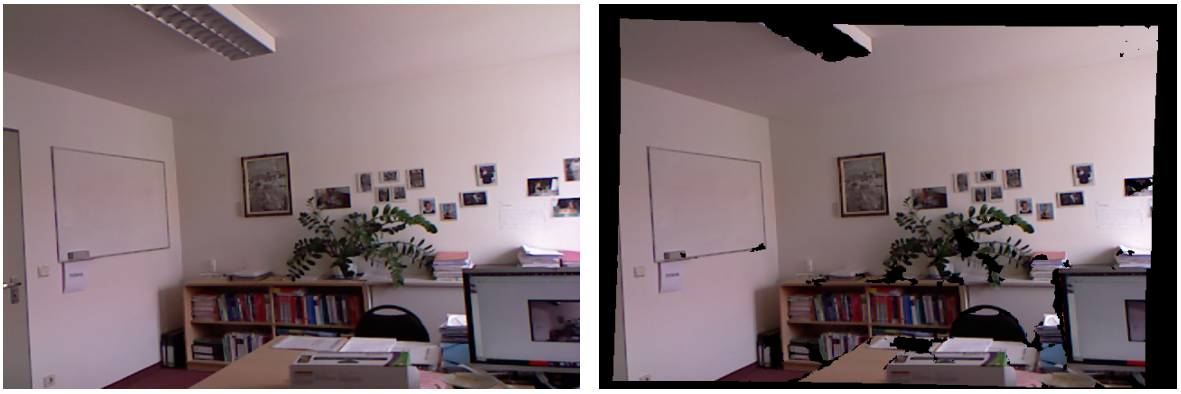
The network bandwidth requirements provide a perspective of how much the system consumes and do the processing stages reduce the bandwidth needed for streaming the live 3D point cloud video. Figure 15a,b prove that the 3D video streaming consumes much less than the actual existed bandwidth. The results show that the proposed streaming system is not being limited by the WLAN bandwidth. Furthermore according to Figure 15b the needed bandwidth is minimized to around 80% after applying filtering and compression to the captured 3D point cloud video stream.

Starting with the whole system as in Figure 1, two filters and an octree compression are used to process the live 3D point cloud video frames. Table 5 represents the processing time amount at the server (PS) and the client (PC) for different frames.



(a) **(b)**

Figure 13. Encoded vs. Decoded 3D Point Cloud Video Streaming. (**a**) Encoded 3D Point Cloud Video; (**b**) Decoded 3D Point Cloud Video.



(a) **(b)**

Figure 14. Original Frame vs. Decoded frame. (**a**) Original raw frame at the Server; (**b**) Decoded frame at the Client.

In Table 6 the processing time is also calculated for the streaming system but only the outlier filter and an octree compression are used to process the 3D point cloud video frame.

Table 7 shows the processing time at the server and the client for the streaming system where only CHC filter and an octree compression are used.

From Tables 5–7 it is clearly noticed that the whole system requires more time to process the 3D point cloud frame since two filters and an octree compression are used, however using only CHC filter with an octree compression requires less time than using outlier and an octree compression, therefore the outlier filter is responsible of consuming more time in processing.

As mentioned in the related work, streaming system in [17] uses only the octree compression and this system is compared to the streaming system designed in this paper. Figure 16a,b are generated to capture the compression ratio and total bytes per point respectively. While the quality SSIM was 0.6843 and PSNR 18.02 dB. The quality is around 0.70 since the noise is not removed from the point cloud frame.

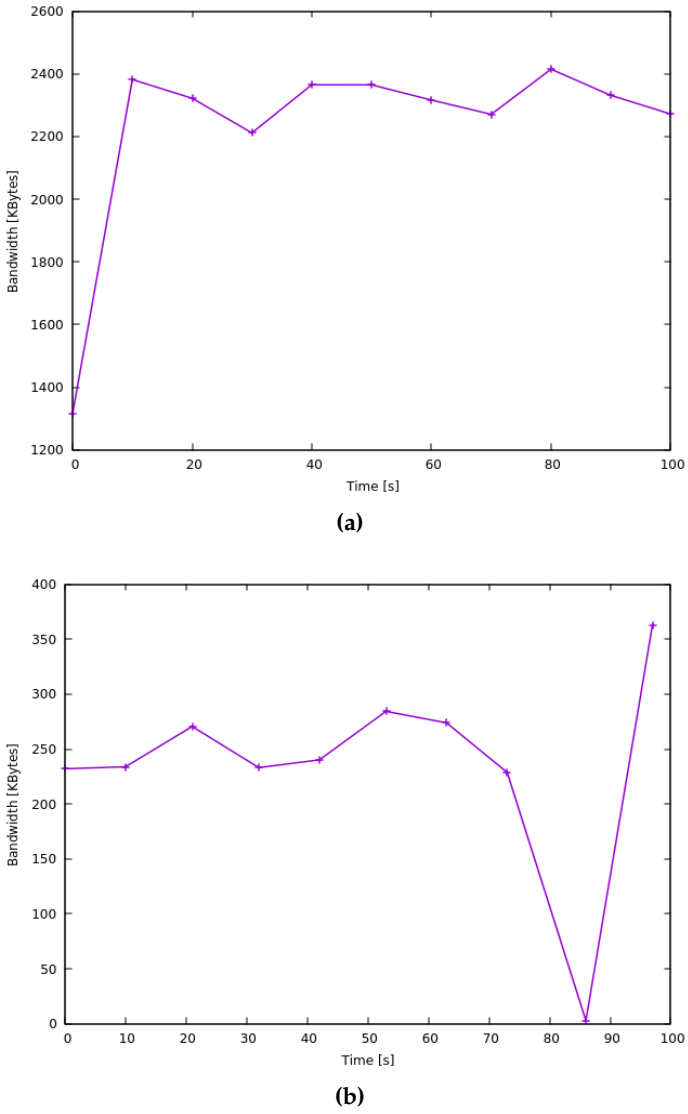


Figure 15. Streaming Bandwidth between Server and Client. (**a**) Bandwidth before Filtering and Compressing; (**b**) Bandwidth after Filtering and Compressing.

Table 5. Processing Time for the Whole System.

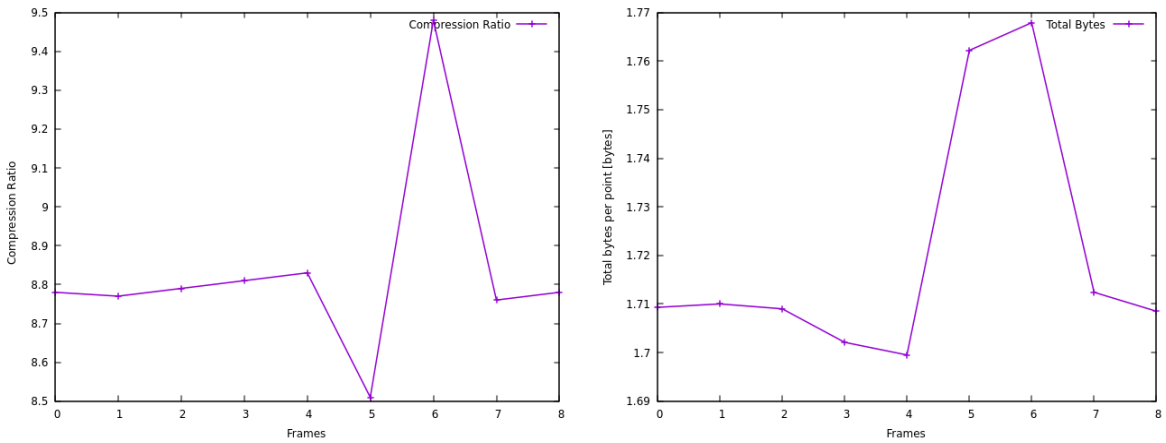
Video Frames	Processing at Server (PS)	Processing at Client (PC)
1	4446.11 ms	33.4 ms
2	4457.77 ms	31.553 ms
3	4370.73 ms	29.962 ms
4	4296.03 ms	28.73 ms
5	4484.38 ms	30.928 ms
6	4329.12 ms	30.33 ms
7	4359.79 ms	34.702 ms
8	4363.15 ms	28.774 ms
9	4343.36 ms	28.623 ms
10	4363.69 ms	38.095 ms

Table 6. Processing Time for Outlier Filter and Octree Compression.

Video Frames	Processing at Server (PS)	Processing at Client (PC)
1	3915.65 ms	32.542 ms
2	4106.22 ms	28.637 ms
3	3743.4 ms	37.06 ms
4	3754.86 ms	30.94 ms
5	3883.92 ms	30.058 ms
6	3772.1 ms	40.143 ms
7	3786.78 ms	36.17 ms
8	3811.39 ms	31.578 ms
9	3774.54 ms	29.793 ms
10	3803.3 ms	28.61 ms

Table 7. Processing Time for CHC Filter and Octree Compression.

Video Frames	Processing at Server (PS)	Processing at Client (PC)
1	1650.13 ms	34.136 ms
2	1673.11 ms	36.14 ms
3	1641.45 ms	40.2 ms
4	1673.61 ms	37.181 ms
5	1582.93 ms	33.404 ms
6	1575.78 ms	36.61 ms
7	1632.06 ms	31.696 ms
8	1626.17 ms	36.906 ms
9	1659.34 ms	32.525 ms
10	1616.04 ms	39.893 ms



(a) **(b)**

Figure 16. Compression Ratio and Total Bytes for the Streaming System in [17]. (**a**) Compression Ratio; (**b**) Total Bytes.

Table 8 represents the processing time for the streaming system in [17].

Table 8. Processing Time for the Streaming System in [17].

Video Frames	Processing at Server (PS)	Processing at Client (PC)
1	1211.84 ms	34.84 ms
2	913.842 ms	30.001 ms
3	1046.93 ms	31.749 ms
4	940.226 ms	30.048 ms
5	926.875 ms	33.837 ms
6	1009.99 ms	28.732 ms
7	914.631 ms	33.043 ms
8	937.886 ms	31.624 ms
9	1010.81 ms	29.143 ms
10	970.159 ms	28.109 ms

6. Conclusion and Discussion

The delivery of a live 3D video to individual users remains a highly challenging problem due to the huge amount of transmitted data, diverse network characteristic and user expectations [38].

This paper presents a live 3D video streaming system that streams a 3D point cloud generated from Microsoft Kinect streamed from an authentic server.

Diffie-Hellman key exchange protocol was designed to solve the issue of securing the communication channel between pairs by exchanging secret shared symmetric key.

Authentication is necessary because how peer Bob would know that it is going to exchange keys with peer Alice? Peer Bob could be exchanging with an eavesdropper who spoofed peer Alice's public key. However, using authentication in form of digital certificates has solved this problem. So using Diffie-Hellman with authentication protocols is more secure [39].

3D point cloud video frames were processed using statistical outlier removal filter, CHC filter and an octree-based compression. Results demonstrate that using filters with compression have increased the compression ratio and decreased the bytes needed to represent each point in the point cloud frame. Number of neighbors and standard deviation multiplier of the outlier removal filter have a significant impact on the streaming quality. In addition, the standard deviation multiplier of CHC filter also affects the color quality of the decoded frame. Therefore, the system has been tested using several test cases to capture the best parameters permutation.

With respect to the paper research objective on reducing the point cloud redundancies on both spatial and temporal level, the following conclusions are derived. First, using outlier removal filter is a mandatory since the Kinect depth camera has a lot of noises where some depth information are missing and therefore the harmony of the generated point cloud is affected. However, using other procedure other than using the mean and standard deviation could provide better noise selection.

Second, the novelty in the designed CHC filter is that the filter conditional statements are adaptive according to the captured 3D point cloud video frame, where the color reduction will differ from frame to frame according to the mean and the standard deviation of the color occurrences in each frame for each color channel. The results show that selecting an optimal standard deviation multiplier threshold will reduce the number of colors while preserving the visual appearance of the video frame. Regarding the network bandwidth, using filters and compression have significantly reduced the bandwidth requirement of a live 3D video streaming system.

Author Contributions: All authors contributed in the overall idea of this manuscript. Rana Al-Tuma suggested the main idea, streaming system architecture and specified the quality measurements. Sandro Wefel suggested the authentication phase, network measurements and provided the network environment. Zainab Sultani designed the proposed streaming system, algorithm and designed/performed the experiments. Zainab Sultani wrote the manuscript. All authors revised the manuscript. All authors have read and approved the final manuscript.

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

References

1. Klaue, J.; Rathke, B.; Wolisz, A. EvalVid—A Framework for Video Transmission and Quality Evaluation. In Proceedings of the 13th International Conference on Modelling Techniques and Tools for Computer Performance Evaluation, Urbana, IL, USA, 2–5 September 2003.
2. Singh, G. Secure Video Conferencing for Web Based Security Surveillance System. Master's Thesis, Indian Institute of Technology, Kanpur, India, 2006.
3. Eisert, P. 3-D Video Conferencing: Challenges, Concepts, and Implementations. In Proceedings of the Visual Communications and Image Processing 2003, Lugano, Switzerland, 8 July 2003; pp. 69–79.
4. Su, G.M.; Lai, Y.C.; Kwasinski, A.; Wang, H. *3D Visual Communications*; Wiley: West Sussex, UK, 2012.
5. Dreier, T. Livestream Explains the Challenges of Streaming Live Video. Available online: http://www.streamingmediaglobal.com/Articles/Editorial/Featured-Articles/Livestream-Explains-the-Challenges-of-Streaming-Live-Video-88010.aspx (accessed on 11 April 2015).
6. Bamber, L. Intel Realiser Technology and the Point Cloud, 2015. Available online: https://software.intel.com/en-us/articles/intel-realsense-technology-and-the-point-cloud (accessed on 27 April 2015).
7. Liu, Y.; Ci, S.; Tang, H.; Ye, Y. Application-adapted mobile 3D video coding and streaming—A survey. *3D Res.* **2012**, *3*, 1–6.
8. Xin, B.; Wang, R.; Wang, Z.; Wang, W.; Gu, C.; Zheng, Q.; Gao, W. AVS 3D Video Streaming System Over Internet. In Proceedings of the 2012 IEEE International Conference on Signal Processing, Communication and Computing (ICSPCC), Hong Kong, 12–15 August 2012; pp. 286–289.
9. Todorov, D. *Mechanics of User Identification and Authentication: Fundamentals of Identity Management*; Auerbach Publications: Boca Raton, FL, USA, 2007.
10. Lott, D. *Improving Customer Authentication*; Federal Reserve Bank of Atlanta: Atlanta, GA, USA, 2015.
11. Stallings, W. *Cryptography and Network Security Principles and Practice*; Pearson Prentice Hall: New York, NY, USA, 2006.
12. Wang, Z.; Bovik, A.; Sheikh, H.; Simoncelli, E. Image Quality Assessment: From Error Visibility to Structural Similarity. *IEEE Trans. Image Process.* **2004**, *13*, 600–612.
13. Thomos, N.; Boulgouris, V.; Strintzis, M.G. Optimized Transmission of JPEG2000 Streams Over Wireless Channels. *IEEE Trans. Image Process.* **2006**, *15*, 54–67.
14. Wang, S.; Ye, L.; Zhong, W.; Zhang, Q. Image Compression Based on Hierarchical Clustering Vector Quantization. In *Multimedia and Signal Processing*, Proceedings of the Second International Conference, CMSP 2012, Shanghai, China, 7–9 December 2012; Wang, F.L., Lei, J., Lau, R.W.H., Zhang, J., Eds.; Communications in Computer and Information Science; Springer: Berlin/Heidelberg, Germany, 2012; Volume 346, pp. 120–128.
15. Azim, M.; Jiang, X. *Wireless Sensor Multimedia Networks Architecture, Protocols, and Applications*; CRC Press: Boca Raton, FL, USA; London, UK; New York, NY, USA, 2016.
16. Kammerl, J.; Blodow, N.; Rusu, R.B.; Gedikli, S.; Beetz, B.; Steinbach, E. Real-time Compression of Point Cloud Streams. In Proceedings of the 2012 IEEE International Conference on Robotics and Automation (ICRA), Saint Paul, MN, USA, 14–18 May 2012; pp. 778–785.
17. Point Cloud Library (PCL). Octree Compression. Available online: http://pointclouds.org/documentation/tutorials/compression.php (accessed on 2 July 2014).
18. Sultani, Z.; Ghani, R. Kinect 3D Point Cloud Live Video Streaming. *Procedia Comput. Sci.* **2015**, *65*, 125–132.
19. Linsen, L. *Point Cloud Representation*; Technical Report Number 2001-3; Fakultät für Informatik, Universität Karlsruhe: Karlsruhe, Germany, 2001.
20. Kinect Sensor, Microsoft. Available online: https://msdn.microsoft.com/en-us/library/hh438998.aspx (accessed on 10 May 2014).
21. PCL/OpenNI tutorial. Available online: http://robotica.unileon.es/mediawiki/index.php/PCL/OpenNI_tutorial_2:_Cloud_processing_(basic) (accessed on 10 July 2014) .
22. Han, J.; Shao, L.; Xu, D.; Shotton, J. Enhanced Computer Vision with Microsoft Kinect Sensor: A Review. *IEEE Trans. Cybern.* **2013**, *43*, 1318–1334.

23. Alhwarin, A.; Ferrein, F.; Scholl, I. IR Stereo Kinect: Improving Depth Images by Combining Structured Light with IR Stereo. In *PRICAI 2014: Trends in Artificial Intelligence*, Proceedings of the 13th Pacific Rim International Conference on Artificial Intelligence, Gold Coast, QLD, Australia, 1–5 December 2014; Pham, D.N., Park, S.B., Eds.; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2014; Volume 8862, pp 409–421.
24. PCL—Point Cloud Library (PCL). Available online: http://pointclouds.org (accessed on 2 July 2014).
25. Mishra, C.; Khan, Z.A. Development and Evaluation of a Kinect based Bin-Picking System. Master's Thesis, Malaedalen University, Västerås, Sweden, April 2015.
26. Point Cloud Library (PCL). Removing outliers using a Statistical Outlier Removal filter. Available online: http://pointclouds.org/documentation/tutorials/statistical_outlier.php (accessed on 3 July 2014).
27. Zhang, L. TraumaBot–3D body reconstruction and recognition. Available online: http://traumabot.blogspot.de/2013/06/octree-based-point-cloudstream.html (accessed on 15 June 2014).
28. Carts, D. *A review of the Diffie-Hellman Algorithm and its Use in Secure Internet Protocols*; 5 November 2001; SANS Institute: Bethesda, MD, USA.
29. Microsoft Tecknet. How TLS/SSL Works. Available online: https://technet.microsoft.com/en-us/library/cc783349(v=ws10).aspx (accessed on 11 May 2015).
30. Anicas, M. OpenSSL Essentials: Working with SSL Certificates, Private Keys and CSRs. Available online: https://www.digitalocean.com/community/tutorials/openssl-essentialsworking-with-ssl-certificates-private-keys -and-csrs (accessed on 12 May 2015).
31. Grigorik, I. Transport Layer Security (TLS). In *High Performance Browser Networking*; O'Reilly Media: Sebastopol, CA, USA, 2013.
32. Rockwel, K. How to Use Histograms. Available online: http://www.kenrockwell.com/tech/histograms.htm (accessed on 1 September 2014).
33. Rockwel, K. How to Use Color Histograms. Available online: http://www.kenrockwell.com/tech/yrgb.htm (accessed on 1 September 2014).
34. Baus, C. TCP CORK: More than you ever wanted to know. Available online: http://baus.net/on-tcp_cork (accessed on 5 June 2015).
35. Open Maniak. IPERF—The Easy Tutorial. Available online: http://openmaniak.com/iperf.php (accessed on 3 May 2015).
36. Varshney, U. The status and future of 802.11-based WLANs. *Computer* **2003**, *36*, 102–105.
37. Qiao, D.; Choi, S. Fast-Responsive Link Adaptation for IEEE 802.11 WLANs. In Proceedings of the IEEE International Conference on Communications, ICC 2005, Seoul, Korea, 16–20 May 2005; pp. 3583–3588.
38. Politis, I.; Lykourgiotis, A.; Dagiuklas, T. A Framework for QoE-Aware 3D Video Streaming Optimisation over Wireless Networks. *Mob. Inf. Syst.* **2016**, *2016*, doi:10.1155/2016/4913216.
39. Internet-Computer-Security.com. Diffie Hellman Encryption Tutorial–Cryptography on Public keys. Available online: http://www.internet-computer-security.com/VPN-Guide/Diffie-Hellman.html (accessed on 10 June 2015).