International Journal of Trend in Scientific Research and Development (IJTSRD)
International Open Access Journal | www.ijtsrd.com
ISSN No: 2456 - 6470 | Volume - 2 | Issue – 6 | Sep – Oct 2018

3D Point Cloud Storage Options: A Comparison with a Kinect Data

Erdal Erdal
Assistant Professor, Department of Computer Engineering,
Kirikkale University, Kirikkale, Turkey

ABSTRACT
In recent years, the concept of 3D point cloud, which is a new approach on this subject, has entered the literature. This concept, which is based on the representation of objects as 3D point for processing, analysis or promotion operations on computer, is preferred with its flexible structure, simplicity, strong representation ability and easy availability. Considering the studies that can be done in this area, studies have been carried out on the development of data collection devices in this area. The Kinect camera, developed by Microsoft, is one of the most preferred devices in this field. The aim of this study is to apply compression algorithms to the data obtained from Kinect applications and to compare the obtained results. In this way, the data obtained by compression can be more easily transmitted and processed. The results obtained also offer suggestions for the applications to be developed after the study.

Keyword: *3D Point Cloud, Kinect, Compression Algorithms, Bzip2*

I. INTRODUCTION
The representation of real-life objects on the computer is a problem that has been studied over many years. In recent years, the concept of 3D point cloud, which is a new approach on this subject, has entered the literature [1-4].This concept, which is based on the representation of objects as 3D point for processing, analysis or promotion operations on computer, is preferred with its flexible structure, simplicity, strong representation ability and easy availability. In the literature, it has become increasingly widespread in many different research areas, including object reconstruction and object recognition problems. Instead of the triangular meshes used before the 3D point cloud, there is no need to maintain or store point cloud topological consistency or polygonal mesh connection. However, the capture process of the 3D point cloud of an object, a human or a part of human brings with it challenges. First problem is capturing the 3D point of a specific object. Thanks to the improved hardware technology, 3D information has become easier to capture. Second problem is storing the captured data. The data from which the data is received provides a large amount of data for the object. Studies in the field of storage and transmission of these 3D data are required. Nowadays, both problem areas are continuing to be studied in the literature.

Considering the studies that can be done in this area, studies have been carried out on the development of data collection devices in this area.3D data collection devices such as Tango, Intel Real Sense and Microsoft Kinect have been developed. With these devices, access to this technology has also been progressed. Nowadays, these devices have become mobile and fit to a compact size, so that they are used daily. Depth sensors have been used by many applications in pattern recognition and computer vision such as 3D facial recognition, 3D object recognition, 3D object recognition and 3D modeling [5-8].

The Kinect camera, developed by Microsoft, is one of the most preferred devices in this field. Originally developed for use in games, the Kinect camera has attracted the attention of researchers with its affordable price and the data it provides.

As shown in Figure 1, Kinect has an RGB camera on the camera, multiple array microphones, and a depth sensor with full-body 3D motion capture capability with face and voice recognition capabilities [9]. There are currently two different Kinect camera versions available for users and developers on the market. The

differences between the two cameras are shown in Table 1 [10, 11].

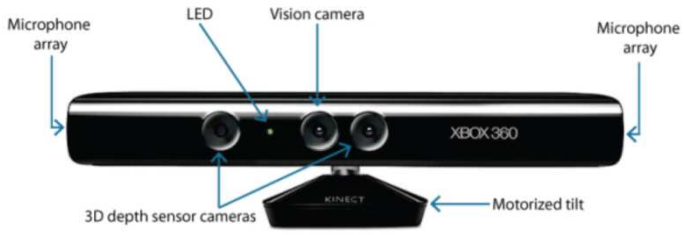

Figure 1: Kinect Camera

Feature	Kinect for Windows v1	Kinect for Windows v2
Color Camera	640 x 480 @30 fps	1920 x 1080 @30 fps
Depth Camera	320 x 240	512 x 424
Max Depth Distance	~4.5 M	8 M
Min Depth Distance	40 cm in near mode	50 cm
Depth Horizontal Field of View	57 degrees	70 degrees
Depth Vertical Field of View	43 degrees	60 degrees
Tilt Motor	yes	no
Skeleton Joints Defined	20 joints	25 joints
Full Skeletons Tracked	2	6
USB Standard	2.0	3.0
Supported OS	Win 7, Win 8	Win 8

Table 1: Kinect Version Feature Comparison

As shown in Table 1, Kinect camera features are being developed day by day. As shown in Table 1, Kinect camera connects the computers via the USB port. In addition, a software development kit (SDK) has been also developed by Microsoft. Thanks to this SDK, all data from the camera is easily transferred to the computer. In addition, Kinect camera-specific development of the skeletal and joint information can be obtained with this SDK. Also, this SDK supports C # language and can be controlled with the desired application. Figure 2 shows the C # code that receives all skeleton data.

```
private void skeletonFrameReady(object sender,
SkeletonFrameReadyEventArgs e)
{
    // Open the Skeleton frame
    using (SkeletonFrame skeletonFrame = e.OpenSkeletonFrame())
    {
        // check that a frame is available
        if (skeletonFrame != null && skeleton != null)
        {
            // get the skeletal information in this frame
            skeletonFrame.CopySkeletonDataTo(skeleton);
        }
    }
    return;
}
```

Figure 2: Skeleton Data in C# [12]

```
foreach (Joint j in s.Joints)
{
    if (j != null
    && j.TrackingState == JointTrackingState.Tracked
    && j.JointType == JointType.Head)
    {
        string js = "Joint: &1 ( &2, &3, &4 )\n";
        js = js.Replace("&1", j.JointType.ToString());
        js = js.Replace("&2", j.Position.X.ToString());
        js = js.Replace("&3", j.Position.Y.ToString());
        js = js.Replace("&4", j.Position.Z.ToString());
        output += js;
    }
}
```

Figure 3: The code that follows the head [12]

As shown in Table 1, Kinect for windows v1 camera provides X, Y and Z coordinates of the 20 joints belonging to the two bodies. This data is transmitted at 60 frames per second. The 20 joint points from the Kinect camera are shown in Figure 4[9, 13].

With all these advantages, features and low cost, it is predicted that Kinect sensors will be preferred by researchers for a long time. When considering the Kinect 1 sensor for Windows, 20 joints of two skeletons, X, Y and Z coordinate information of each insert are taken 30 frames per second. The Kinect sensor produces 240,000 data and 270,000 dots per second, depending on the stage complexity. Therefore, dynamic point clouds contain about 7.5 million points per second, which is a large amount of data [14].

The aim of this study is to apply compression algorithms to the data obtained from Kinect applications and to compare the obtained results.In this way, the data obtained by compression can be more easily transmitted and processed. The results obtained also offer suggestions for the applications to be developed after the study.

Figure 4: Kinect traceable joint points

In this study, it is aimed to compress 3D cloud points taken from Kinect camera. This large data obtained with compression can be transmitted, recorded and managed easily. In order to ensure consistency in the data obtained, all data from the camera were recorded in the XML file via XML tags. This file is then compressed with compression algorithms and the difference between them is examined. In online or offline applications where the Kinect camera or 3D cloud points are used, it is recommended to ensure performance and stability by using this comparison.

II. METHOD

As mentioned the data received are taken over the SDK without any labels in a C# class. Therefore, there is a need for a label to compress or store data. The Extensible Markup Language (XML) seems to be appropriate for storing, transmitting, compressing, and managing data from the Kinect camera. Another reason for choosing XML file format is suitable to cross platform and its platform independent structure. Therefore, all data from the Kinect camera is written to the XML file. It was created with the attention to XML tags so that there is no confusion and for consistent storage of data. The code block that created the XML file was added to the region from which the data was received. In this way, all the data from the SDK is transferred to the XML file without any difference or missing in the data. The code block that generates the XML file is shown in Figure 5.

```
foreach (Joint jn in skel.Joints)
{
    JointOrder++;
    writer.WriteStartElement("Data");
    writer.WriteElementString("TraceId", skel.TrackingId.ToString());
    writer.WriteElementString("FrameOrder", order.ToString());
    writer.WriteElementString("JointOrder", JointOrder.ToString());
    writer.WriteElementString("JointType", jn.JointType.ToString());
    writer.WriteElementString("X", jn.Position.X.ToString());
    writer.WriteElementString("Y", jn.Position.Y.ToString());
    writer.WriteElementString("Z", jn.Position.Z.ToString());
    writer.WriteEndElement();
}
```

Figure 5: XML file generator

There are two types of compression algorithms in the literature. Lossless compression algorithm and lossy compression algorithms are available. According to the application where Kinect camera is used, lossy or lossless compression algorithms may be preferred. There is no tolerance for any data loss in lossless compression algorithms. After decompressing the compressed file, the original file is obtained without any loss of data. However, the compression ratio obtained in lossless compression algorithms is less than lossy compression algorithms. Lossy compression algorithms are preferred in applications where some data loss can be tolerated. Although the compression ratio is higher, the original file cannot be obtained after decompressing the compressed file. The scope of this study will be a comparison of the lossless compression algorithms in which any data loss cannot be tolerated.

There are five lossless compression algorithms included in the study. These are Bzip2, Xz, C# framework built-in compression method (gzip), Zip and 7-zip (7z).First of all, these lossless compression algorithms are examined.

A. Bzip2

The Bzip2 file compression algorithm was launched in 1996[15]. This free compression algorithm has been used for more than twenty years. The basis of the Bzip2 compression algorithm is the Burrows-Wheeler algorithm. The algorithm can compress files but cannot perform archiving process. The .bz2 extension is added to the end of the files compressed with this algorithm. It can be downloaded and used from the internet without charge.Bzip2 only has compression and cannot archive. Therefore, it can compress a single file at the same time. Despite this disadvantage, it is suitable for the scope of this study [15].

B. Xz

Xz lossless compression algorithm and a file format[16]. It uses LZMA / LZMA2 compression algorithms. Xz is a new compression algorithm similar to gzip and bzip2. This algorithm is general purpose and compresses data via command line. It supports various formats to compress files or decompress the file. Although the compression speed is not very good between compression algorithms, the Xz lossless compression algorithm offers a good compression ratio. However, it is not widely preferred and used. The .xz extension is added to the end of the compressed file with the Xz compression algorithm[16].

C. C# built-in compression (gzip)

The C# programming language is an object-oriented programming language. In this programming language, all operations are done with classes[17]. It is possible to compress data using the GZip Stream class in C # framework. A small block of code written using this class and used in this study is as shown in Figure 6.

```
public static byte[] Compress(byte[] raw)
{
    using (MemoryStream memory = new MemoryStream())
    {
        using (GZipStream gzip = new GZipStream(memory,
            CompressionMode.Compress, true))
        {
            gzip.Write(raw, 0, raw.Length);
        }
        return memory.ToArray();
    }
}
```

Figure 6: C# framework compression

D. Zip

ZIP format is the lossless data compression and archive format applied by Phil Katz in 1989 for the

first time[18]. The ZIP file format and its features are published under the public domain. Developed format has long a success. Even, the term "zip" is used instead of the compressed archive term in computer science. The ZIP extension is a popular file compression and archiving format commonly used operating systems such as Microsoft. Zip is a phenomenon that is accepted by people and archive administrators, so it is preferable for files to be dispatched to avoid problems during the unzip phase. This is one of the main reasons why zip format is included in the study.

E. 7-zip

7-zip (7z) is a free and open source archive and data compression software[19]. It released in December 2018[20]. 7z format uses LZMA2 and LZMA compression algorithms. 7z supports large files up to approximately 16 Exbibyte.7z data compression format is a format that rivals zip format.

Five different lossless compression algorithms that are in the scope of this study have been introduced. In the next step, we present the compression ratios obtained from the compression of these five algorithms with the XML file consisting of the points obtained as a result of the Kinect application.

III. EXPERIMENTAL RESULTS

In order for the study to yield more consistent and good results, 4 different XML files have been created with 3D cloud points from the Kinect camera. Details of the files created with the data received from the Kinect camera are shown in Table 2.

Files	Trace Number	File Size	File Size (B)
Kinect Data 1	1.000	3,91 MB	4.106.896
Kinect Data 2	10.000	39,5 MB	41.465.665
Kinect Data 3	100.000	399 MB	418.653.981
Kinect Data 4	1.000.000	3,93 GB	4.226.552.316

Table 2: Test files details

In our study, five lossless compression algorithms were applied to each file in order to evaluate each file individually. In addition, the success of the compression algorithms depending on the file size is shown at the end of this heading.

First, compression algorithms are applied to Kinect Data 1 file. The results obtained are as shown in Table 3.

Kinect Data 1	Original File Size (MB)	Original File Size (B)	Compressed File Size (KB)	Compressed File Size (B)	Compression Ratio (%)
Bzip2	3,91 MB	4.106.896	480	491.836	88,02
Xz			520	532.608	87,03
Gzip			581	595.118	85,51
Zip			581	595.248	85,51
7-zip			520	532.688	87,03

Table 3: Kinect data 1 file compression results

In the second step, compression algorithms were applied to the Kinect Data 2 file and the results have been shared in Table 4.

Kinect Data 2	Original File Size (MB)	Original File Size (B)	Compressed File Size (MB)	Compressed File Size (B)	Compression Ratio (%)
Bzip2	39,5 MB	41.465.665	4,70	4.937.820	88,09
Xz			5,07	5.324.612	87,16
Gzip			5,64	5.919.146	85,73
Zip			5,64	5.919.276	85,72
7-zip			5,07	5.324.691	87,16

Table 4: Kinect data 2 file compression results

In the same way same compression algorithms have been applied to Kinect Data 3 file and the results are listed in Table 5.

Kinect Data 3	Original File Size (MB)	Original File Size (B)	Compressed File Size (MB)	Compressed File Size (B)	Compression Ratio (%)
Bzip2	399 MB	418.653.981	47,2	49.502.369	88,18
Xz			50,6	53.104.156	87,32
Gzip			56,5	59.285.815	85,84
Zip			56,5	59.285.945	85,84
7-zip			50,6	53.104.238	87,32

Table 5: Kinect data 3 file compression results

Finally, compression algorithms have been applied to Kinect Data 4 file, which is the largest file in the list. The results are as shown in Table 6.

Kinect Data 4	Original File Size (MB)	Original File Size (B)	Compressed File Size (MB)	Compressed File Size (B)	Compression Ratio (%)
Bzip2	3,93 GB	4.226.552.316	472	495.255.164	88,28
Xz			506	530.994.140	87,44
Gzip			567	594.677.741	85,93
Zip			567	594.677.891	85,93
7-zip			506	530.994.222	87,44

Table6. Kinect data 4 file compression results

All compression algorithms are applied for four different files and the results are listed in the tables. The tables contain the original size information for each file and the file sizes obtained after compression. In addition, compression ratios have been calculated and added to the tables for a better and more precise result. Compression ratios enable us to obtain more precise and stable results. As seen in all tables, the best compression ratio was obtained from the Bzip2 compression algorithm.

IV. CONCLUSION

In this study, 3D cloud point data received from Kinect camera has been transferred to XML file and different compression methods have been applied. The Bzip2 algorithm is the lossless compression algorithm that provided the best compression ratio according to the obtained compression ratios. However, there is a limit in the study. The data from the Kinect camera has been transferred to the XML file, which is the most suitable format for storing. However, this has led to the need to store XML tags with data from the camera. This situation increased the required storage space. However, this has been ignored since it has an effect on both the original file size and the compressed file size. In addition, the compression times of the algorithms have been excluded from the study. As a result, the work offers a horizon about the compression algorithms that will work with 3D cloud point applications.

REFERENCES

1. Aldoma, A., et al., *Tutorial: Point Cloud Library: Three-Dimensional Object Recognition and 6 DOF Pose Estimation.* IEEE Robotics & Automation Magazine, 2012. **19**(3): p. 80-91.
2. Leberl, F., et al., *Point Clouds.* Photogrammetric Engineering & Remote Sensing, 2010. **76**(10): p. 1123-1134.
3. Rusu, R. B. and S. Cousins, *3D is here: Point Cloud Library (PCL).* 2011: p. 1-4.
4. Saval-Calvo, M., et al., *A Comparative Study of Downsampling Techniques for Non-rigid Point Set Registration Using Color.* 2015. **9108**: p. 281-290.
5. Mademlis, A., et al., *3D object retrieval using the 3D shape impact descriptor.* Pattern Recognition, 2009. **42**(11): p. 2447-2459.
6. Shah, S. A. A., M. Bennamoun, and F. Boussaid, *Keypoints-based surface representation for 3D modeling and 3D object recognition.* Pattern Recognition, 2017. **64**: p. 29-38.
7. Soltanpour, S., B. Boufama, and Q. M. Jonathan Wu, *A survey of local feature methods for 3D face recognition.* Pattern Recognition, 2017. **72**: p. 391-406.
8. Yang, J., et al., *TOLDI: An effective and robust approach for 3D local shape description.* Pattern Recognition, 2017. **65**: p. 175-187.
9. Zhang, Z., *Microsoft Kinect Sensor and Its Effect.* IEEE Multimedia, 2012. **19**(2): p. 4-10.
10. Caruso, L., R. Russo, and S. Savino, *Microsoft Kinect V2 vision system in a manufacturing application.* Robotics and Computer-Integrated Manufacturing, 2017. **48**: p. 174-181.
11. James, A. *Quick Reference: Kinect 1 vs Kinect 2*. 2014 [cited 2018; Available from: http://www.imaginativeuniversal.com/blog/2014/03/05/Quick-Reference-Kinect-1-vs-Kinect-2/.
12. Abramsba. *Kinect Skeleton Tracker*. 2013 [cited 2018; Available from: https://gist.github.com/abramsba/7356123.
13. Motiian, S., et al., *Automated extraction and validation of children's gait parameters with the Kinect.* Biomed Eng Online, 2015. **14**: p. 112.
14. Erdal, E. and E. Atilla, *Kinect Uygulamaları için Veri Transfer Platformu Tasarımı*, K. University, Editor. 2018.
15. McCool, M., A. D. Robison, and J. Reinders, *Bzip2 Data Compression.* 2012: p. 291-297.
16. Utils, X. *XZ Utils*. 2018 [cited 2018; Available from: https://tukaani.org/xz/.
17. Microsoft. *GzipStream* 2011 [cited 2018; Available from: https://referencesource.microsoft.com/#system/sys/System/IO/compression/GZipStream.cs,b228c3ce72d4d124.
18. Salomon, D., *Data Compression*. 4 ed. 2007: Springer. 1092.
19. 7-Zip. *7-Zip*. 2008 [cited 2018; Available from: https://www.7-zip.org/.
20. Wikipedia. *7z*. 2018 [cited 2018; Available from: https://en.wikipedia.org/wiki/7z.