Compression Techniques for 3D SDI

Chengxi Bernad, SIEW and Alias ABDUL RAHMAN, Malaysia

Key words: Compression Technique, XML, GIS spatial data, Spatial Data Infrastructure (SDI)

SUMMARY

This paper discusses 3D compression techniques that have been developed over the last few decades. These techniques were often addressed in terms of geometry and connectivity compression. Basically, almost all techniques aim to provide solution for efficient 3D objects storage. Until recent years when the XML standard was introduced, then, various data sharing standards were being developed for various applications. One significant data standard for 3D city model is CityGML. Due to the nature of XML schema, problem arises for CityGML in terms of data size for sharing data within Spatial Data Infrastructure (SDI). The generic problem of the XML standard is discussed and this leads to various inventions of XML compression techniques within the computing domain. Since the data sharing activity over the internet is getting important, then, XML compression techniques able to provide solution for efficient data transmission over the internet where XML utilization in SDI and the importance of data sharing standard over the web service environment are really needed. However, customization of such techniques for the suitability of 3D SDI is required. This paper describes various data compression techniques that would be appropriate or suitable for 3D SDI domain, in order to achieve efficient transmission for minimum bandwidth usage as well as for the storage usage of 3D spatial objects. This paper also highlights some future works and the outlook of the intended research experiment within the 3D SDI framework.

Compression Techniques for 3D SDI

Chengxi Bernad, SIEW and Alias ABDUL RAHMAN, Malaysia

1. INTRODUCTION

Compression techniques are commonly found in domains such as computer graphics and multimedia. Compression is often applied in image formats such as Joint Photographic Graphics (JPG), Portable Network Graphics (PNG), Bitmap (BMP), etc. Each format defines its own compression algorithm for digital images. Similar to image compression, compression in 3D objects had been extensively discussed in 3D computer graphics domain over the last decade. Various techniques are discussed and proposed to compress 3D objects for efficient storage (Peng, 2005). On the other hand, due to the advancement of internet, web services are being created to provide services for various applications over distributed environment. This advancement of web services leverages the advantages of XML standard. However, the nature of XML schematic structure tends to produce large file size, which is a common problem when describing geographic entity via Geographic Markup Language (GML) and CityGML (Goetz et al., 2010).

Compression techniques are categorized into lossy compression and lossless compression. Lossy compression techniques degrade quality of input data to reach an optimum files size with acceptable quality. This is commonly found in image compression, where a BMP image could be compressed into JPG image with a significant reduction of file size and quality degraded but yet acceptable. Lossless compression emphasizes a compression algorithm which is able to perform exact reconstruction to the original file from the compressed data.

Common lossless compression method for textual compression is GZIP. Lossless compression is used when the decoded data is identical with the encoded data, and deviation or degradation of such encoded data is prohibited. This method is suitable for compression of source codes.

In 3D SDI, CityGML is becoming popular for representing 3D city model and 3D city storage schema. Such schema was proclaimed as Open Geospatial Consortium (OGC) standard in 2009. This standard is commonly used for interoperability within 3D SDI web services (Basanow et al 2007, Zipf et al, 2007). The advantages of CityGML for 3D city model representation had been discussed (Kolbe, 2005; OGC, 2009). However, according to Mao, (2009), data storage schema using CityGML for a large city will reach several Gigabyte (GB) and this could be problematic for effective transmission within 3D SDI web services. This technical problem might degrade the user experience in terms of performance and expectations.

Since CityGML is a textual XML standard based schema, textual compression seems to be highly relevant. However, as CityGML stores spatial information specified for 3D city model, compression in geometry and connectivity will also be useful. Therefore, compression algorithm for semantics from CityGML as well as geometry information can be reviewed and further improved to suit the 3D SDI scenario.

In this paper, various compression techniques from computer graphics field, which is suitable for 3D SDI especially 3D city model, is discussed. In Section 2, compression techniques for geometry and connectivity are reviewed as well as several textual compression algorithms. In Section 3, some comparisons of these techniques are presented while in section 4, conclusions and future directions for this research are discussed.

2. COMPRESSION

Mesh compression is one of the famous compression techniques for compressing 3D objects. Generally it could be categorized into geometry or connectivity, or both geometry and connectivity compression. Connectivity compression refers to the compression in topological way on 3D objects, while geometry compression reduces the size of 3D objects in terms of geometrical approach. Over the last decades, several mesh compression techniques are studied from triangle mesh aspect to polygonal mesh compression aspect. Triangle mesh compression techniques often use conquest triangles technique followed by a set of rules and use operators to store the conquered mesh until the last mesh is coquetted. On the other hand, polygonal mesh compression defers the triangular approach. In (Peng, 2005), triangulation of a 3D object imposes extra cost in computation and efficiency. Triangulation also causes the possible loss of originality of mesh connectivity due to the re-triangulation process. On the other hand, polygonal mesh compression directly encodes the 3D objects without pre-triangulation.

Geometry compression related researches are discussed by Deering, (1995), Isenburg, (2000), Taubin, (1998) and Touma, (1998), whereas connectivity compression which could be sub-divided into edge-based compression and vertex-based compression. Edge-based related techniques have been discussed by Rossignac (1999) and Szymczak (2003) while vertex-based techniques are discussed by Alliez (2001) and Touma (1998). It is noticeable that some of the techniques are using both geometry and connectivity approach to obtain a better compression ratio.

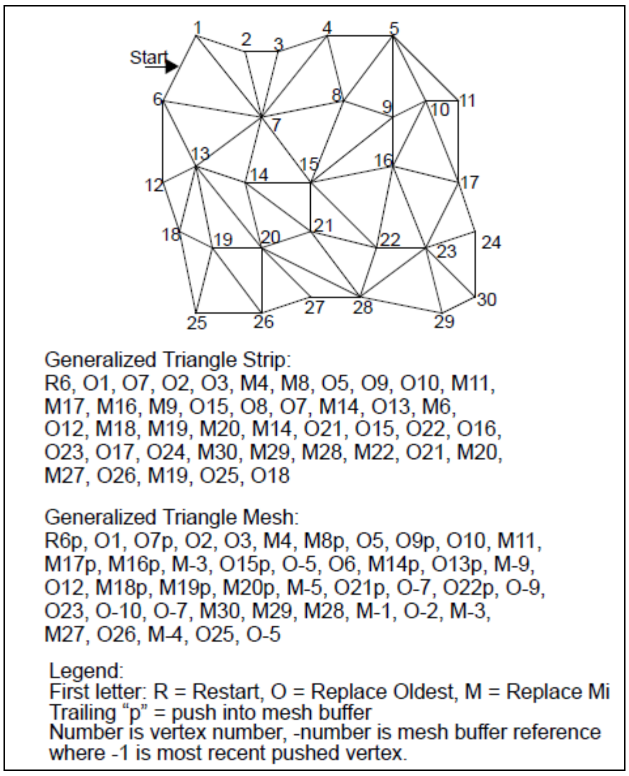
On the other hand, Sakre (2009) studied several textual based compression techniques which are specialized in XML-based compression. Those techniques directly deal with input file in textual form, such as source files for any application. These textual compression techniques are of lossless type.

2.1 Geometry and Connectivity Compression

Generalized Triangle Mesh (GTM) by Deering (1995) is a compression method that constructs a linear triangle strip by converting the targeted triangle data from 3D object. GTM stores vertices and old vertices will be pushed into buffer list or mesh buffer. Such explicit storage for the visited vertices is due to the confined linear strip approach on the model, whereby some interior vertices will appear twice in the strip.

Quantization of geometry coordinate is also one of geometry compression techniques. Quantization is divided into uniform or non-uniform technique, scalar or vector category (Peng, 2005). Giving an example, quantization in uniform way is by fixing the bit number of floating point number within 8bit to 16bit resolution. Deering (1995) and Chow (1997) proposed prediction method where Deering's (1995) method (Figure 1) predicts floating point number of 10 bit to 16 bit resolution whereas the latter predicts 9 to 12 bits quantize resolution. The first method produced results ranging from 17 to 26 bits per vertex (bpv) whereas another method produced results ranging from 13 to 18 bpv. By this comparison, it shows that different quantize resolution produce different bpv which as a result, produce different 3D object size. Quantization is applied due to the limitation of human eye's perceive-ability on IEEE 32-bit floating point number, which means some bit resolution existence do not affect much in terms of visualization.

Compression techniques for graphic object extended from triangle meshes compression to polygonal compression and from single-rate compression to progressive compression, either type resembles lossy or lossless type. A compression technique may be lossless in connectivity, but may cause lossy in geometry, and vice versa.



Generalized Triangle Strip:
R6, O1, O7, O2, O3, M4, M8, O5, O9, O10, M11, M17, M16, M9, O15, O8, O7, M14, O13, M6, O12, M18, M19, M20, M14, O21, O15, O22, O16, O23, O17, O24, M30, M29, M28, M22, O21, M20, M27, O26, M19, O25, O18

Generalized Triangle Mesh:
R6p, O1, O7p, O2, O3, M4, M8p, O5, O9p, O10, M11, M17p, M16p, M-3, O15p, O-5, O6, M14p, O13p, M-9, O12, M18p, M19p, M20p, M-5, O21p, O-7, O22p, O-9, O23, O-10, O-7, M30, M29, M28, M-1, O-2, M-3, M27, O26, M-4, O25, O-5

Legend:
First letter: R = Restart, O = Replace Oldest, M = Replace Mi
Trailing "p" = push into mesh buffer
Number is vertex number, -number is mesh buffer reference where -1 is most recent pushed vertex.

Figure 1: Generalized triangle mesh triangle strip method to push vertices into queue list (Deering, 1995).

Single-rate coding algorithm deals with one-off compression and decompression process while progressive coding emphasizes progressive decompression on compressed data. Single-rate coding may achieve higher compression rate compare to progressive coding, however, when dealing with large file size which required query onto the compressed data, progressive coding may achieve better retrieval time consumption compare to single-rate coding. Therefore, optimization is required to achieve optimize result out of both worlds. Such optimization is especially important for 3D SDI scenario since progressive reconstruction of 3D city from compressed 3D data such as CityGML or X3D should not be time consuming to achieve performance expectation.

Several connectivity compression techniques such as Edgebreaker (Rossignac, 1999), Triangle decomposition (Park, 1999) and Improved Edgebreaker (Jong, 2005) showed that compression result via topological approach is practical and viable. The aforementioned connectivity compression deals with triangulated 3D object and stores triangle information via triangle conquest process.

Edgebreaker (Rossignac, 1999) encode triangles by giving several rules on triangle conquest process. Stored triangle will be represented by series of predefined alphabet which are C, L, E, R, and S. The CLERS string model represents examined 3D object and will be constructed along with compression process. A stored triangle will be represented as series of CCRCRSERCS CRRCRCRCRRRLLR which shows encoded triangle stream (Figure 2). Using this method, Edgebreaker ensures lossless compression for 3D triangulated object connectivity of its meshes.

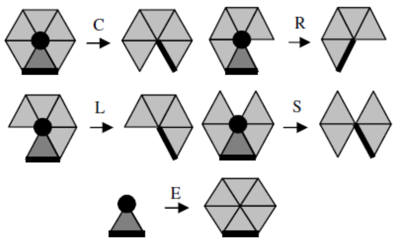


Figure 2: The operators of Edgebreaker algorithm (Rossignac, 1999)

Edgebreaker introduced active gate and active boundary for initialization of compression process. Active boundary means that the entire boundary of the 3D object meshes, while active gate shows the edge of the triangle where the rules of encoding process would be examined to represent the mesh. Third vertex is always determined and the third vertex of every mesh would always be the vertex that opposite the active gate. By these primitive elements for encoding process, CLERS string could be constructed. The rule for C operator is that the third vertex examined is a new vertex. R operator shows that the third vertex is located at the active boundary, and it would then become the vertex for the next active gate. Furthermore, L operator indicates that the third vertex is located at the active boundary, and it was located as previous active gate vertex. Meanwhile S operator shows the third vertex is located at the active boundary, and this vertex is neither the previous vertex nor the next vertex of the active gate. At last, the E operator shows the third vertex is located at the active boundary, and it was the vertex for previous active gate, and it is also the vertex for the next active gate. Rossignac (1999), Jong (2005) (see Figure 3) and Coors (2005) showed that Huffman coding ensures Edgebreaker encodes triangle giving result ranged from 1.5 to 2 bits size.

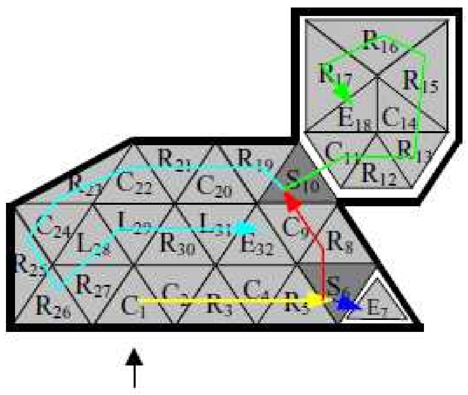


Figure 3: Edgebreaker compression process for a 3D object (Jong, 2005)

On the other hand, Triangle Strip Decomposition (TSD) (Park, 1999) breaks down triangulated object into triangle strips (see Figure 4). While encoding process shows similarity to GTM, however, Park (1999) improved GTM triangle conquest method by introducing vertex chain C to store vertices in a list while connectivity stores in a degree list. Triangle Strip Decomposition method visits all triangles in a set and store chains of vertices and ensures that all triangles are visited. Representation of triangle strip is named T, and vertex chain is represented as C, while v is the vertex. By these indications, a 3D triangulated object M could be encoded into set of $\{T_0, T_1, T_2, \dots\}$ formation.

The chain vertex list in TSD is vital for its parallel decompression process. The relationship between chain C^0 and C^1 is shown in Table 1. Decoding using the relationship of vertex connectivity under vertex degree, offer advantage as the triangle strips between chains can be decoded in parallel way. Also, TSD solved the problem of pocket and hole where such cases refer to sudden gap down in levelling in terms of Digital Terrain Model (DTM).

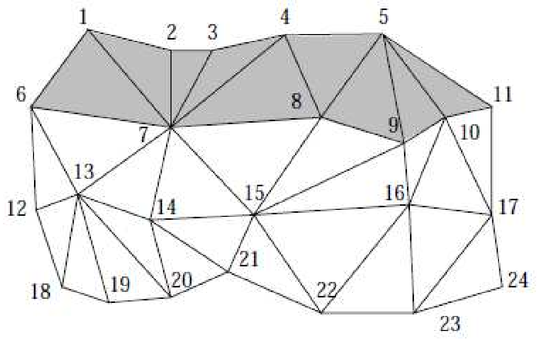


Figure 4: Triangle strips with denoted index for vertices (Park, 1999)

vertex chain	vertex list	vertex degree
C^0	$< 1, 2, 3, 4, 5 >$	$< 0, 0, 0, 0, 0 >$
C^1	$< 6, 7, 8, 9, 10, 11 >$	$< 1, 4, 2, 1, 1, 1 >$
C^2	$< 12, 13, 14, 15, 16, 17 >$	$< 1, 2, 1, 3, 2, 2 >$
C^3	$< 18, 19, 20, 21, 22, 23, 24 >$	$< 2, 1, 2, 2, 2, 2, 1 >$

Table 1: Vertex list in a chain denoted with C (Park, 1999)

2.2 Textual Compression

In textual compression, GZIP is often used as the compressor behind the scene. Common GZIP has been a benchmark of comparison in various studies for improved compression rate as well as encoding time. In Deering (1995), Huffman Coding is used at the last stage as the compressor for the textual data. Similar approaches were discovered in (Liefke et al, 1999), (Tolani et al, 2002), (Min et al, 2003), (Li et al, 2003), (OGC, 2006), and etc. The main idea of textual compression is to reduce redundancy that could be found in the source input for compression.

As CityGML is based on XML encoding standard and it is also textual-based schema. Therefore textual compression method can be used for encoding this standard schema. Common GZIP encoding onto plain XML documents without segmentation is normally applied. However, various researches (Liefke et al, 1999), (Tolani et al, 2002), (Min et al, 2003), (Li et al, 2003), (Ng et. al, 2007) shows that segmentation of XML document and predefine window size of memory buffer for compressor can achieve better result.

Furthermore, studies have shown that binary representation of XML tag also offer better compression rate. Such techniques can be found from BXML (OGC, 2006), Fast InfoSet (ITU-T, 2005) and Efficient XML (EXI) (W3C). However, most of the binary XML techniques could be complicated.

In textual compression, encoding process often takes place by considering factors such as compression rate, compression time, and retrieval (decoding) time. However, generally consideration on compression rate and retrieval (query time) will be more vital. Before using any encoding standard, it is necessary to consider the priority on which side is weighted with higher expectation, that is either priority on compression rate, or priority on query time, or optimize for both aspects.

XMill was developed by Hartmut Liefke and Dan Suciu in 1999, aiming to compress XML document to achieve better result compare to common GZIP. XMill separates data and structure and categorized data into container, then compress. In other words XMill perform split, group and compress job. Containers are instantiated with default 8MB size and XMill uses path processor to store path within documents. Figure 5 is the workflow of the XMill compressor.

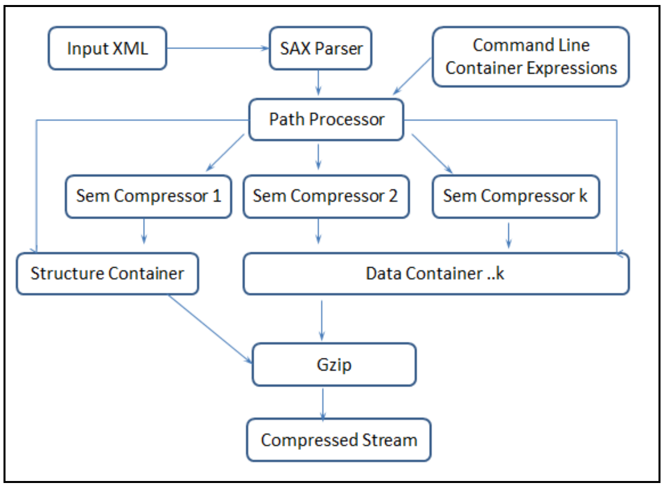


Figure 5: The architecture of the XMill compressor workflow (Liefke et al, 1999)

Besides that, XGrind which was created by Pankaj M. Tolani, and Jayant R. Haritsa in 2002 (see Figure 6) by introducing query ability onto compressed documents. The difference for XGrind is that structure of document is not encoded; only data is used for compression. XGrind added ability of scanning DTD of XML and perform simple encoding scheme via enumeration type of attribute value.

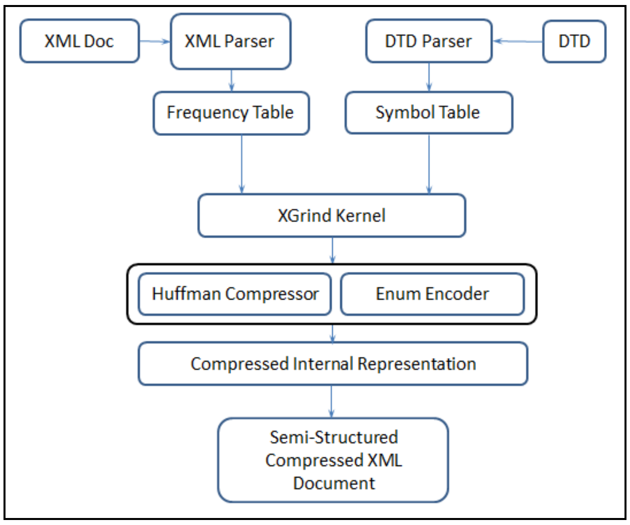


Figure 6: The architecture showing the workflow of XGrind (Tolani et al, 2002)

XPRESS was developed in 2003 by Jun-Ki Min, Myung-Jae Park and Chin-Wan Chung. This is another technique that able to perform query onto compressed documents. XPRESS uses reverse arithmetic encoder, which is a type of binary encoding method. XPRESS uses interval of [0.0 to 1.0] to store path value and uses automatic type inference engine to identify data types automatically. Human intervention on determining data type is avoided in XPRESS. Since interval of path expression is introduced, query can be done by assessing elements indexed by interval figure and decompression can be done based on sequenced interval and this method successfully provide progressive way of decoding an encoded documents. Figure 7 shows the architecture of the XPRESS.

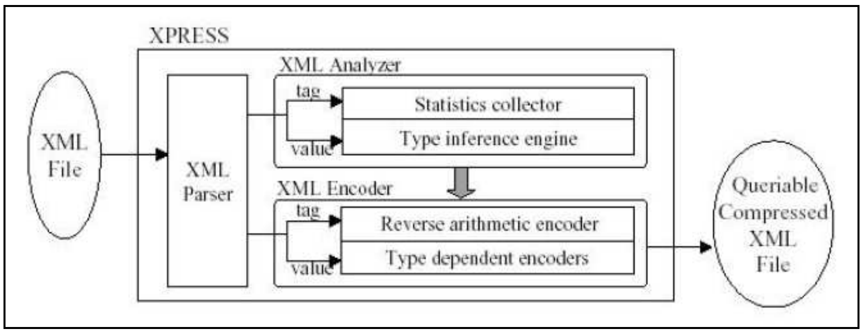


Figure 7: The architecture and workflow of XPress compressor (Min et al, 2003)

Furthermore, XComp (Figure 8) which is developed in 2003 by Weimin Li is an improved version of XMill. XComp uses the separate, join and compress method employed by XMill. The difference of XComp compare to XMill is the container type where XComp uses container type such as structure container, data length container, dictionary container, and data container.

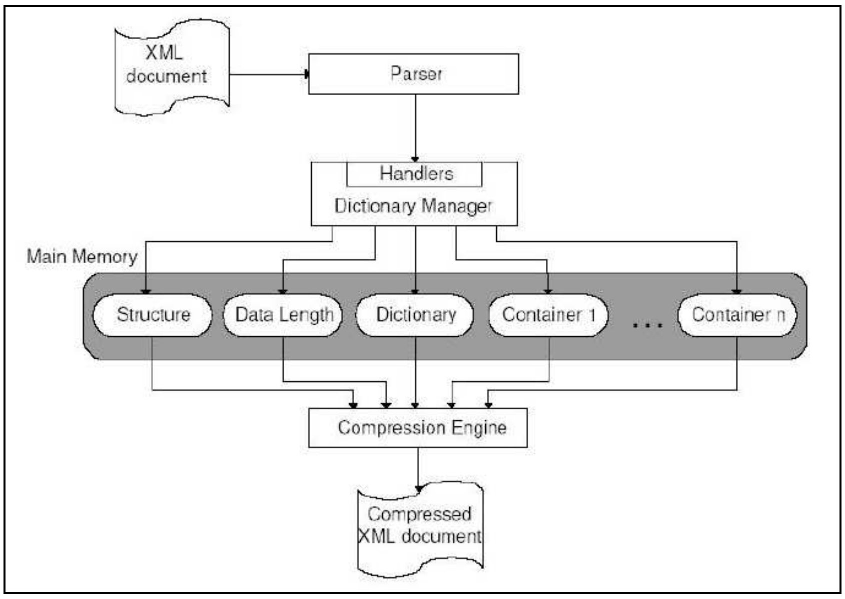


Figure 8: The architecture showing the components and workflow in XComp (Li et al, 2003)

Besides that, XCQ by (Ng et. al, 2006) produces better result by employing Partitioning and Indexing methodology in the compression technique. XML documents are parsed using SAX Parser while DTD Parser retrieve relevant DTD information and build DTD tree for the DSP module. According to Ng et al (2006), the DSP module uses Pseudo-Depth-First strategy instead of conventional depth-first strategy (Figure 9).

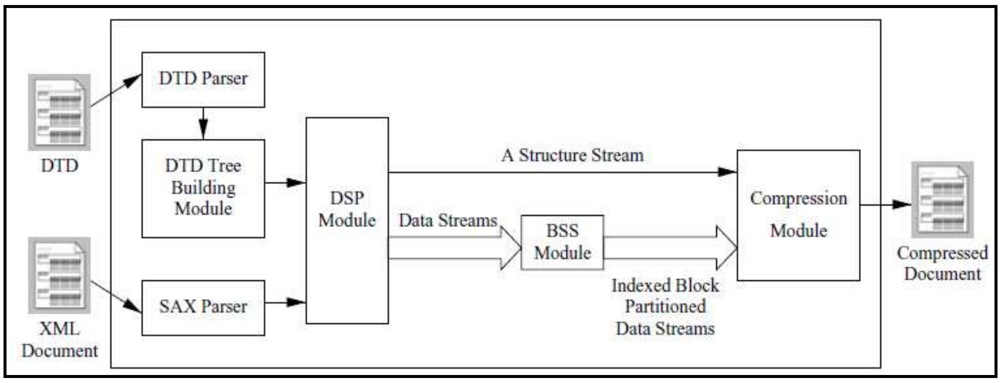


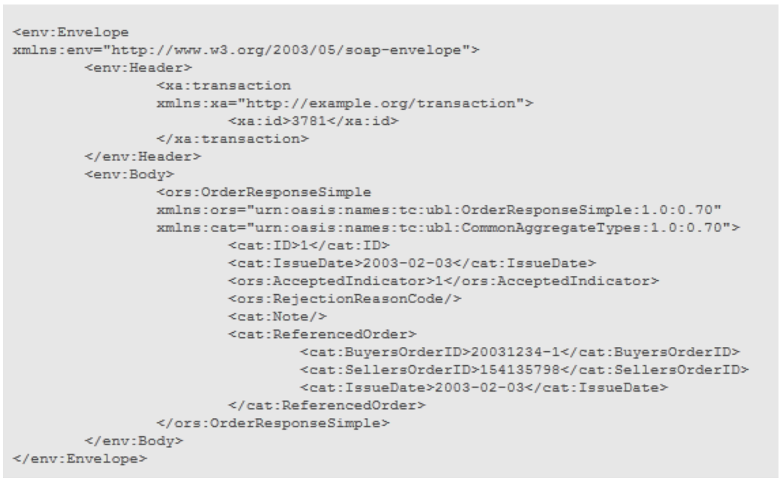
Figure 9: The architecture showing XCQ compression flow (Ng et. al, 2006)

The method employed by XCQ though efficient, but it is not dedicated designed for compressing 3D GIS data especially coordinates and topology, which is very important for GIS domain. From the architecture of XCQ, it shows that the engine does not process coordinates and topologies. Making use of XCQ methodology and adding in coordinates and connectivity scanning and storage engine could be effective for 3D GIS data such as CityGML.

On the other hand, Binary XML (BXML) was discussed in OGC best practise paper (OGC, 2006) and was tested onto Web Feature Service (WFS) version 1.0.0. According to the paper, OGC Web Service (OWS) has to support binary XML capabilities in order to allow WFS clients to choose to encode data during submission or submit via common uncompressed channel.

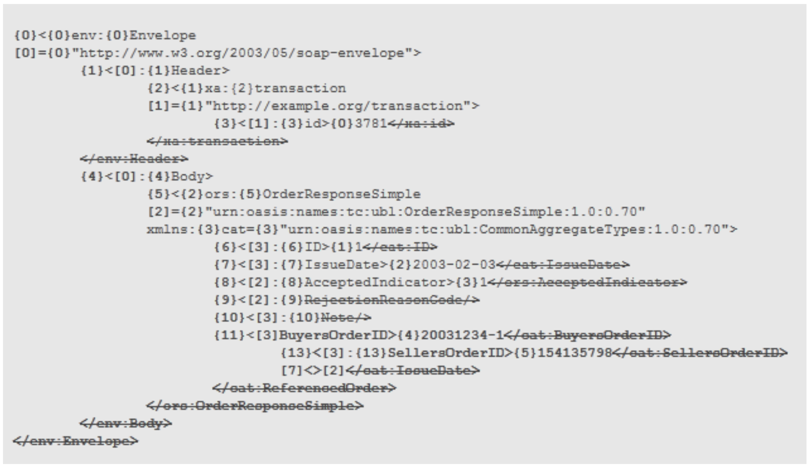
In BXML, document is encoded into header section and body section. Header comprises identifier, version, flags, and body compression notification. Tokens are introduced in BXML which are used to represent several token types that could be discovered in a XML document.

In Fast Infoset (FI), XML document are encoded into Infoset document. FI is an international standard (W3C) for binary XML format. It is also a standard by *ITU-T Rec. X.891*and *ISO/IEC 24824-1*. Figure 10 is a snippet of pre-encoded XML source while Figure 11 shows the encoding process of FI for the same XML source. Furthermore, Figure 12 shows the pre-encoded snippet of X3D code while Figure 13 shows the encoded X3D code using FI.



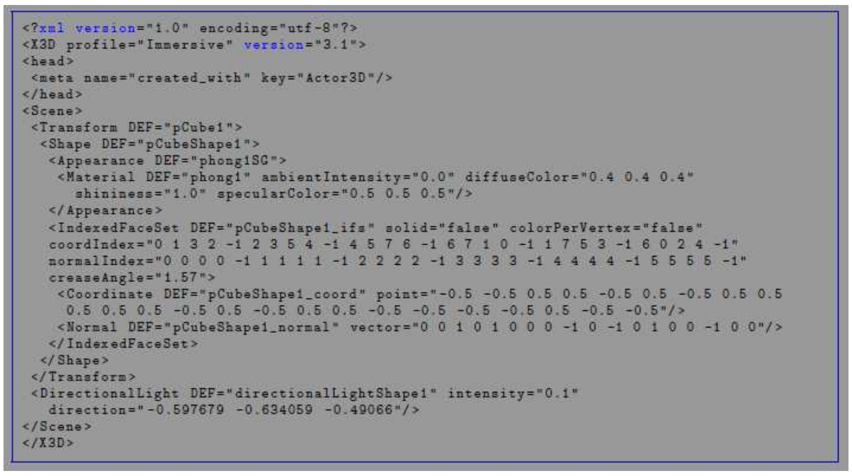
```
<env:Envelope
xmlns:env="http://www.w3.org/2003/05/soap-envelope">
        <env:Header>
                <xa:transaction
                xmlns:xa="http://example.org/transaction">
                        <xa:id>3781</xa:id>
                </xa:transaction>
        </env:Header>
        <env:Body>
                <ors:OrderResponseSimple
                xmlns:ors="urn:oasis:names:tc:ubl:OrderResponseSimple:1.0:0.70"
                xmlns:cat="urn:oasis:names:tc:ubl:CommonAggregateTypes:1.0:0.70">
                        <cat:ID>1</cat:ID>
                        <cat:IssueDate>2003-02-03</cat:IssueDate>
                        <ors:AcceptedIndicator>1</ors:AcceptedIndicator>
                        <ors:RejectionReasonCode/>
                        <cat:Note/>
                        <cat:ReferencedOrder>
                                <cat:BuyersOrderID>20031234-1</cat:BuyersOrderID>
                                <cat:SellersOrderID>154135798</cat:SellersOrderID>
                                <cat:IssueDate>2003-02-03</cat:IssueDate>
                        </cat:ReferencedOrder>
                </ors:OrderResponseSimple>
        </env:Body>
</env:Envelope>
```

Figure 10: The snippet of document before encoding (Oracle, 2007)



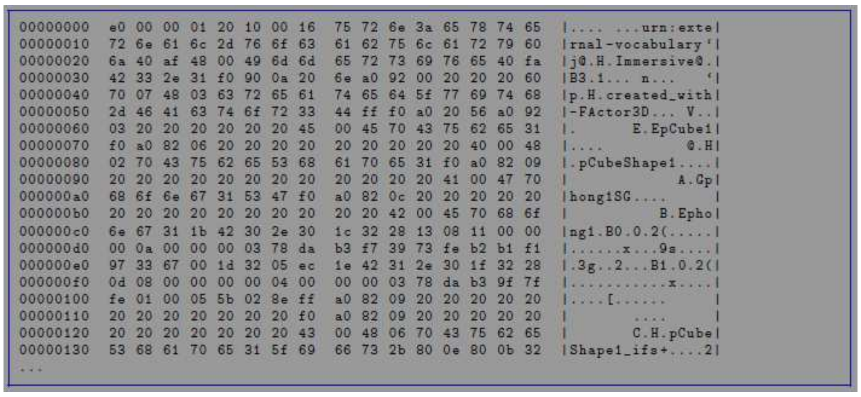
```
{0}<{0}env:{0}Envelope
[0]={0}"http://www.w3.org/2003/05/soap-envelope">
        {1}<[0]:{1}Header>
                {2}<{1}xa:{2}transaction
                [1]={1}"http://example.org/transaction">
                        {3}<[1]:{3}id>{0}3781~~</xa:id>~~
                ~~</xa:transaction>~~
        ~~</env:Header>~~
        {4}<[0]:{4}Body>
                {5}<{2}ors:{5}OrderResponseSimple
                [2]={2}"urn:oasis:names:tc:ubl:OrderResponseSimple:1.0:0.70"
                xmlns:{3}cat={3}"urn:oasis:names:tc:ubl:CommonAggregateTypes:1.0:0.70">
                        {6}<[3]:{6}ID>{1}1~~</cat:ID>~~
                        {7}<[3]:{7}IssueDate>{2}2003-02-03~~</cat:IssueDate>~~
                        {8}<[2]:{8}AcceptedIndicator>{3}1~~</ors:AcceptedIndicator>~~
                        {9}<[2]:{9}~~RejectionReasonCode/>~~
                        {10}<[3]:{10}~~Note/>~~
                        {11}<[3]BuyersOrderID>{4}20031234-1~~</cat:BuyersOrderID>~~
                                {13}<[3]:{13}SellersOrderID>{5}154135798~~</cat:SellersOrderID>~~
                                [7]<>[2]~~</cat:IssueDate>~~
                        ~~</cat:ReferencedOrder>~~
                ~~</ors:OrderResponseSimple>~~
        ~~</env:Body>~~
~~</env:Envelope>~~
```

Figure 11: The snippet of encoded document (FI) (Oracle, 2007)



```
<?xml version="1.0" encoding="utf-8"?>
<X3D profile="Immersive" version="3.1">
<head>
 <meta name="created_with" key="Actor3D"/>
</head>
<Scene>
 <Transform DEF="pCube1">
  <Shape DEF="pCubeShape1">
   <Appearance DEF="phong1SG">
    <Material DEF="phong1" ambientIntensity="0.0" diffuseColor="0.4 0.4 0.4"
      shininess="1.0" specularColor="0.5 0.5 0.5"/>
   </Appearance>
   <IndexedFaceSet DEF="pCubeShape1_ifs" solid="false" colorPerVertex="false"
   coordIndex="0 1 3 2 -1 2 3 5 4 -1 4 5 7 6 -1 6 7 1 0 -1 1 7 5 3 -1 6 0 2 4 -1"
   normalIndex="0 0 0 0 -1 1 1 1 1 -1 2 2 2 2 -1 3 3 3 3 -1 4 4 4 4 -1 5 5 5 5 -1"
   creaseAngle="1.57">
    <Coordinate DEF="pCubeShape1_coord" point="-0.5 -0.5 0.5 0.5 -0.5 0.5 -0.5 0.5 0.5
     0.5 0.5 0.5 -0.5 0.5 -0.5 0.5 0.5 -0.5 -0.5 -0.5 -0.5 0.5 -0.5 -0.5"/>
    <Normal DEF="pCubeShape1_normal" vector="0 0 1 0 1 0 0 0 -1 0 -1 0 1 0 0 -1 0 0"/>
   </IndexedFaceSet>
  </Shape>
 </Transform>
 <DirectionalLight DEF="directionalLightShape1" intensity="0.1"
   direction="-0.597679 -0.634059 -0.49066"/>
</Scene>
</X3D>
```

Figure 12: A standard X3D encoding document (Sons, 2010)



```
00000000  e0 00 00 01 20 10 00 16  75 72 6e 3a 65 78 74 65  |.... ...urn:exte|
00000010  72 6e 61 6c 2d 76 6f 63  61 62 75 6c 61 72 79 60  |rnal-vocabulary'|
00000020  6a 40 af 48 00 49 6d 6d  65 72 73 69 76 65 40 fa  |j@.H.Immersive@.|
00000030  42 33 2e 31 f0 90 0a 20  6e a0 92 00 20 20 20 60  |B3.1... n...   '|
00000040  70 07 48 03 63 72 65 61  74 65 64 5f 77 69 74 68  |p.H.created_with|
00000050  2d 46 41 63 74 6f 72 33  44 ff f0 a0 20 56 a0 92  |-FActor3D... V..|
00000060  03 20 20 20 20 20 20 45  00 45 70 43 75 62 65 31  |.      E.EpCube1|
00000070  f0 a0 82 06 20 20 20 20  20 20 20 20 20 40 00 48  |....         @.H|
00000080  02 70 43 75 62 65 53 68  61 70 65 31 f0 a0 82 09  |.pCubeShape1....|
00000090  20 20 20 20 20 20 20 20  20 20 20 20 41 00 47 70  |            A.Gp|
000000a0  68 6f 6e 67 31 53 47 f0  a0 82 0c 20 20 20 20 20  |hong1SG....     |
000000b0  20 20 20 20 20 20 20 20  20 20 42 00 45 70 68 6f  |          B.Epho|
000000c0  6e 67 31 1b 42 30 2e 30  1c 32 28 13 08 11 00 00  |ng1.B0.0.2(.....|
000000d0  00 0a 00 00 00 03 78 da  b3 f7 39 73 fe b2 b1 f1  |......x...9s....|
000000e0  97 33 67 00 1d 32 05 ec  1e 42 31 2e 30 1f 32 28  |.3g..2...B1.0.2(|
000000f0  0d 08 00 00 00 00 04 00  00 00 03 78 da b3 9f 7f  |...........x....|
00000100  fe 01 00 05 5b 02 8e ff  a0 82 09 20 20 20 20 20  |....[......     |
00000110  20 20 20 20 20 20 20 f0  a0 82 09 20 20 20 20 20  |       ....     |
00000120  20 20 20 20 20 20 20 43  00 48 06 70 43 75 62 65  |       C.H.pCube|
00000130  53 68 61 70 65 31 5f 69  66 73 2b 80 0e 80 0b 32  |Shape1_ifs+....2|
...
```

Figure 13: A byte encoded version of snippet from Figure 12 using FI (Sons, 2010)

3. DISCUSSION

This section will discuss some benefits and disadvantages with regards to those techniques mentioned in previous chapter. Since compression techniques adopted from computer graphics domain are comprehensive, the techniques introduced in this paper were only focusing on triangulated technique, which is suitable for DTM and city model scenario. Since GTM and Triangle Strip Decomposition have tested with DTM as input, these methods seem viable in 3D GIS domain. Furthermore, Edgebreaker was used by (Coors, 2005) to compress 3D buildings. Compression using Edgebreaker to store connectivity for 3D building is useful, but when decoding the encoded meshes using Edgebreaker, single-rate decoding is expected since the nature of Edgebreaker only supports one off decompression process. (Isenburg, 2000) uses The Spirale Reversi algorithm shows that using Edgebreaker compression onto 3D object, the decoding progress can only be done only if the 3D object was fully compressed. Table 2 shows the capability of the discussed techniques in compressing DTM and building, as well as the progressiveness of the compression algorithm.

Techniques	DTM	Building	Progressive

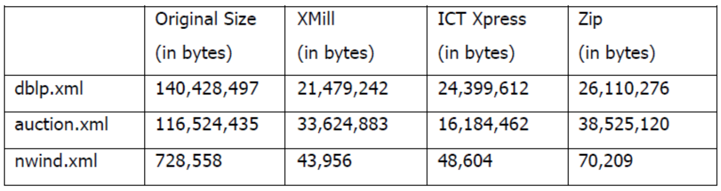
	(Applicable?)	(Applicable?)	
GTM	Yes	No	No
TSD	Yes	Complicated	Yes
Edgebreaker	Yes	Yes	No

Table 2: Compression techniques for triangulated 3D object

On the other hand, XMill and XComp where both algorithms are quite similar, these methods use split, group and compress approach. Such approach does not support progressive decoding as no indexing for file structure is stored and file structure is explicitly compressed in containers. For XMill, result is not significant for data size less than 1MB if comparison was done towards GZIP.

Besides that, XGrind improvised in terms of query availability aspect. XGrind checks validity on DTD of XML and perform indexing on compressed document. The time consumption of XGrind for querying is acceptable and XGrind provides progressive decoding in query predicate. However, XGrind do not support non-equality query operation and also operations such as Join, Aggregation, and Nested Operation is not available.

Xpress also demonstrates query availability on compressed data. Interval sequence is introduced in Xpress and this innovation supports direct evaluation of encoded data with queried data by evaluating interval. According to Sakre (2009), Xpress is 2.83 times faster than XGrind query time, 80% compression ratio better than XMill, and 3.14 times better than GZIP. Table 3 (Sakre, 2009) shows the result of XMill, Xpress and zip algorithm in compressing XML files such as dblp.xml, auction.xml and nwind.xml.



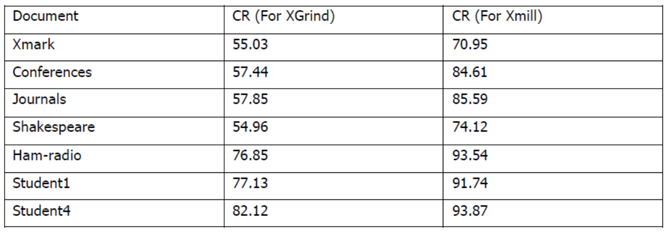
	Original Size (in bytes)	XMill (in bytes)	ICT Xpress (in bytes)	Zip (in bytes)
dblp.xml	140,428,497	21,479,242	24,399,612	26,110,276
auction.xml	116,524,435	33,624,883	16,184,462	38,525,120
nwind.xml	728,558	43,956	48,604	70,209

Table 3: XML files that tested under XMill, Xpress, and Zip (Sakre. 2009)

After averaging and calculating the compression ratio for these 3 files of dblp.xml, auction.xml and nwind.xml, Table 4 is constructed with the result as shown.

Files	XMill (%)	Xpress (%)	Zip (%)
dblp.xml	84.7	82.6	81.4
auction.xml	71.1	86.1	67.0
nwind.xml	93.0	93.3	90.4
AVERAGE	83.0	87.3	79.6

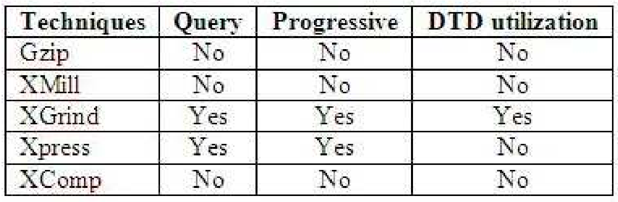
Table 4: Compression ratio in percentage for XMill and Xpress



Document	CR (For XGrind)	CR (For Xmill)
Xmark	55.03	70.95
Conferences	57.44	84.61
Journals	57.85	85.59
Shakespeare	54.96	74.12
Ham-radio	76.85	93.54
Student1	77.13	91.74
Student4	82.12	93.87

Table 5: Compression Ratio (CR) for XGrind and XMill (Sakre, 2009)

Table 5 shows the average compression ratio for XGrind and XMill would be 65.91 % and 84.91 % tested with 7 files. It is significant that XMill still outruns XGrind and Xpress, however, compare to the availability of query for compressed data, XMill does not show any relevance at all (see Table 6).



Techniques	Query	Progressive	DTD utilization
Gzip	No	No	No
XMill	No	No	No
XGrind	Yes	Yes	Yes
Xpress	Yes	Yes	No
XComp	No	No	No

Table 6: Features in each technique.

4. CONCLUDING REMARKS

The paper presented the relevant compression techniques that seem appropriate for transferring large dataset within 3D SDI services. The operation within CityGML can be made simpler by having a dedicated approach, thus enhanced XML-based compression technique.

We have carried out initial experiments on this aspect and the highest redundancies which occurred in the CityGML structure, is when describing geometry for surface members of each building, this hits around 123,622 times for Putrajaya_3D building dataset. On the other hand, the largest space consumption portion in CityGML is the Unique Resource Identifier (URI). This particular part consumes 15.3 Megabytes (MB) out of the original file size of 104 MB. Furthermore, quantized coordinates up to millimeter (mm) accuracy and filtering out repeated coordinates only requires 2.64MB out of original file size.

Given the above outputs, the zipped output was less than 9MB, which is 8.7 %. Compared to the direct zipping method from raw CityGML input, the output was 13.2 MB disk space, which is 12.7 %. However, the XML-based compression technique we employed maintained the connectivity information of coordinates for the surfaces of buildings as well as the Path-expression. This indicates query on the compressed-data could be done directly.

In the future, a Graphic User Interface (GUI) will be developed to enable various compression options for users. Furthermore, the compression engine can be used to integrate into the proposed Malaysian 3D SDI to enable efficient data transmission within web services.

REFERENCES

Alliez P., Desburn M., 2001, Valence-Driven Connectivity Encoding for 3D Meshes, Eurographics '01

Bin-Shyan Jong, Wen-Hao Yang, Juin-Ling Tseng, Tsong-Wuu Lin, 2005, An Efficient Connectivity Compression for Triangular Meshes, Proceedings of the Fourth Annual ACIS International Conference on Computer and Information Science, 0-7695-2296-3/05

Dong-Gyu Park, Yang-Soo Kim, Hwan-Gue Cho, 1999, Triangle Mesh Compression for Fast Rendering, Proceedings. IEEE International Conference on Information Visualization.

H. Liefke and D. Suciu, 1999 XMill: An Efficient Compressor for XML Data. http://sourceforge.net/projects/xmill

Isenburg M., Snoeyin J., 2000, Spirale Reversi: Reverse Decoding of EdgeBreaker Encoding, Proceeding of 12th Canadian Conference on Computational Geometry

Jarek Rossignac, 1999, Edgebreaker: Connectivity compression for triangle meshes, GVU Technical Report GIT-GVU-98-35

Jingliang Peng, Chang-Su Kim, C.-C. Jay Kuo, 2005, Technologies for 3D mesh compression: A survey, J. Vis. Commun. Image R. 16 (2005) pp. 688–733

J. Min, M. Park and C. Chung., 2003, XPRESS: A Queriable Compression for XML Data, In Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data

Kristian Sons, 2010, Fast Infoset (FI) encoding for X3D Speed-up X3D processing, http://www.web3d.org/event/s2010/w3d-x3db.pdf

M. Chow, Optimized geometry compression for real-time rendering, in: IEEE Visualization, 1997, pp. 347–354

Michael Deering, 1995, Geometry Compression, SIGGRAPH '95 Proceedings of the 22nd annual conference on Computer graphics and interactive techniques

M. Goetz, A. Zipf, 2010, Open Issues In Bringing 3D To Location Based Services (LBS) - A Review Focusing On 3D Data Streaming And 3D Indoor Navigation. International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, Volume XXXVIII-4/W15

Oracle Sun, Fast Infoset Documentation, http://java.sun.com/developer/technicalArticles/xml/fastinfoset/

Graig Bruce, 2006, OGC Binary Extensible Markup Language (BXML) Encoding Specification, OGC 03-002r9, OpenGIS Best Practices Document

P. Tolani and J. Harista., 2002, XGrind: A Query-friendly XML Compressor, IN ICDE, pp 225-234

Sherif Sakr, 2009, XML compression techniques: A survey and comparison, Journal of Computer and System Sciences, 75 (2009) pp. 303–322

Taubin, G., Horn, W., Lazarus, F., Rossignac, J. (1998), Geometry coding and VRML, Proceedings of The IEEE, 86(6)

Touma, C., Gotsman, C., 1998, Triangle Mesh Compression, Graphics Interface 98 Conference Proceedings

The Virtual Reality Modeling Language (VRML). ISO/IEC 14772-1, 1997.

Volker Coors, Katja Ewald, 2005, Compressed 3D Urban Models for Internet-Based E-Planning, Proceeding of 1st International Workshop on Next Generation 3D City Models, Bonn, EUROSDR Publication #49

BIOGRAPHICAL NOTES

Alias Abdul Rahman is a Professor at the Department of Geoinformatics and Dean of the Faculty of Geoinformation and Real Estate, Universiti Teknologi Malaysia (UTM), Skudai, Johor. He received a degree in Surveying and Mapping Sciences from North East London Polytechnic, England, UK in 1987, Postgraduate Diploma in GIS from ITC, Netherlands, and MSc in GIS also from ITC, Netherlands. In 2000, he received PhD degree from University of Glasgow, Scotland, U.K. Currently he serves as Chair for ISPRS Commission II/5 from 2008 – 2012 on Multidimensional GIS and Mobile Data Model.

Siew Chengxi Bernad, obtained a Bsc Degree in Geoinformatics in 2009 from Universiti Teknologi Malaysia (UTM), Skudai, Johor. He is a professional programmer in JAVA and C# programming language. He worked as an IT Manager in ABS Innovations Sdn Bhd and a Chief Technical in Effisys Technology, Malaysia. He has been involving in developing various web applications and mobile applications. Currently, he is a PhD researcher in 3D GIS Research Group in the Faculty of Geoinformation and Real Estate, UTM. His current research interests are in the area of distributed web environment for 3D GIS and compression techniques.

CONTACTS

Alias Abdul Rahman
3D GIS Research Lab
Faculty of Geoinformation and Real Estate
Universiti Teknologi Malaysia
81310 Johor Bahru
MALAYSIA
Email: alias@utm.my
Contact: +6013-7490452

Siew Chengxi Bernad
3D GIS Research Lab
Faculty of Geoinformation and Real Estate
Universiti Teknologi Malaysia
81310 Johor Bahru
MALAYSIA
Email: cbsiew2@live.utm.my
Contact: +6017-8888460