Document-Oriented Data Organization for Unmanned Aerial Vehicle Outputs

Suhaibah Azri, Uznir Ujang, and Wan Afifah Wan Embong
3D GIS Research Lab, Geoinformation, Faculty of Built Environment and Surveying, Universiti Teknologi Malaysia, Malaysia
Email: {suhaibah, mduznir}@utm.my, wanafifah41@yahoo.com

Miguel Gonzalez Cuetara and Guillermo Miguel
Ecapture Research and Development S.L., Badajoz, Spain
Email: miguelgonzalezcuetara@gmail.com, guillermo@ecapture3d.com

***Abstract*—Three-Dimensional (3D) point cloud is considered an important geospatial resource for a vast range of applications. This is due to the rapid technology on data acquisition, such as Unmanned Aerial Vehicle (UAV), Mobile Laser Scanning, and Terrestrial Laser Scanning (TLS). Yet efforts to exploit the use of these datasets are increasingly threatened by the massive dataset, data density, and data complexity. Traditional Relational Database Management System (RDBMS) existed years ago, but the capability of relational databases in handling these issues is questionable due to several drawbacks and limitations. To address these challenges, effective storage, querying, and organization is required. Document-oriented databases are becoming more prominent compared to relational database, since it is capable of handling petabytes of data emerging from the Big Data scheme. Thus, this study investigates the capability of the document-oriented database in organizing UAV outputs, such as images and point clouds, via a NoSQL database. There are 103,996,984-point clouds generated from UAV images and stored in the database. Several tests, such as time analysis, insert operation, and storage consumption, are performed and compared with the RDBMS. The results show that the document-oriented database outperforms the relational database during data retrieval, where the document-oriented database response is 37% faster than the relational database. Meanwhile for data updating, the document-oriented database response is 30% faster than a relational database. To retrieve the stored point cloud in the database, the Potree viewer is used to render the data on the web browser. Based on the result, the point cloud data is successfully rendered and can be manipulated for future applications.**

***Index Terms*—relational database, document-oriented database, point cloud, unmanned aerial vehicle**

I. Introduction

Point clouds are datasets that represent objects or space. Each single points represent the X, Y, and Z coordinates an underlying sampled surface. Point clouds are a means of collating a large number of single spatial measurements into a dataset that can then represent a whole object or space. Point clouds are most commonly generated using 3D laser scanners and LiDAR (Light Detection and Ranging) technology and techniques. Here, each point represents a single laser scan measurement. These scans are then stitched together, creating a complete capture of a scene, using a process called "registration". Nowadays, the application of point cloud data has increased dramatically due to the increasing number of data acquisition technologies, such as airborne LiDAR, Mobile Laser Scanning (MLS), Terrestrial Laser Scanning (TLS), Unmanned Aerial Vehicle (UAV), drones and photogrammetry image processing (see [1]-[3]). Using these technologies, discrete and massive point cloud data are recorded, and enormous computing power is required to process the data [4].

The latest standard of inspection embraces drones that master the art of data collection efficiently. UAV technology has encouraged industries in diverse practices. UAVs are utilized in numerous occurrences due to their advancement in safety. With their remote-control abilities, drones can monitor locations, communicate possible hazards, and notify threatening conditions. As a drone's applicability becomes more extensive, their prices also drive towards being more pocket-friendly compared to other devices, such as TLS and MLS. Besides that, a UAV is embedded with high-resolution cameras furnished with top-notch sensors, which means UAVs can take excellent aerial photographs, aerial videos and accumulate large volumes of accurate data. However, these images need to be pre-processed and transformed into a point cloud dataset. Vast amounts of point cloud dataset will be produced and can be up to terabyte and petabyte size. In order to make use of these point clouds, an efficient storage and retrieval system is therefore needed.

Relational Database Management System (RDBMS) stores data in tables and uses Structured Query Language (SQL) for data retrieval. In relational databases, the database schema is defined, and rules are set up to control the relationship between fields. The data may be stored in separate tables but associated using a join function. According to [5], the traditional relational database is

Manuscript received May 19, 2021; revised September 1, 2021.

doi: 10.12720/jait.12.4.267-278

somehow relatively inefficient in handling and processing massive spatial data. It has been proven that relational databases are inadequate to deal with the immense volume of data and would cause delay during data transformation and retrieval. However, even though SQL server and relational databases are considered popular databases, it could not cater to present-day processing needs. Thus, document-oriented databases are highly recommended by developers and users due to their ability to store unstructured and heterogeneous data [6].

This study proposes the point cloud dataset set to be organized using a document-oriented database. The main concept of a document-oriented database is formed from the notion of a document. Documents in a document-oriented storage are roughly equivalent to the programming concept of an object. They are not required to adhere to a standard schema, nor will they have all the same sections, slots, parts, or keys. This concept seems to be in line with the point cloud attributes, which are unstructured and scattered. Thus, the aim of this paper is to test the efficiency of a document-oriented database in handling a point cloud dataset. The rest of this paper is organized as follows. Section II describes related literature review and studies related to a document-oriented database. Section III discusses the methodology of point cloud data acquisition using UAV images and photogrammetry technique and data organization in a document-oriented database. The results and discussions are discussed in Section IV.

II. Relational Database and Document-Oriented Database

Relational databases have been around for over 30 years. They have stable and richly functional software compared to a document-oriented database [7]. However, several drawbacks have been identified and these explain why the relational database is not always the best choice. Some issues that have been raised by the researcher is that the relational database is not compelling, flexible, and expensive to purchase [8], [9]. Compared to a document-oriented database, most of the packages are open source and free [10]. The open-source nature offers opportunities for researchers to investigate and enhance the database features. This effort provides cheaper storage for users that cannot afford proprietary database models. Another limitation of relational databases is the architecture is usually scaled up. This means that the server hardware must be upgraded to make it more efficient, and this causes the amount of administrator's effort to increase [7]. It will be more challenging if the hardware is fixed by design and cannot be altered. For example, some hardware manufacturers have fixed the amount of the maximum Random-Access Memory (RAM) and it cannot be modified. This situation shows that relational databases are able to scale up, but subject to hardware's restrictions. Meanwhile, document-oriented databases, such as NoSQL, scale the hardware horizontally and are not significantly affected by hardware limitations because smaller, cheaper, and less powerful server machines can be combined to offer higher levels of scalability, instead of having one expensive server [11], [12]. This ability gives advantages to the document-oriented database as commodity servers in scenarios where actual hardware cannot be acquired and at the same time, avoids degrading the database performance. With the current social media trend, high levels of scalability are required, and this is not well addressed in relational databases but in document-oriented databases [13].

In addition, the main drawback of a relational database is handling massive datasets, especially for enormous data from the web, where the growth of data was almost 25 times from the year 2007 to 2010 [14]. On the off chance the data need to be connected and the columns on the table need to be expanded, this would prompt small table relationships to be more complex in relational databases. According to [15], volumes of data from internet applications that need to be handled by databases have increased. The emergence of Web 2.0 and 3.0 has increased the volume and variety of data that need to be stored. However, internet data has failed to be handled by relational databases due to the large volumes of data coming from these sources. Several companies, such as Google, Facebook, and Yahoo, have migrated to document-oriented databases (NoSQL), and have shown that this database excels at handling large volumes of data [16], [17]. Several studies proved that a document-oriented data organization database is efficient at handling big data. [18], reported that they had stored various sorts of information in document-oriented databases, such as collect weather information, digital images and videos, transaction information, and others. The size of stored information is almost 2.5 quintillion bytes of data. Furthermore, document-oriented data organization is important for Content Management Systems (CMS), web analytics, e-commerce applications, and real-time analytics. Today, document-oriented databases are increasingly popular, especially for massive data applications and real-time web applications.

The document-oriented data organization stores data differently from the relational database. The database does not require any kind of fixed table schemas, unlike the SQL databases. Other than that, document-oriented data organization can be referred to as structured storage that consists of a relational database as the subset, and it generally scales horizontally and avoids major join operations on the data. According to [19], [20], document-oriented data organization has overcome the limitations of relational databases. Relational databases store and retrieve data from interrelated tables, while document-oriented databases can store an entire object in a single JSON document, making it faster to retrieve. In comparison with relational databases, document databases support the addition of fields to JSON documents. This means the user does not need to define changes at the initial stage. Besides that, these databases support dynamic data that can be changed at any time.

There are several databases that are categorized as document-oriented data organization (NoSQL), such as MongoDB, CouchDB, and ArangoDB. However, this study specifically chose to organize the UAV outputs in

MongoDB. Despite its popularity and stability, MongoDB performed better in terms of performance for queries Create Read Delete Update (CRUD). According to [21], ArangoDB performed better when creating data but performed worse when reading, updating, and deleting data. Besides, [21] also reported that MongoDB had the lowest average query response time for the read and delete operations compared to ArangoDB and CouchDB. To compare the MongoDB database with relational databases, this study used PostgreSQL. Compared to other relational databases, such as MySQL, SQL server, and SQLite, PostgreSQL is truly open-sourced and community-driven. Meanwhile, MySQL requires licensing for database operations. MySQL moves old data to a separate area called rollback segments, which will cause performance deteriorations during bulk insertion. Since this study will apply point cloud bulk insertion, MySQL seems to be the less suitable choice, and this is where PostgreSQL shines. Besides that, MySQL does not work well with long-running selections, and it is best suited to smaller selections, especially the ones covering clustered indexes. Some of the other disadvantages include a lack of full-text search and slow concurrent read-writes. A study by [22] shows a comparison of insert operation between PostgreSQL and SQLite. The study concluded that PostgreSQL is the best competitor of SQLite and it was ahead of SQLite during the insert operation. Another drawback of SQLite is its handling of writes operations, which are serialized. This can be a major bottleneck for applications that require concurrency. As SQLite is a file-based DBMS, it can cause performance issues with larger datasets because of file system limitations. Thus, in this study, MongoDB and PostgreSQL are chosen to represent document-oriented data organization and relational database for UAV output, respectively. The differences between document-oriented data organization (NoSQL) and relational database (PostgreSQL) are summarized in the following Table I based on data storage, schemas, scalability, and integrity compliance.

TABLE I. OVERVIEW OF DIFFERENCES BETWEEN A RELATIONAL DATABASE (POSTGRESQL) AND DOCUMENT-ORIENTED DATABASE (MONGODB)

Characteristic	Relational Database (PostgreSQL)	NoSQL (MongoDB)
Data storage	Store information in the forms of tables. Each row has information about one specific entity and column store separate data items.	Stored in different formats in various databases.
Schemas	Creation of table based on schemas and it complex to alter the schema once it defines.	Schema is dynamic and information can be changes easily. It very flexible compared to relational database.
Scalability	Scaling is vertical. It possible to scale a RDBMS across multiple servers and it also difficult and time-consuming process.	Scaling is horizontal. More servers can be added to increase the performance.
Cost	Expensive approach for data storage	Cheaper as it is open source and inexpensive upgrade

III. METHODOLOGY

This section describes the methodology for this research. The methodology is divided into several sections, which are Section A. UAV Data Acquisition, Section B. Data Processing, Section C. Data Organization, Section D. Data Rendering and Section E. Measuring the Performance.

A. UAV Data Acquisition

The Unmanned Aerial Vehicle (UAV), DJI Phantom 4 (see Fig. 1), is utilized to capture images of ground surfaces. The study area for this study is around Lingkaran Ilmu, Universiti Teknologi Malaysia. The UAV is flown at a flight altitude of 150 meters. Prior to flying, the study areas are divided into ten small areas. Fig. 2 shows the study area and the ten boundaries of the study areas. Each study area must be overlapped with the neighboring boundaries. The percentage of overlapping must be between 60% to 80% of area (see Fig. 2). The UAV flew area by area to make sure all areas are covered and images from the same areas can be stitched together during data processing. Each area may have more than 50 images and every area covers around eight to ten minutes of every flight. Later, these images are processed batch by batch to produce 3D point clouds. Then, these outputs are stored and organized in the database.

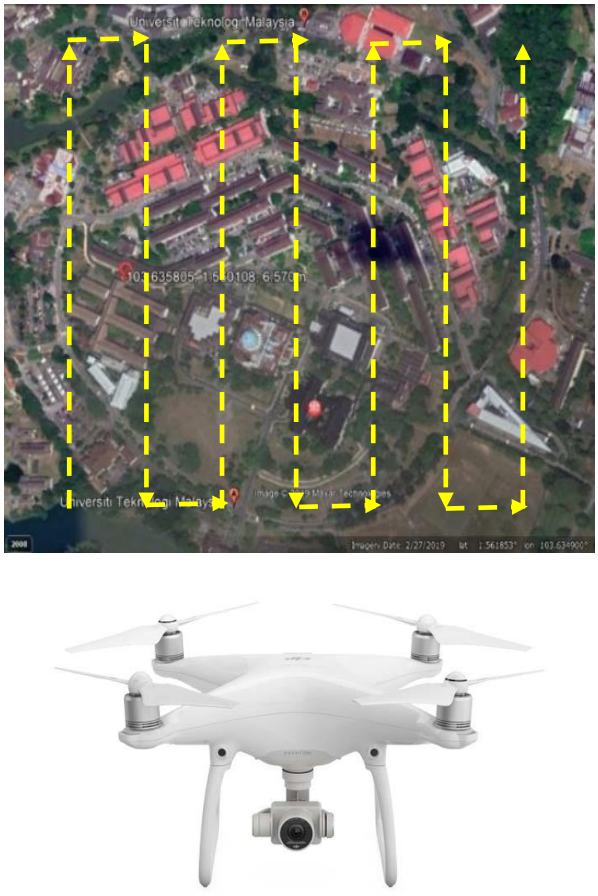


Figure 1. DJI Phantom 4 and flight plan.

Figure 2. Ten boundary areas with 60% overlap between images.

B. Image Processing

The main output from UAV are images. These images need to be processed to produce 3D point clouds. According to the photogrammetry concept, to produce 3D point clouds from images, three orientations need to be determined, which are interior, relative, and absolute orientation. In the interior orientation, several parameters are needed, which are the coordinates of image center, focal length, and image frame resolution. Objects are referenced to an image coordinate system. At least two images are needed to transform into model coordinates. With the absolute orientation, model coordinates can be transformed into any exterior reference system, such as Universal Transverse Mercator (UTM). In this study, we used the World Geodetic System 1984 (WGS84) as a coordinate system.

The relative orientation creates a stereo model that contains 3D information with local coordinates. Relative orientation can be achieved when the images are taken on a perfect line of flight direction or y-parallax is equal to zero. To calculate the y-parallax, P_y = y' – y", which indicates the difference between the feature position and the flight direction. However, in real practice, the zero y-parallax is never a case of reality. Thus, relative orientation is needed. To calculate the orientation, [23] suggested using rotation matrices. The following Fig. 3 shows the two vectors (O1, P) and (O2, P). When two images are intersected with each other, it means that the y-parallax is zero. Small errors will occur on the x-parallax, and it will affect the stereo model's height. Based on this, the coordinates can be obtained by using the trigonometry calculations as follows.

$$x: \frac{x}{x'} = \frac{z}{-c} \rightarrow x = x' \times \frac{z}{-c} \quad (1)$$

$$y: \frac{y}{y'} = \frac{y}{y''} = \frac{z}{-c} \rightarrow y = y' \times \frac{z}{-c} = y'' \times \frac{z}{-c} \quad (2)$$

$$z: \frac{z}{-c} = \frac{b}{x' - x''} \rightarrow z = -c \times \frac{b}{x' - x''} \quad (3)$$

Since $P_x = x' - x''$ then:

$$z = -c \times \frac{b}{P_x}$$

Substituiting *z* in Equations (1) and (2) gives:

$$x = x' \times \frac{b}{p_x} \; and \; y = y' \times \frac{b}{p_x}$$

Based on the equation, camera constant c represents *z* coordinate and variable *b* is the distance from each camera center (see Fig. 3).

In this study, the commercialized cloud system, eyesCloud3D, is used to process the images and turn them into 3D point clouds. eyesCloud3D is a cloud server-based system that processes images and videos using the photogrammetry concept to produce 3D point clouds automatically. The result is highly accurate and can be used in various applications. A study by [24] shows that eyesCloud3D can be used with various devices and mobile phone models.

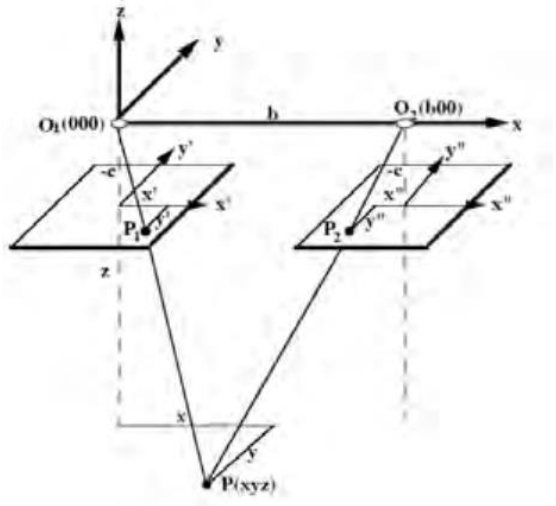


Figure 3. Orientation of two images [23].

The study produced a comparative analysis using various tools to produce 3D models. The following Fig. 4 shows the web interface of eyesCloud3D. Since the cloud server could only produce 50 images at a time, the images are processed batch by batch for all ten areas. It took less than 24 hours to complete the process. There are ten groups of outputs produced by eyesCloud3D. These outputs need to be merged and require further processing, such as noise removal, classification, and segmentation

for analyses. However, this study only focuses on organizing these datasets in the database. Fig. 5 shows the output (point clouds) from the processed images. Each area produced a different number of point clouds. The generated number of point clouds depends on the number of images, quality of the image, size of the area, and overlap percentage. For example, Area 3 and Area 10 (see Fig. 5) produced the highest number of point clouds due to the size area and total number of images produced by the UAV. For Area 3, 115 images are captured and for Area 10, 121 images are captured. The total generated point clouds for the whole study area are 103,996,984 points.

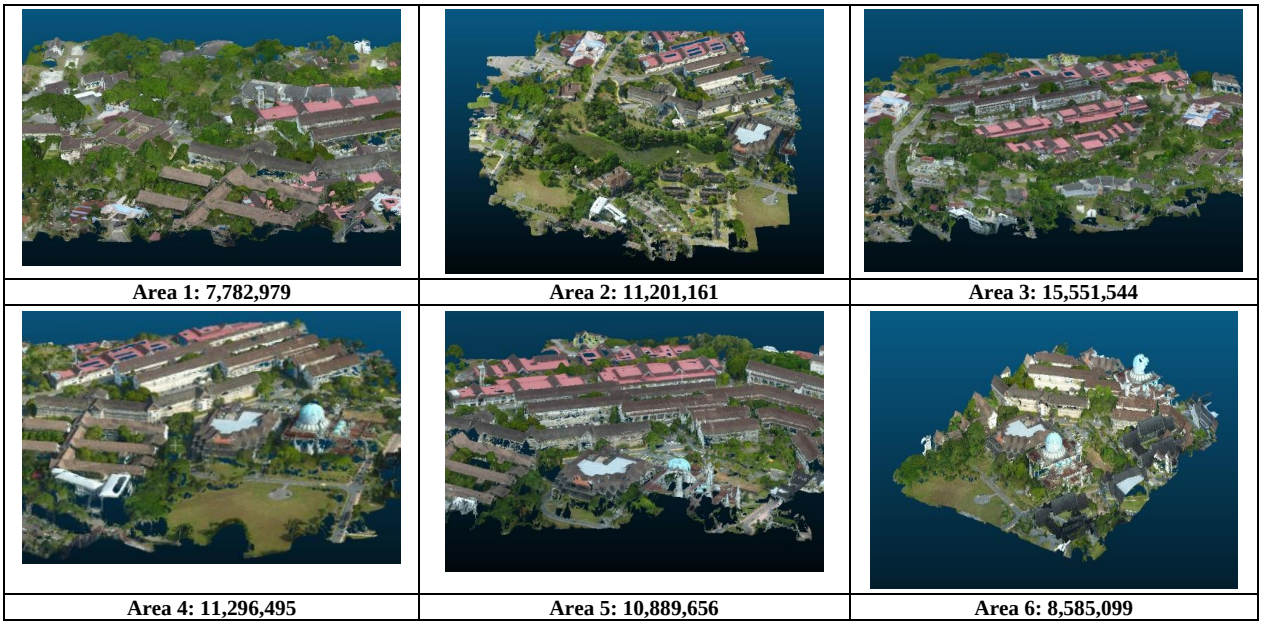
Figure 4. eyesCloud3D web interface.

C. Populating Data into Document-Oriented Database

In this study, there are two types of data stored in the document-oriented database, which are images and 3D point clouds. These datasets are stored in MongoDB, one of the popular document-oriented databases. Images and point clouds are stored differently in MongoDB. The images are stored using programmatic access due to the maximum file limit by MongoDB, which is 255 kB for each file size. Thus, the GridFS function is used to store bigger file sizes. The files will be divided into chunks and each chunk is stored separately. Each chunk is limited to 255 kB in size. This means that the last chunk is normally either equal to or less than 255 kB. When reading from GridFS, the driver reassembles all the chunks needed. This means that sections of a file can be read as per query range. The scenario is more or less like listening to a segment of an audio file or fetching a section of a video file. Meanwhile, for the point cloud dataset, data has been added using Mongo Compass Community. Using MongoDB Compass, the data can be added into a document or file. It also can be used to create a new database and collection. Using MongoDB compass, the data can be in JSON or CSV format. In this study, point cloud data are converted into CSV format and can be viewed directly. The following code shows the python code for programmatic access using GridFS and the following Fig. 6 shows stored point cloud in MongoDB.

D. Data Rendering

In this study, stored point clouds in the database are rendered using the Potree web platform. Potree is a free open-source WebGL plugin based on a point cloud renderer [25]. Point clouds need to be converted into an octree structure for rendering the point cloud within Potree. The converter converter.sln needs to be compiled the with Visual Studio and run to generate the output file. It will generate three output files, including octree.bin, metadata.json, and hierarchy.bin.



Area 1: 7,782,979	**Area 2: 11,201,161**	**Area 3: 15,551,544**
Area 4: 11,296,495	**Area 5: 10,889,656**	**Area 6: 8,585,099**

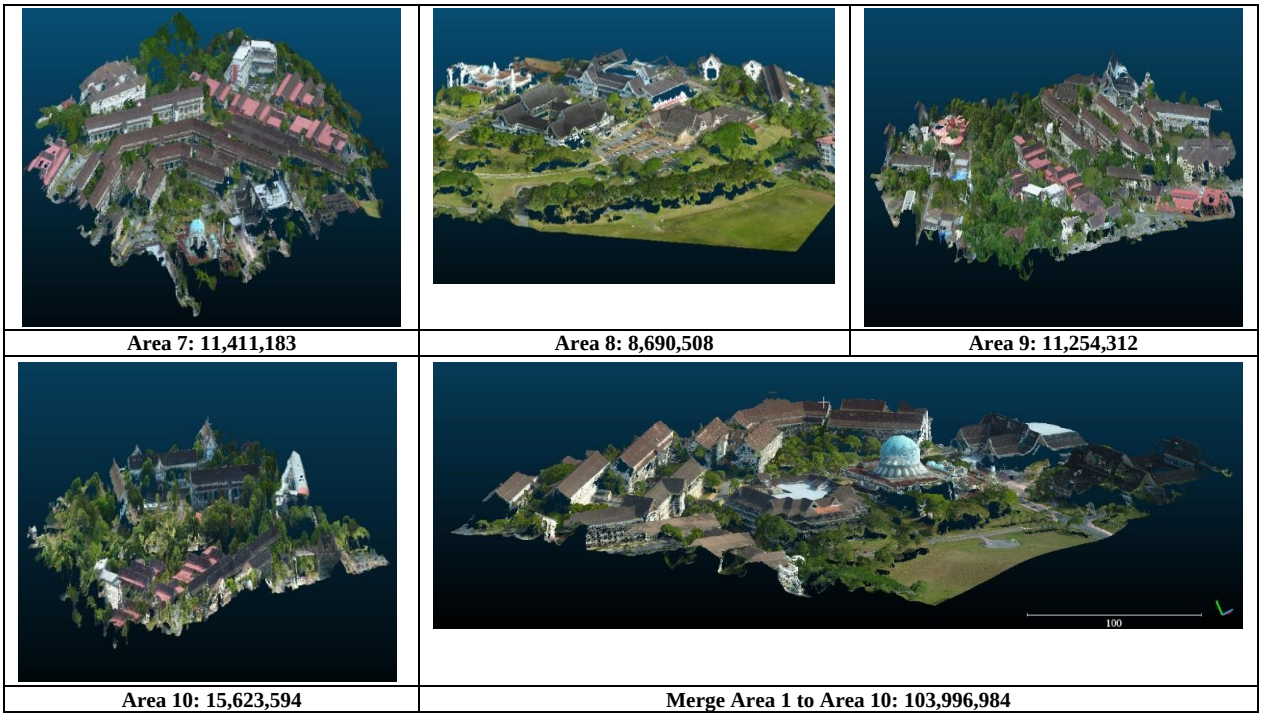


Figure 5. Generated point clouds from the image.

```
From pymongo import MongoClient
Import gridfs

#connect to database
Client = MongoClient ('mongodb://localhost:27017')
Db = client.data_psm
Print ("connected to MongoDB successfully!")

#create a new gridfs object
Fs = gridfs.GridFS(db, collection = 'uavimg')

{"_id":ObjectId("6091f834bd2f50f0fee2b2db"),
  "geometry": {
        "type": "Point",
        "coordinates": [
              101.43946226515568,
              3.048990563388066,
             6.048990563388066
        ]
  }
```

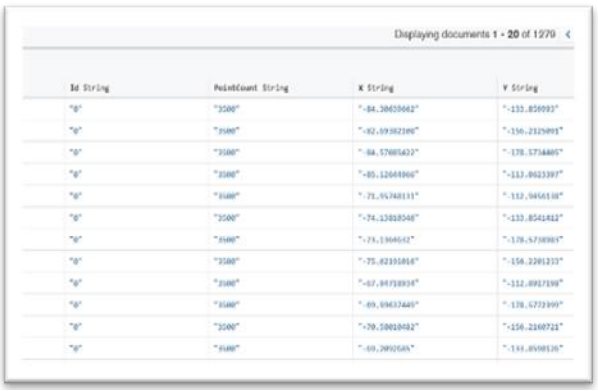


Figure 6. Stored point clouds in MongoDB.

To render the point cloud in a web-based platform, a connection to the database needs to be established using Node.js. A build server with Node.js is used to initiate the Potree renderer on the localhost by uploading the Potree folder with all point clouds and HTML files to the web server. Some other JavaScript libraries or API are also imported through Node.js with the command "npm i" to achieve the function. The node module or JavaScript library, express, ejs, mongoose, multer, multer-gridfs-storage, gridfs-stream, methode-override, bodyparser. The overall framework between Potree, Node.js, and MongoDB can be described in the following Fig. 7.

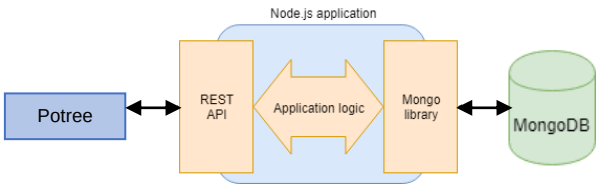


Figure 7. Node.js, MongoDB and Potree framework.

E. Measuring the Performance

Measuring the performance of databases is important to determine its stability, speed, and responsiveness. NoSQL is known as a cross-platform, document-oriented database that provides high performance, high availability, and easy scalability. NoSQL works on the concept of collection and document. Thus, it is important to identify its ability in handling UAV outputs. In this study, the Apache JMeter application is used to record the execution time for each comparison. JMeter is an open-source software. It is developed 100% based on the Java application platform and it is designed to load test

functional behaviors and measure database performance. Originally, it was designed for testing web applications, but has since expanded to other test functions. The following code shows the test plan script to connect JMeter and MongoDB and Fig. 8 shows the example of application performance index by JMeter and the test plan.

```
import com.mongodb.*
import com.mongodb.client.MongoClients;
import com.mongodb.client.MongoClient;
import com.mongodb.MongoClientSettings;
import com.mongodb.ServerAddress;
import com.mongodb.client.MongoCollection;
import com.mongodb.client.MongoDatabase;
import org.bson.Document;
import java.util.Arrays;try {
 MongoClientSettings settings =
MongoClientSettings.builder()
  .applyToClusterSettings {builder ->
   builder.hosts(Arrays.asList(new
ServerAddress(vars.get("mongoHost"),vars.get("mongoP
ort").toInteger())))}
  .build();

 MongoClient mongoClient =
MongoClients.create(settings);

 MongoDatabase database =
mongoClient.getDatabase(vars.get("databaseName"));
 MongoCollection<Document> collection =
database.getCollection(vars.get("collectionName"));

 vars.putObject("collection", collection);

 return "Connected to " + vars.get("collectionName");
}
catch (Exception e) {
 SampleResult.setSuccessful(false);
 SampleResult.setResponseCode("500");
 SampleResult.setResponseMessage("Exception: " + e);
}
```

There are three metrics used in this study to measure the database performance, which are storage comparison, update, and insert operations. These metrics are important for point cloud data organization. The insert operation performance is measured to see how the database responds to any updates. For example, in certain cases, the size of the study area will be changed for certain reasons, and it requires an additional UAV flight plan. This additional output from the UAV needs to be updated in the database. Thus, by having an insight into insert and update operations, expected specifications on the hardware can be made prior to the project development. These steps are important to reduce additional costs and at the same time, it may help identify problems during development. The same reason is applied to storage comparison. It is important to identify which database offers minimal and cost-effective data storage. Massive point cloud datasets require efficient storage space. Thus, in this study three metrics are used to compare the database performance efficiency between NoSQL database (MongoDB) and relational database (PostgreSQL). The following Section IV describes each test and the results.

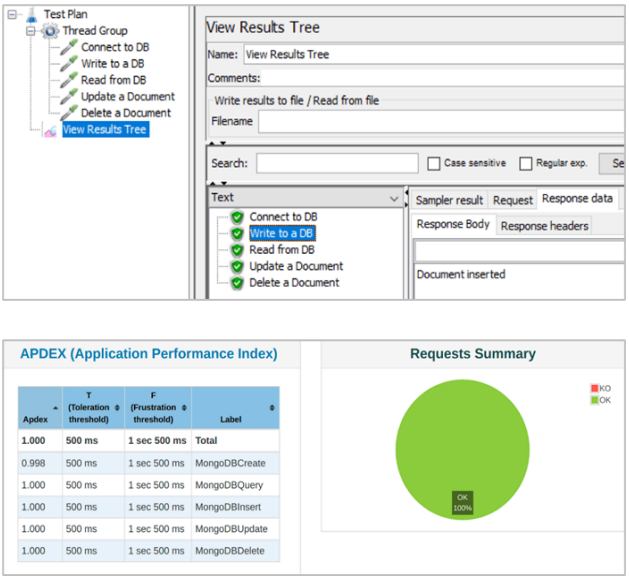


Figure 8. MongoDB test plan and application performance index.

IV. Results and Discussion

To analyze the efficiency of the document-oriented database in handling outputs from the UAV, such as images and point clouds, data are also stored in the relational database, PostgreSQL, for comparison purposes. Further analysis on data updating, such as insertion and data retrieval, is analyzed to measure the performance of both databases in handling UAV outputs, including the storage capacity.

A. Data Updating - Insert Operation

To test the time complexity on data updating for both databases, this study has performed an experiment based on data updating. By using ten groups of data (from Fig. 5), the execution time for the insert operation is recorded. The first group of point clouds (7,782,979 number of points) is stored in the NoSQL (MongoDB) and relational database (PostgreSQL). The execution times for both databases are recorded. Then, the second group of data, which is 11,201,161 points, is inserted into both databases and the times are recorded. This step is repeated for all ten groups of datasets.

To compare the execution time for each group and database, the results are profiled in Fig. 9. From the figure, it is observed that both the NoSQL and relational databases show an increasing trend of time execution for the insert operation. This is due to the number of data increasing for each cycle of operation. It is also identified from the profile that major spikes happen during the insertion of Group 3 and Group 10 for the relational database. The spikes happen due to the high number of point clouds loading to the database, which are 15,551,544 for Group 3 and 15,623,594 for Group 10.

MongoDB, by design, tries to keep all of its data in memory and relies on the Operating System (OS) to swap the memory-mapped files. Thus, when large documents are stored in NoSQL, it will occupy more memory and affect the time performance. However, NoSQL profiling shows that the trend is increasingly proportional, even though a high number of points are loaded for Group 3 and Group 10. Based on these findings, it is believed that this situation happened due to a non-schematic procedure offered by the document-oriented database (NoSQL). Meanwhile, for the relational database (PostgreSQL), operations such as data insertions consume more time due to schematic rules and conditions. Data need to be organized based on structures such as tables, key identification, and other table condition. Users also tend to get some errors due to unmatched data types in the same field. This explains why schematic structure applied by relational database consume more time compared to document-oriented database. Without schematic structure data are dumped into document-oriented database as a file and document.

In terms of time efficiency for data updating, the results indicate that NoSQL (MongoDB) is more timesaving compared to the relational database (PostgreSQL). From Fig. 8, the result shows that even though NoSQL and relational databases are loaded with the same number of point clouds, the execution time for the NoSQL database is lower than the relational database. NoSQL has better performance in terms of time for insert operation, where the insertion time is 30% faster than the relational database. Thus, it can be concluded that NoSQL has better performance for point cloud data updating. This is an important key point in organizing UAV outputs since huge data will be produced, especially for large scale projects.

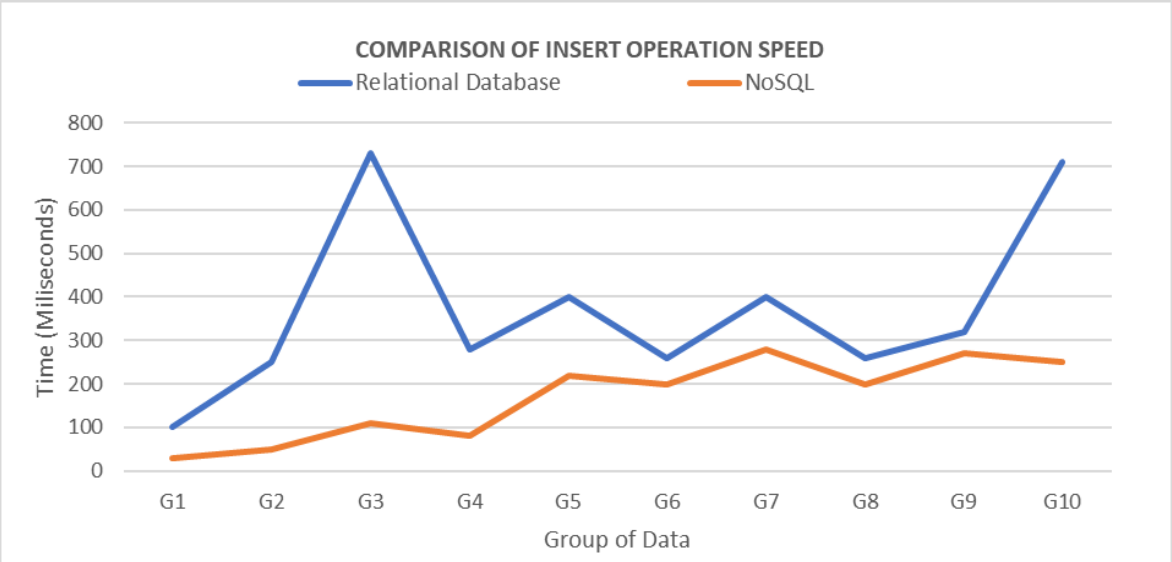


Figure 9. Comparison of insert operation speed (millisecond).

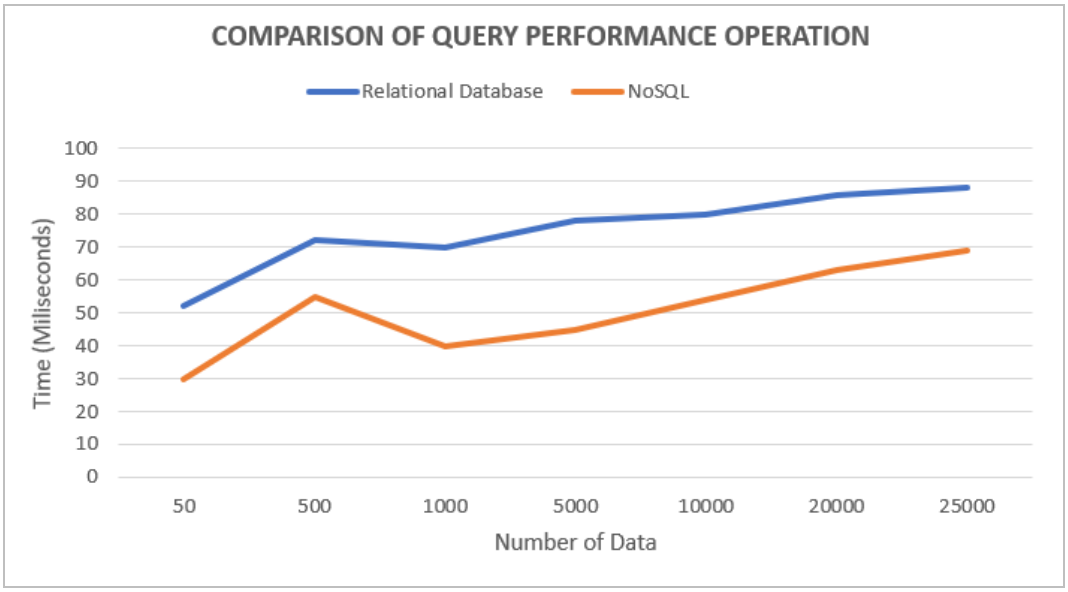


Figure 10. Comparison of query operation speed (millisecond).

B. Query Operation

To analyze the efficiency of both databases in retrieving point clouds, this study performs a comparative analysis for query performance speed. All datasets from Group 1 to Group 10 are loaded into the NoSQL and relational databases. The total number of point clouds loaded in the databases is 103,996,984. Several samples of point clouds are retrieved from the database and the response time is recorded. There are seven samples with different numbers of point queries used in this experiment, which are 50, 500, 1,000, 5,000, 10,000, 20,000 and 25,000. Each sample is queried explicitly for each database and the response time is recorded.

To compare the response time for each sample, the response times are plotted in Fig. 10. From the figure, it is shown that the trend of response time for the NoSQL database and the relational database are increasing proportionally to the number of point queries. However, in terms of performance, NoSQL outperforms the relational database, where the response time is 37% faster. Even when the size of data increases, NoSQL maintains its performance. This is happening due to the information processing of a relational database. Relational databases require much more time to process information, which causes slow data retrieval. The performance of NoSQL improves further since the data is directly retrieved from volatile memory, unlike relational databases that retrieve data from non-volatile memory. By design, volatile memory is faster than non-volatile memory. This explains the results of query performance for both databases.

Even though the relational database could not offer better performance than NoSQL, it is identified in this study that the relational database has a strong foundation on the query language, which is Structured Query Languages (SQL). SQL is the only data manipulation language that is widely used in all relational databases. On the other hand, this foundation still lacks in NoSQL, as it relies on object-oriented API for data manipulation. Thus, it is important to know what kind of query that needs to be performed on the UAV outputs before organizing the data in specific databases.

C. Storage Comparison

Another factor that needs to be considered in handling UAV outputs is storage consumption since the device produces large data sizes. As mentioned previously, NoSQL limits the file size for storing the files. The limit is 16 MB of storage file. In this study, point cloud data with the images are stored in the database and require a larger size than 16 MB. Thus, the GridFS function is used to overcome the limitations. By using GridFS, the data can be stored in several files as required. Moreover, it is possible to retrieve a particular information from massive data files without necessarily loading huge documents in the memory. The image data is stored using GridFS and divided into chunks. All the chunks are stored in a separate document with a maximum of 255 kB from the last chunk. For relational databases, issues in storing point clouds are manageable, but for UAV images, storage is limited to 1 GB.

To compare the response time for each sample, the response times are plotted in Fig. 10. From the figure, it is shown that the trend of response time for the NoSQL database and the relational database are increasing proportionally to the number of point queries. However, in terms of performance, NoSQL outperforms the relational database, where the response time is 37% faster. Even when the size of data increases, NoSQL maintains its performance. This is happening due to the information processing of a relational database. Relational databases require much more time to process information, which causes slow data retrieval. The performance of NoSQL improves further since the data is directly retrieved from volatile memory, unlike relational databases that retrieve data from non-volatile memory. By design, volatile memory is faster than non-volatile memory. This explains the results of query performance for both databases.

Even though the relational database could not offer better performance than NoSQL, it is identified in this study that the relational database has a strong foundation on the query language, which is Structured Query Languages (SQL). SQL is the only data manipulation language that is widely used in all relational databases. On the other hand, this foundation still lacks in NoSQL, as it relies on object-oriented API for data manipulation. Thus, it is important to know what kind of query that needs to be performed on the UAV outputs before organizing the data in specific databases.

D. Storage Comparison

Another factor that needs to be considered in handling UAV outputs is storage consumption since the device produces large data sizes. As mentioned previously, NoSQL limits the file size for storing the files. The limit is 16 MB of storage file. In this study, point cloud data with the images are stored in the database and require a larger size than 16 MB. Thus, the GridFS function is used to overcome the limitations. By using GridFS, the data can be stored in several files as required. Moreover, it is possible to retrieve a particular information from massive data files without necessarily loading huge documents in the memory. The image data is stored using GridFS and divided into chunks. All the chunks are stored in a separate document with a maximum of 255 kB from the last chunk. For relational databases, issues in storing point clouds are manageable, but for UAV images, storage is limited to 1 GB.

To test the storage consumption for point clouds, the disk size is assessed for each group of data (Group 1 until Group 10). Fig. 11 shows the comparison of storage sizes between the relational database and NoSQL (MongoDB). Based on the results in Fig. 11, NoSQL (MongoDB) consumes larger storage size compared to the relational database. The reason for this behavior is that the data stored in NoSQL (MongoDB) are in GeoJSON format and each record consists of many extra characters and an auto-generated unique id. Compared to the relational database, the data stored are in CSV format, which is smaller than GeoJSON. However, the difference in size consumption between these two databases is very small

and can be considered as the same size in the storage consumption.

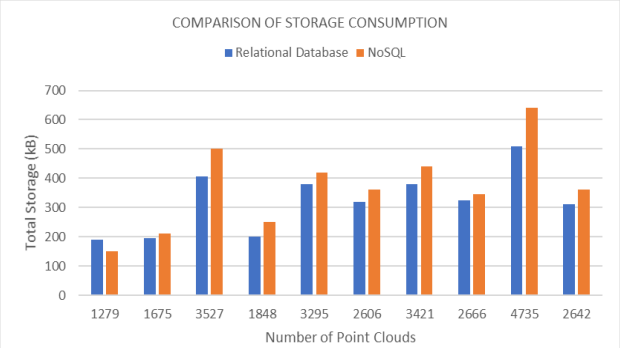
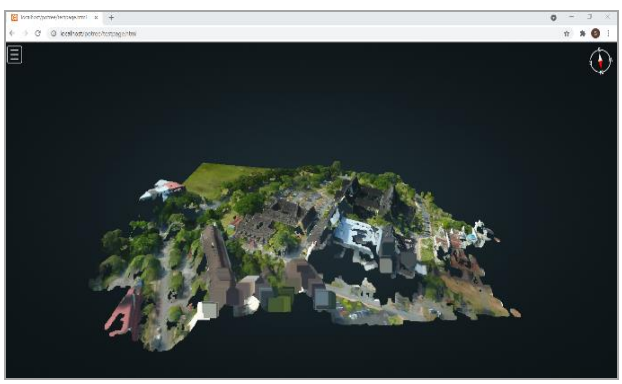


Figure 11. Comparison of storage size.

E. *Visualization of Point Clouds*

Standard visualization tools or viewers provided by the document-oriented database (NoSQL) are not capable of visualizing the 3D objects or dataset. Therefore, this study utilized a free open-source WebGL plugin based on a point cloud renderer called Potree. The data organization has been modified from the beginning until the latest, but based on Modifiable Nested Octree (MNO), which is a hierarchical model that supports level-of-detail rendering and improves rendering performance effectively. Several rendering techniques, including WebGL, node.js, cesium.js, and three.js, are applied with Potree to develop the rendering tool. Document-oriented databases serve as a database server to store and manage point cloud data. With WebGL and three.js, Potree viewer can successfully render the point cloud. Point cloud data is read and piped with the gfs.createReadStream function to a url. The following Fig. 12 shows the result of point cloud rendering in the Potree viewer. The available viewer comes with several features for point cloud manipulation. Users are also allowed to do some customization according to their needs. Based on the available features, direct measurements such as area and distance, can be performed on the rendered point cloud. However, direct query from the viewer to the database is not performed in this study. The query has to be performed on the document-oriented database. Further application customization is required to process this operation.



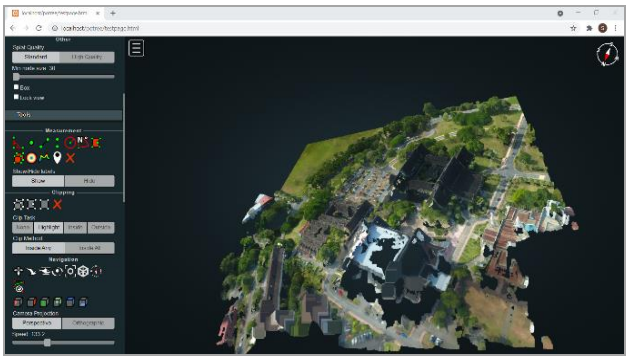


Figure 12. Point cloud rendering on the web browser.

V. Conclusion

In conclusion, the Relational Database (RDBMS) technique used in most database systems is seen as inadequate in terms of its implementation for point cloud and image data from a UAV. In any relational database, the database schema must be created first before inserting the data into the database system. For example, tuples need to be constructed to classify the ID, X coordinate, Y coordinate, Z coordinate, as well as the RGBD information for the point cloud. Next, these tuples need to establish their relationship with other tuples in other tables. Based on the point cloud data characteristics, this is seen as unnecessary because the point cloud data are massive. It does not require any relationship with other tables or tuples. Besides that, the performance of query operation using NoSQL (MongoDB) database outperforms the relational database (PostgreSQL). This is proof that document-oriented databases retrieve data efficiently using document storage styles. It is proven fast, effortless, and more practical in executing data updating and data queries without affecting the storage size of the data used. Even though it consumes more data storage than the relational database (PostgreSQL), the percentage is still relatively small, which is around 2% to 3% storage size. Therefore, the document-oriented data organization technique described in this study clearly provides an advantage in addressing this weakness. For future recommendations, we propose to manipulate the UAV outputs for further analysis, such as neighboring analysis, and organize the outputs based on NoSQL data model, such as aggregation.

Conflict of Interest

The authors declare no conflict of interest.

Author Contributions

Suhaibah Azri, Uznir Ujang and Wan Afifah Wan Embong contributed equally to the conceptualization of the manuscript. Methodology and validation were contributed by Suhaibah Azri and Wan Afifah Wan Embong. Meanwhile, the formal analysis and investigation section were contributed by Suhaibah Azri and Uznir Ujang. The Original Draft Preparation was prepared by Suhaibah Azri and writing continuation and

review was prepared by Uznir Ujang. Miguel Gonzalez Cuetara and Guillermo Miguel provided the methodology and tools for image processing.

ACKNOWLEDGEMENT

This research was partially funded by UTM Research University Grant, Vot Q.J130000.3652.02M78 and Vot Q.J130000.2652.15J95.

REFERENCES

[1] L. Hinge, J. Gundorph, U. Ujang, S. Azri, F. Anton, and A. Rahman, "Comparative analysis of 3d photogrammetry modeling software packages for drones survey," *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. XLII-4/W12, pp. 95-100, 2019.

[2] A. Hairuddin, S. Azri, U. Ujang, M. G. Cuétara, G. M. Retortillo, and S. M. Salleh, "Development of 3D city model using videogrammetry technique," *Int. Arch. Photogramm. Remote Sens. Spatial Inf. Sci.*, vol. XLII-4/W16, pp. 221-228, 2019.

[3] S. Anuar, *et al.*, "3D geometric extraction using segmentation for asset management," *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. XLIV-4/W3-2020, pp. 61-69, 2020.

[4] S. Azri, U. Ujang, F. Anton, D. Mioc, and A. Rahman, "Review of spatial indexing techniques for large urban data management," in *Proc. International Symposium & Exhibition on Geoinformation*, 2013.

[5] X. Xudong and G. Rui, "Research on storage and processing of mongodb for laser point cloud under distribution," in *Proc. the 3rd International Conference on Materials Engineering, Manufacturing Technology and Control*, 2016.

[6] S. Agarwal and K. Rajan, "Analyzing the performance of NoSQL vs. SQL databases for Spatial and Aggregate queries," *Free and Open Source Software for Geospatial (FOSS4G) Conference Proceedings*, article 4, 2017.

[7] M. Abourezq and A. Idrissi, "Database-as-a-Service for big data: An overview," *International Journal of Advanced Computer Science and Applications*, vol. 7, 2016.

[8] A. Zaki, "NoSQL databases: New millennium database for big data, big users, cloud computing and its security challenges," *International Journal of Research in Engineering and Technology*, vol. 3, pp. 403-409, 2014.

[9] W. Kim, "Web data stores (aka NoSQL databases): A data model and data management perspective," *International Journal of Web and Grid Services*, vol. 10, pp. 100-110, 2014.

[10] H. Phiri and D. Kunda, "A comparative study of NoSQL and relational database," *Zambia ICT Journal*, vol. 1, pp. 1-4, 2017.

[11] A. Singh, "NoSQL: A new horizon in big data," *International Journal of Scientific Research in Science, Engineering and Technology*, vol. 2, 2016.

[12] S. Sharma, R. Shandilya, S. Patnaik, and A. Mahapatra, "Leading NoSQL models for handling big data: A brief review," *International Journal of Business Information Systems*, vol. 22, no. 1, pp. 1-25, 2016.

[13] J. Kepner, *et al.*, "Associative array model of SQL, NoSQL, and NewSQL databases," in *Proc. IEEE High Performance Extreme Computing Conference (HPEC)*, Sept. 2016, pp. 1-9.

[14] C. Tauro, S. Aravindh, and A. B. Shreeharsha, "Comparative study of the new generation, agile, scalable, high performance NoSQL databases," *International Journal of Computer Applications*, vol. 48, pp. 1-4, 2012.

[15] A. B. M. Moniruzzaman and S. Hossain, "NoSQL database: New era of databases for big data analytics - Classification, characteristics and comparison," *Int. J. Database Theor. Appl.*, vol. 6, 2013.

[16] A. Nayak, A. Poriya, and D. Poojary, "Type of NOSQL databases and its comparison with relational databases," *International Journal of Applied Information Systems*, vol. 5, pp. 16-19, 2013.

[17] Jatin and S. Batra, "MongoDB versus SQL: A case study on electricity data," in *Emerging Research in Computing, Information, Communication and Applications*, Springer, 2016, pp. 297-308.

[18] R. Kumar, S. Charu, and S. Bansal, "Effective way to handling big data problems using NoSQL database (MongoDB)," *J. Adv. Database Manag. Syst.*, vol. 2, pp. 42-48, 2015.

[19] G. Bathla, R. Rani, and H. Aggarwal, "Comparative study of NoSQL databases for big data storage," *International Journal of Engineering & Technology*, vol. 7, 2018.

[20] W. Ali, M. Majeed, A. Raza, and M. U. Shafique, "Comparison between SQL and NoSQL databases and their relationship with big data analytics," *Asian Journal of Computer Science and Information Technology*, vol. 4, pp. 1-10, 2019.

[21] R. Gunawan, A. Rahmatulloh, and I. Darmawan, "Performance evaluation of query response time in the document stored NoSQL database," in *Proc. 16th International Conference on Quality in Research (QIR): International Symposium on Electrical and Computer Engineering*, 2019.

[22] M. I. Hossain, S. Mahmud, and T. D. Santa, "Oracle, MySQL, PostgreSQL, SQLite, SQL Server: Performance based competitive analysis," project report, Faculty of Information Technology, Daffodil International University, 2019.

[23] A. Boberg, "Digital 'height models'," in *Geodetic Photogrammetric Measurement Calculation Technique*, Stockholm: Lanmäteriet, 2013, pp. 243-245. (In Swedish)

[24] N. Ahmad, S. Azri, U. Ujang, M. Cuétara, G. Retortillo, and S. Salleh, "Comparative analysis of various camera input for videogrammetry," *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. XLII-4/W16, pp. 63-70, 2019.

[25] M. Schütz, "Potree: Rendering large point clouds in web browsers," doctoral dissertation, Technical University of Vienna, Vienna, 2016.

Suhaibah Azri is currently a Senior Lecturer at Geoinformation, Faculty of Built Environment and Surveying, Universiti Teknologi Malaysia (UTM). Prior to her recent appointment at the UTM, she was a post-doctoral researcher at Technical University of Denmark (DTU). Dr. Suhaibah Azri received her PhD in 3D GIS from UTM. She is also an active member of International Society of Photogrammetry and Remote Sensing (ISPRS). Her research activities are currently twofold: while the first research activity is set to explore the other type from relational database to organize geospatial data; the second major research theme that she is pursuing is focused on 3D GIS and spatial analysis with various applications especially urban planning, smart cities etc.

Uznir Ujang, Head of 3D GIS Research Group, Faculty of Built Environment and Surveying, Universiti Teknologi Malaysia (UTM). Interest in 3D city modelling, 3D GIS, topology, and Geographical Information Science (GISc). He published more than 90 publications in international journals, conferences, and books. Actively involved with scientific committee activities all around the globe and being acknowledged by having more than 150 publications reviewed within 45 journals, books, and conferences worldwide. He is an active member of the International Society for Photogrammetry and Remote Sensing (ISPRS), the Royal Institutions of Surveyors Malaysia (RISM), Institution of Geospatial and Remote Sensing Malaysia (IGRSM) and Professional Technologist at Malaysia Board of Technologist.

Wan Nur Afifah Wan Embong is a GIS Associates at private company in Malaysia. She received her Undergraduate Degree in GIS from Universiti Teknologi Malaysia (UTM) in 2020 and her Diploma Degree in Land Survey in 2017. Her research interest is more on non-relational database and point cloud data organization. She is also has been involved with several industrial projects related to GIS data management and analysis.

Miguel Gonzalez Cuetara is a CEO of the eCapture3D company. He has a more than 20 years of professional experience in working with different companies nationally and internationally. Currently, eCapture focus on their latest product eyescloud3D Web platform for the automatic generation of 3D models through photos and / or videos, from any camera, including smartphones.

Guillermo Miguel Retortillo is a Technical Director at eCapture3D, with studies in Computer Engineering and in Management and Direction of Technology-Based Companies. He is also a technical coordinator of R&D projects at regional, national and European level, working with research centers and public and private entities in areas focused on 3D Technology, Computer Vision, Artificial Intelligence, Web Technologies and Cloud Computing.