Ilkogretim Online - Elementary Education Online, 2021; 20 (3): pp. 1160-1166
http://ilkogretim-online.org
doi: 10.17051/ilkonline.2021.03.128

Development of a human-motion database for intelligent wearable robots

Yoon-Jae Chae, *Researcher, Computer Engineering, Sunmoon Univ., ASAN, KS002, Republic of Korea, kodeep11@gmail.com*
***Kyung-Oh Lee,** *Professor, Computer Engineering, Sunmoon Univ., ASAN, KS002, Republic of Korea, leeko@sunmoon.ac.kr*
Yoon-Yong Park, *Professor, Computer Engineering, Sunmoon Univ., ASAN, KS002, Republic of Korea, yypark@sunmoon.ac.kr*
In-Ae Kang, *Master Student, Computer Engineering, Sunmoon Univ., ASAN, KS002, Republic of Korea, inae5004@gmail.com*
Ji-Bum Jung, *Master Student, Computer Engineering, Sunmoon Univ., ASAN, KS002, Republic of Korea, 2377325@gmail.com*
*Corresponding Author

Abstract. Human-motion data need to be stored and processed to manage wearable robots. Given the considerable data volume associated with sensor data, we designed a database that efficiently stores sensor data from human motion and can be used by intelligent wearable robots. We designed a database for sensor data from human motions for use in intelligent wearable robots. We used functions to preprocess and transform the data into JSON files. Additionally, we created a function to store and process data files in the database. A database of human-motion data is necessary to determine whether robot movements (e.g., by wearable robots or robot simulations) are similar to those of humans. The most common methods for collecting motion data from people have been markerless optical systems. However, these methods are less accurate than motion capture using a non-optical system. We designed and implemented a database to collect and store accurate human-motion data using a non-optical inertial motion-capture system. Our database can be applied to walking robots used for people with paraplegic disabilities who need walking assistance, and to robots and equipment that mimic human movements for people working in high-risk environments. Our system can also be applied to humanoid robotics.

Keywords: human-motion data, inertial motion capture, JSON, robot simulations, database

Received: 08.12.2020 Accepted: 13.01.2021 Published: 05.02.2021

INTRODUCTION

Movement dysfunction is becoming a major issue in aging populations across the world as quality of health improves and individuals live longer. Pollution in big cities is a major challenge, and related congenital disorders influencing somatosensory and motor control disorders in newborn children have led to mobility challenges. Assistive-gait wearable robots are being studied to address these disorders[1], as well as devices and wearable robots for paraplegic patients [2, 3]. Cloud databases are increasingly used to store data generated by such hardware[4]. Although data for robotic movement are important, human-motion data are also critical because all of these solutions are deeply entwined with people. Therefore, it is necessary to collect and store data on human movement using motion-capture equipment and apply the results to robots. There are two different methods for capturing human movement: optical and non-optical. Optical systems use data captured from image sensors to triangulate the 3D position of a subject between two or more cameras calibrated to provide overlapping projections. Among the most popular optical systems, markerless methods were created through the development of computer vision technology and capture motion using various imaging analysis techniques[5]. These systems typically use Microsoft's Kinect sensor [6]. Markerless systems are cheaper and easier to access than other motion capture methods. However, human-related equipment, such as wearable or assistive-walking robots, can be dangerous to users if the data are not accurate. Therefore, we collected human-motion data using non-optical inertial motion-capture equipment and preprocessed the data. Then, the data were formatted as JavaScript Object Notation (JSON) files and stored in a database. We investigated how motion data stored in a database as big data can be applied to intelligent wearable robots. The rest of the paper is as follows: section 2 summarizes related literature; section 3 outlines the design of the human-motion database for intelligent wearable robots; and section 4 summarizes our findings.

1. Related Work

2.1. Big data–based database

Big data are generated not only by the internet, mobile phones, and social media but also in other areas including healthcare and finance. Strong calculation techniques are needed to deduce the patterns of such data. Sensor data are especially diverse and high in volume; therefore, sensor data should be approached like big data[7]. We used the NoSQL database program MongoDB to store the data. MongoDB is a cross-platform document-oriented database and uses the same documented schema as JSON. The main function of a database is to return specific document fields through temporary queries and to support regular-expression searches. Databases also provide high data availability through replication. MongoDB is a good choice for JSON because this file format is often used to process big data and requires a database that can manage and store these types of files well[8].

2.2. JSON

JSON is an open standard file and data interchange format that uses human-readable text to store and transmit data objects consisting of attribute–value pairs and array data types (or any other serializable value). It is a very common data format, with a diverse range of applications, such as serving as a replacement for XML in AJAX systems[9]. When used in MongoDB, JSON is called binary JSON (BSON). BSON is a computer data interchange format. It is a binary form for representing simple or complex data structures, including associative arrays (also known as name–value pairs), integer-indexed arrays, and a suite of fundamental scalar types. BSON originated in 2009 at MongoDB. Several scalar data types are of particular interest as transfer formats for MongoDB, but BSON can be used independently of MongoDB. Relative to JSON, BSON is designed to be efficient both in terms of storage space and scan speed. Large elements in a BSON file are prefixed with a length field to facilitate scanning[10]. Motion data have inherently high volume due to data from different body parts. Therefore, motion-sensor data should be stored and managed as JSON types to facilitate analyzing such large amounts of data.

2.3. Human-motion database

To communicate and interact with humans using gestures, humanoid robots need to both act and look like humans. A previous study used humanoid robots to create human-like arm movements in real time and collected motion data with a markerless optical system; however, the accuracy of the data was too low to produce the desired movements[11]. Another study measured swimming behavior using gyro sensors, identified regular and irregular data, stored them in motion-capture databases, and compared swimming behavior between professional athletes and ordinary people[12]. However, although inertial sensors were used, the number of inertial motion sensors was lower than other inertial motion sensors, resulting in lower data extraction. Other studies have analyzed motion-capture data stored in databases using machine learning[13, 14]; such studies have also collected motion data using optical systems, resulting in inaccurate and unreliable data. To date, most studies of human motion using databases have stored data captured with markerless optical systems. Even when the desired motion data were collected or achieved a high recognition rate through data processing, the resulting data were less accurate than inertial motion capture data[11, 14, 15, 16]. Therefore, we used a non-optical inertial motion capture system to collect and process motion data, achieving greater accuracy than with markerless optical methods.

2. Database of human-motion data for intelligent wearable robots

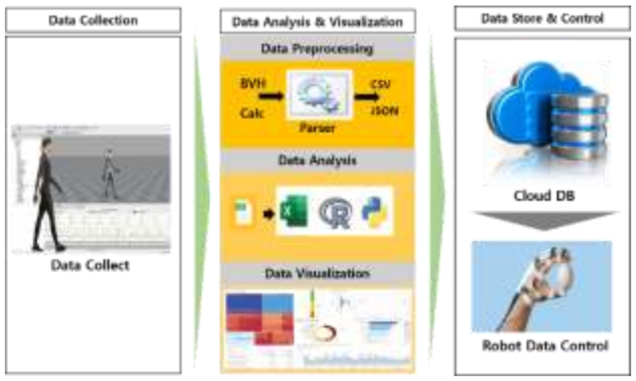


Figure 1. Data collection, processing, storage, and usage of a human-motion database for intelligent wearable robots

Figure 1 shows a general workflow for collecting, processing, and storing motion data. First, human-motion data are collected using inertial motion-capturer equipment and then cleaned and preprocessed. The preprocessed data are converted into JSON and CSV file formats. CSV files of motion data can be analyzed using Microsoft Excel, R, or Python and visualized as graphs or charts, while JSON files of motion data are stored in a cloud database and used to operate or control intelligent wearable robots.

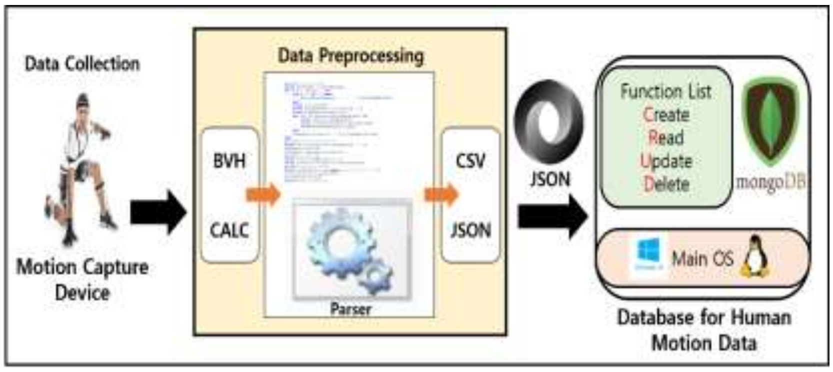


Figure 2. Design of the human-motion database intelligent wearable robots used in the present study

Figure 2 outlines our study design. The motion data were collected using an inertial motion-capture device (Perception Neuron; Notiom, Ltd, Beijing, China)[17]. The collected data were exported to BVH or calc files and preprocessed using a parser function. Data were then converted into JSON and CSV files. We stored and processed the JSON files in MongoDB using an IWR create, read, update, delete (CRUD) function.

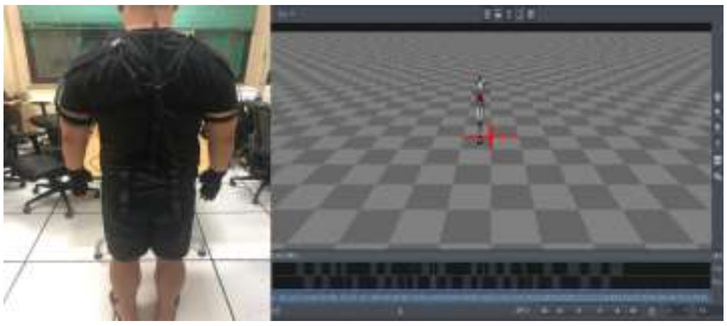
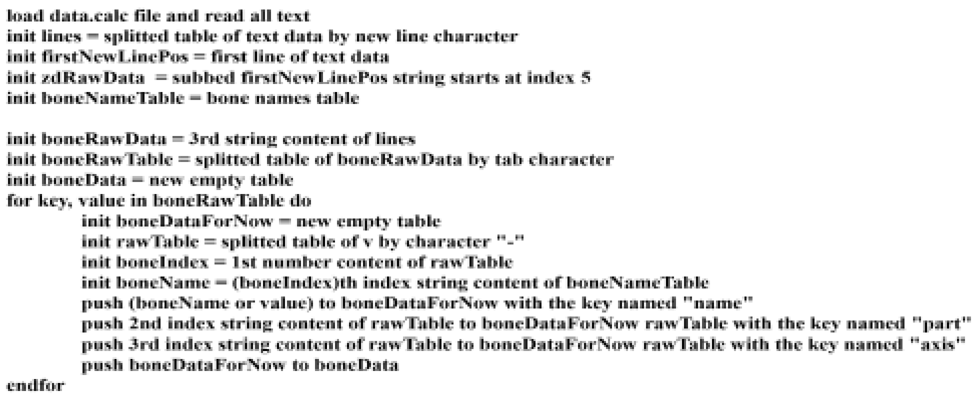


Figure 3. The motion-capture device and data collection

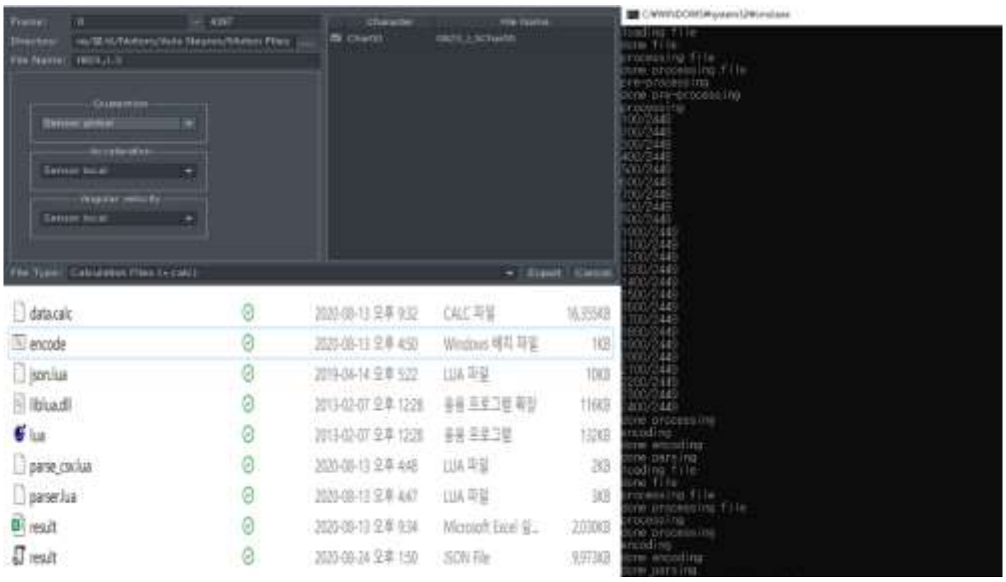
Figure 3 shows the process of wearing the motion-capture equipment and collecting data. Data collection was performed in full-body sensor mode to measure all human-motion data. Once connected to the network, the joint shape of the 3D model was captured through calibration. Then the movements of the model were captured, and the motion data were collected.



```
load data.calc file and read all text
init lines = splitted table of text data by new line character
init firstNewLinePos = first line of text data
init zdRawData  = subbed firstNewLinePos string starts at index 5
init boneNameTable = bone names table

init boneRawData = 3rd string content of lines
init boneRawTable = splitted table of boneRawData by tab character
init boneData = new empty table
for key, value in boneRawTable do
        init boneDataForNow = new empty table
        init rawTable = splitted table of v by character "-"
        init boneIndex = 1st number content of rawTable
        init boneName = (boneIndex)th index string content of boneNameTable
        push (boneName or value) to boneDataForNow with the key named "name"
        push 2nd index string content of rawTable to boneDataForNow rawTable with the key named "part"
        push 3rd index string content of rawTable to boneDataForNow rawTable with the key named "axis"
        push boneDataForNow to boneData
endfor
```

```
init frameData = new table with table embedded named "frames"
for k = 4 to length of line do
        init line = (k)th string content of lines
        init dataRawTable = splitted table of line by tab character
        init frameDataForNow = new empty table
        for key, value in dataRawTable do
                init data = numbered v2
                init boneDataForNow = (key)th string data of boneData
                push data to frameDataForNow with (key)th index
        endfor
        push (k-4)th index with frameDataForNow to frameData.frames
endfor

init Zd = splitted table of zdRawData by tab character
push Zd to boneDataForNow with the key named "Zd"
push boneData to boneDataForNow with the key named "boneData"

init jsonData = string encoded frameData by json format

write jsonData to result.json
```
Figure 4. Parser function pseudo-code used for preprocessing the data

We developed a function that deletes unnecessary data, labels the remaining data, and converts the calc file into a JSON file (pseudo-code representation in Fig 4.). First, we loaded the "data.calc" file, read the data stored in it, and separated the text data. There were three variable values: the position value, a code for five characteristics[18], and the name of each bone. Originally, raw data consisted of three strings. We then separated the "-" through an iteration statement and put each string into "name", "part", and "axis" tables. Each table was converted into a dictionary-type dataframe. The final results were formatted as JSON-type data.



```
data.calc       2020-08-13 오후 9:32    CALC 파일              16,355KB
encode          2020-08-13 오후 4:50    Windows 배치 파일          1KB
json.lua        2019-04-14 오후 5:22    LUA 파일                  10KB
liblua.dll      2013-02-07 오후 12:28   응용 프로그램 확장        116KB
lua             2013-02-07 오후 12:28   응용 프로그램             132KB
parse_csv.lua   2020-08-13 오후 4:48    LUA 파일                   2KB
parser.lua      2020-08-13 오후 4:47    LUA 파일                   3KB
result          2020-08-13 오후 9:34    Microsoft Excel 쉼...   2,030KB
result          2020-08-24 오후 1:50    JSON File              9,973KB
```

```
C:\WINDOWS\system32\cmd.exe
loading file
done file
processing file
done processing file
pre-processing
done pre-processing
processing
100/2448
200/2448
300/2448
400/2448
500/2448
600/2448
700/2448
800/2448
900/2448
1000/2448
1100/2448
1200/2448
1300/2448
1400/2448
1500/2448
1600/2448
1700/2448
1800/2448
1900/2448
2000/2448
2100/2448
2200/2448
2300/2448
2400/2448
done processing
encoding
done encoding
done parsing
loading file
done file
processing file
done processing file
processing
done processing
encoding
done encoding
done parsing
```
Figure 5. Exporting and parsing data

Figure 5 shows the results of the JSON file conversion. First, we exported the calc file and used the "encode.bat" executable file (Fig. 4) to preprocess the data and proceed with encoding. When the run was complete, the data were converted to CSV and JSON files. The CSV files were used for analyses, and the JSON files were stored in the database.

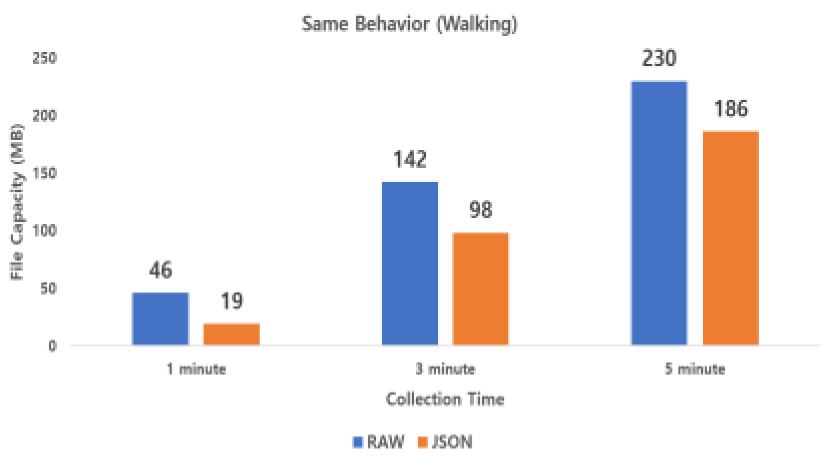


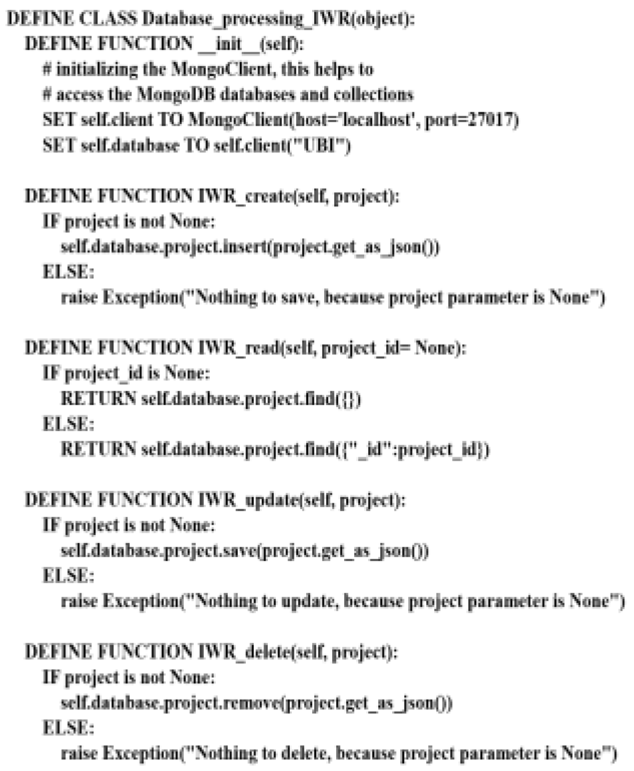
Figure 6. Comparison of file size using the parser

Figure 6 compares the sizes of resulting data files (see Fig. 5). Walking-motion data were collected at 1-, 3-, and 5-minute intervals. We compared the file sizes of raw data and the JSON files that were converted using the parser. After being compressed by the parser, the JSON files had a smaller file size than the raw data. Therefore, to reduce storage use, motion data from various body parts should be converted to JSON files rather than using typical data types.

```
DEFINE FUNCTION Store_HM_DB(db_name, collection_name, json_name):
  # Making Connection
  SET myclient TO MongoClient("mongodb://localhost:27017")

  # database
  SET db TO myclient[db_name]

  # Created or Switched to collection
  SET Collection TO db[collection_name]

  # Loading or Openig the json file
  with open(json_name) as file:
    SET file_data TO json.load(file)

  # Inserting the loaded data IN the Collection
  # If JSON contains data more than one entry
  # insert_many is used else insert_one is used

  IF isinstance(file_data, list):
    Collection.insert_many(file_data)
  ELSE:
    Collection.insert_one(file_data)

SET if __name__ TO '__main__':
  SET db_name TO "IWR_HM"
  SET collection_name TO "J_Walk_Square"
  SET json_name TO "J_Walk_square.json"
  Store_HM_DB(db_name, collection_name, json_name)
```

Figure 7. Pseudo-code for the database storage function

Figure 7 shows the pseudo-code for storing JSON files in the database. The function "Store_HM_DB" had parameters "db_name," "collection_name," and "json_name" and set the client port number as well as the database and collection. Finally, JSON files were read and saved to the collection.



```
DEFINE CLASS Database_processing_IWR(object):
  DEFINE FUNCTION __init__(self):
    # initializing the MongoClient, this helps to
    # access the MongoDB databases and collections
    SET self.client TO MongoClient(host='localhost', port=27017)
    SET self.database TO self.client("UBI")

  DEFINE FUNCTION IWR_create(self, project):
    IF project is not None:
      self.database.project.insert(project.get_as_json())
    ELSE:
      raise Exception("Nothing to save, because project parameter is None")

  DEFINE FUNCTION IWR_read(self, project_id= None):
    IF project_id is None:
      RETURN self.database.project.find({})
    ELSE:
      RETURN self.database.project.find({"_id":project_id})

  DEFINE FUNCTION IWR_update(self, project):
    IF project is not None:
      self.database.project.save(project.get_as_json())
    ELSE:
      raise Exception("Nothing to update, because project parameter is None")

  DEFINE FUNCTION IWR_delete(self, project):
    IF project is not None:
      self.database.project.remove(project.get_as_json())
    ELSE:
      raise Exception("Nothing to delete, because project parameter is None")
```

Figure 8. Pseudo-code for the data processing function of the human-motion database for intelligent wearable robots

Figure 8 shows the CRUD function pseudo-code that processed the saved motion data. We defined a class containing the CRUD function, which processed the JSON-formatted motion data stored in the human-motion database for intelligent wearable robots.

3. Conclusions

Human-motion data are critical for assistive-gait walking robots, wearable robots, and equipment. Such devices can be dangerous to users if they are designed with low-quality data. Since existing optical-based systems have poor data accuracy, we collected motion data using non-optical inertial motion-capture equipment, resulting in higher data accuracy. We then designed a big-data database for use by intelligent wearable robots. Motion data can accumulate in the database and be used to assess whether wearable robots, humanoid robots, and robot simulations behave similarly to human movements. Our database can be applied to several areas of robotics, including walking robots used by people with paraplegic disabilities who need walking assistance, robots and equipment that mimic human movements for people working in high-risk environments, and humanoid robotics.

Acknowledgements

This work was supported by the Technology Innovation Program (Robot Industry Core Technology Development Project) (20003914, Development and commercialization of motorized exoskeleton robots for the assistant of daily life exercise) funded By the Ministry of Trade, Industry & Energy (MOTIE, Korea)

References

[1] Abouhossein, Alireza, Uriel Martinez-Hernandez, Mohammed I. Awad, Imran Mahmood, Derya Yilmaz, and Abbas A. Dehghani-Sanij. Assistive Gait Wearable Robots—From the Laboratory to the Real Environment. *Reinventing Mechatronics*, 75-92, 2020.

[2] Yang, Mingxing, Xingsong Wang, Zhiyong Zhu, Ruru Xi, and Qingcong Wu. Development and Control of a Robotic Lower Limb Exoskeleton for Paraplegic Patients. *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science* 233(3), 1087-098, 2018

[3] Auberger, Roland, Robert Riener, Michael Friedrich Russold, and Hans Dietl. Energy Recuperation at the Hip Joint for Paraplegic Walking: Interaction Between Patient and Supportive Device. *2019 IEEE 16th International Conference on Rehabilitation Robotics (ICORR)*, 938-943, 2019

[4] Wu, Yi, Guang-Zhong Cao, Su-Dan Huang, and Yeping Peng. Distributed Database of Cloud Platform for the Lower-limb Exoskeleton Robot. *2020 17th International Conference on Ubiquitous Robots (UR)*, 322-326, 2020

[5] “Motion capture”, Wikipedia, last modified Aug 18, 2020, accessed Aug 20, 2020, https://en.wikipedia.org/wiki/Motion_capture.

[6] “Kinect”, Wikipedia, last modified Aug 19, 2020, accessed Aug 20, 2020, https://en.wikipedia.org/wiki/Kinect.

[7] George, Gerard, Martine R. Haas, and Alex Pentland. Big data and management. *Academy of Management (ACM)*, 321-326, 2014

[8] “MongoDB”, Wikipedia, last modified Aug 25, 2020, accessed Aug 30, 2020, https://en.wikipedia.org/wiki/MongoDB.

[9] “JSON”, Wikipedia, last modified Aug 17, 2020, accessed Aug 20, 2020, https://en.wikipedia.org/wiki/JSON.

[10] “BSON”, Wikipedia, last modified Aug 18, 2020, accessed Aug 20, 2020, https://en.wikipedia.org/wiki/BSON.

[11] Kim, Seungsu, Changhwan Kim, and Jong Park. Human-like Arm Motion Generation for Humanoid Robots Using Motion Capture Database. *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 3486-3491, 2006

[12] Yani, Muhammad Aksa Hidayat, Sandi Bayu Aji, Ika Fadhilah Ariyanti, Sritrusta Sukaridhoto, Muhammad Agus Zainuddin, and Achmad Basuki. Implementation of Motion Capture System for Swimmer Athlete Monitoring. *2019 International Electronics Symposium (IES)*, 400-405, 2019.

[13] Chentanez, Nuttapong, Matthias Müller, Miles Macklin, Viktor Makoviychuk, and Stefan Jeschke. Physics-based Motion Capture Imitation with Deep Reinforcement Learning. *Proceedings of the 11th Annual International Conference on Motion, Interaction, and Games*, 1-10, 2018.

[14] Bergamin, Kevin, Simon Clavet, Daniel Holden, and James Richard Forbes. DReCon. *ACM Transactions on Graphics (TOG),* 38(6), 1-11, 2019

[15] Ofli, Ferda, Rizwan Chaudhry, Gregorij Kurillo, Rene Vidal, and Ruzena Bajcsy. Berkeley MHAD: A Comprehensive Multimodal Human Action Database. *2013 IEEE Workshop on Applications of Computer Vision (WACV)*, 53-60, 2013

[16] Kapadia, Mubbasir, I-Kao Chiang, Tiju Thomas, Norman I. Badler, and Joseph T. Kider. Efficient Motion Retrieval in Large Motion Databases. *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games (I3D 13)*, 19-28, 2013.

[17] “Neuronmocap”, Noitom, (n.d). accessed Aug 24, 2020, https://www.neuronmocap.com/products/perception_neuron.

[18] “Neuronmocap”, Noitom, (n.d). accessed Aug 24, 2020, https://neuronmocap.com/system/files/software/Axis%20Neuron%20User%20Manual_V3.8.1.5.pdf