Revista Romana de Interactiune Om-Calculator 10 (3) 2017, 216-230

Gesture-based Visual Analytics in Virtual Reality

Mihai Pop, Adrian Sabou

Computer Science Department, Technical University of Cluj-Napoca
Str. G. Baritiu, Nr.28, 400027, Cluj-Napoca
E-mail: mihai.m.m.pop@gmail.com, adrian.sabou@cs.utcluj.ro

Abstract. This paper presents an approach to dynamic data visualization and manipulation using virtual reality as a means of display and the Leap Motion controller in order to interact with the virtual scene. The data and related APIs provided to the virtual reality application are based on a separate web application which provides REST APIs in order to get and further manipulate existent data. The algorithm pipeline is managed by using the Unity Game Engine, which provides a means to express control and logic over the sensory inputs of the Leap Motion controller. The application is built by using the Unity Game Engine and it is streamed on any Android phone through RiftCat Vridge software solution for streaming application over Wi-Fi, while interpreting the phone's sensor input and translating it into virtual head movements. The issues from the standpoint of a corporate need for a more interactive and natural experience in viewing and interacting with big data.

Keywords: Visual Analytics; Virtual Reality; Leap Motion; Gestures detection; RiftCat Vridge; Node server application.

1. Introduction

Over the last couple of years, the virtual reality scene has become more and more popular. Part of the reason is that computational power has become less expensive and readily available to those who study and create projects with such needs. From its early cumbersome beginnings, Virtual Reality has become widely spread in the game industry through simulation games using handheld controllers and a head-mount for displaying the virtual scene. Though this has been the main use case for virtual reality applications, new use cases are emerging lately from the corporate world.

When compared to the conventional means of analyzing big data, virtual reality data analytics is more appealing and visually pleasing than scrolling through endless table rows. There are large datasets which need to be sorted, viewed, compared, enhanced. A virtual reality solution would offer the user a means of interacting with the data in a more natural manner; through gestures, and physical user interface objects which mimic the behavior of

real-life buttons, sliders and knobs.

As a response, this need of a more convenient and natural way to approach data analytics has fueled the development of an application designed to receive large sets of data and display and manipulate them in order to fulfill the user's data queries and study.

The context for this virtual reality application is set by the corporate need of a better data analytics visualization and manipulation approach. Having this in mind, an auxiliary server-side application has been developed in order to substantiate real-world server behavior that we plug into our analytics application. The server application provides a basic REST API for querying and filtering data.

Then, the users queries data by requesting it with the appropriate request URL. The requesting of data can be done through the use of hand gestures in front of the Leap Motion controller. For testing purpose, the data is currently represented as a 3D array of cubes. This array of cubes is generated at runtime through an instantiation algorithm which takes into account all data from the server application and maps it accordingly to Unity game objects with attached data storing scripts for the server-side data.

So, by mapping the server-side data into compatible Unity game objects, we can apply our desired logic onto the virtual scene providing the user many features for an interactive and immersive experience in order to suit his goals through simple hand gestures.

The proposed solution makes use of the phone's gyroscopic information in order to track head movements and to supply this information to the cameras in the virtual scene, allowing them to rotate and translate with respect to the tracked head movements. Also, we use the Leap motion controller for its hand tracking and development library which provides a toolkit for defining custom gestures to be detected and many user interface functionalities which make the application user friendly and smooth.

The chosen solution uses custom gestures built upon predefined elements for certain actions which are all described as Unity classes or logic gates between several gesture detectors from the Leap Motion Controller library.

This paper elaborates on a dual application solution which tries to solve the issue of big data manipulation using natural gestures in order to achieve this goal.

2. Related Work

A variety of techniques for virtual reality interaction already exist, due to the high demand in the gaming market for such applications and games. As a result, several iterations of relevant libraries and APIs have been created to suffice the development of virtual reality applications. As virtual reality further matures, it will only get easier to start developing and creating an interactive virtual reality application due to the advances of previous developers and the documentation of their features.

Ridder et al. used Virtual Reality and Augmented reality coupled with gestures to create an immersive environment for visualizing fMRI data. fMRI or Functional magnetic resonance imaging consists of a functional neuroimaging procedure using MRI technology in order to measure brain activity by detecting changes related with blood flow. They suggested that VR/AR can potentially allow a reduction in visual clutter and lets users keep their focus on visualizations by allowing them to navigate the data abstractions in a natural way.

Moran et al. used interactive Virtual Reality to manipulate Twitter datasets and to visualize them in the original geospatial domain. By using emerging technologies, they created a fully immersive tool that promotes visualization and interaction and that can help ease the process of understanding and representing big data.

Olshannikova et al. provide a multidisciplinary overview of the research issues and achievements in the field of Big Data and its visualization techniques and tools. They discuss the impacts of new technologies, such as Virtual Reality displays and Augmented Reality helmets on the Big Data visualization as well as to the classification of the main challenges of integrating the technology.

Hackathorn and Margolis outline the objectives for analytical reasoning and immersive data spaces, followed by suggestions for the design and architecture of data worlds. Finally, they describe current work for building data worlds.

3. The Leap Motion Controller

The Leap Motion Controller device consists of two cameras and three infrared LEDs. These track the infrared light with a wavelength of 850 Motion Controller doesn't generate a depth map – instead it applies advanced algorithms to the raw sensor data.

The Leap Motion Service is the software on your computer that processes the images. After compensating for background objects (such as heads) and ambient environmental lighting, the images are analyzed to reconstruct a 3D representation of what the device sees.

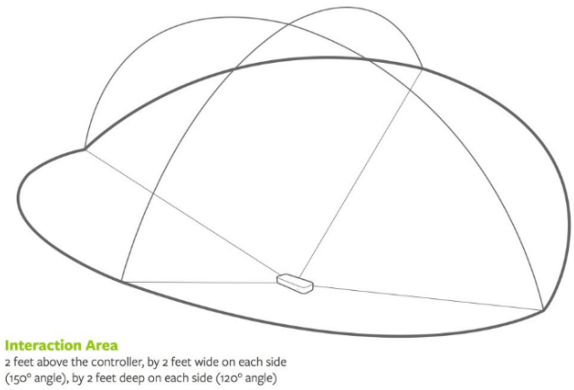

Figure 1. The Leap Motion interaction area.

Next, the tracking layer matches the data to extract tracking information such as fingers and tools. Our tracking algorithms interpret the 3D data and infer the positions of occluded objects. Filtering techniques are applied to ensure smooth temporal coherence of the data. The Leap Motion Service then feeds the results – expressed as a series of frames, or snapshots, containing all of the tracking data – into a transport protocol.

Thanks to its wide-angle lenses, the device has a large interaction space of 0.226 cubic meters, which takes the shape of an inverted pyramid – the intersection of the binocular cameras' fields of view (Figure 1).

With the Orion beta software (the current iteration of the software development kit for the Leap Motion device), this has been expanded to 2.6 feet (80 cm). This range is limited by LED light propagation through space, since it becomes much harder to infer your hand's position in 3D beyond a

certain distance.

At this point, the device's USB controller reads the sensor data into its own local memory and performs any necessary resolution adjustments. This data is then streamed via USB to the Leap Motion tracking software.

The data takes the form of a grayscale stereo image of the near-infrared light spectrum, separated into the left and right cameras. Typically, the only objects you'll see are those directly illuminated by the Leap Motion Controller's LEDs. However, incandescent light bulbs, halogens, and daylight will also light up the scene in infrared. You might also notice that certain things, like cotton shirts, can appear white even though they are dark in the visible spectrum.

Once the image data is streamed to your computer, it is time for complex mathematical computations. Despite popular misconceptions, the Leap

4. Node REST application with MongoDB database

As mentioned in the introduction, there is an auxiliary application which interacts with the virtual reality application, providing it with the dynamic data and advanced query capabilities. In the following, there is a brief description of the core concepts of such an application.

One of the most used approaches is Representational State Transfer – REST – because it is an open approach for lots of conventions that are used for consumers of your API. The way this transfer is made is determined by the resources provided by your API.

RESTful applications use HTTP requests to perform four operations termed as CRUD (C: create, R: read, U: update, and D: delete). Create and/or update is used to post data, get for reading/listing data, and delete to remove data.

MongoDB is an open source database that uses a document-oriented data model. MongoDB is one of several database types to arise in the mid-2000s under the NoSQL banner. Instead of using tables and rows as in relational databases, MongoDB is built on an architecture of collections and documents. Documents comprise of sets key-value pairs and are the basic unit of data in MongoDB. Collections contain sets of documents and function as the equivalent of relational database tables.

So, the server-side application is a REST application which uses Node for server emulation. Node being a server-side solution for JavaScript, and in

particular, for receiving and responding to HTTP requests. The application is written in JavaScript language, for a clean and rapid development.

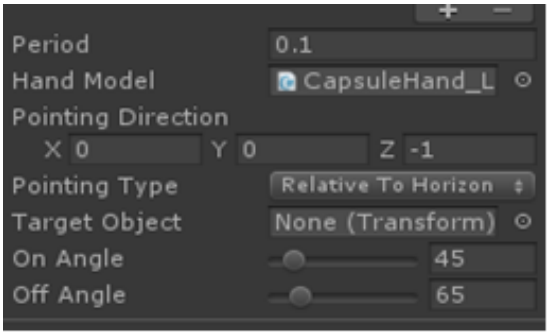

Figure 2. The configuration of the Palm Direction Detector

5. Leap Motion gestures

The Leap motion software development library by itself does not contain definition for complex gestures. The Leap motion API contains definitions for the mapped human body parts (Hand, Arm, Bone, Finger, etc.) and also definitions for concepts like Frame (the clipped viewport in which a Leap object exists) or Controller (you can access the actual controller attributes and methods) or other non-physical control concepts.

So, in order to define more complex gestures, we can make use of the Leap Motion Detection Utilities Module. The detection utilities are a set of scripts in the core asset package that provide a convenient way to detect what a user's hand is doing. For example, you can detect when the fingers of a hand are curled or extended, whether a finger or palm are pointing in a particular direction, or whether the hand or fingertip are close to one of a set of target objects.

Detectors can be combined together in order to declare complex gestures. This can be done by using a Logic Gate. The Detector Logic Gate provided by this module is in itself a detector which combines two or more detectors to determine its own state.

Detectors ultimately dispatch standard Unity events upon activation or deactivation. This provides a convenient means to hook different function handlers.

For instance, when we would want to detect if the user has his hand

camera-facing and open we would assemble the following detectors:

1. Extended Finger Detector (Figure 3) - configure the component so that all fingers must be extended.

2. Palm Direction Detector (Figure 2)
   a. Pointing direction = (0, 0, -1)
   b. Pointing type = Relative to Horizon
   c. On and off angles: set as desired

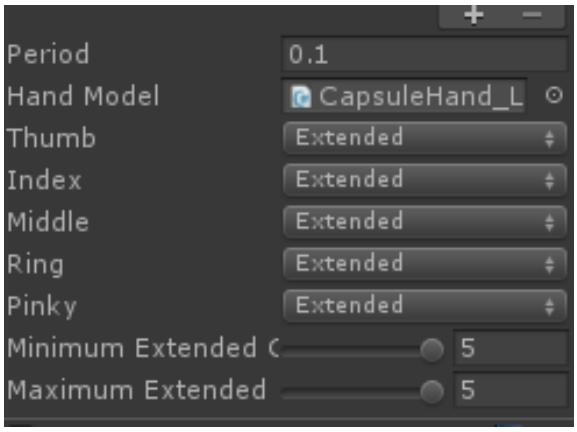

Figure 3. The configuration of the Extended Finger Detector

After that, we can link these two detectors in the Detector Logic Gate and link other C# scripts and functions to further our desired logic or use case (Figure 4).

This minimal setup can facilitate powerful results. The above example links the detection of a camera-facing open hand to a function call from a separate file, which provides an UI attachment to the left palm of the user (Figure 5).

Another such gesture is pinching which is defined similarly through the use of detectors. The effects of pinching with the left hand while looking at a given cube will cause it to highlight and display a panel over it with its mapped data from the server-side application (Figure 6).

The data from the server-side application is parsed before attaching it in the form of data scripts to Unity game objects. The parsed JSON data is further integrated in the application by being mapped as a custom list of

ResourceNode objects, which are a custom class linked to the project namespace.

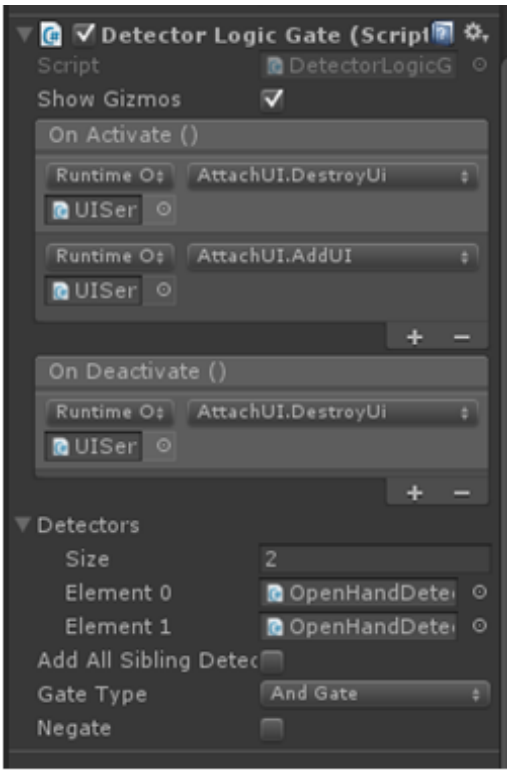

Figure 4. Detector Logic Gate configuration

If the user would pinch with his right hand while looking at any certain cube, that cube will change position in the Unity scene hierarchy and shall be moved under the RTS node which stands for Rotate Translate Scale node (Figure 7).

This Node has scripts attached to it which rotate, translate and scale any mounted game object. It also has capabilities for one and two-handed rotation.

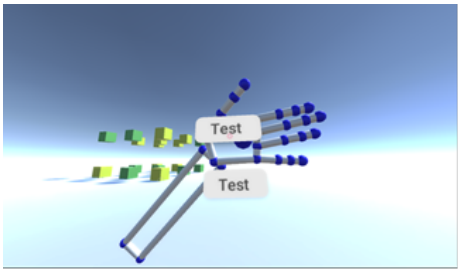

Figure 5. The instantiation of UI components on camera-facing open hand gesture

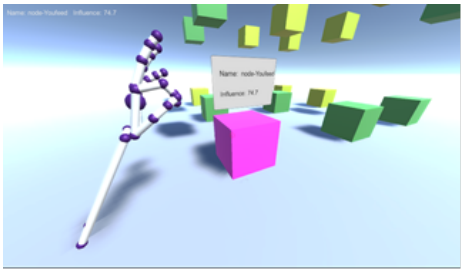

Figure 6. Pinching with left hand highlights cube and displays its information

6. Server application in depth

The server application uses a MongoDB database on port 27017, the standard MongoDB port, this can obviously be changed if conflicting settings. The application runs on port 3000 and has different URL mappings, in order to provide consumption to its API.

Of course, in order to start the MongoDB local service on port 27017 we would need to execute this statement in a terminal: mongod --dbpath "C:\data". This statement just starts the "mongod" process having as database path, the above-mentioned path.

The server mappings are a collection of URLs for getting information, which support search parameters. Examples of the URL mappings are as follows:

1. http://localhost:3000/api/data
   This is the main GET URL, which gets all data in the database (Figure 8)
2. http://localhost:3000/api/data?name=Youfeed
   This gets the node with the name Youfeed.
3. http://localhost:3000/api/data?influenceGt=20&influenceLt=40
   This gets the nodes with the influence key between 20 and 40 (Figure 9).

On the root mapping, "http://localhost:3000/", there is exposed a list of all the possible request patterns in order for developers to visualize the capabilities of the existing API and quickly determine how to send requests for specific filtered data. (Figure 10)

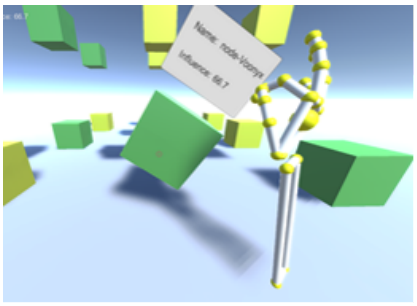

Figure 7. Pinching with right hand mounts the looked at cube in the RTS node

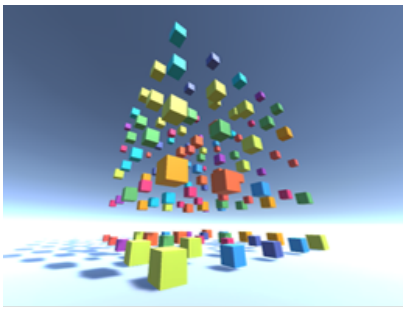

Figure 8. Main get URL for getting all database entries

These filtering operations can essentially be endless because MongoDB offers rich functionality for collection manipulation.

6.1 Server data mapping

In order to represent the server-side data as cubes, there must be defined some transitory objects between the primitive cube game objects and the actual raw data.

The typical pipeline of processing server data in our application is:

1. Execute Start void function from Instantiation Service script

This function calls another function which does the actual request.

2. Check if erroneous response from the function which made the server call. If everything is ok, we call a function which builds the data

3. The function which builds the data does the parsing of the JSON server response. Each value is parsed as the appropriate data type it should have. A list of Resource Nodes are created, which aid in mapping the JSON data. Resource Node is a custom class which belongs to the project namespace After parsing, call a function which creates nodes.

4. The function which creates the nodes loops over the list of Resource Nodes and creates new Cube object which is also a custom class that has attributes for parsed values and methods for setting them and also a render method which adds a game object to the scene with mapped properties

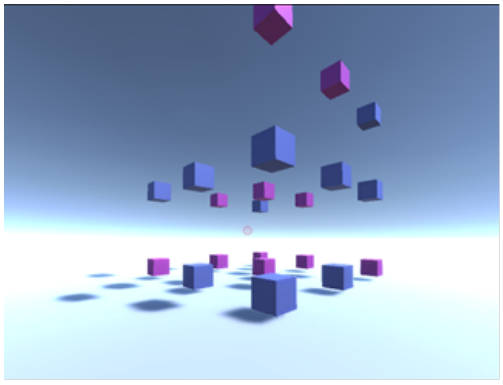

Figure 9. Nodes between influence range [20, 30]

5. The position of each cube is computed by the place cubes function which basically tries to construct a cube of cubes.

VR-Analytics API

Unfiltered resource endpoint:

```
localhost:3000/api/data
```

Querying capabilities:

- Search for nodes with certain colors

```
http://localhost:3000/api/data?color=<color>
http://localhost:3000/api/data?colorIn=<color1>,<color2>
```

e.g.
http://localhost:3000/api/data?color=cddc39
http://localhost:3000/api/data?colorIn=cddc39,3f51b5

- Search for nodes with a certain influence

```
http://localhost:3000/api/data?influence=<influence>
http://localhost:3000/api/data?influenceLt=<influence>
http://localhost:3000/api/data?influenceGt=<influence>
http://localhost:3000/api/data?influenceLt=<influence1>&influenceGt=<influence2>
http://localhost:3000/api/data?influenceIn=[<influence1>,<influence2>,...]
```

- Search for nodes with a certain name

```
http://localhost:3000/api/data?name=<name>
```

Figure 10. Server API interface list querying data

The raw data (Figure 11) received on a sent request is a JavaScript array of

objects which contain relevant keys, in order for the mapped 3d objects to respect and help to visually determine the characteristics of this data. Because we are using MongoDB for our database solution, a supplementary "_id" parameter is inserted into the array objects. This parameter is used by MongoDB to uniquely define a resource; it does not interfere in any way in our mapping mechanism.


```
[
  - {
      _id: "590f5325c99ccfd5d451a904",
      influence: 74.7,
      name: "Youfeed",
      color: "cddc39"
    },
  - {
      _id: "590f5325c99ccfd5d451a905",
      influence: 40.3,
      name: "Wordtune",
      color: "2196f3"
    },
  - {
      _id: "590f5325c99ccfd5d451a906",
      influence: 35,
      name: "Fadeo",
      color: "3f51b5"
    },
  - {
      _id: "590f5325c99ccfd5d451a907",
      influence: 44.9,
      name: "Realbridge",
      color: "2196f3"
    },
  - {
      _id: "590f5325c99ccfd5d451a908",
      influence: 79,
      name: "Geba",
      color: "cddc39"
    },
  - {
      _id: "590f5325c99ccfd5d451a909",
      influence: 75.2,
      name: "Blogpad",
      color: "cddc39"
    },
  - {
      _id: "590f5325c99ccfd5d451a90a",
      influence: 77.8,
      name: "Demizz",
      color: "cddc39"
    },
```


Figure 11. Raw data received on a request

7. Implementation

The Unity game engine

For the implementation, there was an extensive research as to how to assure compatibility the many components at play in this project. The final versions chosen for the components are:

1. Leap Motion SDK 3.1.2
2. Leap Motion Core SDK 4.1.4
3. Leap Motion Attachments Module 1.0.4
4. Leap Motion UI input module 1.0.0
5. Unity Game Engine 5.3.4f1
6. RiftCat Vridge 1.3.3
7. Oculus Runtime 0.8

The reasoning for choosing these compatible versions was in order to stream the application on any android phone and make use of its sensors by not faking sensor input and using the actual phone's sensor information.

In order to aid the user in his visualizing experience, some custom UI elements have been defined:

1. Information Canvas (Figure 12)

This element serves as an informative panel which attaches to the currently viewed 3d mapped resource. Currently, it provides information about the resource name and its influence value. These pieces of information are not constrained in any way and can be mapped to any property on the received request array object.

Figure 12. Information Canvas

2. Query Builder (Figure 13)

The query builder panel serves as a graphical utility in order to make custom requests to the server application, by using virtual buttons and sliders.

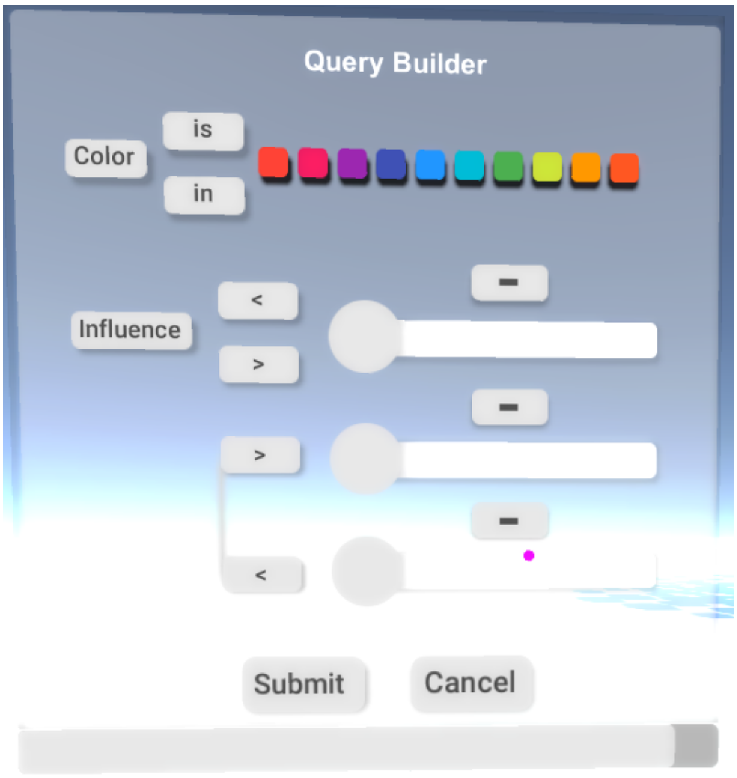

Figure 13. Query Builder

8. Conclusions

In the world of virtual reality, technology is advancing drastically and new implementations and solutions to problems we do not know we have arrive every day. The rise of the virtual reality world has fueled other industries to peer down and indulge into the advantages this industry brings.

Therefore, the use cases associated with virtual reality got more complex and more 'big players' got into the virtual reality game. In consequence, there is a fueled need in the corporate world for a solution to common reporting problems through the means of virtual reality.

Trying to provide a user friendly and straightforward solution, this application could be the start of the modern way to visualize and manipulate data analytics by using virtual reality.

This paper presented a dual application solution in order to give a user the opportunity to do his queries and tasks through a virtual interface in which everything is controlled by gestures and intuitive design.

Future work includes a better user experience and integrating with email and text-to-speech APIs.

References

M. de Ridder, Y. Jung, R. Huang, J. Kim and D. D. Feng. Exploration of Virtual and Augmented Reality for Visual Analytics and 3D Volume Rendering of Functional Magnetic Resonance Imaging (fMRI) Data. 2015 Big Data Visual Analytics (BDVA), TAS (2015), 1-8.

A. Moran, V. Gadepally, M. Hubbell and J. Kepner. Improving Big Data visual analytics with interactive virtual reality. 2015 IEEE High Performance Extreme Computing Conference (HPEC), (2015), 1-6.

Ekaterina Olshannikova, Aleksandr Ometov, Yevgeni Koucheryavy and Thomas Olsson. Visualizing Big Data with augmented and virtual reality: challenges and research agenda. Journal of Big Data 2, 1 (2015), 22.

R. Hackathorn and T. Margolis. Immersive analytics: Building virtual data worlds for collaborative decision support. 2016 Workshop on Immersive Analytics (IA), (2016), 44-47.

Alex Colgan. How Does the Leap Motion Controller Work?. Leap Motion Blog. (2014), Available: http://blog.leapmotion.com/hardware-to-software-how-does-the-leap-motion-controller-work/.

Leap Motion Inc. Leap Motion. (2017), Available: https://www.leapmotion.com/.

MongoDB Inc. MongoDB. (2017), Available: https://www.mongodb.com/.

Leap Motion Inc. Using the Tracking API. Leap Motion Developer (2017), Available: https://developer.leapmotion.com/documentation/python/devguide/Leap_Guides2.html.

Node.js® JavaScript runtime (2017), Available: https://nodejs.org/en.