Australia

School of Computer Science and Engineering

The University of New South Wales

Scalable 3D virtual reality visualization of biological data

by

Jianfu Li and Yu Yao

Thesis submitted as a requirement for the degree of Bachelor of Engineering in Computer Engineering

Submitted: 29, May 2018

Supervisor: Dr Joshua Ho

Assessor: Dr Eleni Giannoulatou

Student ID: z3485848, z3458929 Topic ID: 3820

Abstract

With the development of RNA sequencing, flow cytometry and mass cytometry technology, volume of single cell data is growing explosively. It becomes much more difficult for scientists to use existing tools to efficiently visualize the large volume of high dimensional data sets generated by these single-cell technologies. Under such circumstances, selecting an efficient method to enhance intuitive data understanding is becoming very important. An efficient data visualization method should enable users to easily interact with large data sets and understand the structure within them. As a result, we built a low-cost and cross-platform virtual reality (VR) visualization application which provides a powerful means to explore large single-cell data sets. It allows data analysts to have an immersive view of the global structure of data sets, while interactively explore profiles of a single cell.

Contents

Introduction 5

Related application 7

Single cell variables characterization technologies 7

Bulk RNA-seq 7

Single cell RNA-seq 7

Flow cytometry 8

Mass cytometry 8

Screen-based 3D visualization 8

Existing visualization tools 8

ParaView 8

VisIt 9

Mayavi2 9

NIA array analysis tool 9

Gecko 10

VR visualization 10

VR technology 10

VR equipment 10

VR software development kit (SDK) 13

WebVR 13

Google VR 13

OpenVR 13
Existing visualization tools 13
MetNet3D 13
SkinExplorer 14
BRAINtrinsic 14
IViz 14
Microcopy data visualization tool 15
CellexalVR 15
Summary 15
Related frameworks 17
WebGL2.0 17
Unity 17
PlayCanvas 17
Babylon.js 18
Three.js 18
A-FRAME.js 18
Solution 20
Software developments 21
Input data 21
Input format 21
Content requirements 21
Upload method 22
Data pre-process 23

Data visualization 23
3D geometry visualization 23
High dimensional features visualization 24
3D compass 25
Interaction 26
GUI 26
Input method 28
Keyboard and iCade remote controller 28
Mouse and gaze-based interactions 30
Voice control 30
Evaluation 32
Loading time test 32
Rendering ability test 33
Further work 36
Movement animation generation 36
Support point recluster 36
Support VR gloves controller 36
Conclusions 38
Bibliography 39

Introduction

Data visualization, as a key component in the process of mining data, establishes a link between the quantitative content of data and human intuition, thereby providing a scientific path from data to knowledge and understanding. It is also crucial in terms of bioinformatics. For example, in the process of analysing sequencing data, data visualization can be used to identify inconsistent cluster annotation, which will directly affect analysis result. However, data sets generated by single-cell RNA-seq (scRNA-seq) usually has both high dimension (~20,000 genes) and high volume (~100,000 single cells). Moreover, the volume is still increasing with the development of new single cell experimental protocols, with the possibility of profiling a million cells very soon. Although the dimension can be reduced by applying dimensionality reduction methods such as the principal components analysis (PCA), effectively displaying hundreds of thousands of data points on a two-dimensional plot is still challenging.

A key task for single cell data analysis is clustering and visualization of all the single cells to identify cell subpopulations. A good visualization tool should enable clear visual separation of different clusters and allow exploration of their gene/protein expression profiles. Current single cell data visualization tools are mainly designed to visualize up to thousands of cells but are not designed to scale up to millions of cells. Moreover, high dimensional profiles of individual cell are invisible.

Our application, starmap, introduces a scalable visual design that combines the benefit of a three-dimensional (3D) scatter plot for exploring clustering structure, and the benefit of star plots (also known as radar charts) for multivariate visualization of an individual cell. In addition, starmap provides a lot of interactions to drive into data sets with an intuitive graphical user interface (GUI). Moreover, starmap is designed to utilise low-cost VR headsets starting from 20 US dollars. We reason that an immersive visual experience will likely improve the navigation and exploration of hundreds of thousands of cells.

In this thesis report, the literature review section will briefly explain cell variables' characterization methods, VR technology and related frameworks. On top of that, this section will present some previous work of screen-based and immersive VR 3D data visualization tools. Meanwhile, reasons for choosing VR in addition to advantages and disadvantage of these tools will also be illustrated. The subsequent section will present our

solutions based on problems with existing tools, followed by the software development. To test the usability of our application, evaluation is divided in two different aspects including performance evaluation and user testing.

Related application

Single cell variables characterization technologies

Bulk RNA-seq

Bulk RNA-seq (RNA sequencing) is one of the most widely used technologies in high-throughput sequencing technology**,** with which the average expression level for each gene across a large population of input cells can be calculated (Vladimir, et al., 2017) RNA-seq can help bioinformaticians to understand the variations in the expression of all genes under different conditions. For instance, detecting the differences between normal and tumor tissues, differences in gene expression prior to and after drug treatment or differences in the gene expression of different tissues at different stages of their development. The basic steps include the preparation of a sequencing library and sequencing (Sho, et al., 2014). By applying this method, an n-dimensional data set (matrix) is obtained, with each dimension (a row of the matrix) representing a different gene, and each sample (a column in the matrix) representing the expression of a biological sample. The number of the dimension is in the order of tens of thousands (i.e., the total number of transcripts). There are usually hundreds of samples in a data set) and each sample consists of a pool of cells (usually millions of cells).

Single cell RNA-seq

Single cell RNA-seq (scRNA-seq), as a new technology, has been increasingly widely adopted since early 2010s. Comparing to bulk RNA-seq, it measures the distribution of expression levels for each gene in individual cells instead of the population average, providing a better insight for understanding changes in transcriptome-wide expression changes, and identifying subpopulations of cells, as well as the heterogeneity of cell responses (Fuchou, et al., 2009; Vladimir, et al., 2017). Each sample of a data set generated in an scRNA-seq experiment is a single cell. There could be up to a million cells in a scRNA-seq experiment.

Flow cytometry

Flow cytometry is a high-throughput biophysical technology that can be used for cell counting and clustering. Physical and chemical properties of cells can be analysed by passing cells that rapidly flow in a fluid through electronic detection instruments. Nevertheless, there is spectral overlap between the emission spectrum of multiple fluorophores by using traditional fluorescence-based flow cytometry. Data compensation is used to address this problem, but with increasing number of parameters, compensation calculations become more complex. As a result, the number of parameters that can be quantified is limited. In recent years, 30-parameter flow cytometers have become commercially available. (Saeys, et al., 2016).

Mass cytometry

The mass spectrometry cytometric technique can quantify approximately 70-100 parameters (Sean, et al., 2012) which is much greater than using flow cytometry. By employing time of flight (TOF) mass spectrometry, heavy metal-conjugated antibodies can be detected and quantified. With its help, the number of parameters collected simultaneously is greatly expanded and a high dimensional protein-abundance profile for each individual cell is obtained (Matthew & Garry, 2016).

Screen-based 3D visualization

Existing visualization tools

ParaView

ParaView is an open source, scientific data visualization tool for general purposes. The project was started in 2000. It is based on Visualization Toolkit (VTK) library and has the ability to render 3D graphs. It was first released on October 2002. Various clients are available for different platforms. Users are allowed to manipulate their visualized data such as slicing, contouring and rotation via GUI. Moreover, the python script can be generated for reproducible visualization. The advanced features of ParaView include tile display support, parallel rendering and distributed computing. These features can enhance its own ability to process massive data sets (Andy, et al., 2006).

VisIt

VisIt is another distributed, parallel visualization and graphical analysis tool for 2D and 3D scientific data. It was also initiated in 2002 (Eric, et al., 2005). Both software architecture and functionality of VisIt and ParaView are extremely similar. Meanwhile, VisIt also support Java application programming interface (API).

Mayavi2

Mayavi2 is also a tool based on open source used for 2D and 3D visualization in many general circumstances. Implemented in Python and supporting various types of data such as scalar, vector and tensor data, it provides 2 ways of manipulating visualized data sets, python scripting or via its intuitive GUI. It offers higher flexibility for users with different preferences. Moreover, it provides more convenience for interactive work compared to ParaView and VisIt due to the integration of Ipython. However, ParaView and VisIt have better performance than Mayavi2 because Mayavi2 only supports serial processing (Ramachandran & Gaël, 2011). Instead of focusing merely on particular fields, designers of those tools want to create applications that can provide users with general solutions to many similar problems in various fields of academic research, which is their primary motivation. Without doubt, of course, there are edge cases for each research field that cannot be solved by general solutions. In those cases, the importance of designing tools for some specific purposes should not be ignored.

NIA array analysis tool

In 2005, the United States Institute of Health developed a tool for both data processing and visualizing microarray data. This half web-based data analysis tool allows microarray data to be uploaded in required formats in which it can assess the statistical significance of different gene expressions. It provides many helpful functions for the analyzation of microarray data via mathematical means which include true-error variance estimation with different error models, hierarchical clustering and principal component analysis etc. In addition, it is able to plot basic static 2D PCA graphs with their web-based visualization tool. In order to generate interactive 3D PCA graphs which allow rotation and

displacement, users must download additional VRML viewer like freeWRL and then import the files that was exported by NIA Analysis tool (Alexei A., et al., 2005).

Gecko

Gecko, another software for analysis of gene expression, also has a strong computational engine with 50 different analysis methods including clustering methods and PCA. It is an analysis system based on a client-server architecture as well. A client application for analyzing, containing the embedded browser which is only compatible to windows, is required to be downloaded for this application. Users are allowed to upload data files and send requests for applying different analysis methods to the server via the client. Gecko integrates the Spotfire visualization platform to render 3D data (Joachim, et al., 2004). Undeniably, both Gecko and NIA provide many methods for data procession. However, it is hard to customize their virtualization system to be identical to their specific data types by only integrating a third-party virtualization engine designed for end users. Consequently, the volume of information that can be observed and abstracted by virtualization of processed data is limited.

VR visualization

VR technology

VR, also known as virtual environment, is the use of computer simulation to produce a three-dimensional virtual world space, which makes users feel more immersed by providing sensory stimulation. Despite some drawbacks, visual simulation is the most widely used method that can achieve this goal. Users are able to explore the artificial world by using various VR devices. In recent years, VR devices such as VR-BOX, Google Cardboard, HTC Vive etc. have become more affordable, lightweight and flexible, making the technology more applicable in many aspects including data visualization.

VR equipment

There are 3 main types of VR devices, prices of which are significantly different from one type to another. Most expensive type of devices are usually equipped with large screens (cave, cave2) (Figure 1). Key customers for this kind of devices are usually universities or research institutes. As the equipment prices always exceed the limited budgets of ordinary

customers and are too cumbersome for ordinary users. Another type of VR devices is a kind of headset that are much cheaper and more flexible. They are connected to PCs via cables, along with a headset with a built-in screen that are situated very close to the users' eyes when the users put on the headsets (Figure 2). Prices of those headsets range between 200 US dollars and over 1000 US dollars. VR headsets for mobiles is the third type (Figure 3). This type of equipment usually costs around 20 US dollars. Mobile devices are being used to replace the built-in screen of VR headset for PC. It is the most portable solution for the experience of VR. Moreover, with the increasing performance of mobile devices, it has become the most cost-efficient choice for ordinary customers.

Fig.1 Cave2 VR equipment. *https://www.*monash*.edu/miv*

Fig.2 VR headset for PC. *https://www.vive.com/us/product/vive-virtual-reality-system/*

Fig.3 VR headset for mobile. *https://www.buyvrguide.com/vr-headsets/vr-box-2-0/*

VR software development kit (SDK)

WebVR

WebVR is a necessary JavaScript API for implementing an immersive VR web-based application. The first public version of it was released in 2016 and till now, it is still in experimental stage. Due to this reason, it is not supported by common web browsers. Its development team, therefore, has built a WebVR-polyfill to overcome the problem, making it possible for this tool to support most of web browsers and conduct the strong scalable ability for both low-end or high-end VR devices. (WebVR, 2017)

Google VR

Google VR SDK is used for providing support to their own VR devices. Developers are able to create the standalone mobile VR applications for both Android and IOS via google VR API. It also offers plugins for Unity and Unreal game development platforms respectively. With it, developers are able to create applications for google VR devices on these platforms. (Google, 2016)

OpenVR

OpenVR is another VR SDK developed by Valve. OpenVR is designed for supporting different VR headset devices among different brands instead of relying on a specific hardware vendor's SDK like Google VR. It is an open source library that is mainly implemented by C++. It is primarily used to develop standalone VR applications both for PCs and mobile devices. (Valve, 2015)

Existing visualization tools

MetNet3D

MetNet3D is a great metabolic network visualization tool developed by Iowa State University in 2005. The invention of the tool is very challenging since the VR technology was still at its early stages during 2005. It allows users to interact with the stereoscopic view of gene expression data on a flat screen or immersive VR environment. The equipment they used for VR is composed of six large screen setups on the wall based on the CAVE system. The users can navigate through the network and choose a specific

node or edges to display the detailed information by using a six degree-of-freedom head tracker and a remote controller (Yuting, et al., 2005).

SkinExplorer

In 2013, a skin exploration VR system was implemented. The system supports both large-screen 3D TV and desktop screen. For the VR mode on large-screen 3D TV, A ART SmartTrack is used to track the user's head movement and user can use an ART Flystick3 controller to interact with the GUI. For flat screen mode of the desktop, head movement is tracked by Kinect and a Joystick takes the place of Flystick3. Furthermore, users can enter data with either built-in auto-oriented 3D widgets on screen or virtual keyboards of remote devices linked to the system via TCP/IP sockets. In such cases, interactions including the modification of 2D transfer functions can be conducted more easily (Marie-Danielle, et al., 2013).

BRAINtrinsic

BRAINtrinsic is a web-based VR visualization tool developed in 2015 for visualizing connectome data. In order to comprehensively discover the brain's intrinsic geometry, users are able to choose different topological spaces where the connectome is embedded. In addition, it allows users to focus on interested brain areas by hiding other brain areas (Conte, et al., 2015). BRAINtrinsic derives a solution of using lightweight VR device (Oculus rift) to render the data when compared to MetNet3D and SkinExplorer. However, it is not completely compatible with VR. In VR mode, users are only able to interact with scene via mouse. Moreover, the users are only able to change settings in flat screen mode. After testing, the performance of BRAINtrinsic is lower than expected.

IViz

IViz is an immersive and collaborative data visualization VR system implemented by the Unity 3D platform. Oculus Rift is needed to experience VR. Leap Motion, 3D mouse or Kinect is used to interact with the scene. Iviz can render up to approximately 1,000,000 data points. The user can select a particular point to show its information on the screen and add comments to the point. In addition, points clustering and outliers searching are also supported. Another useful feature of IViz is the collaborative visual data exploration which allows users to share their views with other users. (Donalek, et al., 2014).

Microcopy data visualization tool

In 2017, a microcopy data visualization system was implemented by BMC Bioinformatics using Unity game engine. This system supports two interaction methods. The first method is interacting with the GUI with hands by tracking users' hands movement with Leap Motion. Another method is that users can use a gamepad and an Oculus Rift VR-headset with head tracking function (gaze). By using the VR-headset, users are able to control the cursor rendered in the centre of the screen through head movement. Cursor can also be leveraged to select the elements in the scene and confirm the selection using gamepad. In addition, the system also provides region selection tools in selection mode. Users can obtain a 2-dimensional Boolean array with z-position after selecting their interested region (Theart, et al., 2017).

CellexalVR

CellexalVR (Oscar, et al., 2018) is a standalone VR single-cell gene expression data visualization tool. Implemented by Unity, it contains many features and interactions which aid bioinformaticians in analysing data. For example, it provides multiple multidimensional scaling (MDS) plots in a VR scene to allow users to easily compare the differences generated by different dimensional reduction algorithms. Heatmaps for genes expression of defined groups can also be generated. The HTC Vive controller is used to grab and select both the data and the GUI. To make it easier for users to input data into CellexalVR, scripts are provided to convert data from an R session into the required input format. Key shortcomings of CellexalVR include that it is not supported on mobile devices, requires expensive equipment (e.g. a HTC Vive, and a performant desktop), and has a limited capability to provide stable rendering of points over 15000 points.

Summary

Compared to flat screen 3D data visualization tools, VR data visualization tools has stronger ability to immerse scientists in their datasets. VR not only provides 360 degrees of view, but also provides more natural interaction (gaze, hand tracking etc.) for data exploration compared to 3D data visualization. Above all, there are several researches

indicate that it is more effective in terms of abstracting useful information from data sets and error reduction (Nelson, et al., 1999; Laha & Doug A., 2012).

Except Gecko and BRAINtrinsic, standalone packages need to be downloaded. The significant problem of standalone applications however, are their low capability of multi-platform support. Developers have to implement their applications for various operation systems and devices. None of above are both compatible for mobile devices and desktops. Although Gecko and BRAINtrinsic are web-based applications, they do not provide solutions for efficient data visualization as mentioned previously.

One of the biggest issues of VR data visualization tools is not cost effective. Especially MetNet3D and SkinExplorer, they are hardly affordable by ordinary users. Even the cheapest VR devices used by above applications are over 200 US dollars whereas the cheapest commercial VR devices are less than 20 US dollars. In addition, their applications are not easy-access because none of them fully support the VR experience with mobile devices.

For all applications discussed above, none of them can visualize high dimensional features of data sets intuitively. However, in some circumstance, these features are also important for data analyzation.

Despite repeated research, attempts to find an efficient single cell data visualization tools have failed to yell any positive results.

Related frameworks

WebGL2.0

WebGL2.0 is a highly used, low level JavaScript library for rendering 2D and 3D graphics in web application. WebGL2.0 can be considered as the JavaScript version of OpenGL ES 3.0, which is used to create standalone application. No plugin is required to be installed for compatible browsers, since it has already become one of the web standards for browsers with hardware acceleration which can enhance the performance of rendering (WebGL, 2017).

Unity

Unity is a high level, cross-platform 3D graphics engine primarily used to develop standalone games of Windows, MacOs, Linux or mobile game based on iOS and Android. It also can be applied to develop VR application with various VR plugins such as Google VR and Open VR downloaded. Integrated Development Environment (IDE) of Unity provides a powerful GUI which makes it easier for beginners without much programming background to interactively edit their applications. A code editor is also integrated by the IDE. Additionally, Unity provides many valuable features including gamepad support and smart Physic Engine. Recent years, the project developed in Unity also can be exported to WebGL application by cross-compiling C++ code into JavaScript. However redundant code will be generated in this process which can influence the rendering performance. Besides, it is hard to add extra JavaScript code if there are some features that have to be implemented in a lower level. Moreover, Unity claimed that WebVR supporting are still in progress with time needed for its development being uncertain (Unity, 2017).

PlayCanvas

PlayCanvas is a high-level 3D graphic engine based on WebGL and it is very popular among web-based game developers. It provides one-stop services for creating interactive web contents. An advanced online editor has been offered to users to build their applications efficiently. It can be used for projects' visual building and real time collaborative development to enhance production rates. Users are also allowed to use it as

a project assertion management tool. Furthermore, the application created by PlayCanvas can be deployed in one click (PlayCanvas, 2017).

Babylon.js

Babylon.js has been developed by Microsoft and was launched in 2013. Being a newcomer, its target focuses more on web-based game development compared to its competitors. One of the key features of Babylon.js framework is to provide many built-in game modules including physics system, fire, road and cloud which makes the development of games much easier. The community of Babylon.js is still relatively small, but it has been growing quickly in recent years (Microsoft, 2017).

Three.js

Compared to PlayCanvas and Babylon.js, Three.js is a lower level 3d graphics framework that focuses on creating GPU enhanced 3D graphics for general purpose. The main disadvantage of lower-level framework is that developers have to take care of each part of the code. Thus, total development time will be longer than those of PlayCanvas and Babylon.js. On the other hand, one of benefits of Three.js is its higher degree of customization. It gives developers better control of their codes and more sufficient access to functions of WebGL level. It is not the most productive, but it is the safest library for consideration with performance compared to others. Furthermore, Three.js has excellent documentations attached to a lot of samples and online community of Three.js is much more mature than those of other WebGL based libraries (Threejs, 2017).

A-FRAME.js

A-Frame is a high-level web framework built on top of Three.js. In order to maintain the customizability, it provides full access to three.js as well. A-Frame allows users to render 3D graphics by coding in HTML format, which is an efficient and productive feature for rendering static objects. An entity component system implemented by A-Frame is also an advanced feature for code structure optimization and unit testing. By using it, the code can be wrapped into components based on functionalities. Moreover, the component system is designed to be an ecosystem of A-Frame. There are a lot of useful custom open source components published on GitHub, for instance, line, points and mouse-cursor component.

Besides, A-Frame has integrated a lot of useful libraries such as remote controller support and WebVR-polyfill (A-FRAME, 2017).

Solution

In this project, we aim to build an immersive, cross-platform and low-cost high dimensional biological data visualization tool. We will develop a web-based application using JavaScript and VR technology.

One of main objectives of this system is to allow wider populations to have access and to enjoy the VR service. With this respect, the application is designed to be compatible with mobile phone VR devices, both for IOS and Android, which are cheaper compared to PC VR devices, but are still able to provide satisfactory VR experience. At the same time, in order to increase the universality, the application will also both support flat screen display of desktop and some expensive VR devices. In addition, Users are allowed to navigate their data to get better observation via different input techniques such as mouse, keyboard, gaze, voice and different remote hand controller.

Another main objective is to explore gene/protein expression profiles of single cells. Intuitive visualization of high dimensional features requires a new visual design. We achieve this goal, we develop a new visual representation that combines a three-dimensional scatter plot with radar charts for visualization of a very large number of multivariate data points. A radar chart is a graphical method for displaying multivariate data by providing an axis for each variable, and these axes are arranged radially around the centre point of a 2-dimensional chart with equal spacing. Features can be compared along their own axis for an individual, moreover overall differences among each individual point can be obtained by observing shape of the polygons. When a user selects a specific point, that point and its nearest points will be turned into radar charts showing the gene expression values of these cells.

In order to provide a smooth and stable website for visualizing large data sets, GPU hardware acceleration is applied to enhance the performance.

Software developments

We built a new smartphone-enabled VR application called Starmap, that enables immersive visualization of single-cell data for hundreds of thousands of cells using a mobile-enabled web browser and low-cost VR head mount device. Users can choose to view the application in a virtual environment (VR mode) using most modern VR devices, including VR-Box, Google CardBoard, VIULUX and VIVE, or on a flat screen via a PC (flat screen mode). The following subsections describe the detailed implementations of each component.

Input data

To view their data, users upload their local files stored on a PC or remote files stored in a cloud server onto our application. We use FileReader, a Web API, to read the input files. We focus on visualization of single-cell RNA-seq, flow cytometry and mass cytometry data after dimensionality reduction.

Input format

The accepted data input file format is a Comma-Separated Values (CSV) format, and a ZIP compression of the CSV format. Since the file translation of file sizes larger than 30MB (approximate 8 attributes for each point and 50K points totally) is time-consuming (i.e. from Cloud server to application), the user can upload a ZIP format file that contains only one input file. To decompress ZIP files, JSZip.js has been employed. Additionally, two demos both with a different sample size and data type are available at the landing page.

Content requirements

There are some content requirements for the input data. The first row of data must indicate a maximum of fifteen data attribute labels, containing at least the ‘x’, ‘y’, ‘z’ coordinates of 3D points and a ‘cluster’ attribute. Custom label names for high dimensional features can also be included into the first row. The value of features for each point should be a numeric value. A value of ‘-1’ in the cluster column indicates outliers (Figure 4).

x	y	z	cluster	CD25_FITC	CD45Ra_PerC	CD8_PE	CCR7_PeCy7	CD4_APCCy7	CD127_BV421
3.36419985	-1.0436875	3.57677757	5	273.312952	0	8180.5753	0	0	1384.61858
-1.034279	-1.8800266	-0.4666802	2	630.931576	6171.11468	0	6276.74438	1018.43632	470.135935
-1.0609177	-1.9071946	-0.8883475	2	796.604611	6179.01821	39.9384386	7580.45544	448.803914	314.558042
-1.3296678	-1.2584404	-0.1300179	2	496.523525	4675.46901	13.748088	7238.24949	1878.80259	806.909855
-1.2084918	-1.3650264	-0.2207991	2	613.059883	4639.68783	172.390328	7506.49217	1421.58813	680.516744
3.09855854	-0.5358364	2.38103802	5	0	0	8588.24673	3943.51252	0	1214.80227
2.66827336	0.21447621	-0.9374078	-1	1641.0813	6167.22576	7913.01791	6852.10789	234.50115	3653.38122
-0.9460879	-1.8653397	-0.5532112	2	773.696899	6151.04212	112.059131	6517.23538	701.363536	606.476922
-0.9613159	-1.3154819	-1.2213289	2	2669.0768	6650.59732	281.312393	8012.84762	367.859922	1612.71471
-0.9321766	-1.7986658	-0.7604751	2	720.184094	6124.96976	213.345776	7185.50585	498.665947	671.098802
-0.6506342	-0.7931387	-0.6541428	2	732.651931	6981.81416	1633.56175	6028.71965	3167.15325	1481.92665
-0.0129613	-0.6026124	0.52513183	-1	493.982583	6342.7914	132.202917	2010.49143	134.827681	6558.24582
-1.0139719	-2.0884088	-0.9590884	2	713.381516	6558.29062	81.9670312	7336.96535	429.373202	0
3.70486196	0.10899766	2.76676068	-1	323.520421	1304.89999	8688.09026	650.174587	0	4596.83202
-0.0590378	-2.681072	1.13944393	2	1053.92029	6363.84433	282.304905	473.02878	132.847552	997.987718
2.95714329	1.40101873	-1.6148758	1	1562.64546	6969.97844	8249.19651	7668.48822	15.4925452	7038.49887

Fig.4 An example of input file content.

Upload method

For PC, Android, or iOS 11+ devices, users can either use a local file system or cloud storage (e.g. iCloud, Google Drive, OneDrive, etc.) to upload input data. For earlier versions of iOS devices, cloud storage is the only uploading method since the web browsers cannot access local files directly (Figure 5).

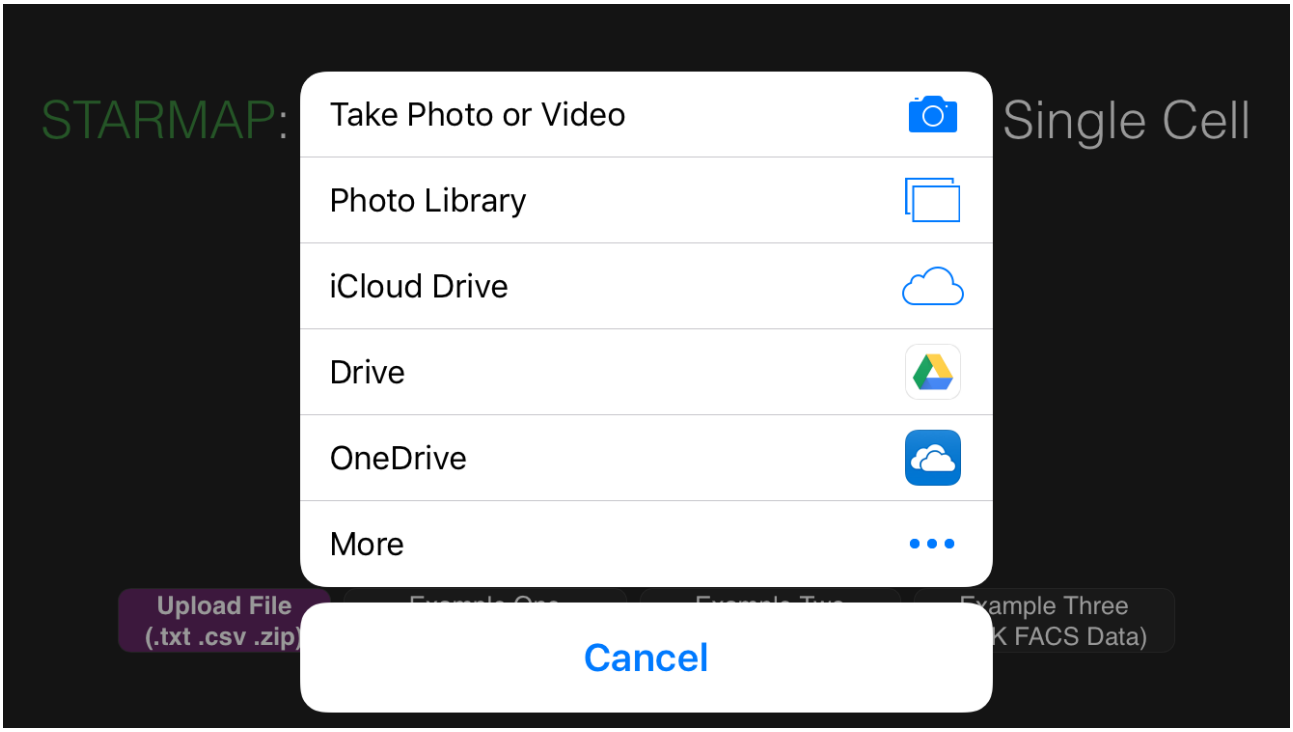

Fig.5 Upload input data via cloud storage.

Data pre-process

After an input data file has been uploaded, data pre-processing is crucial for getting better visualization effects. This involves rearranging a range of 3D coordinates and high dimensional features and using an initial camera (eyes) position to give a global view of the input data. Instead of using native looping to do calculations, we use matrix operations provided by numJs.js, which offers faster pre-processing speed especially on data with a large volume.

Data visualization

3D geometry visualization

We use A-Frame.js, cooperating with Three.js, for data visualization. After the rendering starts, a scene with a perspective camera is created in order to provide a smooth and stable visualization of large data sets, we use billboards, 2D planes that can automatically rotate their front face towards the camera, to represent 3-dimensional structures of the input data. In addition, THREE.buffergeometry is used as a computationally efficient method to reduce the cost of passing data (vertex positions, colour, etc.) to the GPU.

Every inlier point has a ball-shaped texture and size attenuated according to its distance to the camera. An outlier point represented by coloured dot is much smaller than an inlier point. (Figure 6). Semi-transparent bounding spheres for clusters are also rendered.

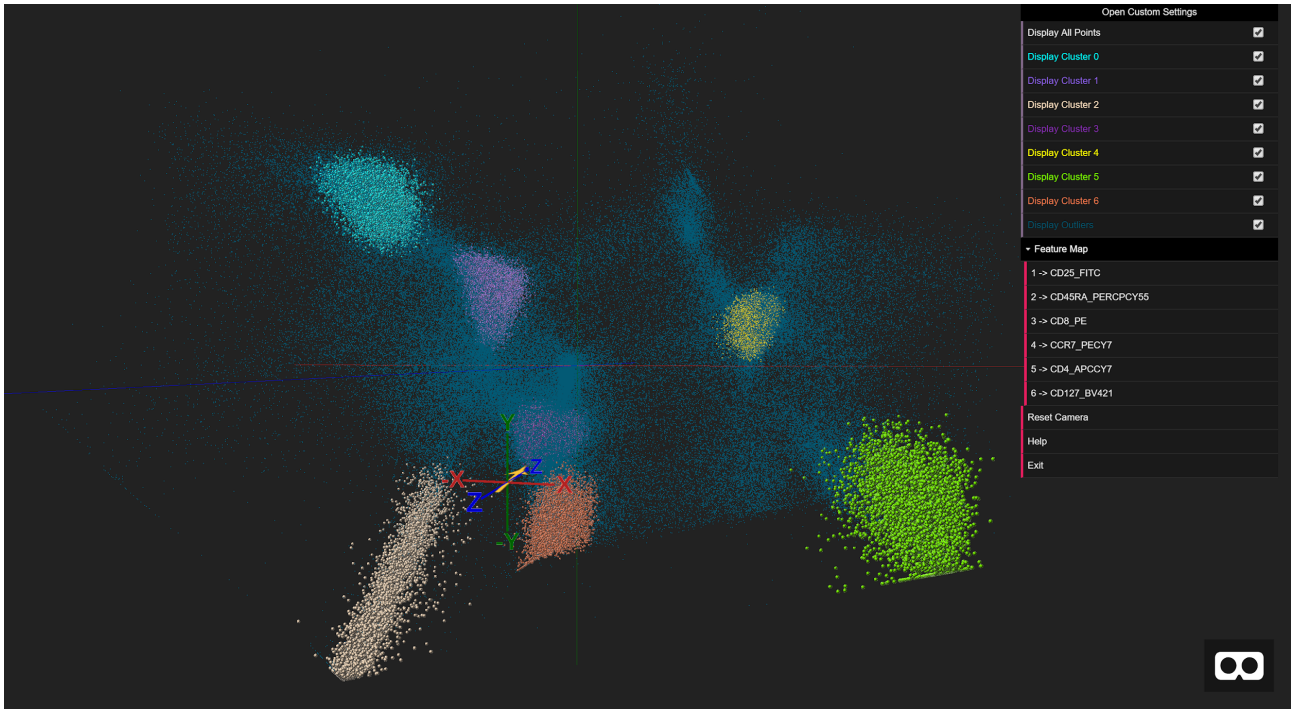

Fig.6 Global view of flow cytometry data contains outliers.

High dimensional features visualization

When an Inlier point has been selected, a radar chart implemented by a combination of 2D planes and lines will be covered on the point. Each spoke corresponds to a feature of the point. The data length of a spoke is proportional to the magnitude of the variable for the data point, relative to the maximum magnitude of the variable across all data points. A line is drawn connecting the data values for each spoke (Figure 7).

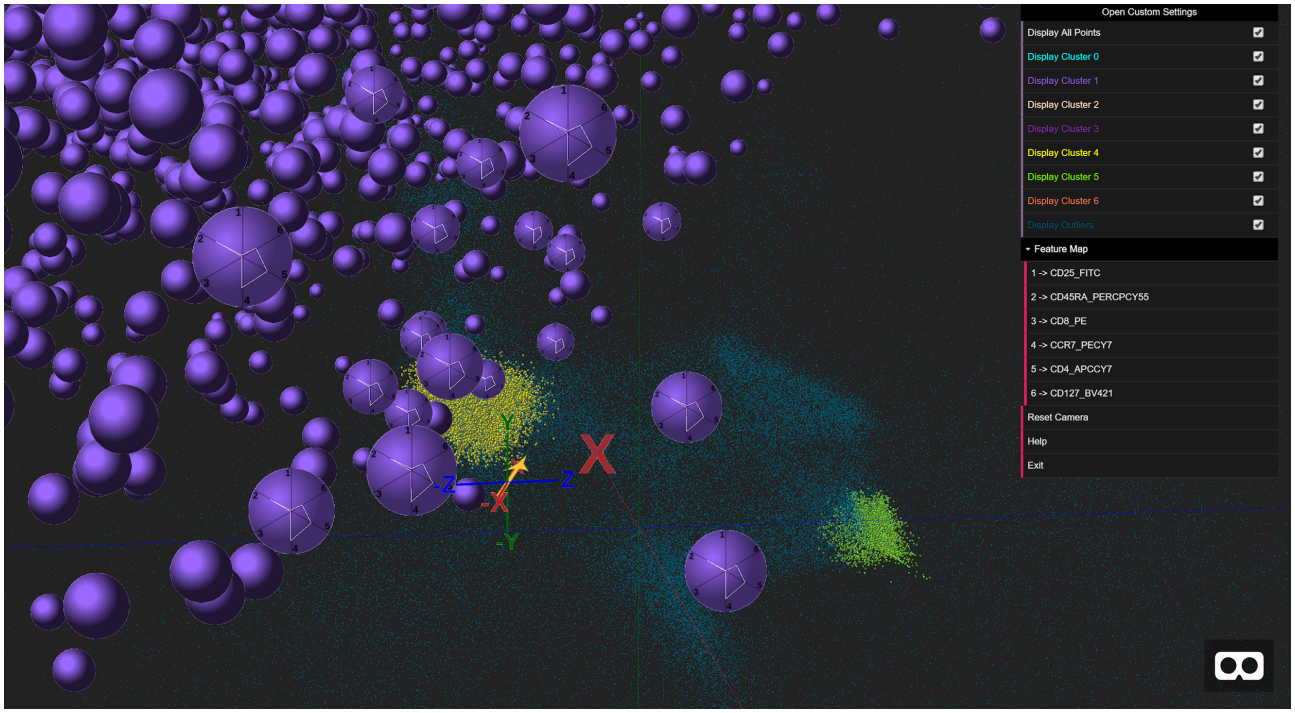

Fig.7 High dimensional features in flat screen mode.

3D compass

To prevent losses in large datasets while immersing into data for exploring detailed information, we implemented a 3D compass to indicate the orientation of the x, y, and z axes. (Figure 8). The x, y, z axes shown in red, green, and blue respectively, indicate the local coordinate system of data sets. A yellow arrow always points towards the camera's look at direction. The compass is updated automatically with any changes in the viewing angle.

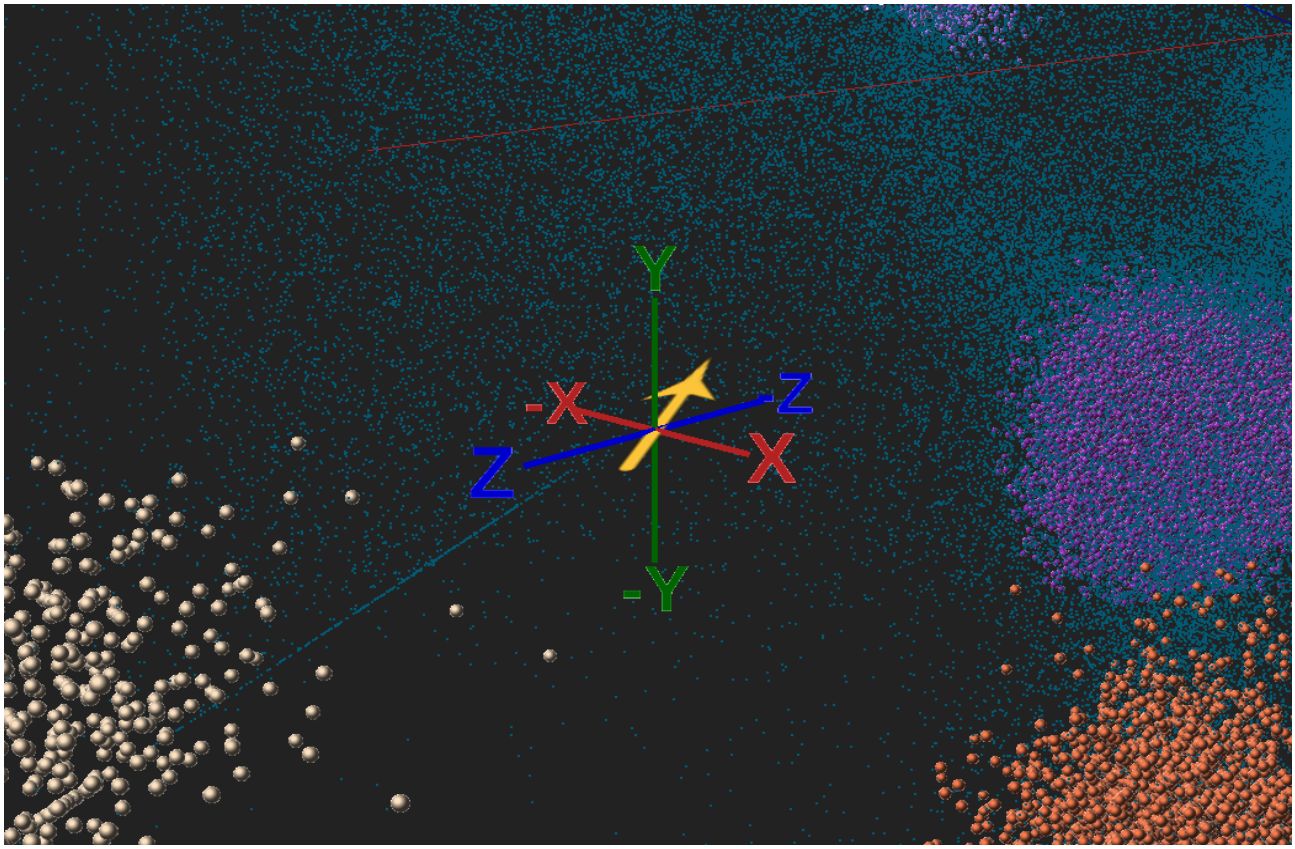

Fig.8 3D compass.

Interaction

Starmap provides a lot of interactions for manipulating with the scene. The camera can be moved to the left, right, forwards, or backwards, along the x and z axis of its own coordinate system. By changing the camera's position in the scene, users are able to drive into their input data to observe its detailed inner structure. The viewing perspective can also be changed by updating either the 3D rotation of the camera or data. In order to get a suitable points' density, Starmap allows users to scale the distance between points. When an inlier point is selected, high dimensional features are displayed on itself and its surrounding points.

GUI

To enhance user experience (UX) and enable more interactions with observing data, we set up a GUI (Figure 9) for both the VR and flat screen mode, based on modification of Dat.GUI.js and Dat.GUIVR.js. Bounding spheres of all clusters will be displayed if "Display

All Bounding Sphere" option is selected (Figure 10). If users are only interested in some clusters, we added options to select specific display clusters. In addition, the font colour of a 'Display Cluster' label indicates the colour of the corresponding point of the cluster in the application scene. A list called 'Feature Map' maps the axes label of a radar chart with the corresponding high dimensional features name. Moreover, users can reset the camera position by selecting the 'Reset Camera' button. A "Help" function is also available in the GUI to remind users of starmap's control instructions.

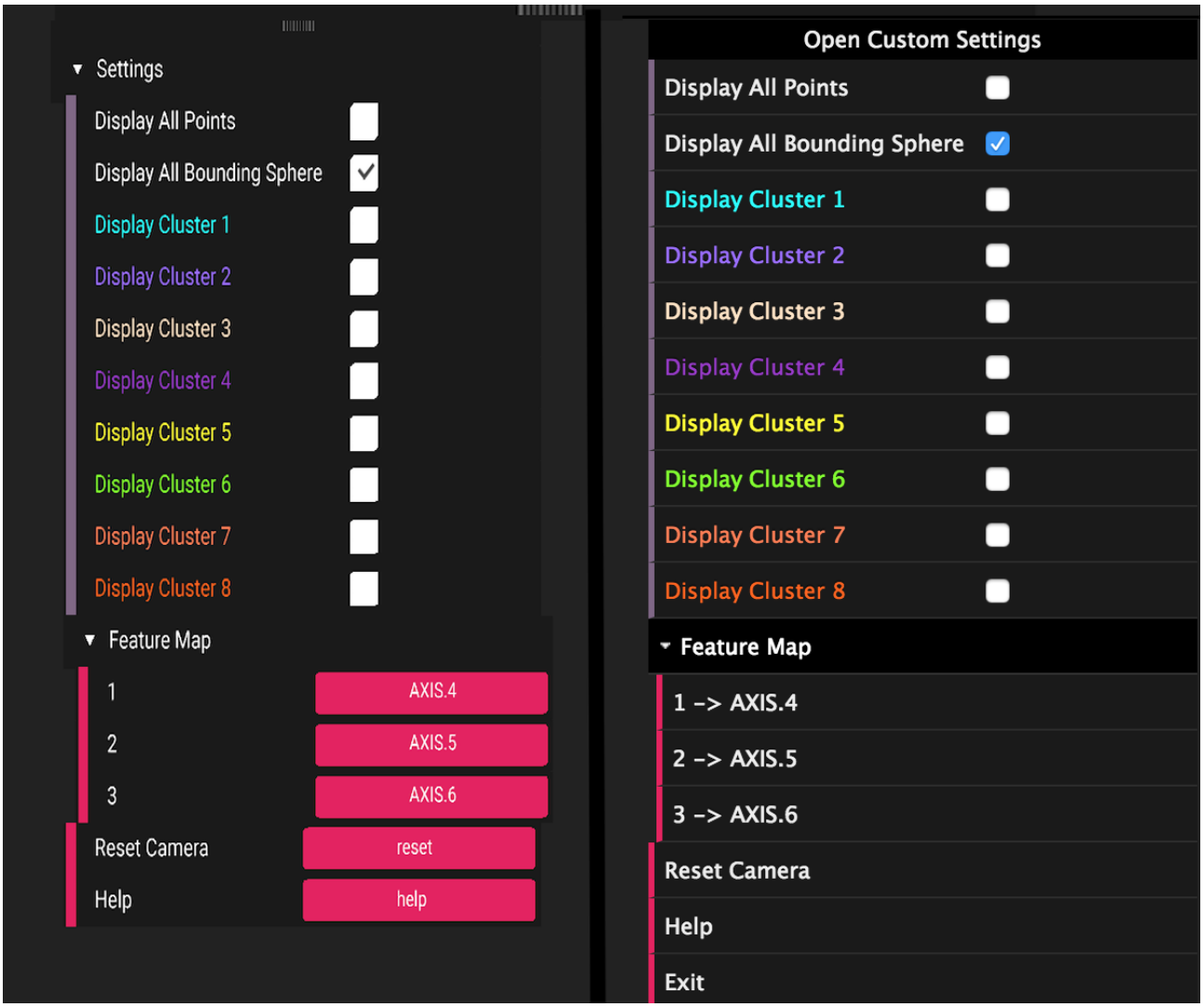

Fig.9 GUI in VR (left), GUI in flat screen (right).

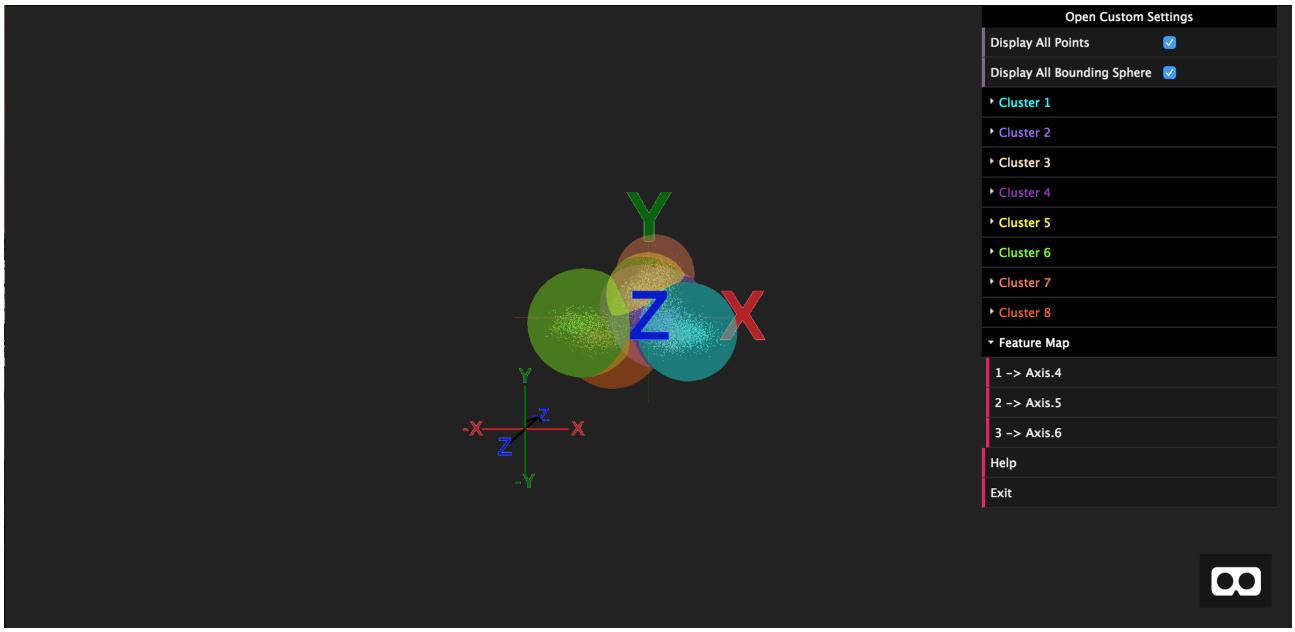

Fig.10 Display bounding spheres for all clusters.

Input method

By using JavaScript's "addeventlistener" method, Starmap can monitor signals sent by various different input technologies to interact with the scene or GUI.

Keyboard and iCade remote controller

The keyboard (Figure 11) is used to scale, rotate data, and move the camera when using the flat screen mode. When the "keydown" event of JavaScript is detected, a bound command will be executed once. The iCade Remote Controller (Figure 12) substitutes the keyboard when in VR mode. These controllers are usually sold as a package with VR headsets for approximately 20 US dollars. The implementation logic of iCade controller is different from a keyboard because of different technical standards. When a button is pressed (down and up), a pair of sequential "keydown" events are emitted. Between this pair of "keydown" event, bound commands are executed continuously.

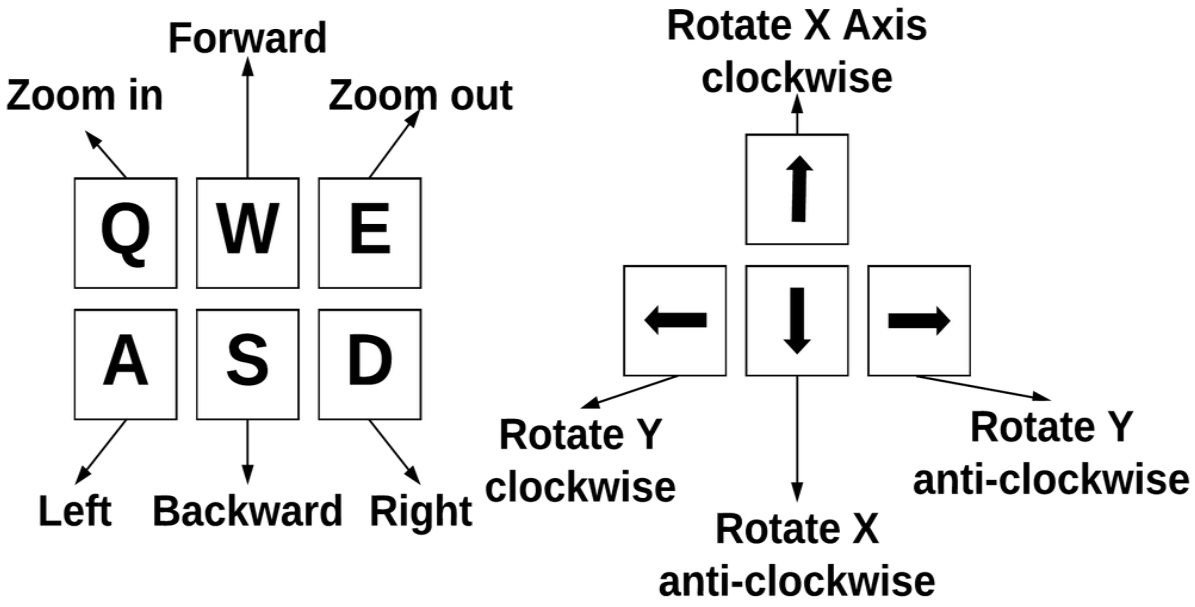

Fig.11 Keyboard Mappings

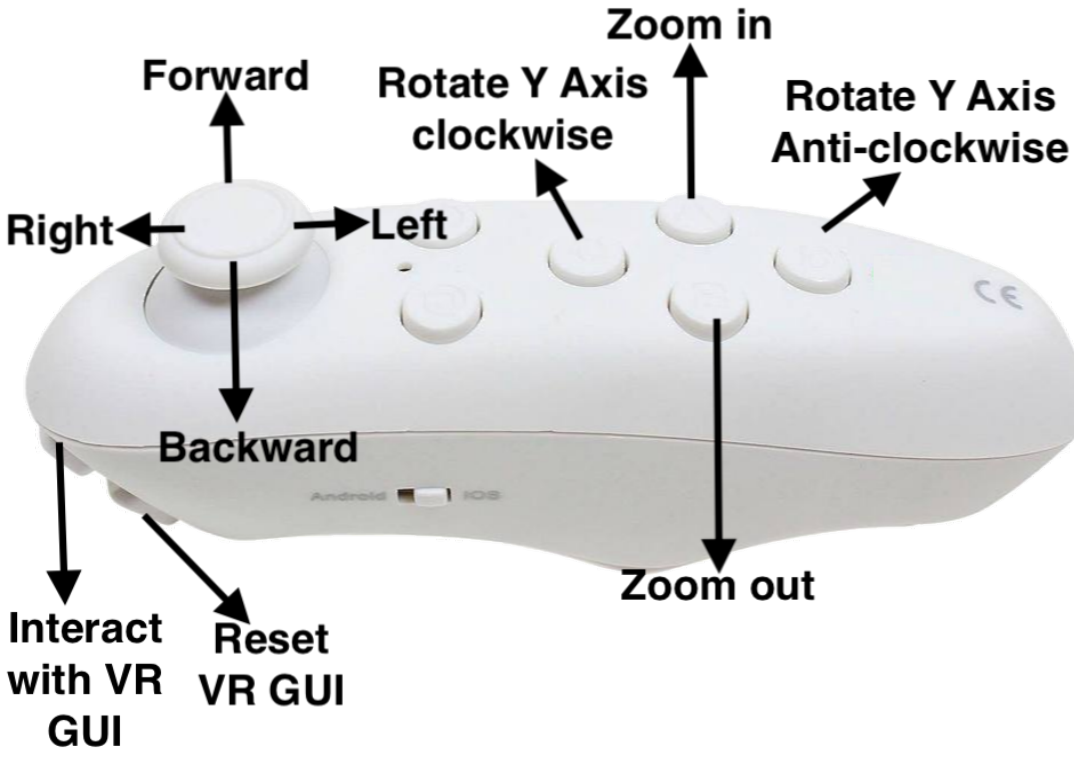

Fig.12 Remote controller Mappings

Mouse and gaze-based interactions

In the flat screen mode, the mouse is used to interact with the GUI- specifically to select points and control movement direction. The position of the mouse is detected by calling JavaScript's "mouse event" event method.

In VR mode, gaze is a substitution of the mouse control in VR mode (Figure 13) and uses a cursor which is fixed on centre of the screen. Users can control the cursor by changing the view angle with their head movements. The advantage of gaze interaction is that it is naturally supported by all VR devices. However, complex interactions may require excessive head movements from users. The "Raycaster" class of Three.js is used for detecting points of intersection for both the mouse and cursor.

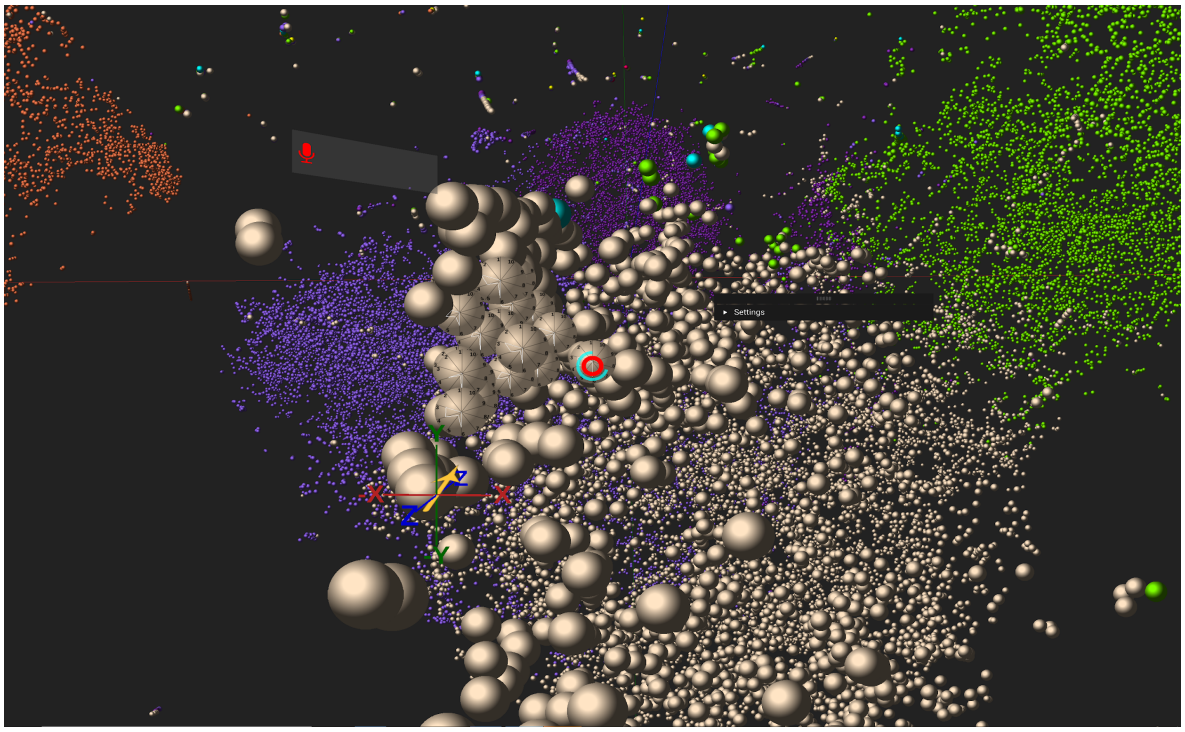

Fig.13 Gaze to select a point in VR mode

Voice control

By using both voice and gaze-based control, Starmap can be used without any extra input devices. Voice control, which is based on Annyang.js, is added in VR mode for all devices that support web speech APIs, and contains all functions of the remote controllers.

Corresponding functions are triggered when a correct voice command is recognized by Starmap. To increase the accuracy, each voice command consists of a single word (Figure 14).

Voice Command	
Forward	'forward'
Backward	'backward'
Left	'left'
Right	'right'
Zoom in	'in'
Zoom out	'out'
Rotate Y axis anticlockwise	'rotate'
Stop	'stop'
Click with VR GUI	'select'
Reset VR GUI	'reset'
Reset Camera	'init'

Fig.14 Voice command instruction

Evaluation

The evaluation focuses on the performance tests of local data loading time and rendering ability tests among popular mobile devices and desktop devices. Input data with different volumes (from 200 thousand to 1.5 million points) were generated using a Python CSV writer. All tests were run in Chrome.

Loading time test

The loading time for local files were calculated by the time difference between uploading start and rendering success. Since number of points in real data can be greater than 200 thousand, it was important to evaluate the loading speed of Starmap for different data sizes.

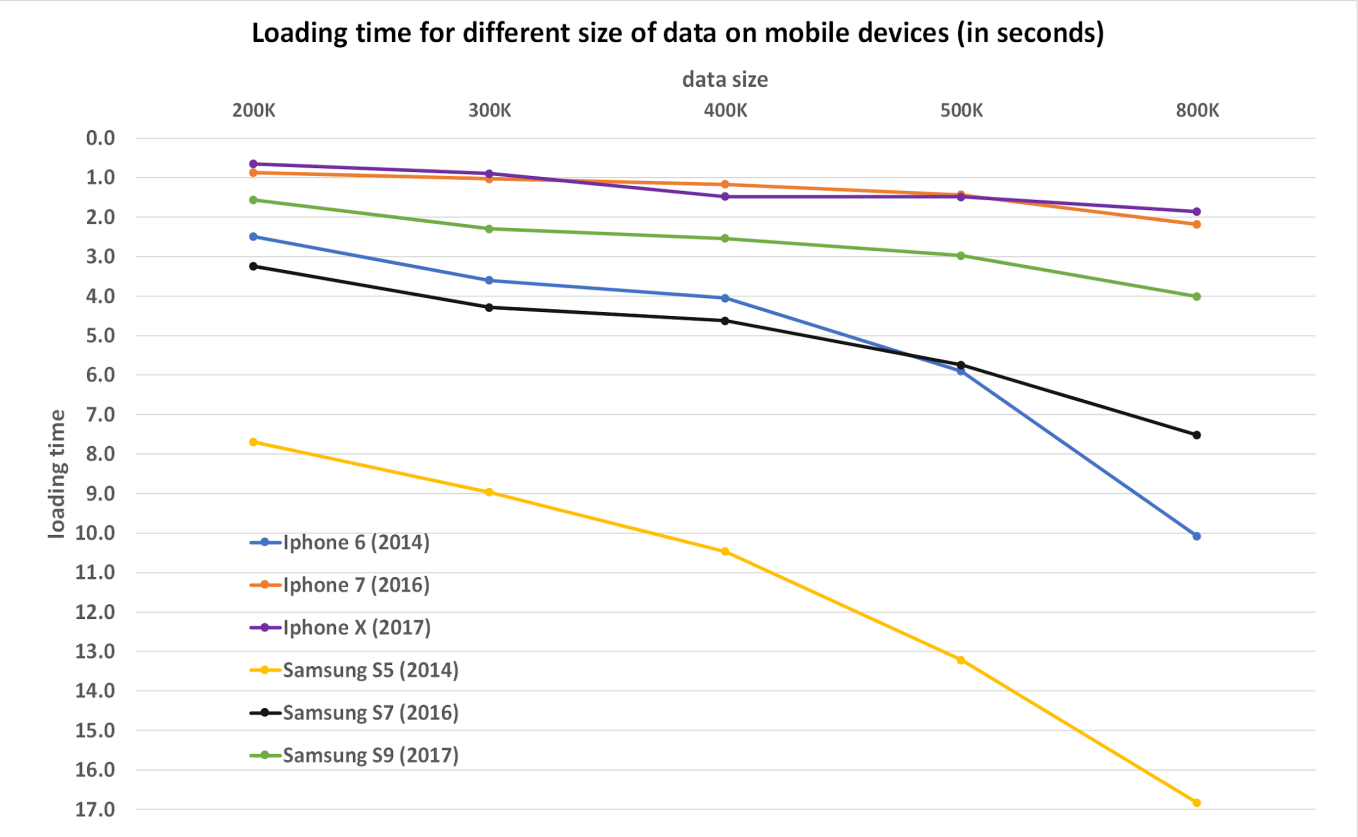

Fig.15 Loading time of different sizes of data on mobile devices

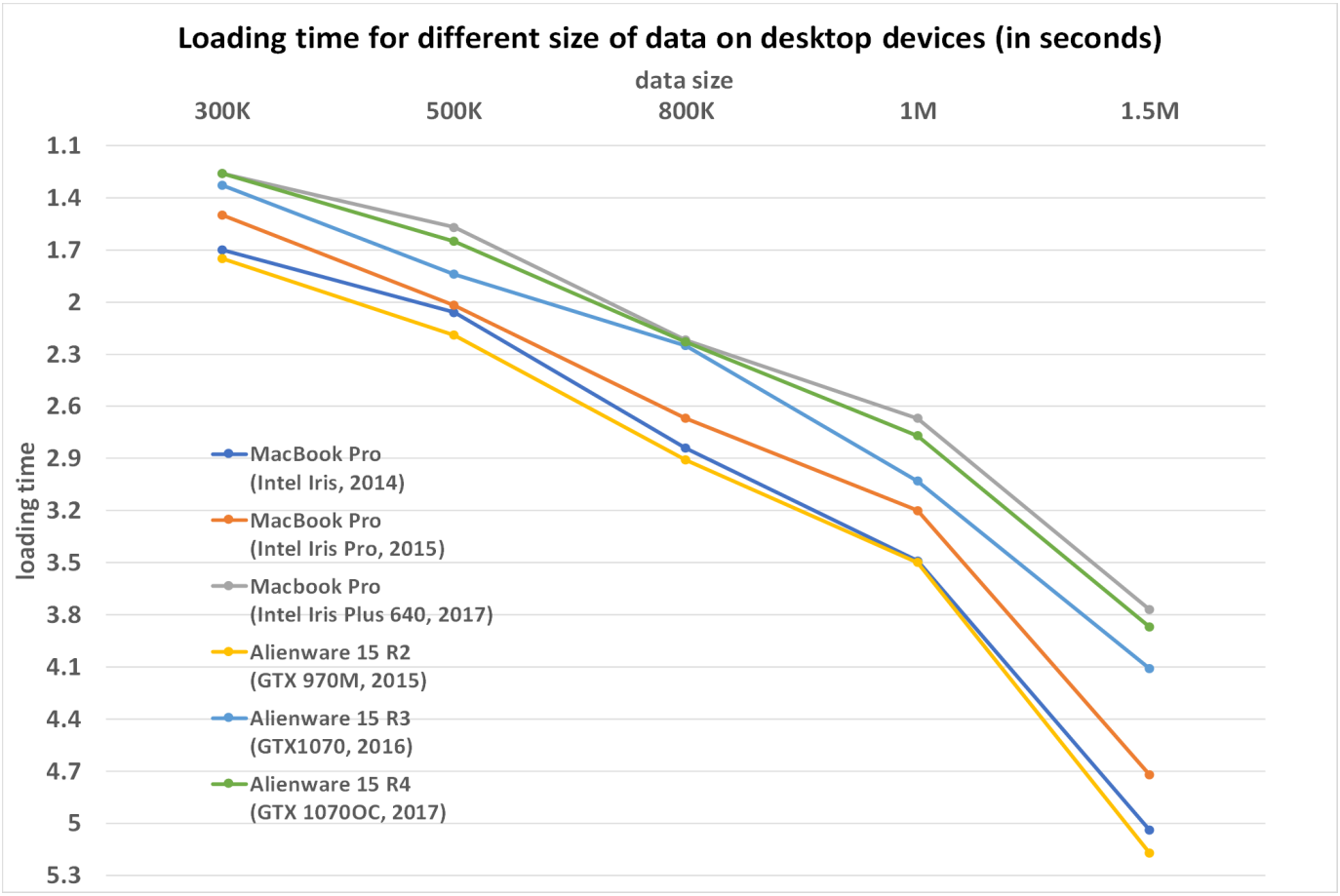

Fig.16 Loading time of different sizes of data on desktop devices

Figure 15 shows that later mobile devices are able to load large volume of data in a short time, especially the iPhone X and iPhone 7. Both were able to complete the task within 2 seconds despite the data file containing 800 thousand points. Similar to figure 15, figure 16 shows that the later desktop devices use less time to load. Moreover, all desktop devices were able to load 1.5 million points within 6 seconds.

Rendering ability test

A-Frame.js' stats component was used to test the rendering ability to monitor frames per second (FPS). The aim was to evaluate whether devices are able to obtain a stable visualization at 30 FPS (a comfortable FPS for human eyes) for different volumes of data.

Amount of points can be rendered on mobile devices with FPS ≥ 30					
data size / mobile devices (name, years)	200K	300K	400K	500K	800K
Iphone 6 (2014)	✓	✓	✓	✗	✗
Iphone 7 (2016)	✓	✓	✓	✓	✗
Iphone X (2017)	✓	✓	✓	✓	✓
Samsung S5 (2014)	✓	✓	✓	✓	✗
Samsung S7 (2016)	✓	✓	✓	✓	✓
Samsung S9 (2017)	✓	✓	✓	✓	✓

Fig.17 The amount of points which can be rendered on mobile devices with FPS ≥ 30

Amount of points can be rendered on destop devices with FPS ≥ 30					
data size / desktop devices (name, GPU, yrs)	300K	500K	800K	1M	1.5M
MacBook Pro (Intel Iris, 2014)	✓	✓	✓	✗	✗
MacBook Pro (Intel Iris Pro, 2015)	✓	✓	✓	✗	✗
Macbook Pro (Intel Iris Plus 640, 2017)	✓	✓	✓	✓	✓
Alienware 15 R2 (GTX 970M, 2015)	✓	✓	✓	✓	✗
Alienware 15 R3 (GTX1070, 2016)	✓	✓	✓	✓	✓
Alienware 15 R4 (GTX 1070OC, 2017)	✓	✓	✓	✓	✓

Fig.18 The amount of points which can be rendered on desktop devices with FPS ≥ 30

Figure 17 and figure 18 show that the rendering ability is highly dependent on the performance of the GPU because Starmap makes use of the GPU acceleration. Thus, the results show that desktop devices usually have better rendering power than mobile devices, with later devices having better rendering power in comparison to earlier devices. All the desktop devices could reach at least 800,000 points with 30 FPS, and 400,000 points for mobile devices. In addition, most of the later desktop devices were able to reach a stable rendering of over 1.5 million points, while later mobile devices could reach 800 thousand points.

Further work

Movement animation generation

We will implement an algorithm that can automatically find a camera movement path to nicely describe the input data. The key spots (i.e. a position for observing global structure of data) and meaningful paths (i.e. decision boundaries between clusters) should be included in the generated movement path. In addition, collisions between the camera and points should be avoided.

Support point recluster

There are a lot of clustering methods, and the results produced by them may significantly different. Starmap currently does not have the ability to swap clusters of points between these results. Thus, a function will be implemented to recluster points based on the selection of different cluster results contained in input data via the GUI.

Support VR gloves controller

To make interactions in VR more intuitive and natural, VR gloves (Figure 19) will be integrated into Starmap for tracking hands movement in real time. Users will be able to display profiles of a cell and interact with the GUI by touching, and potentially set the visibility of an area by circling with their fingers.

Fig.19 VR gloves *https://manus-vr.com/#product-anchor*

Conclusions

In order to visualize multivariate single cell data with large volume to help biologists efficiently explore important information, a feasible solution was carried out after summarizing advantages and problems based on current methods and investigations on available technologies and development weapons. Starmap, a single cell data visualization tool, was also implemented, providing not only efficient multivariate data visualization, but also enabling a widespread adoption of VR data visualization by supporting low-cost VR devices.

Bibliography

A-FRAME, 2017. *Introduction-A-FRAME.* [Online]
Available at: https://aframe.io/docs/0.7.0/introduction/
[Accessed September 2017].

Alexei A., S., Dawood B., D. & Minoru, S., 2005. A web-based tool for principal component and significance analysis of microarray data. *Bioinformatics,* 15 May, 21(10), pp. 2548-2549.

Andy, C. et al., 2006. Remote large data visualization in the paraview framework. *EGPGV,* 11 May.pp. 163-170.

Conte, G. et al., 2015. BRAINtrinsic: A virtual reality-compatible tool for exploring intrinsic topologies of the human brain connectome. In: *Brain Informatics and Health.* s.l.:Springer International, pp. 67-76.

Donalek, C. et al., 2014. Immersive and collaborative data visualization using virtual reality platforms. *Big Data,* October.pp. 609-614.

Eric, B. et al., 2005. A contract based system for large data visualization. In Visualization. *Visualization,* October.pp. 191-198.

Fuchou, T. et al., 2009. mRNA-Seq whole-transcriptome analysis of a single cell. *Nature methods,* 6 April, 6(5), pp. 377-382.

Google, 2016. *Google VR.* [Online]
Available at: https://developers.google.com/vr/
[Accessed October 2017].

Joachim, T. et al., 2004. GECKO: a complete large-scale gene expression analysis platform. *BMC bioinformatics,* 10 December, 5(1), p. 195.

Karl, P., 1901. LIII. On lines and planes of closest fit to systems of points in space. *Philosophical Magazine,* 2(11), pp. 559-572.

Laha, B. & Doug A., B., 2012. Identifying the benefits of immersion in virtual reality for volume data visualization. pp. 1-2.

Marie-Danielle, V.-D.et al., 2013. Virtual reality for skin exploration. *In Proceedings of the Virtual Reality International Conference: Laval Virtual,* 20 March.p. 5.

Matthew, H. S. & Garry, P. N., 2016. Mass Cytometry: Single Cells, Many Features. *Cell,* 5 May, 165(4), pp. 780-791.

Microsoft, 2017. *BabylonJS Documentation.* [Online]
Available at: http://doc.babylonjs.com/
[Accessed September 2017].

Nelson, L., Dianne, C. & Carolina, C.-N., 1999. Xgobi vs the c2: Results of an experiment comparing data visualization in a 3-d immersive virtual reality environment with a 2-d workstation display. *Computational Statistics,* 14(1), pp. 39-52.

Oscar, L. et al., 2018. CellexalVR: A virtual reality platform for the exploration and analysis of single-cell gene expression data. 24 May. Manuscript submitted for publication.

PlayCanvas, 2017. *User Manual|Learn PlayCanvas.* [Online]
Available at: https://developer.playcanvas.com/en/user-manual/
[Accessed October 2017].

Ramachandran, P. & Gaël, V., 2011. Mayavi: 3D visualization of scientific data. *Computing in Science & Engineering,* 13(2), pp. 40-51.

Saeys, Y., Sofie, V. G. & Bart, N. L., 2016. Computational flow cytometry: helping to make sense of high-dimensional immunology data. *Nature Reviews Immunology,* 20 June, 16(7), pp. 449-462.

Sean, C. B., Garry, P. N., Mario, R. & Pratip, K. C., 2012. A Deep Profiler's Guide to Cytometry. *Trends in immunology,* July, 33(7), pp. 323-332.

Sho, M., Ayako, S., Sumio, S. & Yutaka, S., 2014. RNA Sequencing: From Sample Preparation to Analysis. *Transcription Factor Regulatory Networks: Methods and Protocols,* Volume 1164, pp. 51-65.

Theart, R. P., Ben, L. & Thomas, R. N., 2017. Virtual reality assisted microscopy data visualization and colocalization analysis. *BMC bioinformatics,* 18(2), p. 64.

Threejs, 2017. *three.js docs.* [Online]
Available at: https://threejs.org/docs/index.html#manual/introduction/Creating-a-scene
[Accessed September 2017].

Unity, 2017. *Unity-Solution.* [Online]
Available at: https://unity3d.com/solutions
[Accessed October 2017].

Valve, 2015. *OpenVR.* [Online]
Available at: https://github.com/ValveSoftware/openvr/wiki/API-Documentation
[Accessed September 2017].

Vladimir, K., Tallulah, A., Davis, M. & Martin, H., 2017. *Introduction to single-cell RNA-seq.* [Online]
Available at: https://hemberg-lab.github.io/scRNA.seq.course/

WebGL, 2017. *WebGL 2.0 Specification.* [Online]
Available at: https://www.khronos.org/registry/webgl/specs/latest/2.0/
[Accessed October 2017].

WebVR, 2017. *WebVR.* [Online]
Available at: https://w3c.github.io/webvr/spec/latest/
[Accessed October 2017].

Yuting, Y. et al., 2005. Integration of metabolic networks and gene expression in virtual reality. *Bioinformatics,* 15 September, 21(18), pp. 3645-3650.