A Virtual Reality-based Multiplayer Game using Fine-grained Localization

Tom De Schepper, Bart Braem and Steven Latré
Department of Mathematics and Computer Science, University of Antwerp - iMinds, Belgium
Email: {firstname.lastname}@uantwerpen.be

***Abstract*—The popularity and availability of Head Mounted Displays (HMDs) has known an increase over the past few years. In general those devices need to be connected to a computer to function properly. Moreover, applications that project images on such an HMD respond typically only to user inputs like mouse and keyboard actions and head movement. In this paper we go a step further by proposing a framework in which the movements of a person are translated into actions in a virtual world. This allows a user to walk around freely while wearing an HMD, increasing the reality of the virtual experience. We will focus in detail on the challenges related to the localization of the user and discuss different options like existing wireless localization technologies, Bluetooth Low Energy (BLE) and an object recognition algorithm. The latter is used in the final solution. Furthermore, we present an experimental and mobile setup to use an HMD in a portable manner. Our solution is demonstrated through a multiplayer game, in which two players compete against each other to capture a flag.**

Keywords*—*Virtual Reality, Multiplayer Game, Oculus Rift, Wireless Localization, Object Tracking.

I. Introduction

While the concept of Virtual Reality (VR) was originally only known to most people through some Hollywood blockbusters, there has been an increase in the commercial availability over the last decade. This phenomenon can be explained by the technological improvements and the resulting dramatic drop in the cost of those systems. The Microsoft Kinect and the Nintendo Wii are two examples of VR devices that are now used by many around the globe.

A part of this evolution is the surfacing of a new generation of Head Mounted Displays (HMDs), like the Oculus Rift (OR). Those devices are in most cases connected with a computer as they require power and an HDMI input to present a 3D view of an image. This means that a user is restricted to a close area around that computer. Furthermore, the current HMD applications respond to only two kinds of user inputs: the traditional mouse and keyboard actions and the head movements of the person wearing the HMD. These characteristics limit the VR experience due to a low immersiveness: the user has the feeling of controlling an alter ego, rather than being the person in the VR world himself.

In this paper, we present a mobile framework in which a user can move around freely, while wearing an HMD. This increases the immersiveness and correspondingly creates a more realistic experience. The contributions of this paper are the following: we analyse multiple approaches for localizing a person or object. We build a framework, implementing the most suited localization method, that tracks the user's movements and translate those movements into corresponding actions in a virtual world. This setup can be used in both indoor and outdoor environments and is mobile and scalable. Furthermore, we consider different options for using an HMD in a portable manner. We will demonstrate our proposal by means of a multiplayer game in which two players compete against each other to capture a flag that is located in a labyrinth. A schematic representation of the high-level concept is shown in Figure 1.

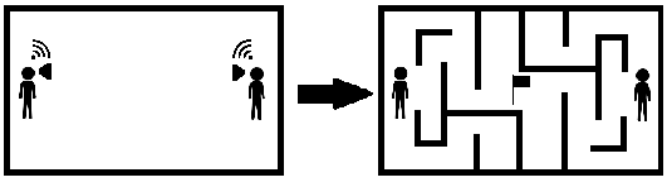

Fig. 1: *Schematic representation of the concept*

Summarized, we can distinguish three main challenges: first of all, we will need to find a way to track an object, in this case a player, in order to know its current location. Secondly, we will need to implement the multiplayer game itself and finally, we will need to make everything work in a mobile context.

Besides gaming or entertainment purposes, the presented framework could be used for training purposes in both military and civil areas. It offers a more realistic experience than existing simulators and has the advantage that the virtual world can be easily adapted to the specific needs of the training. Another promising application is to offer people the chance of realistically exploring a virtual model of a building or city. This way architects could check if a proposed design matches the requirements and expectations of a customer. Tourists could walk through an replica of an ancient temple in an identical manner.

The remainder of this paper is structured as follows: we start by giving an overview of the current state in this field in Section II. Next, we present a solution to the localization problem in Section III and describe the overall framework in Section IV. We finish with discussing the results in Section V and stating conclusions in Section VI.

II. Related Work

Tracking a user's movement is often performed by means of InfraRed (IR) cameras. Those cameras are capable of providing a very accurate location of a user but they are quite

978-1-4673-7707-2/15/$31.00

expensive, can only be used indoors and are not suited for a mobile setup. Marks et al. [1] describe the development and operating principles of such a system. Other researchers have implemented this idea and managed to let a user walk around in a small area while wearing an OR [2]. At first their solution involved connecting the OR to a laptop that was being carried by a second person walking behind the one wearing the HMD. Later this was improved by strapping a battery pack and a wireless HDMI receiver to the test subject. The video is sent to the receiver by a powerful computer connected with a wireless HDMI transmitter.

Other projects focus on the combined usage of an OR and a Microsoft Kinect [3], [4]. The primary goal of [3] is providing some visual transformations of the real world, for instance providing color blind filters for dichromacy color blindness types, a night vision mode or a tool to accurately calculate the user's distance to a certain item. This project was inspired by a Microsoft sponsored research project called Kinect Fusion. The second project had the ambition of creating an interactive virtual building walkthrough. They successfully used the OR to create a fully immersive experience and the Kinect for realistic environment interaction using limb tracking.

Finally, we would like to mention a promising product, called Virtuix Omni [5]. It consists of a special platform that simulates the motion of walking. A user standing on the platform wearing some special equipment and an OR will be able to traverse a virtual world by moving his or her feet.

III. Localization

In this section we address the problem of tracking the location of a moving object or person. Since these location estimates will be used to adapt the view on a virtual world, we require the solution to be as accurate and well-performing as possible to avoid abrupt jumps in the video output. In terms of accuracy we are talking about a magnitude of millimetres or a few centimetres. First we will look at wireless technologies for localization and secondly we will look towards image processing.

A. Wireless Localization

Given the mobile and flexible demands for our framework, we are interested in the usage of wireless localization. Furthermore, many wireless technologies are by default implemented in the latest smartphones. Since those devices are widely available, fairly cheap and easily carried by a person, localizing players through their phones looks like a promising possibility.

Over the years a variety of systems have been developed with the capability of estimating the location of a device transmitting a signal [6], [7]. In general four distinct techniques have been implemented: trilaterion, the proximity technique, scene analysis techniques and dead reckoning techniques. Except for dead reckoning systems, wireless technologies are also used in collaboration with these techniques. We will shortly discuss a selection of the available wireless technologies and the achieved accuracy:

- The commercially most successful localization system is GPS. While its accuracy can be up to 3.5 m in an outdoor scenario, the poor coverage of satellite signal for indoor environments makes it unsuitable for indoor location estimation [7]. Other satellite navigation systems, like GLONASS, behave in a similar matter.
- In the previous section we mentioned some research projects using IR cameras to localize a person with an accuracy of 1 cm. We opted not to use these cameras because of their price and immobility.
- The most used technology in proposed localization systems because of its wide availability is Wi-Fi. The localization with Wi-Fi can achieve an accuracy in the range of 1-5 m, with the best consistently between 1-2 m [6], [7].
- A last option is Bluetooth. Compared with Wi-Fi it has a shorter range (typically 10-15 m) but it is also less power consuming, which is interesting for us because of the mobile context. The accuracy turns out to be comparable with Wi-Fi, so at best 1 m [6], [7].

We can conclude that none of the options above is suited for our framework. A more recent technology called Bluetooth Low Energy (BLE) [8]. This newest version of Bluetooth uses less power and has a lower cost, while maintaining the same range. This makes it ideal for all kinds of monitoring purposes, for example, in healthcare or sports. As BLE's performance in terms of fine-grained localization has not been studied yet in depth, we conducted a number of small experiments.

The setup of this series of tests is basic. We place a Texas Instruments SensorTag (ST) [9] - a small device that transmits a BLE signal - at certain locations. The transmitted signal is captured by a receiver that is placed at a know distance from the ST. Next, we acquire an estimate of the distance between the two devices by using the Received Signal Strength Indicator (RSSI), a measurement for the power of the signal:

$$RSSI = -(10n \log_{10} d + A). \quad (1)$$

Where n is the signal propagation constant, d the distance and A the received signal strength at 1 m.

The first experiment took place indoor with an ST placed at four different distances. First, we measured the RSSI of the signal at a distance of 1 m and used the average as reference (variable A in the equation). For the signal propagation constant n we used a calibration technique to get an accurate value [10]: we calculated the value for n in each situation and used once again the average (in this case 2.74) for the distance calculations. When comparing the acquired estimates in Table I with the actual distances, we see an error in the worst case of 0.98 m

A second experiment was conducted outdoor with an ST placed at six different distances. The results are shown in Table II. The manner of calculation was identical to the indoor test and a value of 1.1 was used for n . When comparing the results of the second experiment with the results of the indoor experiment, it is clear that there is not a big difference in the signal strength. Only at the very short distances, 0.5 m and 1 m, there is a noticeable decrease in signal strength in the outside situation. One can also see that the measured RSSI in the cases of 7.5 m, 10 m and 12 m m are very similar, so it is very hard to distinguish between the different locations. This results in a larger overall worst case error of 3.26 m.

The results of the conducted experiments are comparable with those of other research [11], [12] and clearly indicate the large potential of using BLE in localization systems. However, the achieved accuracy is insufficient for our goals and for

$Distance$	$AverageRSSI$	n	$Estimate$
1 m (reference)	-57.52 dBm	N/A	N/A
0.2 m	-49.87 dBm	1.09	0.53 m
0.5 m	-48.76 dBm	2.91	0.48 m
2 m	-70.52 dBm	4.32	2.98 m
5 m	-75.95 dBm	2.64	4.71 m

TABLE I: *Indoor experiment*

$Distance$	$AverageRSSI$	n	$Estimate$
1 m (reference)	-68.59	N/A	N/A
0.5 m	-66.43 dBm	0.72	0.64 m
2 m	-73.67 dBm	1.69	2.90 m
5 m	-74.42 dBm	0.83	3.39 m
7.5 m	-79.78 dBm	1.28	10.42 m
10 m	-80.94 dBm	1.23	13.26 m
12 m	-79.81 dBm	1.04	10.48 m

TABLE II: *Outdoor experiment*

that reason we will describe an alternative in what follows. Note that our tests were focused on gathering preliminary accuracy information for our specific purposes and were not very detailed. To draw more general conclusions we believe that a more thorough study should be conducted.

B. Image Processing

As wireless communication-based methods proved not to be feasible, we now turn our attention towards image processing and object recognition in particular. The general idea is to place cameras around the four sides of the area in which the players will move. The captured video stream can then be used as the input to an algorithm that is capable of recognizing an object or a person throughout the different images. If we know where the object is located on a frame in terms of pixels, we basically know its position in one dimension. Combined with the results acquired from a second camera feed of the other dimension, we thus are able to know the position of that object in a 2D coordinate system. This position can then be translated into a position in a virtual scene.

As image processing has much been studied before, we went looking for solutions without the need of developing custom algorithms. We opted to use the OpenCV library, a collection of specific functions and algorithms for real-time computer vision purposes [13]. One of those algorithms is Camshift, originally proposed by G. Bradski [14]. It is a face and colored object tracking algorithm that is histogram backprojection-based and internally uses an older algorithm called Meanshift. Histogram backprojection is the reverse process of calculating a histogram. This method uses a given histogram to search regions in a certain image that match the histogram distribution. This means that an object can be tracked throughout a video stream, given the histogram of that object.

The algorithm is fairly simple and computationally efficient [14], [15]. Because of its simplicity there are some limitations. E.g., the algorithm only works with objects that do not change color or form. This also means that the light of the environment must not fluctuate too much. Another limitation is the fact that the tracked object should not disappear from the scene. If no similar area can be found in the scene, the algorithm stops. This is something we will need to take into account for our setup since it is possible that both players are standing

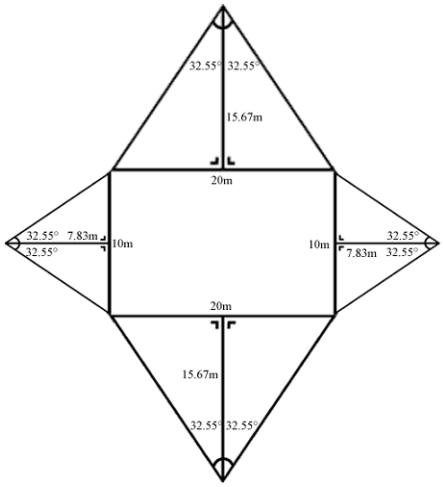

Fig. 2: *Example field setup*

directly on one line with the camera. In this case one player is blocking the view of the camera on the other one, which causes the algorithm to stop. We solved this by starting to track the second player, the moment the first one becomes invisible. While tracking the second user, we continuously restart the first algorithm until it does not immediately stop. If that happens, this means that the original player being tracked is once again visible and tracking the second user can stop. Furthermore, since the algorithm is color based, we need to make sure that the players can be distinguished from the background and from each other. For this reason, we propose to let the two competitors wear fluorescent safety vests in different colors. These jackets are available in almost all colors, so depending on the environment different colors can be chosen.

The accuracy of this approach depends on the size of the field and on the resolution of the camera. We have chosen to use mobile devices with a built-in camera. The latest smartphones are able to record with Ultra HD resolution at 30 or 60 frames per second (FPS). For a side of 10 m this results in a density of at least 1 pixel per centimetre, giving us the required accuracy. Furthermore, it is important to take into account the angle of view (AOV) of the cameras. Mobile devices have a different angle depending on the model and on how the phone is turned. The horizontal angle (with the mobile phone in landscape mode) is the largest, so we will assume the device is landscape mode throughout the rest of the project. This AOV has consequences for our setup because we need to place a camera at a certain distance to be able to cover an entire side of the field. We can calculate the needed distance by using basic trigonometry:

$$d = \frac{\frac{x}{2}}{\tan\frac{a}{2}}, \tag{2}$$

where d is the required distance, a the horizontal AOV and x the length of the field side. If we fill in this formula for an example with a size of 10 m and an AOV of 65.1 degrees, as illustrated by Figure 2, we get a result of 7.83 m. This is a bit more then 75% of the input and means that there is extra space required besides the game field itself. Given those dimensions, there are not many indoor locations where this setup would fit. Only buildings like a sports hall or a hangar would qualify.

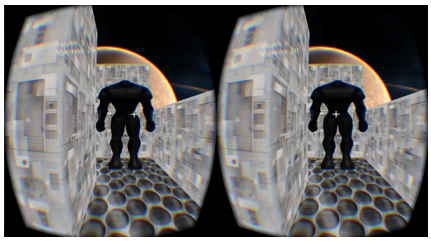

Fig. 3: *Client view*

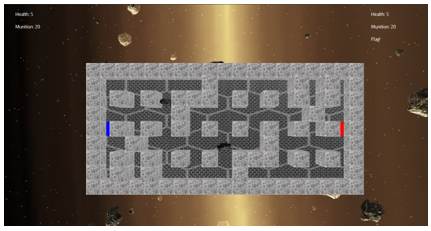

Fig. 4: *Spectator view*

Fig. 5: *User-equipment*

Fig. 6: *Selfie stick*

IV. Framework Overview

We will now describe the complete framework and our solution for the two remaining challenges. We already mentioned that we will use smartphone cameras for the tracking part. We have opted to write an Android application that handles the tracking and processes the captured frames. After the Camshift algorithm has determined the new location of the player, this information is sent to the (game) server.

To create the multiplayer game we have opted to use the jMonkeyEngine [16], an open source java based engine, for a number of reasons: the offering of free OR support, the availability of very detailed tutorials covering all features and the intuitive development environment based on Netbeans focused on programmers instead of graphic designers.

The architecture of the game is fairly simple since we just have a server that communicates with a few clients, a so-called star network. We will have one (for testing purposes) or two clients that will connect to a server. Furthermore, we added a separate client for a spectator, where a bird's-eye view of the game map is visible, to follow the actions of the competitors in the virtual world. An example of the client and spectator interface can be seen in Figures 3 and 4, respectively. The SpiderMonkey API [17] is used to handle the communication between the different components. This is a high-level networking library that is embedded in the jMonkeyEngine and uses the Java.net package. It supports, among others, the sending and receiving of messages over both UDP and TCP. The messages in our application that are sent to update the location of a player will be sent over UDP. This is because if one message does not arrive, the next one will make the lost message obsolete and synchronizes the state again. Given the nature of the game, messages related to the location make up for 99 percent of all network traffic. All other messages will be sent over TCP, this includes messages needed to start and setup the game correctly or those that are sent when a player picks up the flag or fires a shot at the competitor.

By using the tracking algorithm, explained in the previous section, we can detect movements in four directions. However, a player should also be able to rotate and to shoot at the other player. To make this kind of actions possible, we have chosen to let a user hold a normal computer mouse. The left button is used to fire a shot, while the scroll wheel allows a player to rotate the body of his alter ego. Note that at the beginning of the game, the player will need to look at a specified direction in reality (to the center), for allowing synchronized rotating in both the real and the virtual world. The rotation of head is registered by the sensors in the HMD itself.

During the development of the game we encountered a new type of problems for game behaviour related to the difference between the real world and the virtual environment. An example of such an issue concerns the walls of the maze. In the game world those walls are solid (by using collision detection), because players should not be able to walk through walls. However, in reality those walls simply do not exist. This means that a player could continuously (and deliberately) walk against the wall in VR, while moving forward in the real world. Eventually, he or she can walk out of the scope of the cameras. Within the scope of this project it was safe to count on the good intentions of the people testing out the framework, however for a more commercial product this should be solved. There are a number of similar issues, like how to handle a player who dies in the game, which we will not discuss in this paper.

The last big challenge we faced was providing a solution in which an HMD, in our case an OR, can be used in a mobile way. Practically, we needed to find a manner in which the device receives power and video, while a player is walking around the game field. An initial approach involved connecting the OR with a mobile phone for video and with a battery pack for power. This would result in a very portable and light-weight solution. However, connecting the devices without introducing a lot of overhead turned out to be technologically impossible. Due to new developments and products, like the Samsung Gear VR [18], this is something to keep in mind for future research. After dismissing other options like the usage of minicomputers (such as the Intel NUC and the Raspberry Pi), we concluded that it is currently almost impossible to use an HMD without it being connected to a regular computer, given the boundaries of our project. This is why we decided to equip the players

with a backpack containing a laptop, as can be seen in Figure 5. We can run a game client on this machine, that sends output to the connected OR. Since most modern laptops are not heavy, the weight of carrying the backpack should not be a big issue. This can not be said of the cooling of laptop and the air circulation, especially because of the intensive use of the graphical processing capabilities. To avoid overheating, we simply suggest to not use an entirely closed backpack.

Finally, two practical notes about our experimental setup: first of all, the mobile phones used for tracking should be placed in a stable manner around the field at a suited height to capture the players. We decided to use selfie sticks to achieve this, since those products are specifically made to hold any smartphone in a steady way. Moreover, they are quite cheap and are collapsible, making them exactly what we need for our mobile setup. We will strap them to a pole, tripod or even a garden chair to hold them upright. An example of such a setup can be seen in Figure 6. Secondly, we also need to make sure the different devices are connected to each other. Given the mobile and local nature of our project, our first idea was to use the mobile ad-hoc mode of 802.11. However, the Android operation system does not natively support connections to ad-hoc networks. We decided to use a battery powered wireless Wi-Fi access point instead.

V. Results

In this section we will evaluate the proposed framework. First, we will look at the performance of our tracking application, because this is the most critical part of our framework in terms of operating speed. Afterwards we will discuss our experiences with the setup and the game. The 2014 version of the Moto G smartphone of Motorola [19] is used in the test setup. This device has a screen resolution of 720 by 1280 pixels and can record video in this format at 30 FPS. This means that there will be an accuracy of 1.28 pixels per centimetre for a field side of 10 m. The angle of view of this device amounts to 65.1 degree, this means that the device needs to be placed at a distance of 7.83 m to capture the entire field size.

Device	*Run*	*Avg FPS without tracking*	*Avg FPS with tracking*
Moto G (720 x 1280 px)	Run 1	13.44	2.80
	Run 2	15.24	4.27
	Run 3	15.21	4.32
Samsung G S5 (1080x1920 px)	Run 1	17.88	5.94
	Run 2	28.73	6.43
	Run 3	26.93	5.69
Samsung G S5 (720x1080 px)	Run 1	28.96	10.46
	Run 2	16.27	8.91
	Run 3	16.50	8.03

TABLE III: *Average FPS values*

The results in Table III clearly indicate a large drop in the FPS when the actual object tracking occurs. This was expected since the object recognition takes more resources and time than simply showing the captured frames on the screen. In general the application is able to send by approximation four locations per second (on the Moto G phones we used in the setup). This is very low compared with the default 60 FPS that modern games are realizing. However, it is quite acceptable given the context of this framework, especially taking into account the observation that a user will move around very cautiously in a VR world. To get a broader view on the performance we also ran the application a few times at different resolutions on another device, the Samsung Galaxy S5, which is more powerful and expensive. The results of the six tests indicate the same general behaviour as before, but a better performance can be observed. In this case the more powerful hardware increases the frame rate by roughly a factor of two. More general performance results of image processing applications on different mobile devices can be found in the work of M. A. Hudelist et al. [20].

The current implementation of the tracking application is completely written in Java and relies on the java interface of the OpenCV library. This is the recommended way of using OpenCV on Android, according to the documentation [21]. Since the library is written in C++, this means that every time we call a Java function, a Java Native Interface (JNI) call occurs internally to access the corresponding C++ method. Those calls are in general slower than regular methods calls [22]. Because in our software per frame we need to call multiple functions provided by the OpenCV Java interface, the overhead of the JNI calls becomes relevant. We tried reimplementing the tracking part completely in C++, making sure that only one JNI call would happen per frame. Unexpectedly this resulted in an significant lower performance. This can be explained by the fact that the Java interface of OpenCV relies on asynchronous initialization with an externally installed version of the library on the system [21]. This version of the library is targeted specifically to mobile devices and includes NEON support. NEON is an additional instruction set for ARM architectures like those typically used in mobile devices, that allows faster matrix, vector and floating point operations [23]. However, when we rewrote a part of the implementation in C++, we needed to link this against the OpenCV library that can be freely downloaded from the website. This version can be used for ARMv7 platforms, but does not include this additional NEON support. This explains why we were unable to further increase the performance of the tracking application.

To accurately track the two players and calculate location estimates, the smartphones need to be placed on specific positions, based on the size of the game terrain and the angle of view of the mobile devices themselves. When developing the framework we did not give this too much thought and assumed it to be possible. However, it turned out during testing that it was actually not straightforward to place the different components at the exact positions: first of all, we need to make sure that the game field is measured out and delimited correctly. While measuring out one field side itself was doable, it was harder to make sure that the entire field was a rectangle with rectangular corners. Furthermore, the selfie sticks holding the smarthpones should be placed in such a way that the camera lens of the mobile device is at the middle of the corresponding field side. The mobile devices should also be placed at the correct distance of and parallel with the border of the field. From this enumeration it is clear that the setup is quite fragile and that it is very likely to introduce small errors.

We also stated that it was necessary to choose the right color of safety jacket, depending on the environment, to allow the Camshift algorithm to work correctly. In practice this was not always convenient, especially in an outdoor environment where the background consisted of multiple colors. For instance, we came across the situation where there was once again some flora in the setting, however this time there were both brown and green plants. This caused both the orange

and yellow safety jacket to be inadequate, since both orange and yellow are close to respectively brown and green in the color spectrum. Not only the background made it hard to correctly track the players in an outdoor scenario, but also the fluctuations of the sunlight. The reason for this is that everything looks brighter when the sun was not (partially) blocked by clouds. This means the difference between colors becomes smaller, making it harder for the color based tracking algorithm to recognize the player. Note that both of these problems were not encountered in an indoor scenario.

One of the key concerns was whether the framework would be responsive enough to avoid uncomfortable latency towards the users. In other words, whether the four location updates on average per second would be sufficient and how big the influence of the network delay would be. When we ran our application indoors the result was as good as we had hoped. While a user could still notice a fraction of delay, it did not interrupt the players' movement and the virtual world felt quite reactive. In an outdoor environment this was much less the case. The images received by a player were more shaky and often it looked like the screen jumped from one location to another. A reason for this could be the longer distances between different components and the access point. We believe that additional tests and effort are required to get a better view on the behaviour of the framework outdoors.

While the tracking and the location updates work in an acceptable manner indoor, the rotation actions are less comfortable. The reason for this is the difference between the rotation of the head and the body: if a player moves his head in a certain direction, this movement is registered by the sensors in the OR and instantly the same rotation takes place in the virtual world. However, this is not the case when a player turns his body completely, for example, to walk onto a path in an other direction in the maze. To see a response in the VR the player needs to manually rotate his alter ego by using the scrolling wheel of the mouse that he or she is holding.

VI. Conclusion

In this work we have presented a mobile framework to track the movements of users and translate them into corresponding actions in a virtual reality environment. We implemented a multiplayer first person shooter game as proof of concept. We identified and solved three main challenges: the tracking of an object or a person, the development of a multiplayer game and the use of an HMD in a mobile context. For the first challenge we looked at the currently available wireless technologies that can be used for object localisation. We were especially interested in BLE We conducted some experiments to obtain an indication of the accuracy of localization when using BLE and decided afterwards not to use this technology. Instead we opted to use image processing and the OpenCV Library with the Camshift algorithm. Next, we discussed briefly how we implemented the multiplayer game and described how we solved the issue of a portable Oculus Rift by equipping a player with a backpack containing a laptop. To conclude, we evaluated the proposed framework in terms of performance, accuracy and user experience. We showed that a frame rate of approximately 4 FPS was achieved and that this result could be increased by using more powerful hardware. Furthermore, we demonstrated that the general ideas behind our setup work in practice. Future work might include the use of context management techniques, that could be based on memorizing the game context, or the use of cloud offloading methods.

VII. References

[1] S. Marks, J. E. Estevez, and A. M. Connor, "Towards the holodeck: fully immersive virtual reality visualisation of scientific and engineering data," *ACM*, 2014.

[2] Computerphile. (2014) Real life holodeck with an oculus rift. [Online]. Available: https://www.youtube.com/watch?v=7ZPs7knvs7M

[3] S. Baulch, N. Fuchs, and B. Walker, "Oculuskinect: Real time augmented reality visual assistance system," 2013, a student research project, https://github.com/bpwalker/OculusKinect.

[4] W. Woodard and S. Sukittanon, "Interactive virtual building walkthrough using oculus rift and microsoft kinect," University of Tennessee, Martin Department of Engineering, Tech. Rep., Feb. 2013.

[5] Virtuix omni official product website. [Online]. Available: https://www.virtuix.com

[6] D. Stojanović and N. Stojanović, "Indoor localization and tracking: Methods, technologies and research challenges," *Facta Universitatis, Series: Automatic Control and Robotics*, vol. 13, no. 1, pp. 57–72, 2014.

[7] H. Liu, H. Darabi, P. Banerjee, and J. Liu, "Survey of wireless indoor positioning techniques and systems," *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, vol. 37, no. 6, pp. 1067–1080, Nov 2007.

[8] C. Gomez, J. Oller, and J. Paradells, "Overview and evaluation of bluetooth low energy: An emerging low-power wireless technology," *Sensors*, vol. 12, no. 9, pp. 11 734–11 753, 2012.

[9] Sensortag. texas instrument. official product website. [Online]. Available: http://www.ti.com/ww/en/wireless_connectivity/sensortag2015/index.html?INTC=SensorTag&HQS=sensortag

[10] E.-E.-L. Lau, B.-G. Lee, S.-C. Lee, and W.-Y. Chung, "Enhanced rssi-based high accuracy real-time user location tracking system for indoor and outdoor environments," *International Journal on Smart Sensing and Intelligent systems*, vol. 1, no. 2, pp. 534–548, 2008.

[11] R. Faragher and R. Harle, "An analysis of the accuracy of bluetooth low energy for indoor positioning applications," in *Proceedings of the 27th International Technical Meeting of The Satellite Division of the Institute of Navigation (ION GNSS+ 2014)*, September 2014, pp. 201–210.

[12] X. Zhao, Z. Xiao, A. Markham, N. Trigoni, and Y. Ren, "Does btle measure up against wifi? a comparison of indoor location performance," in *European Wireless 2014; 20th European Wireless Conference; Proceedings of*. VDE, 2014, pp. 1–6.

[13] S. Brahmbhatt, *Practical OpenCV*. Apress, 2013.

[14] G. R. Bradski, "Computer vision face tracking for use in a perceptual user interface," 1998.

[15] D. Exner, E. Bruns, D. Kurz, A. Grundhofer, and O. Bimber, "Fast and robust camshift tracking," in *Computer Vision and Pattern Recognition Workshops (CVPRW), 2010 IEEE Computer Society Conference on*, June 2010, pp. 9–16.

[16] jmonkeyengine. official website. [Online]. Available: http://jmonkeyengine.org/

[17] Spidermonkey, jmonkeyengine wiki. [Online]. Available: http://wiki.jmonkeyengine.org/doku.php/jme3:advanced:networking

[18] Samsung gear vr. official product page. [Online]. Available: http://www.samsung.com/global/microsite/gearvr/gearvr_features.html

[19] Motorola moto g 2nd generation. official product page. [Online]. Available: https://www.motorola.com/us/smartphones/moto-g-2nd-gen/moto-g-2nd-gen.html

[20] M. A. Hudelist, C. Cobârzan, and K. Schoeffmann, "Opencv performance measurements on mobile devices," in *Proceedings of International Conference on Multimedia Retrieval*. ACM, 2014, pp. 479–483.

[21] Android development with opencv. opencv wiki. [Online]. Available: http://docs.opencv.org/doc/tutorials/introduction/android_binary_package/dev_with_OCV_on_Android.html

[22] S. Liang, *The Java Native Interface: Programmer's Guide and Specification*. Addison-Wesley Professional, 1999.

[23] Neon. arm. official website. [Online]. Available: http://www.arm.com/products/processors/technologies/neon.php