information

MDPI

*Article*

# Cloudification of Virtual Reality Gliding Simulation Game

**Rytis Buzys [1], Rytis Maskeliūnas [1], Robertas Damaševičius [2,*], Tatjana Sidekerskienė [3], Marcin Woźniak [4] and Wei Wei [5]**

1 Department of Multimedia Engineering, Kaunas University of Technology, 51368 Kaunas, Lithuania; rytis.buzys@gmail.com (R.B.); rytis.maskeliunas@ktu.lt (R.M.)
2 Department of Software Engineering, Kaunas University of Technology, 51368 Kaunas, Lithuania
3 Department of Applied Mathematics, Kaunas University of Technology, 51368 Kaunas, Lithuania; tatjana.sidekerskiene@ktu.lt
4 Institute of Mathematics, Silesian University of Technology, 44-100 Gliwice, Poland; marcin.wozniak@polsl.pl
5 Xi'an University of Technology, Xi'an 710048, China; weiwei@xaut.edu.cn
* Correspondence: robertas.damasevicius@ktu.lt; Tel.: +370-609-43772

Received: 6 November 2018; Accepted: 20 November 2018; Published: 22 November 2018


**Abstract:** Cloud gaming provides cloud computing-based game as a service. In this paper we describe the development of a virtual reality base gliding game as a proof-of-concept. In the cloud, a cloud gaming platform is hosted on cloud servers with two principal components: game logic engaged in the implementation of game mechanics and game interactions, and video renderer that generates the game frames in real-time. The virtual gliding game was realized in the Unity gaming engine. To ensure smooth playability, and access for remote players, the computationally-intensive parts of the game were offloaded to a physically remote cloud server. To analyze the efficiency of the client-cloud interaction, three cloud servers were setup. The results of cloudification were evaluated by measuring and comparing computation offloading performance, network traffic, the probability of service drop, perceptual quality and video quality.

**Keywords:** game-as-a-service; cloud computing; virtual reality; cloud gaming

## 1. Introduction

Cloud computing delivers an adaptable and practical approach for hosting and providing services over the Internet to remote end users on a large-scale [1]. Cloud can be seen as a distributed computing system composed of a stack of interconnected and virtualized machines that are dynamically managed as affiliated resource(s) adhering to Service Level Agreements (SLA) between the cloud service provider (CSP) and cloud users [2]. Different models of cloud services exist, such as Software-as-a-Service (SaaS), Platform-as-a-Service (PaaS), Infrastructure-as-a-Service (IaaS), and Backend-as-a-Service (BaaS). The latter specifically is a cloud computing service model that allows the game programmers to link their custom apps to cloud services by using Application Programming Interfaces (API) and Software Developer Kits (SDK).

Cloud gaming [3] is related to a gaming BaaS that employs cloud computing services to achieve better gaming performance. First introduced in 2009 by Ross [4], cloud gaming enables workload distribution among numerous cloud servers and game clients. The main advantage of game clouding is in efficient exploitation of cloud resources to store and manage components of games, while decreasing workload at multiple and heterogeneous gamers' devices and improving overall efficiency characteristics [5]. The combination of gamification and BaaS ushered in Game Backend as a Service

(GBaaS) and Game as a Service (GaaS) [6]. Mobile backend-as-a-service (MBaaS) allows game programmers to connect the backend of their software systems to the cloud, as well as to enable integration with social networks [7]. A player can interface with the cloud-hosted application using a thin client model, which controls the displaying of the game's video frames from the cloud server and for responding to the player's commands and responding back to the cloud [8]. These cloud services aim to provide powerful real-time tools specifically dedicated for gaming, with applications emerging in other areas, such as medicine, marketing and education as well.

The characteristics of mobile devices recently were rising rapidly in terms of computational abilities, memory, features, and number of apps. However, mobile apps are still relatively inferior by bounded battery life, lack of stability of wireless communications, and prone-to-interruptions connections and constrained bandwidth, and lack of local storage and computing power. By relocating (off-loading) local device tasks to the cloud servers, efficient and ambient cloud services can be accomplished, thus guaranteeing cloud gaming ecosystems with a large number of mobile game players and mobile devices working remotely, but in coordination [9]. Since mobile devices have only limited memory and computational resources, information stored on local devices is synchronized with shared data centers, therefore the system's availability is a demanding factor.

The problems faced by cloud gaming include the server provisioning problem [10], with the goal of slashing server running and data storage costs, as well as measuring and reducing the latency [11]. There is also some work focusing on graphics rendering and video encoding for network bandwidth reduction in the video transmission [12]. The cloud resource allocation problems in order to minimize the interaction latency among interactive clients were researched in Reference [13]. A promising approach utilized by many authors to mitigate cloud-related problems can by implemented by computational offloading, thereby the remote computers are enrolled to execute the computationally costly parts of mobile games, while the mobile clients only serve as front-ends to receive and visualize the results. Computation offloading allows to liberate local CPU resources on the mobile device and also to maintain lower consumption of battery power. However, the implementation of offloading requires more complex design of app architecture, as well as becomes vulnerable to network delays, which, in the worst case, can slows down the game, as well as may require more complex maintenance actions to support the offloading-based game infrastructure [14].

When evaluating cloud games, Quality-of-Experience (QoE) metrics are used that represent a measure of game playability, and are closely related to network Quality-of-Service (QoS) parameters, such as packet loss, jitter, and latency are known to influence the user experience, system responsiveness or game playability [15].

The latter is especially relevant to virtual reality (VR) simulators and trainers, which create complex interactive 3D worlds, but demand high computing resources to simulate realistic graphics, while the lack of smooth rendering and delays may lead to simulation sickness [16]. The evolution of VR technologies increases the ability to move costly and dangerous activities, such as extreme sports or medical training into the virtual world without compromising subject's health or risking losing high-priced equipment. Simulation games that try to portray one or another of the real-world activities must be based on certain laws of the real world.

Here we describe the development of cloud-based gliding simulator based on real-world physics laws with a real-time data management system that supports multiplayer mode. In order to ensure a smooth game on the network, it is necessary to optimize the amount and stability of the data transmitted, thus minimizing the consequences of delay in data transmission and avoiding the effects of simulation sickness in a fast-moving game.

## 2. Methodology

### *2.1. Game Story*

The idea of a game is racing with a gliding suit. The goal of the game is to reach the finish and do it faster than the opponents or improve the personal track record. In order to make the game as realistic as possible, it is advisable to use VR glasses, but the game can be played without them.

The game can be played with by a single player or with other players, racing against each other. With the help of the server, players' positions and other information are synchronized so that players can see each other. Track records, and other player information, are stored in the database. Joining the game is possible using the Facebook social network, while the results can be shared on the social network as well.

A glider is a complex physical body, so it would be too complicated to calculate the exact forces of this body during gliding. The purpose of the landing suit is to give the person as much aerodynamic as possible in order to reach the maximum distance traveled the way down the distance traveled. Similar gliding characteristics can be obtained from a physical body called an airfoil. To achieve the realistic gliding experience requires performing complex computations based on multiple factors, which account for example for air resistance, therefore computation offloading to the cloud is necessary.

### *2.2. Cloud Gaming Model*

We have considered two alternative architectures for implementation of our cloud game: Thick-client, which provides rich functionality without the need of server functionality, and thin-client, which strongly depends upon server resources. Considering the requirements of our game, which has to provide VR experience using a mobile device (a smartphone or a tablet), and at the same time needs to provide a realistic simulation of gliding, which introduces a significant computational burden on a device, we have selected the thin-client approach. The cloud gaming approach we have adopted is based on a classical thin-client model (see Figure 1) [17], which offloads the most computationally intensive operations related to video content generation to the cloud server [8]. Such a model has been used for cloud gaming before [18–20].

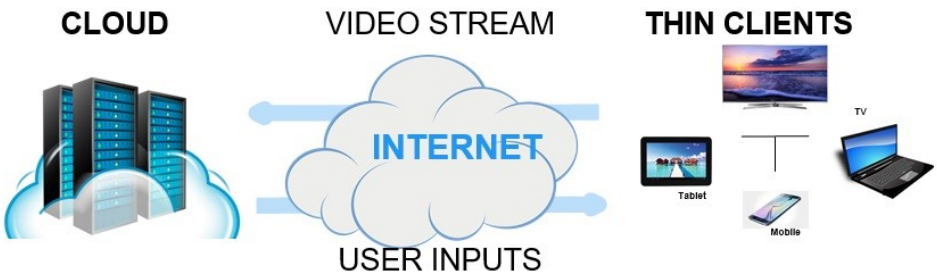


**Figure 1.** Cloud gaming model.

A cloud gaming platform runs (i) game logic that is responsible for game mechanics and game interactions received from game server API from game instances, and (ii) Graphics Processing Unit (GPU) renderer that creates game-world scenes in real-time. The generated video frames, after encoding and compression, are streamed to mobile devices, which handles decoding and output to a game player.

The model is detailed in Figure 2. The purpose of the cloud gaming server is to perform physics computations and transfer player information to other players participating in the same race. The created server works on the client-server principle. The players do not communicate directly with each other, they transmit information to the server, and the server sends information to the other players for processing.

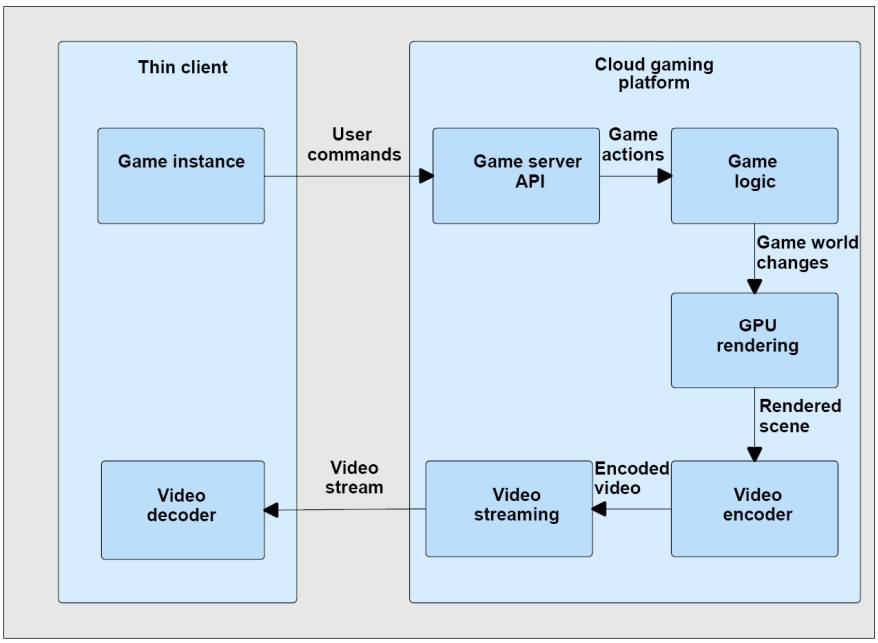


**Figure 2.** Cloud based game architecture (adapted from Reference [8]).

The server logic has three layers: Data receiving (thin client interaction) and sending (video streaming); the abstraction of data processing and their assignment to specific clients (graphics rendering), and game logic (processing of changes in a virtual game world). The game logic layer takes place with specific game-related operations. This layer stores information about the current state of play and makes decisions during the game. This layer also processes the information sent by the players, and changes are made to the game status and sends signals to other players.

### *2.3. Game Virtualization*

We assume that there can be multiple geo-distributed remote players connecting to the game platform. The game instances are executed on globally distributed data centers and then, the game video frames are streamed to the users. To ensure smooth gaming and small network delay, virtualization of cloud resources from different remote locations, who are network-connected and require thin clients, e.g., smart-phones or tablets.

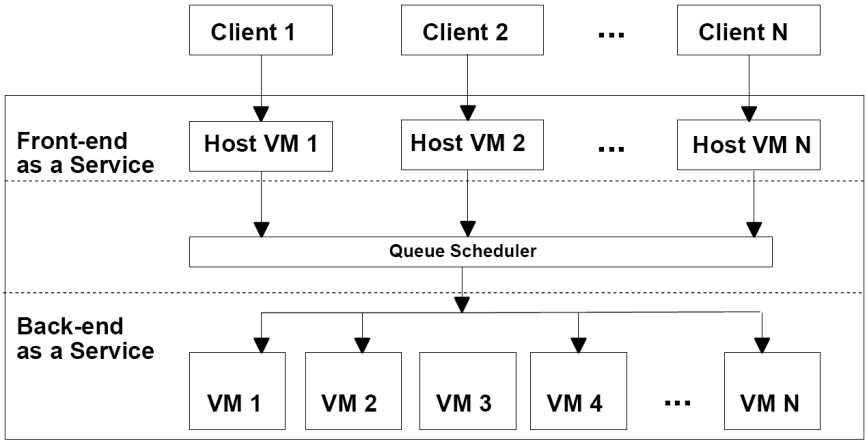


**Figure 3.** Virtualization of cloud resources in a physical machine (PM). VM, virtual machines.

Virtualization abstracts computer hardware to an operating system (OS) known as virtual machines (VM) which includes the operating system, the application, and the configurations of

a physical server [21]. Virtualization enables a physical machine (PM) to be logically segmented into numerous VMs executing scheduled tasks in their own memory space. This dedication of resources to match their actual task requirements allows avoiding the resource under-utilization. For virtualization of cloud resources, we have adopted a model proposed in Reference [22] as follows (see Figure 3). Sending client requests to the cloud gaming platform and executing game actions among different cloud servers are represented. The game requests, first, are submitted to a task queue scheduler, which handles the requests on the first-in, first-out basis (FIFO).

### *2.4. Service Resource Queuing*

For game scheduling, we use the GI/M/1/K-type queueing model described in detail in Reference [23] and recapitulated briefly below. We assume that the arriving game client requests for service are handled following an exponentially distributed service time with mean $\mu^{-1}$ by the First In First Out (FIFO) service. The number of system requests is constrained by a constant value K, which represents buffer capacity. The process begins with the advent of the first request, while the server starts processing when the number of requests stored in the FIFO queue reaches N, where $1 \leq N \leq K$. When the server process is idle it enters hibernation and wakes up again when it finds N requests accumulated in the queue. Each of incoming requests can resign with probability $1 - \beta$, or be accepted into the queue with probability $0 < \beta \leq 1$. If incoming request discovers that the queue is full, it leaves the system without service (i.e., the request is dropped). Let us denote the number of requests size by n, defined as the number of game requests from different game clients, here n is a geometrically distributed random variable with a parameter p in the case of cloud gaming.

Following Reference [20], the probability $P_r$ of having the number of requests i in the requests queue is calculated as:

$$P_r(n = i) = (1 - p_s)(1 - p)^{i-1}p + p_S\delta, \tag{1}$$

here $p_S$ is the probability of getting a single game client request, $\delta$ is the Dirac delta function, and p is the parameter of the geometric distribution.

This type of modeling provides a dedicated strategy to ensure a better Quality of Service (QoS) and enable better playability in game management.

### *2.5. Game Physics: Gliding Simulation*

By applying the laws of physics to virtual gliding simulation, the aim is to achieve the most realistic gliding simulations. To make the gliding effect realistic, the player sees the image with the glider's eyes. Using Virtual Reality (VR) goggles and sound effects can have a fairly realistic effect. Using VR goggles, gliding control is performed using head movements, and while playing without glasses of virtual reality, it is possible to control a glider using a joystick. All force calculations are performed in the local gliding coordinate system, that is, the coordinates begin with the position of the gliding suit, and the coordinate axes coincide with the gliding suit axes.

The glider simulation is based on the simplified model adopted from Reference [24]. All airborne objects have forces that can be divided into three groups: Lifting force L, drag force D, and the gravity force G. In Figure 4, the lifting vector is always perpendicular to the motion vector, and the resistance vector is always parallel to the motion vector, but directed to the opposite direction. The lifting force is calculated as follows:

$$L = C_l \times \frac{\rho \times v^2}{2} \times A, \tag{2}$$

here L—lifting force (N), $C_l$—lifting factor, $\rho$—air density ($kg/m^3$); v—velocity (m/s); A—area ($m^2$).

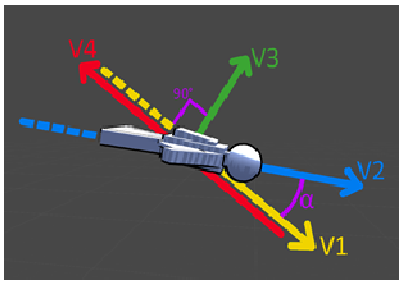


**Figure 4.** Vectors of forces of the glider: V1 is the motion vector, V2 is the glider direction vector, V3 is the lifting force vector, V4 is the resistance force vector, and α is the attack angle between the direction of the glider and the direction of movement.

The landing suit has three main parts-two arm fenders and one wing connecting both legs, but the human body also plays the role of a wing. The arms and legs of the wings help rotate, rise and drop down. As the gyroscope of the mobile device will be used for control, the composition of the gliding suit has been simplified to one large wing, the shape of which resembles a square, so its area is calculated by the formula:

$$A = a \times b, \tag{3}$$

here a is the length of the gliding suit (m), and b is the width of the gliding suit (m).

The lifting factor ($C_l$) for a symmetrical wing (airfoil) is calculated as follows:

$$C_l = C_{l0} + 2 \times \pi \times \alpha, \tag{4}$$

here $C_{l0}$ is the lifting factor when the angle of attack is zero, and α is the angle of attack. Using this formula, we can obtain realistic results only at an effective angle of attack. The effective angle of attack of the gliding suit is approximately from 0° to 48° degrees. A typical relationship between lifting factor and angle of attack is illustrated in Figure 5.

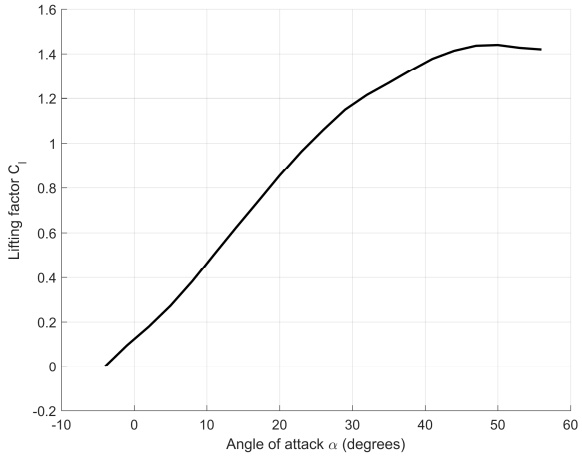


**Figure 5.** Dependence of lifting factor on the angle of attack for glider simulation.

The air resistance force is calculated as follows:

$$D = C_d \times \frac{\rho \times v^2}{2} \times A, \tag{5}$$

here D is the resistance force (N), $C_d$ is the resistance (drag) coefficient, ρ is air density, v is velocity (m/s), and A is the surface area of the gliding suit ($m^2$).

The resistance coefficient is calculated with a lifting factor:

$$C_d = C_{d0} + \frac{{C_l}^2}{\pi \times A_R \times e}, \tag{6}$$

here $C_d$ is the resistance coefficient, $C_{d0}$ is the coefficient of resistance at zero load factor, $C_l$ is the lifting factor, $A_R$ is the ratio of the wingspan and wing area, e is the coefficient of efficiency (constant), which is approximately equal to 0.7.

The wing ratio $A_R$ is calculated according to the formula:

$$A_R = \frac{s^2}{A}, \tag{7}$$

here s is the wing length (m), and A is the wing area ($m^2$).

To model air resistance and wind force, we used U.S. Standard 1976 atmosphere model, which provides air density and pressure up to 85 km height. To model wind speed profile at a different height, we adopted the wind model described in Reference [25].

To obtain the glider simulation equations, a mathematical model defined by the Newton laws of motion with six Degrees of Freedom (DOF) as follows:

$$\dot{X} = U(\cos\phi\cos\theta) + V(\sin\phi\sin\theta\cos\psi - \cos\phi\sin\psi) + W(\cos\phi\sin\theta\cos\psi + \sin\phi\sin\psi), \tag{8}$$

$$\dot{Y} = U(\sin\phi\cos\theta) + V(\sin\phi\sin\theta\sin\psi + \cos\phi\cos\psi) + W(\cos\phi\sin\theta\sin\psi + \sin\phi\cos\psi), \tag{9}$$

$$\dot{Z} = -U(\sin\theta) + V(\sin\phi\cos\psi) + W(\cos\phi\cos\theta), \tag{10}$$

here $\dot{X}$, $\dot{Y}$, $\dot{Z}$ are the change of the aerodynamic forces X, Y, Z along the body x, y, z-axis, respectively, U, V, W are the axial speeds (m/s), and $\phi$, $\theta$, $\psi$ are the Euler angles, which represent the rotation around the x, y, z-axis, respectively.

Axial acceleration is expressed as follows:

$$\dot{U} = R \times V - Q \times W - g \times \sin\theta + X/m, \tag{11}$$

$$\dot{V} = P \times W - R \times U + g \times \cos\theta \times \sin\phi + Y/m, \tag{12}$$

$$\dot{W} = Q \times U - P \times V + g \times \cos\theta \times \cos\phi + Z/m, \tag{13}$$

here $\dot{U}$, $\dot{V}$, $\dot{W}$ are the change in acceleration, P, Q, R are the angular speeds (rad/s), g is the gravitational acceleration constant, and m is the glider weight (kg).

Solving Equations (8)–(13) yields the derivatives of the Euler angle's in Equations (14)–(16), which describes the orientation of the glider on the fixed coordinate system, as follows:

$$\dot{\phi} = P + \tan\phi(Q\sin\phi + R\cos\phi), \tag{14}$$

$$\dot{\theta} = Q\cos\phi - R\sin\phi, \tag{15}$$

$$\dot{\psi} = (Q\sin\phi + R\cos\phi)/\cos\theta, \tag{16}$$

here $\dot{\phi}, \dot{\theta}, \dot{\psi}$ are the change in the Euler angles.

### *2.6. Update of Glider Position*

To ensure smooth graphics, the glider position has to be updated. To reduce network traffic and to decrease the load on a mobile device, the position is only updated if the calculated glider position changes more than a predefined threshold. Consider position update for a game client A. Let $t_{last}$ denotes the timestamp of the previous cloud server update for game client A, with $l_{last}$ and $v_{last}$ are the

position and speed of a glider in A, accordingly. Let d be the network delay of the connection between the server and the game client. Game client A continues updating the server about the glider's position regularly. The server follows the difference of the position between the glider's state at the cloud server and its state at the game clients. A new request is initiated at time t when the cloud server receives a new game client update on A with position $l_{new}$ speed, $v_{new}$ Let $\delta$ be an updating threshold. We can describe the position update equation as recommended in Reference [26] as follows:

$$\parallel l_{new} + v_{new}d, \quad l_{last} + v_{last}(t - t_{last} + d) \parallel > \delta. \tag{17}$$

### *2.7. Evaluation of Cloud Game Performance*

Quantifiable performance measures help to evaluate the efficiency of software architectures and systems. Here we considered the evaluation of most important game characteristics entertaining the Quality of Experience (QoE) requirement [27] as follows. Game smoothness is the ability to play the game without interruptions, while game responsiveness is the updating delay updates in responding to the player input, represented in real-world time (in ms) [14].

To evaluate their characteristics, we calculate the performance of computational offloading (in ms) and compare it with the performance of the non-offloaded version of the game. We also measure both the incoming and the outgoing network traffic to check if the values are acceptable of network bandwidth limitations. We also evaluate the probability of service drop, which occurs then network delay exceeds a predefined value and game client sends a new request.

To evaluate game playability, we have adopted the playability model [28], which relates perceptual quality Q to game video bit rate R as follows:

$$Q(R) = \frac{\left(1 - e^{-\kappa(\frac{R}{R_{max}})^{0.55}}\right)}{(1 - e^{-\kappa})}, \tag{18}$$

here $\kappa$ is the parameter characterizing the type of video content, $2 \leq \kappa \leq 6$ (i.e., from motion-intensive to stationary videos), and $R_{max}$ is a network bandwidth assigned to a single client.

To ensure smooth playability, the quality of video stream is adapted dynamically with respect to detected network bandwidth and connection delay. The quality drop introduced by compression is evaluated using the Peak Signal to Noise (PSNR) metric, which is evaluated for each video frame separately:

$$PSNR = 10\log_{10}\left(\frac{255^2}{(1/MN)\sum_{i=1}^{M}\sum_{j=1}^{N}\left(x_{ij} - y_{ij}\right)^2}\right) \tag{19}$$

where M and N are the height and width of the input image, $x_{ij}$ and $y_{ij}$ denote the pixels of the original image generated at the game server, and the decoded image on game client's device, respectively.

## 3. Results

### *3.1. Implementation and Testing Environment*

The experiments used the developed game as the cloud gaming evaluation platform. The game has two modular elements: A server and a client. As a client, a PC with AMD FX-6300 6-core processor, 8 GB of RAM, a 1TB 7200 RPM hard drive, and AMD Radeon 4670 GPU set up in the SANTAKA valley building of Kaunas University of Technology (Kaunas, Lithuania). The network access is provided by LITNET ISP through a wired connection with a maximum connection speed of 866 Mb/s for download and 943 Mb/s for upload (ping time 1ms). Additionally, 30 workstations were used to host 30 clients. The 30 game clients implemented the simulation scenario of 6 teams of 5 players (30 players, in total),

simultaneously racing in the virtual environment, which is a realistic use-case for this type of game. The experiment consisted of three game sessions.

The game was realized in the Unity gaming engine. The client part of the game, executed on individual devices of the users, is responsible for rendering graphics and playing audio, which present the game state for the player, accepts VR control commands to control the players' in-game character, and connects with the cloud server to allow the player to interact and race with other players within the same shared game environment.

Each game client connects to the cloud server. The cloud server synchronizes and updates the game state (simulated glider world) between multiple connected users who interact within the same simulated environment. To analyze the performance of cloud platform, we have performed measurement and analysis of network traffic from the three cloud servers, which are located in Lithuania (Kaunas), Poland (Gliwice), and China (Xi'an). We performed the latency measurements between Kaunas (where the cloud hosted) and Gliwice and Xi'an, respectively, every hour from 8:00 a.m. to 5:00 p.m. The median delay between the game client and the game server in Kaunas is about 6 ms, Kaunas and Gliwice is about 21 ms, while they are 75 ms between Kaunas and Xi'an. The latency could be improved using more geo-distributed cloud servers set-up close to the game clients.

The gliding simulation was implemented using a physics engine from the Nvidia PhysX integrated with the Unity engine (Unity Technologies, Lithuania). For playing in the VR mode, we used HTC Vive© VR headsets (HTC, Xindian District, New Taipei City, China). The device was selected because of a high level of immersion achieved, due to the availability to use VR stick controls that players users to control the glider within the VR environment. The screenshots of the virtual glider game are presented in Figure 6, while a photo of a player playing the game is given in Figure 7. The example of simulated glider flight trajectory is presented in Figure 8.

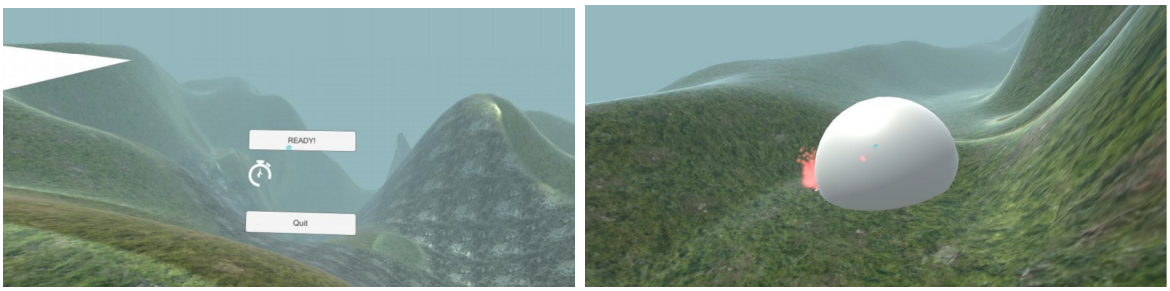


**Figure 6.** Snapshot of a game menu and game finish.

**Figure 7.** A photo of a player playing a virtual gliding simulator game (Source: VRLab, Kaunas University of Technology).

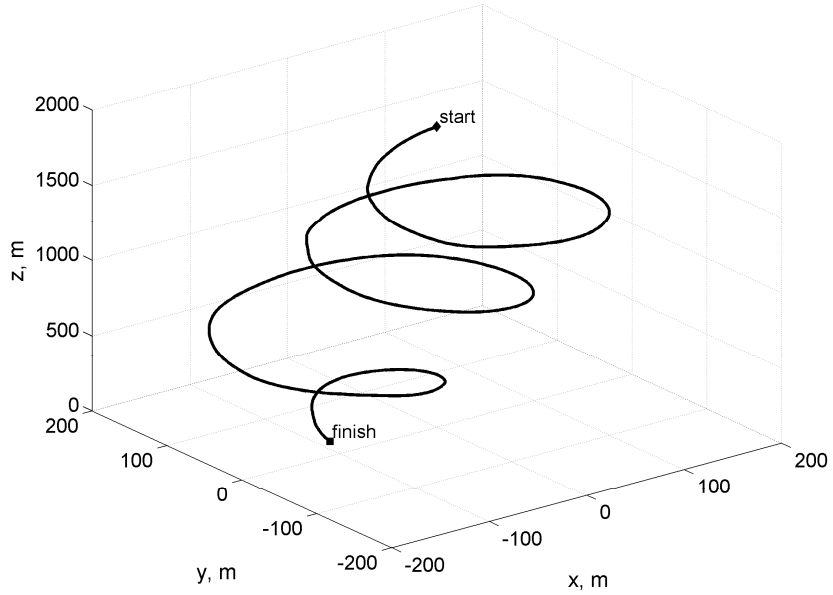


**Figure 8.** Snapshot of a game representing finish target.

To collect game evaluation metrics, we developed a Python script, which collects performance measures during game playing. The script was executed both on the cloud server and on game client sides. The results were processed and visualized using MATLAB (MathWorks Inc., Natick, MA, USA, 2017).

### *3.2. Evaluation of Computational Offloading*

Computational loads hinder and slow down. Our measurement results show that by enabling computational we can considerably improve the performance of the game (see Figure 9). The off-loadable parts consist of the gliding computation and video generation functions.

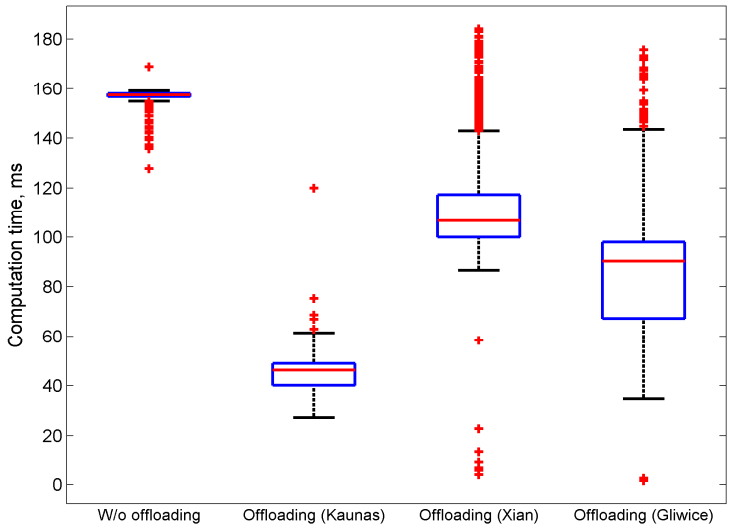


**Figure 9.** Performance comparison of computational offloading using local (Kaunas) and remote (Xian and Gliwice) cloud servers.

From the results presented we can see that the mean computation time remains below 120 ms while using all servers, and rarely spikes to more than 180 ms, while a latency of 1000 ms is an acceptable delay for real time strategy games [29], and latency below 200 ms is considered good for moderate-paced games [30]. Once the connection latencies exceed the corresponding thresholds, any further improvement would not deliver a noticeable improvement in the player's perceived playability [30].

### *3.3. Evaluation of Network Traffic*

We evaluated the amount of network traffic for each of the tested game clients (1 through 30), while their requests are sent and scheduled on the game server hosted on the cloud (i.e., 30 instances of game clients are interacting in the simulation). Figure 10 plots the incoming and outgoing network with respect to the number of simultaneously interacting game clients.

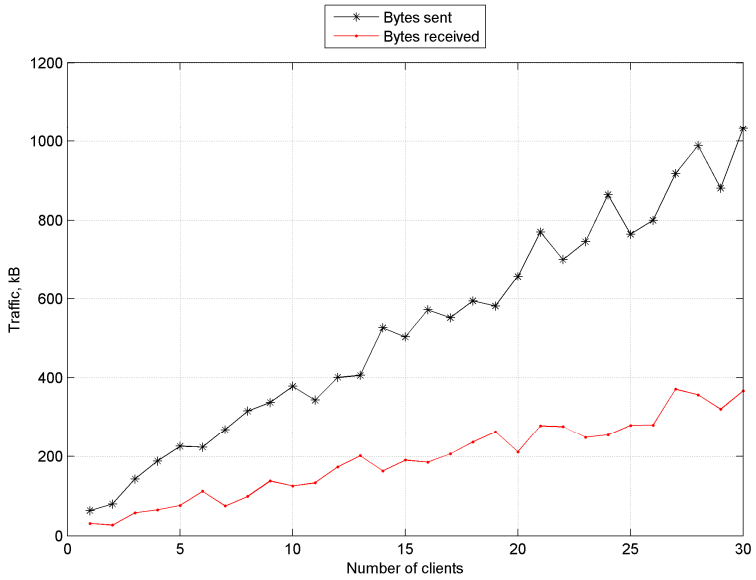


**Figure 10.** Network bytes sent to and received from a cloud server vs. the number of clients.

### *3.4. Evaluation of Scheduling*

We calculate the probability of service drops using the adopted service queuing model. The results are presented in Figure 11. The results show that the probability of service drop increases with the number of clients connected, and becomes larger than 0.5 if more than 20 remote clients are connected to the game cloud server. Note that it varies considerably because of the packet drop caused by the network congestion.

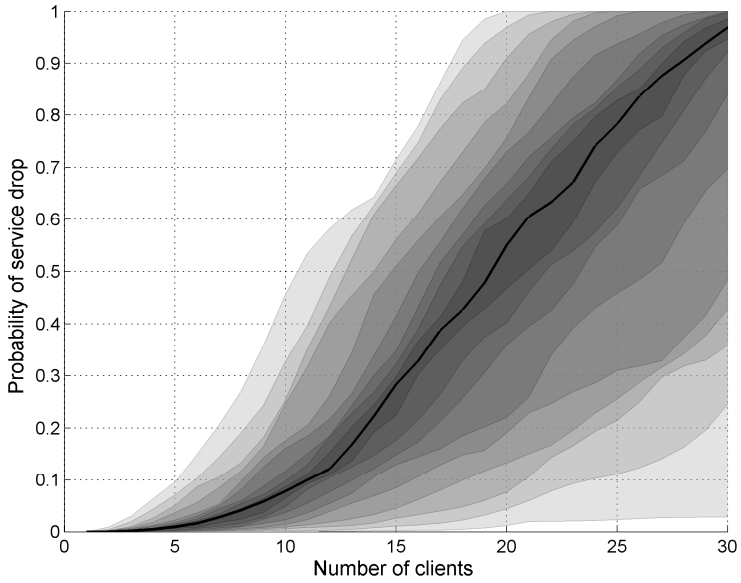


**Figure 11.** Probability of service drop with respect to the number of clients: Median value and deciles are shown.

### *3.5. Evaluation of Playablility*

In the evaluation of game's playability, we set up 256 kB/s as the maximum bandwidth of a Wi-Fi network. A gamer's visual system can tolerate a certain degree of video compression without noticeable visual degradation even for moderate-to high motion videos. However, the videos of our game can be characterized as slow motion. The observed results presented in Figure 12 (according to perceptual quality model) and Figure 13 (using PSNR) are in line with the ones presented in Reference [28].

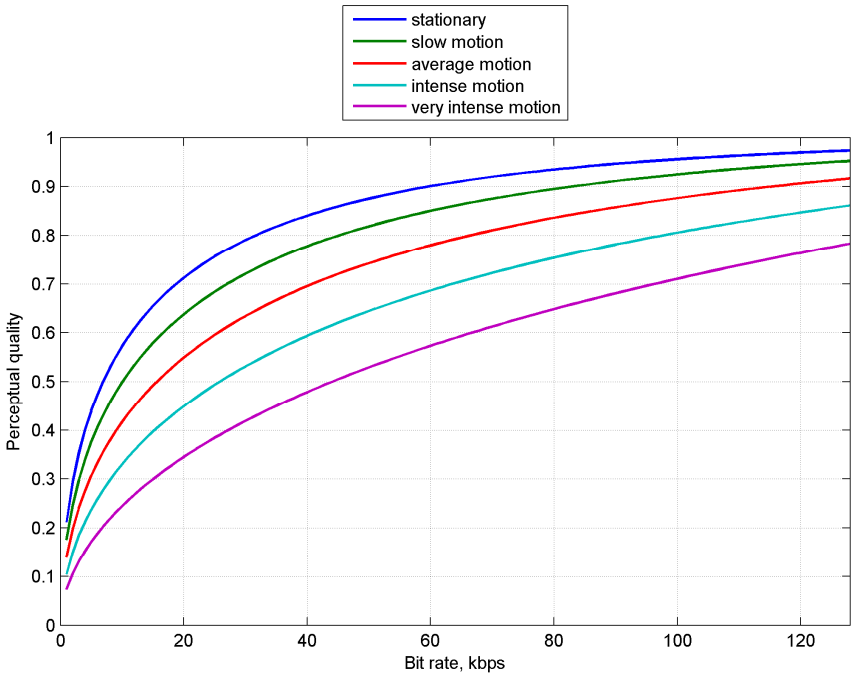


**Figure 12.** Evaluation of perceptual quality with different bit rates.

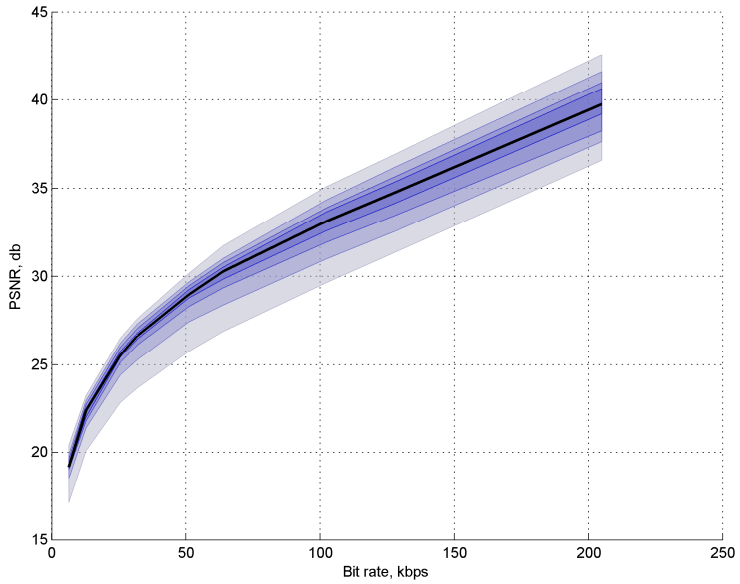


**Figure 13.** Video quality (Peak Signal to Noise—PSNR) vs video bit rate.

## 4. Discussion and Conclusions

Cloud gaming is evolving rapidly with many exciting possibilities promised, allowing to deliver advanced 3D graphics content to computationally less feasible mobile devices. The trend is especially relevant for the use of Virtual Reality (VR) games in mobile gaming, which is expected to grow in coming years. Cloud computing offers added value for VR games as it may supply physically remote graphics rendering resources, which reduce the load on mobile devices. Nethertheless, the technologies must shift towards new cloud gaming architectures, which ensure dynamic partitioning and offloading of computations based on the use of Artificial Intelligence (AI) techniques ensuring elasticity and scalability of gaming services [31,32].

We have developed a virtual reality-based glider simulation game, which allows teams of remote players to join and enjoy the simulation of gliding, while competing with each other. The game provides a real-world experience by using real-world physics laws to calculate realistic gliding trajectories. To improve game playability and ensure smoothness of graphics, we have offloaded complex physics computations to cloud infrastructure. As a result, the players can enjoy smooth game graphics even on the computationally weaker devices.

In order to analyze and evaluate the characteristics and effectiveness of cloud-client architecture and implementation, we considered various characteristics, such as computation offloading performance, network traffic, the probability of service drop, perceptual quality and video quality. All these metrics are important to the gaming experience. The results show that the players can still enjoy a good game experience while using Wi-Fi or 3G mobile internet connection. However, our experiments were only performed using quite stable Wi-Fi/ Local Area Network (LAN) connection for the game client. In reality, most players will likely use 3G, 4G, or even less stable public Wi-Fi networks, which can require additional measures to ensure the robustness of the game experience.

**Author Contributions:** Formal analysis, M.W.; Software, R.B.; Validation, W.W.; Visualization, T.S.; Writing—original draft, R.D.; Writing—review and editing, R.M.

**Funding:** This research received no external funding.

**Acknowledgments:** This work was supported by the National Key R&D Program of China under Grant 2018YFB0203901 and the Scientific Research Program Funded by the Shaanxi Provincial Education Department under Program 2013JK1139 and the China Postdoctoral Science Foundation under Grant 2013M542370 and the Specialized Research Fund for the Doctoral Program of Higher Education of China under Grant 20136118120010.)

**Conflicts of Interest:** The authors declare no conflict of interest.

## References


1. Deng, Y.; Li, Y.; Seet, R.; Tang, X.; Cai, W. The server allocation problem for session-based multiplayer cloud gaming. *IEEE Trans. Multime,* **2018**, *20*, 1233–1245. [CrossRef]
2. Buyya, R.; Yeo, C.S.; Venugopal, S.; Broberg, J.; Brandic, I. Cloud computing and emerging IT platforms: Vision, hype and reality for delivering computing as the 5th Utility. *Future Gener. Comp. Syst.* **2009**, *25*, 599–616. [CrossRef]
3. Cai, W.; Chi, Y.; Zhou, C.; Zhu, C.; Leung, V.C.M. UBC Gaming: Ubiquitous cloud gaming system. *IEEE Syst. J.* **2018**, *12*, 2483–2494. [CrossRef]
4. Ross, P.E. Cloud computing's killer app: Gaming. *IEEE Spectr.* **2009**, *46*, 14. [CrossRef]
5. Cai, W.; Shea, R.; Huang, C.-Y.; Chen, K.-T.; Liu, J.; Leung, V.C.M.; Hsu, C.-H. A Survey on Cloud Gaming: Future of Computer Games. *IEEE Access* **2016**, *4*, 7605–7620. [CrossRef]
6. Al-Rousan, N.M.; Cai, W.; Ji, H.; Leung, V.C.M. DCRA: Decentralized Cognitive Resource Allocation Model for Game as a Service. In Proceedings of the IEEE 7th International Conference on Cloud Computing Technology and Science (CloudCom), Vancouver, BC, Canada, 30 November–3 December 2015; pp. 218–225.
7. Costa, I.; Araujo, J.; Dantas, J.; Campos, E.; Silva, F.A.; Maciel, P.R.M. Availability Evaluation and Sensitivity Analysis of a Mobile Backend-as-a-service Platform. *Qual. Reliab. Eng. Int.* **2016**, *32*, 2191–2205. [CrossRef]
8. Shea, R.; Liu, J.; Ngai, E.; Cui, Y. Cloud gaming: Architecture and performance. *Netw. IEEE* **2013**, *27*, 16–21. [CrossRef]
9. Kim, H.; Kim, K.J. Optimized state update for mobile games in cloud networks. *Cluster Comput.* **2017**, 1–7. [CrossRef]
10. Li, Y.; Deng, Y.; Tang, X.; Cai, W.; Liu, X.; Wang, G. Cost-efficient server provisioning for cloud gaming. *ACM Trans. Multimed. Comput. Commun. Appl.* **2018**, *14*, 55. [CrossRef]
11. Choy, S.; Wong, B.; Simon, G.; Rosenberg, C. A hybrid edge-cloud architecture for reducing on-demand gaming latency. *Multimed. Syst.* **2014**, *20*, 503–519. [CrossRef]
12. Ahmadi, H.; Zad Tootaghaj, S.; Reza Hashemi, M.; Shirmohammadi, S. A game attention model for efficient bit rate allocation in cloud gaming. *Multimed. Syst.* **2014**, *20*, 485–501. [CrossRef]
13. Wang, H.; Shea, R.; Ma, X.; Wang, F.; Liu, J. On design and performance of cloud-based distributed interactive applications. *IEEE Comput. Soc.* **2014**, 37–46. [CrossRef]
14. Jiang, M.H.; Visser, O.W.; Prasetya, I.S.W.B.; Iosup, A. A mirroring architecture for sophisticated mobile games using computation-offloading. *Concurr. Comput. Pract. Exp.* **2018**, *30*, e4494. [CrossRef]
15. Mishra, D.; El Zarki, M.; Erbad, A.; Hsu, C.-H.; Venkatasubramanian, N. Clouds+Games: A multifaceted approach. *IEEE Int. Comput.* **2004**, *18*, 20–27. [CrossRef]
16. Lee, J.; Kim, M.; Kim, J. A Study on Immersion and VR Sickness in Walking Interaction for Immersive Virtual Reality Applications. *Symmetry* **2017**, *9*, 78.
17. Hwang, G. Supporting cloud computing in thin-client/server computing model. *ISPA* **2010**, 612–618. [CrossRef]
18. Cuervo, E.; Wolman, A.; Cox, L.P.; Lebeck, K.; Razeen, A.; Saroiu, S.; Musuvathi, M. Kahawai: High-Quality Mobile Gaming Using GPU Offload. In Proceedings of the 13th Annual International Conference on Mobile Systems, Applications, and Services-MobiSys, Florence, Italy, Octorber–December 2015; pp. 121–135.
19. Amiri, M.; Al Osman, H.; Shirmohammadi, S.; Abdallah, M. Toward delay-efficient game-aware data centers for cloud gaming. *ACM Trans. Multimed. Comput. Commun. Appl.* **2016**, *9*, 12. [CrossRef]
20. Basiri, M.; Rasoolzadegan, A. Delay-aware resource provisioning for cost-efficient cloud gaming. *IEEE Trans. Circuits Syst. Video Technol.* **2018**, *28*, 972–983. [CrossRef]
21. Jain, R.; Paul, S. Network virtualization and software defined networking for cloud computing: A survey. *IEEE Commun. Mag.* **2013**, *51*, 24–31. [CrossRef]
22. Mondesire, S.C.; Angelopoulou, A.; Sirigampola, S.; Goldiez, B. Combining virtualization and containerization to support interactive games and simulations on the cloud. *Simul. Model. Pract. Theory* **2018**. In press. [CrossRef]
23. Kempa, W.M.; Woźniak, M.; Nowicki, R.K.; Gabryel, M.; Damaševičius, R. Transient solution for queueing delay distribution in the GI/M/1/K-type mode with "queued" waking up and balking. In *Artificial Intelligence and Soft Computing*; Rutkowski, L., Korytkowski, M., Scherer, R., Tadeusiewicz, R., Zadeh, L., Zurada, J., Eds.; Springer: Cham, Switzerland, 2016; Volume 9693, pp. 340–351. [CrossRef]

24. Demircali, A.A.; Uvet, H. Mini Glider Design and Implementation with Wing-Folding Mechanism. *Appl. Sci.* **2018**, *8*, 1541. [CrossRef]
25. de Santana, R.A.S.; Dias-Júnior, C.Q.; do Vale, R.S.; Tóta, J.; Fitzjarrald, D.R. Observing and Modeling the Vertical Wind Profile at Multiple Sites in and above the Amazon Rain Forest Canopy. *Adv. Meteorol.* **2017**, *2017*, 1–8. [CrossRef]
26. Yu, Y.; Li, Z.; Shi, L.; Chen, E.Y.-C.; Xu, H. Cross-Layer Optimization for State Update in Mobile Gaming. *IEEE Trans. Multimed.* **2008**, *10*, 701–710. [CrossRef]
27. Olokunde, T.; Misra, S.; Adewumi, A. Quality Model for Evaluating Platform as a Service in Cloud Computing. In *Information and Software Technologies*; Damaševičius, R., Mikašytė, V., Eds.; Springer: Zug, Switzerland, 2007; Volume 756, pp. 280–291.
28. Xu, Y.; Shen, Q.; Li, X.; Ma, Z. A cost-efficient cloud gaming system at scale. *IEEE Netw.* **2008**, *32*, 42–47. [CrossRef]
29. Raaen, K.; Grønli, T.-M. Latency thresholds for usability in games: A survey. In Proceedings of the 27th Norsk Informatikkonferanse, Halden, Norway, 17–19 Novenber 2014.
30. Jarschel, M.; Schlosser, D.; Scheuring, S.; Hoßfeld, T. Gaming in the clouds: QoE and the users' perspective. *Math. Comput. Model.* **2013**, *57*, 2883–2894. [CrossRef]
31. Wang, Y.; Chen, I.-R.; Wang, D.-C. A Survey of Mobile Cloud Computing Applications: Perspectives and Challenges. *Wirel. Pers. Commun.* **2014**, *80*, 1607–1623. [CrossRef]
32. Cai, W.; Shea, R.; Huang, C.-Y.; Chen, K.-T.; Liu, J.; Leung, V.C.M.; Hsu, C.-H. The Future of Cloud Gaming [Point of View]. *Proc. IEEE* **2016**, *104*, 687–691. [CrossRef]