Survey VR Digital Toy Car Twins

Amirabbas Sherafatian
University of Tartu, Estonia.
amirabbas.sherafatian@ut.ee

***Abstract* – as the Digital Twins moving towards modeling real objects in virtual environment, it requires us to consider the way the data is exchanged, optimized, stored and moreover, which methodology to follow. Indeed, before step into broad area of Digital Twins, a survey is inevitable and the aim of this paper is to conduct a survey considering definitions, together with the importance of data exchange by going through a methodology for Digital Twins. This paper also tries to keep the focus on the Toy Cars, by exploring Donkey Car S1 and introduce applications to teleoperate the Donkey Car.**

I. INTRODUCTION

It is true if we say Digital twins (DTs) are the result of integration of AI (Artificial Intelligence), IoT (Internet of Things) and Big Data.

The term Artificial Intelligence first introduced by John McCarthy in 1956 [1] and afterwards the definitions came out: "The theory and development of computer systems able to perform tasks normally requiring human intelligence, such as visual perception, speech recognition, decision-making, and translation between languages." (English Oxford Living Dictionary), "concerns the theory and development of computerized systems able to imitate and simulate human intelligence and behavior'" (Merriam-Webster Dictionary). Since 1956, there have been prominent progress in developing intelligent systems led to continuous advances in Big Data processing, Machine Learning, Pattern Recognition and so forth. Now, it is a fundamental part of our life and research teams in Google, Facebook, IBM and more has focused on it.

Emergence of IoT highly influenced the way of exchanging data. "Internet of Things (IoT) plays the role of an expert's technical tool by empowering physical resources into smart entities through existing network infrastructures" [2]. Indeed, by having sensors, cameras, GPS, and many more devices connected to the internet we run into the term Big Data where the three V's are considerable.

"Big data is often characterized by the three V's:

- the large *volume* of data in many environments;
- the wide *variety* of data types frequently stored in big data systems; and
- the *velocity* at which much of the data is generated, collected and processed." [3]

So truly, the integration of AI models, Big Data Analytics and IoT, will yield another field of as Digital Twins in technology.

In this paper, we describe the definition of DTs and review the design methodology. Afterwards, we employ a Donkey Car S1 to experiment making communication with a Raspberry Pi and introduce two applications to teleoperate the car.

II. DEFINITIONS

A DT can be defined as a virtual model of a physical object or a process and while there are similarities between simulation and DT, they differ. Simulations are digital models mapped in computer-aided environments and imitating operations and processes of a system. Then the performance analysis of the system can be facilitated and tested for optimization. On the other hand DTs are virtual models connected to the physical objects not to simulate them but also to receive data from them in real-time and regularly. For example, Tesla automobiles has a digital twins installed on cars to transfer the real-time data to be used for improvement and optimization.

In addition, in some cases, there might be several DTs connected together to enrich the virtual model by exchanging data leading to more performant and optimized physical model. DT goes further to make a two-way communication to apply the optimization on real object.

Therefore, we can consider the characteristics of DT as Real-time simulations, Optimize real-world products and processes and enhancing product design – e.g. Boeing with the help of DTs analyzed the use of diverse material for the aircrafts and achieved 40% improvements in the quality of certain parts [4].

Some more definitions: “A DT is a living, intelligent and evolving model, being the virtual counterpart of a physical entity or process. It follows the lifecycle of its physical twin to monitor, control, and optimize its processes and functions.” [5], “A digital twin is a virtual representation of an object or system that spans its lifecycle, is updated from real-time data, and uses simulation, machine learning and reasoning to help decision-making.” [6].

DTs applications cover broad industries. While there are many applications in the areas of Healthcare; Supply Chain; Construction; Retail, the area of Manufacturing is the one benefits from DTs massively, as the Tesla example was already mentioned. The sub areas of product development, optimization and design customizations (Boeing example) are highly influenced by the advantages of DTs. DTs have made it possible to start evaluating, designing and testing the products even before the physical counterparts exist and as a result it reduces the cost of production effectively [7], [8].

III. METHOD

The DT platforms consist of three environments: physical environment, virtual environment and the environment to exchange data that connects first two environments together.

Real-world objects are in physical environment, which have functionalities, behaviors, characteristics, rules and more. Obviously, these features are actually the data, which there is a need of an environment to transfer this data to the virtual environment. In addition to the data representing the physical objects, there is also dynamic data representing the status of the objects (enriched by the sensors, actuators, cameras, GPS, and so forth) at the current moment. DT in the virtual environment continuously receives the data to mirror, simulate, monitor, control and analyze the objects. Afterwards, it will reflect the optimization achieved by data analysis to the physical objects. In some cases, there are another type of data collected from other DTs that the current virtual model is connected to – there is a chain of digital twins in this manner. The following figure [15] depicts the environments [9], [10].

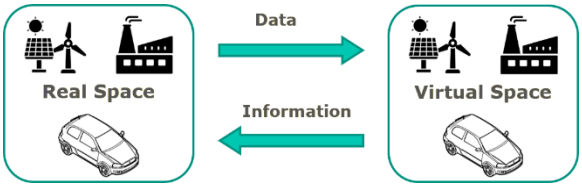

Figure 1 – DT Environments

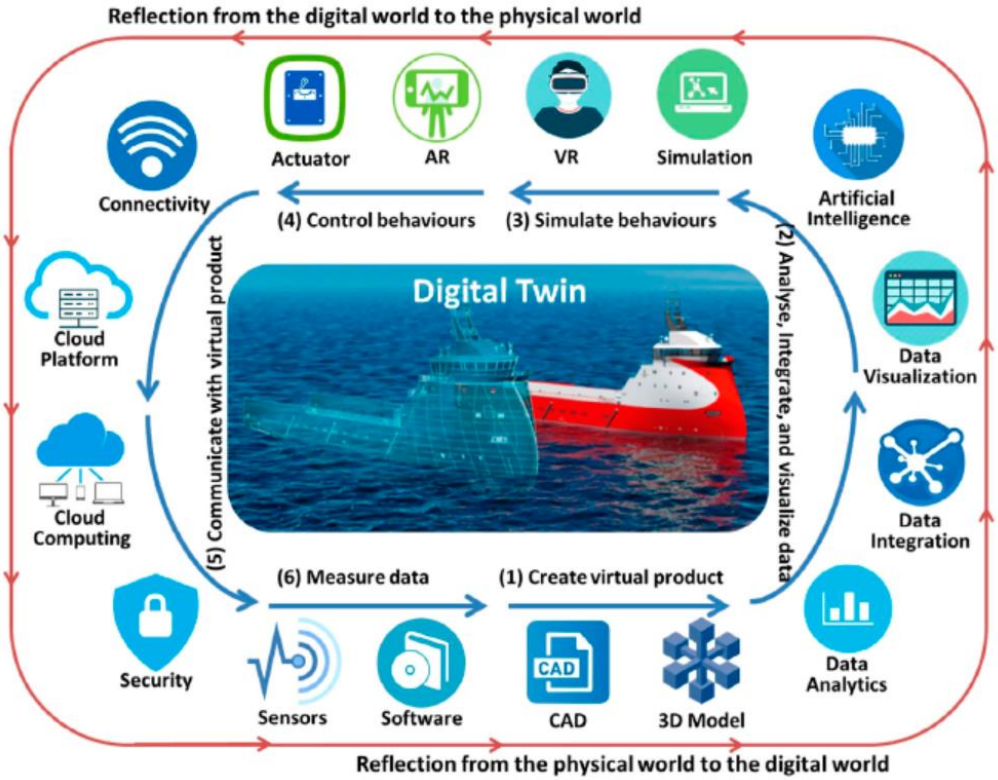

Figure 2 – Enabling technology of Digital Twin

A. Process of building digital twins

As the figure 2 [12] shows, it takes six steps to build a Digital Twin and it is possible to go through these steps concurrently. At the first step, computer-aided design (CAD) and 3D modeling are employed to create the virtual model. The second step is where the data is collected, analyzed and integrated. Since the data is received from various sources continuously, we are dealing with Big Data; hence, in this layer Big Data analytics, ML and AI algorithms (feature selection and feature extraction), Data Fusion, and the techniques are defined to combine data as necessary as possible while the informative data is kept. "data fusion techniques combine data from multiple sensors and related information from associated databases to achieve improved accuracy and more specific inferences than could be achieved by the use of a single sensor alone." [11] (Hall and Llinas). Thus, this layer allows extracting and storing useful data that leads to cost reduction. During the third step, the behavior of the real-world objects with the use of simulation and virtual reality (VR) is built – with VR users are able to directly interact with the DT. In step four, physical objects are enriched to be capable of adjust its function, behavior and structure in the physical world – for the digital twins, sensors and actuators are two fundamental backbones (sensors facilitate the sensing the external world while the actuators executing the adjustment sent by the DT). With the step five, a real-time, two-way, and secure connection between physical and virtual environment are established: Networking technologies e.g. Wi-Fi, Bluetooth, etc. allow the products to send it data and the cloud computing makes

the virtual model deployed on cloud so that it is accessible anywhere through the internet. As the last step, data is collected from various sources. There are types of data that should be processed by the DT: product data, environmental and interactive data.

IV. EXPERIMENT

A Donkey Car S1 (shown in the figure 3 [13]), is employed to experiment making communication with raspberry pi through the Godot Engine and Javascript.

Figure 3 – Donkey Car S1

A. Hardware

The Donkey Car S1 comes with a standard Raspberry Pi 4 and a controller (Robohat MM1) to power the Raspberry Pi and ESC (Electronic Speed Controller – the Electronic circuit that controls and regulates the speed of an electric motor by receiving the signal from MM1). Additionally, there is a camera installed on the car connected to the Raspberry Pi to stream the status of the physical environment. Thanks to the power of Raspberry Pi and the Donkey Car, we are capable of taking the control of the car programmatically via the built-in installed package through web socket connections. Furthermore, the car comes with a mobile application making it possible to teleoperate the car.

B. Exploration

Considering the Raspberry Pi installed on the car, we were able to make a SSH connection to the car and run the built-in package written in python. By running the package, a web server is launched and stays listening so that it is possible to send the request to the server (car). Another way of lunching the web server is through the web interface accessible at http://<IP>:8888/lab when the car is booted up – it is true that the device running the web interface should be in the same network with the car and the <IP> would be the IP of the car. By running the web server there is an enriched repository exposing services for varying purposes, which main ones are to consume the camera streaming and a service to steer the car.

When the web server is running, it is possible to receive the camera output at http://<IP>:8887/video. In fact, the web server continuously writes the images taken by the camera to the response and it is enough to represent the written images in the front end.

Moreover, to steer the car a service at <IP>:8887/wsDrive is accessible. Via this endpoint, we would be able to make a two-way web socket connection and send the messages to steer the car. The message is sent to the car should be in JSON string format and required attributes are “angle”, “throttle”, “recording”, and “drive_mode”. The “angle” and “throttle” are accepting a number between -1 and 1 to steer the car. When the number for the attribute “angle” is negative, the car is set to go to the left. While when it is positive, then orders the car to go to the right. In fact, angle would order the front wheels. Similarly, we can steer the car forward by setting the negative number for the attribute “throttle”. And steer the car backward by setting a positive number. The attribute “recording” cause the images from

the camera to be saved to train the car for self-driving and the attribute "drive_mode" should be set to "user" to indicate the user is steering the car and it is not in self-driving mode.
By having these services and being capable of web socket connections, we are free to choose every preferred programing language to teleoperate the car.
The negative points we run into was the lack of the detailed technical documentation for above mentioned services and the battery issue. The lack of technical documentation around web socket messages made difficulties to be on the track. To overcome the issue we had to go over the sources written in python and built-in Javascript sources to discover how the messages should be constructed and which services are available, which made an overwhelming situation and the progress slow. Actually, the lack of the technical documentation in these regards make some confusion for beginners. Truly, when the weak documentation is combined with the life of the original battery comes with the car, then beginners have to be prepared for the continuous cycle of recharging, waiting, discovering by experiment. For these experiments, we used two more powerful (1700 mAh) batteries instead of original battery (1100 mAh).
To go further with experiment, we wrote two applications one with Godo Engine and another one with Javascript.

C. Godot Based Application

Godot is a free and open source game engine under MIT licence, which firstly was released in 2014. It was first for 2D environments and after that they started supporting 3D on top of 2D. Godot is following a hierarchical based project structure and new features, as new nodes, are added to currently available node (the scene is considered as the root node). The original language for Godot is GDScript - a dynamically typed language, similar to python meaning no compilation needed. On the other hand, the disadvantage is that there is no compiler to recognize the compile time mistakes. Godot is supporting C# as the second language, although it is not recommended yet for production. As it was expected, setting up the environment to support C# is tricky enough and it needs the original environment to be coupled with Visual Studio Code or Visual Studio. Also, to enable debugging there are some difficulties but with some tricks it is possible to overcome the issue. In our prototype, the user can drive the car via keyboard and the main job is done via the object "WSMessage" from the Godot library. This object make the socket connection and after the connection is established, accepts the message and queues the message via the method "PutPacket" as follows:

```
var msg = wsMsg.stringify();
var err = ws.GetPeer(1).PutPacket(msg.ToUTF8());
```

wsMsg is an instance of a class representing the actual message structure. Then by calling "stringify" method, we are converting the message into JSON and then it is queued. Finally, in every game loop, it is checked to send the message to the server if it is available by the method "Poll". The complete source code is available at Github [16] and a demo of the application is available at https://www.dropbox.com/s/by6ax1004xkxfc5/VID_20211212_133307.mp4?dl=0

D. Javascript Based Application

The same application as Godot application is provided with Javascript but the camera is also added to the user interface and consumed. In javascript, the main object responsible to establish web socket connections is the WebSocket object, while the JSON object is used to serialize the message Into JSON format:

```
var data = { "angle": angle, "throttle": throttle,
"recording": false, "drive_mode": "user" };
this.Socket.send(JSON.stringify(data));
```

After having the socket connection established, as the above code shows, the message is converted to the JSON format and sent by the method “send” to the web server. In another part of the code, we listen to the key press via JQuery library to send respective messages to the server to steer the car. As a result, the user with its browser can see the environment continuously via the camera installed on the car and drive the car with the keyboard. The complete source code is available at Github [17] and a demo of the application is available at https://www.dropbox.com/s/d29oa0ewfx0iirf/VID_20211212_134049.mp4?dl=0

V. CONCLUSION

Digital twin is going forward sharply with the demand for its applications in varying areas such as industry mainly. Depends on the application and the area, there are different kind of approaches and frameworks customized based on the needs. Donkey car is truly exciting in terms of hardware and software, but the detailed technical documentation regarding web socket connections needs to be paid attention and beginners would need spend some more time to discover this area by doing experiments. Lastly, Donkey Car S1 has an enriched repository to smoothly teleoperate the car via available services. Both Godot and Javascript is smooth to drive the car but Godot seems to be more robust in socket connections.

REFERENCES

[1] B. Marr. (Feb. 14, 2018). The Key De_nitions Of Arti_cial Intelligence (AI) That Explain Its Importance. Available online:https://www.forbes.com/sites/bernardmarr/2018/02/14/the-key-definitions-of-artificial-intelligence-ai-that-explain-its-importance/?sh=51fc9b074f5d

[2] Md Arafatur Rahman and A. Taufiq Asyhari. (2019). The Emergence of Internet of Things (IoT): Connecting Anything, Anywhere.

[3] Bridget Botelho, Stephen J. Bigelow. (2021) .Big Data. Available online: https://searchdatamanagement.techtarget.com/definition/big-data

[4] exorint. (2020) . Available online: https://www.exorint.com/en/blog/what-is-the-difference-between-a-simulation-and-a-digital-twin

[5] BARBARA RITA BARRICELLI, ELENA CASIRAGHI, DANIELA FOGLI. (2018). A Survey on Digital Twin: Definitions, Characteristics, Applications, and Design Implications.

[6] Maggie Mae Armstrong. (2020) . Available online: https://www.ibm.com/blogs/internet-of-things/iot-cheat-sheet-digital-twin/

[7] behrtech . Available online: https://behrtech.com/blog/digital-twins-for-industry-4-0/

[8] Cem Dilmegani . (2021) . Available online: https://research.aimultiple.com/digital-twin-applications/

[9] Iñigo Ortega. Available online: https://cicenergigune.com/en/blog/digital-twins-heat-management-industry-4-0

[10] A. R. Al-Ali 1, Ragini Gupta, Tasneem Zaman Batool, Taha Landolsi, Fadi Aloul, and Ahmad Al Nabulsi. (2020) . Digital Twin Conceptual Model within the Context of Internet of Things.

[11] DAVID L. HALL, SENIOR MEMBER, IEEE, AND JAMES LLINAS. (1997) . An Introduction to Multisensor Data Fusion.

[12] Fei Tao, Fangyuan Sui, Ang Liu, Qinglin Qi, Meng Zhang, Boyang Song, Zirong

Guo, Stephen C.-Y. Lu & A. Y. C. Nee. (2018) . Digital twin-driven product design framework.
[13] Donkey Car S1 website. Available online: https://courses.10botics.com/
[14] Donkey car guide and documentation. Available online: https://docs.donkeycar.com
[15] Digital Twins represent a disruptive technology for the analysis of systems or industrial processes such as those requiring heat management. Available online: https://cicenergigune.com/en/blog/digital-twins-heat-management-industry-4-0
[16] Godot application source code at Github: https://github.com/sherafatian-amirabbas/GodoPrototype
[17] Javascript application source code at Github: https://github.com/sherafatian-amirabbas/DonkeyCar-WebInterface