Technical Report: Measuring fidelity of DTs –The case of a Lego car–

Paula Muñoz, Javier Troya, Antonio Vallecillo
ITIS Software
Universidad de Málaga (Spain)

Manuel Wimmer
CDL-Mint
Johannes Kepler University (Austria)

Contents

I Introduction

This technical report presents the results of the fidelity analysis of the Digital Twin (DT) of a Lego car. The goal is to show how the alignment algorithm works when snapshots contain various attributes of different types (numeric, boolean, strings).

To validate the alignment algorithm, we define several scenarios. Each one corresponds to a different situation in which we want to study how our algorithm behaves. To focus on the algorithm itself, we created synthetic traces for both the DT and the PT in each scenario.

1) **One lap: synchronized**. The DT and the PT follow the same trajectory during one lap, both being fully synchronized.
2) **One lap: timestamp uncertainty**. Both the DT and the PT follow the same trajectory during one lap, and at the same speed, but the timestamps of the snapshots may contain some inaccuracies due to imprecision of the clocks.
3) **One lap: DT accelerated**. Both the DT and the PT follow the same trajectory 7during one lap, but at different speeds: the PT maintains a constant speed while the DT accelerates and decelerates.
4) **One lap vs Two Laps**. The speed of the DT is double that of the PT. Therefore, the PT performs one lap while the DT performs two in the same time.
5) **Two laps: synchronized**. The DT and the PT are fully synchronized during two laps.

The structure of this report is as follows. After this introduction, Section II-A describes the Digital Twin System (DTS) used to conduct the experiment. Then, Section II-B presents the alignments produced by our algorithm and the results of the fidelity metrics for each scenario.

II Fidelity analysis of a Lego Mindstorms NXT Car at the ITIS Software Institute

A. System description

The Atenea Research Group is a part of the ITIS Software Institute in Málaga. Their research is focused on applied research on modeling software systems, and recently, they started studying how to apply modeling concepts to Digital Twins. In this context, they developed a conceptual architecture and applied it to two cyber-physical systems: a Lego Mindstorms NXT [1] and an Arduino Braccio [2].

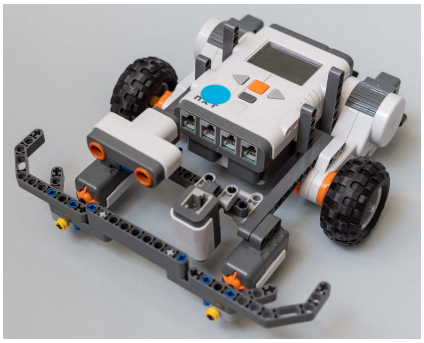

Fig. 1: The Lego car.

The Lego car is a small vehicle equipped with sensors, as shown in Fig. 1. It can move, detect obstacles, and interact with its surroundings in various ways, such as following a colored line on the floor. The car communicates with a computer via Bluetooth, exchanging data and commands, including information about its current state (position, speed) and sensor readings.

The car is equipped with a pose-provider that determines its planar coordinates and the angle of its current direction. It also has an engine that enables it to move forward and rotate its direction at a certain angle. Additionally, the car has three types of sensors: an ultrasonic sensor that can detect the distance to the object in front of it, a light sensor that can determine the color of the ground beneath it and two touch sensors that act as a bumper to determine whether the car has collided or not.

The car is equipped with software that controls its movements. This software can be programmed to perform different behaviors. For instance, the car can be set to follow a black line on the ground using the `LineFollowingBehavior`. If the car detects a color other than black, it means that it has deviated from the line.

In this case, the car stops and starts turning left and right, increasing the angle each time until it detects the black line again. Once the line is found, the car resumes moving forward. Other behaviors can also be programmed into the car, such as turning a certain number of degrees when it hits an obstacle or when the distance to an object in front is too close. It is important to note that this software is an integral part of the car, meaning it is physically built into the vehicle.

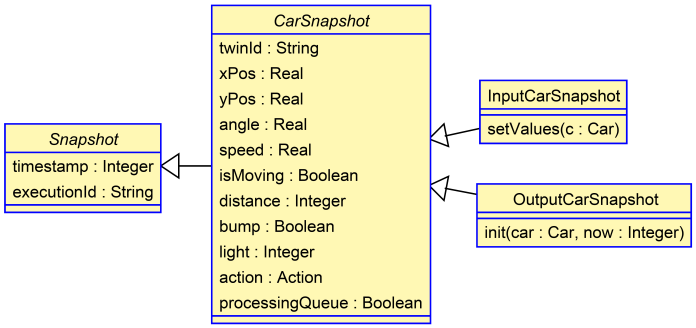

Fig. 2: Snapshots capturing the Lego car state.

We used this car as the initial prototype to test our approach. This example is also very useful to illustrate how snapshots can include non-numeric attributes and how they reflect on the alignment results.

The snapshot contains a specific set of attributes. These attributes are as follows:

- `timestamp`: the moment in time when the state of the system (DT or PT) was captured. It is represented by an integer value using the POSIX notation [3].
- `executionId`: the unique identifier of the execution (the database may contain several executions of the same DT).
- `twinID`: the unique identifier of the digital twin.
- `xPos, yPos`: the position of the car in Cartesian coordinates.
- `angle`: the angle in degrees towards which the car is heading.
- `speed`: the speed at which the car is moving.
- `isMoving`: a boolean attribute that determines whether the car is moving or not.
- `distance`: the distance to any object in front of the car, as provided by the ultrasonic sensor.
- `bump`: indicates whether the car has collided or not, as provided by the touch sensor.
- `light`: the color of the ground beneath the car, as detected by the light sensor.
- `action`: the action that the car is executing, which can be either Forward or Rotate.
- `processingQueue`: indicates whether the snapshot has been processed by the database.

All attributes are of interest for the alignment, except `executionId`, `twinId`, and `processingQueue`. We did not want to consider the `timestamp` attribute here, because assigning a MAD to it would mean establishing temporal windows, something that we wanted to avoid in this example.

To emulate the car's behavior, we developed a Digital Twin using the USE modeling tool [4]. However, when we repeatedly executed the same deterministic behavior of the Lgo car, it was unable to perform a similar trajectory. It deviated from the intended behavior too often. As a result, we decided to use it as an illustrative example for our work [1] using synthetic traces instead of actual traces. Therefore, in the following, the DT will represent the theoretically expected behavior of the Lego car, while the PT will represent different variations of the expected behavior, synthetically produced.

B. Scenarios

Verifiability: The alignments performed in this section are available at https://github.com/atenearesearchgroup/fidelity-measure-for-dts/blob/main/src/resources/output/nxj/

Since these traces are artificially created, and the scenarios were designed to demonstrate how parameters of various datatypes (such as enumerations, booleans, and numeric values) interact with each other, we did not conduct a MAD tuning. However, we will use the accuracy of the sensors to determine the MAD for each of the parameters of interest. By default, the MAD value is taken as three times the corresponding sensor accuracy. Therefore, the MAD values applied for the numerical values are:

Parameter	Accuracy	MAD
`xPos, yPos`	0.50 cm	1.50 cm
`angle`	0.05 $degrees$	0.15 $degrees$
`speed`	0.50 cm/s^2	1.50 cm/s^2
`isMoving`	-	-
`distance`	0.50 cm	1.50 cm
`bump`	-	-
`light`	0.05	0.15
`action`	-	-

To align a snapshot, the boolean attributes (`bump`, `isMoving`) and the enumeration types (`action`) must all have the same value. If any of these attributes have different values or if the numerical values differ by more than their corresponding MAD values, the snapshots will not be matched and the comparison function will return 0.

1) One Lap - Synchronized

In this case, PT and DT are fully synchronized, and no uncertainty or noise is included in PT readings, as shown in Figure 3. The fidelity metrics produced the following results: % of matched snapshots (%MS) 100%, Frèchet distance (FD) 0.0067, and Euclidean distance (ED) 0.004. Distances are not zero because there is a little noise in the synthetic trace.

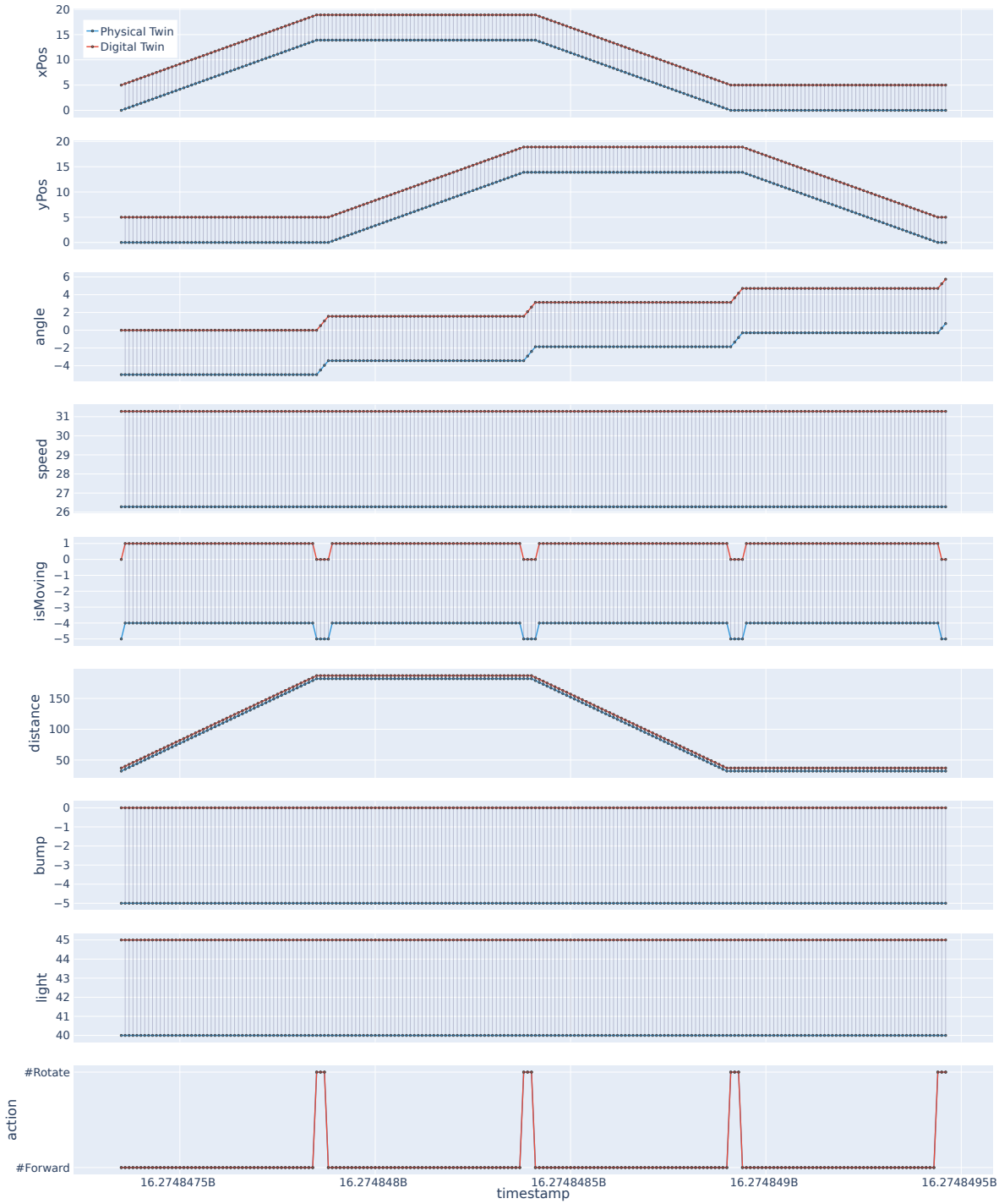

Fig. 3: Alignment for the scenario *One Lap - Synchronized*.

The figure includes a 5-unit difference in each numerical plot to improve visualization.

2) One Lap - Timestamp Uncertainty

In this scenario, both the DT and the PT perform the same trajectory. The only difference between the traces is that the PT includes certain variations in the `timestamp` attribute to emulate the uncertainty of a real system, where clocks are inaccurate and some noise is unavoidable. The fidelity metrics produced the following results: % of matched snapshots (%MS) 99.53%, Frèchet distance (FD) 0.007, and Euclidean distance (ED) 0.004. In Figure 4, we see how most of the snapshots are aligned without noticeable gaps.

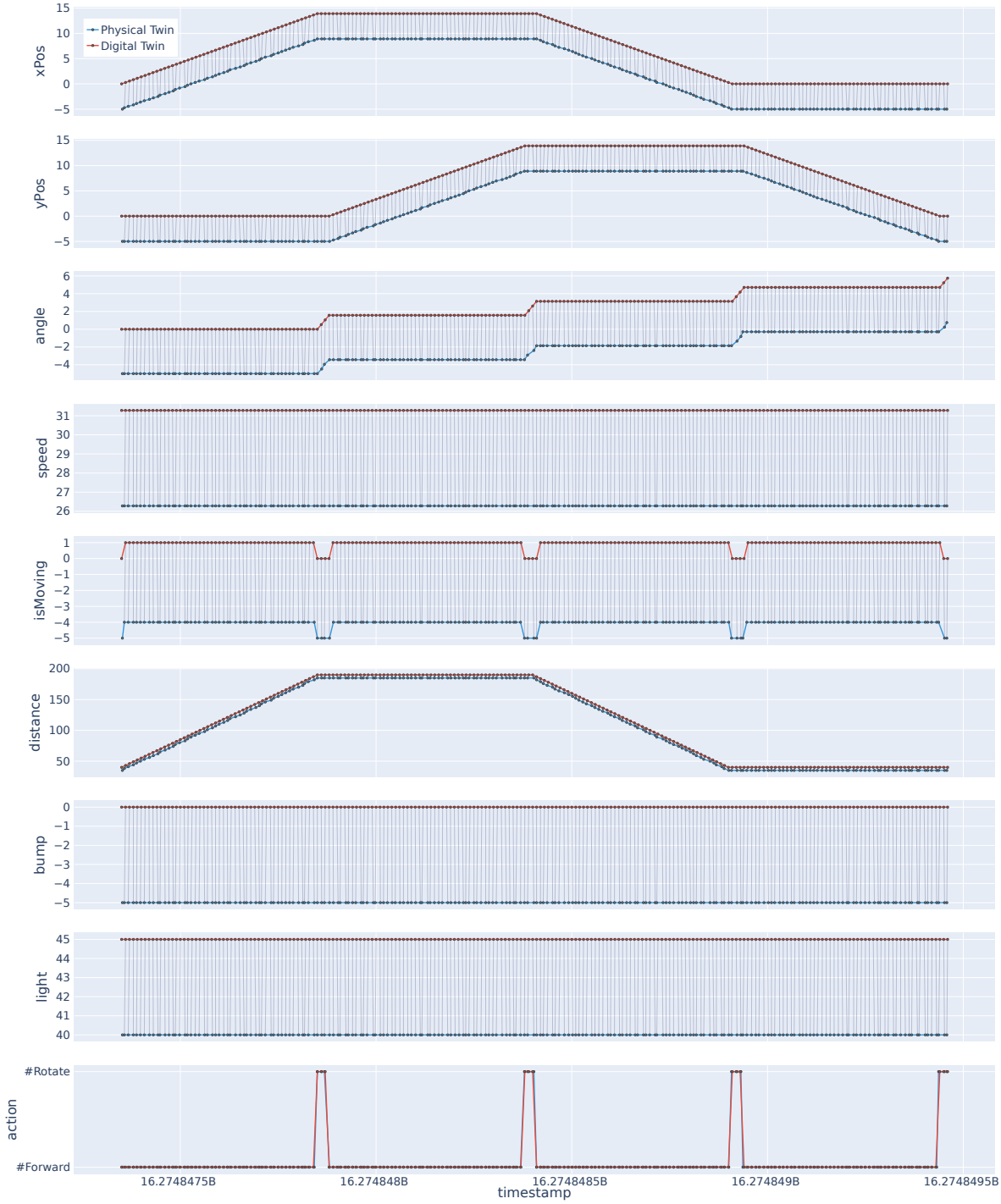

Fig. 4: Alignment for the scenario *One Lap - Timestamp Uncertainty.*
The figure includes a 5-unit difference in each numerical plot to improve visualization.

3) One Lap - DT accelerated

In this scenario, we wanted to simulate how the PT increases its speed in the lap. This led to inconsistencies in the alignment, wherein we were only able to align the parts where both speed and positions matched. The results are shown in Figure 5. We can see that the gaps and mismatches are located in the parts where either the `yPos`, `xPos`, or `speed` differ. The fidelity metrics produced the following results: % of matched snapshots (%MS) 49.05%, Frèchet distance (FD) 1.975, and Euclidean distance (ED) 0.758.

Fig. 5: Alignment for the scenario *One Lap - DT accelerated*.
The figure includes a 5-unit difference in each numerical plot to improve visualization.

If we double the MAD value of these parameters to 3, we can increase the percentage of matched snapshots at the cost of increasing distance, as shown in Figure 6. The fidelity metrics produced the following results: % of matched snapshots (%MS) 91.04%, Frèchet distance (FD) 2.91, and Euclidean distance (ED) 1.63.

Fig. 6: Alignment for the scenario *One Lap - DT accelerated* doubling the MAD of `xPos`, `yPos` and `speed`.
The figure includes a 5-unit difference in each numerical plot to improve visualization.

However, if the inconsistency is in enumerated or boolean values, we do not have the option of increasing the MAD, and the algorithm is forced to find a combination of snapshots that have the same values for all the attributes, including those to make a match. In Figure 7, we can observe the impact on the alignments when the error is in an enumerated attribute. The algorithm includes gaps and mismatches in the snapshots

that correspond to the fact that the `#Rotate` action is not correctly located. The fidelity metrics produced the following results: % of matched snapshots (%MS) 85.37%, Frèchet distance (FD) 2.91, and Euclidean distance (ED) 1.643.

Fig. 7: Alignment for the scenario *One Lap - DT accelerated* doubling the MAD of `xPos`, `yPos` and `speed` altering the sequence of actions.

The figure includes a 5-unit difference in each numerical plot to improve visualization.

4) One Lap vs Two laps

In this particular scenario, we are attempting to align the trace of a DT, which performs one lap around the circuit, with a PT, which performs two laps in the same amount of time. However, this results in several inconsistencies in the attributes, including numerical, boolean, and enumeration types. As a result, the algorithm is unable to align any of the snapshots, as shown in Figure 8. The fidelity metrics resulted in 0% matched snapshots (%MS), 97.6% mismatched snapshots, 0 Frèchet distance (FD), and 0 Euclidean distance (ED) due to the lack of aligned snapshots.

Fig. 8: Alignment for the scenario *One Lap vs Two laps*.
The figure includes a 5-unit difference in each numerical plot to improve visualization.

5) Two laps - Synchronized

In this scenario, both PT and DT perform two laps around the circuit in synchronization, as shown in Figure 9. The fidelity metrics produced the following results: % of matched snapshots (%MS) 100%, Frèchet distance (FD) 0.01, and Euclidean distance (ED) 0.006.

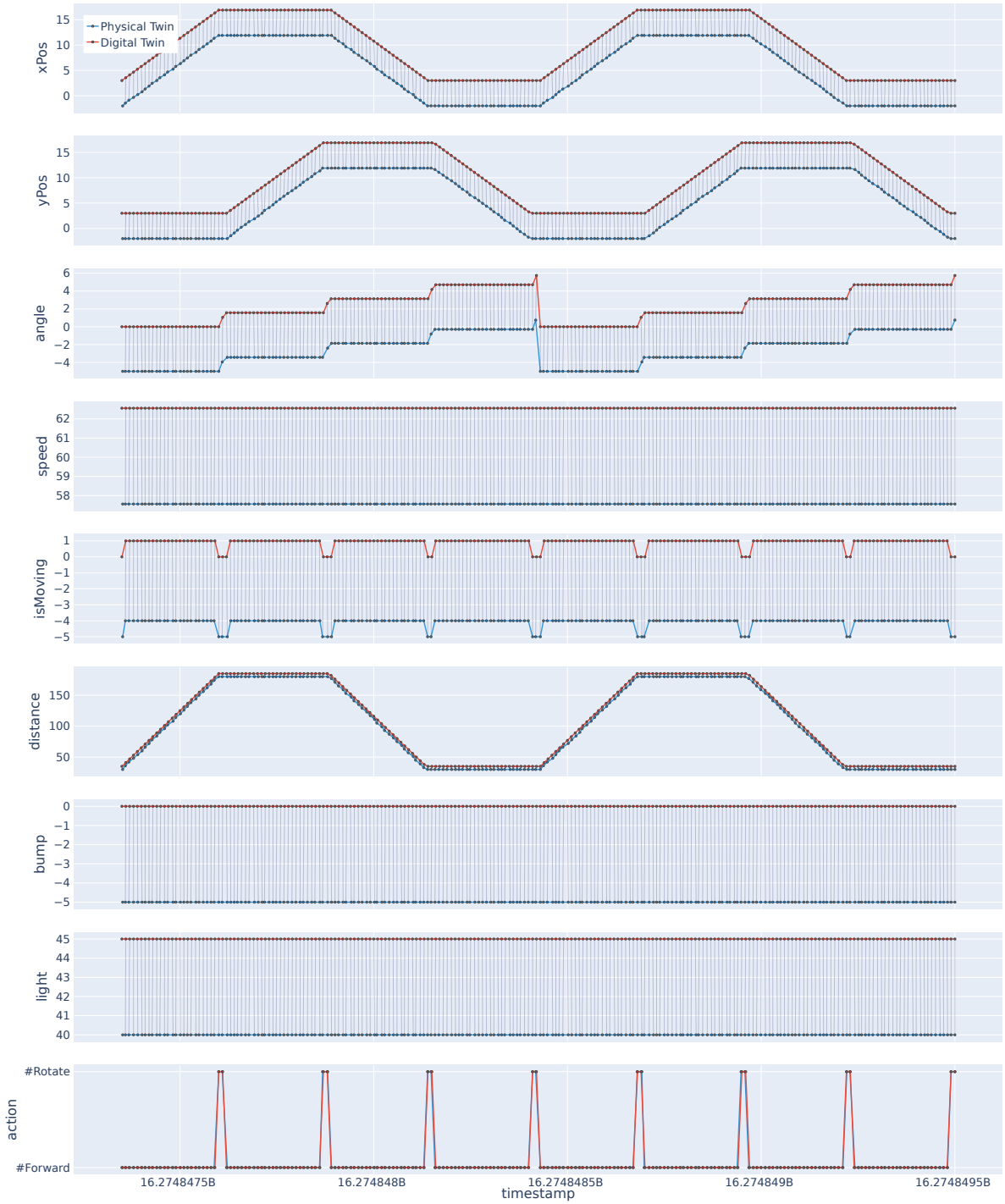

Fig. 9: Alignment for the scenario *Two laps - Synchronized*.

The figure includes a 5-unit difference in each numerical plot to improve visualization.

References

[1] P. Muñoz, J. Troya, and A. Vallecillo, "Using UML and OCL Models to Realize High-Level Digital Twins," in *Proc. of ModDiT2021@MODELS'21*. IEEE, 2021, pp. 212–220.

[2] D. Pérez-Porras, P. Muñoz, J. Troya, and A. Vallecillo, "Key-Value vs Graph-based data lakes for realizing Digital Twin systems," in *Proc. of MeSS@STAF'22*, 2022.

[3] IEEE Std 1003.1-2008, *The Open Group Base Specifications. Issue 7, Sect. 4.16, Seconds Since the Epoch*, 2016.

[4] M. Gogolla, F. Büttner, and M. Richters, "USE: A UML-based specification environment for validating UML and OCL," *Sci. Comput. Program.*, vol. 69, pp. 27–34, 2007.