DejaVu: Visual Diffing of Cyber-Physical Systems

Fergus Leahy
Department of Computing, Imperial College London, London, UK
fergus.leahy@imperial.ac.uk

Naranker Dulay
Department of Computing, Imperial College London, London, UK
n.dulay@imperial.ac.uk

Abstract

In this paper we present DejaVu, a novel 3D virtual world co-simulator for 'visual diffing' of cyber-physical system deployments in indoor and outdoor environments. Using faster-than-real-time simulation and efficient recording DejaVu can record days of simulation data, including environmental, sensor and network data for later replay and analysis. DejaVu enables developers to replay and visually compare multiple simulations simultaneously using different visual diffing techniques, including ghosts, paths, colour and size, highlighting differences between runs, including energy consumption, radio metrics, movement, etc. We demonstrate several of these visual diffing techniques in an CPS-enhanced evacuation case study.

Categories and Subject Descriptors

I.6.7 [**Simulation and Modeling**]: Simulation Support Systems – Environments

General Terms

Design, Experimentation, Performance

Keywords

Cyber-Physical Systems, Visual Diffing, Environment

1 Introduction

Cyber-physical systems and IoT devices are becoming increasingly integrated into buildings around us, providing building climate-control, fire detection, security, lighting and numerous other services for homes and offices. Building and integrating these systems quickly becomes complex, with many different types of devices, vendor's ecosystems and protocols to deal with [5]. Real-world end-to-end testing of such systems, to ensure correctness, robustness and energy-sustainability, is time-consuming, expensive and often impractical or inconvenient, e.g., requiring repeated access to office space to set-up, test and move networks of nodes.

Simulation tools provide part of the solution to these issues, enabling software testing on virtualised target devices, whilst remaining accurate enough for a successful deployment in the real-world. However, traditional test-beds and simulation tools have limited facilities for simulating the environment, post-simulation analysis and data-processing. A typical simulation utilises scripted inputs or traces to provide input to running tests, recording all outputs and events to log files. Developers can observe simulated behaviour visually (through network maps, timelines and visual sensor feedback, such as LEDs) whilst it is running. After which, the recorded data can be viewed and compared to other runs textually, or through ad-hoc processing into visual graphs and plots etc. Although simulation runs can be recorded into video, this only provides a single fixed viewpoint to observe the simulation from. Thus, the tools and techniques for supporting developers to test and analyse cyber-physical systems intuitively in simulation remains limited.

In this paper we:

- Present the concept of visual diffing of CPS simulations and explain how our CPS simulator, DejaVu, implements several visual diffing features (section 3).
- Demonstrate the use of visual diffing in the setting of a CPS-assisted fire evacuation scenario, utilising several visual diffing features of DejaVu to highlight differences and points of interest between multiple simulated runs (section 4).
- Show that visual diffing using DejaVu is efficient and able to run on a modern desktop machine, simultaneously recording, viewing and visual diffing hours of simulation data (section 3.5.2 and 5).

2 Related Work

Compared with the current WSN and CPS simulators [15, 12, 7], including Cooja and Tossim, which provide only manual sensor input or fixed traces, DejaVu adds a full physics-based 3D debugging environment and support for visualisation of human mobility traces.

As modern desktops and graphics card become more powerful and capable of supporting complex and dynamic 3D worlds, research has taken to utilising 3D game engines and frameworks for environmental and CPS simulation: Kim et al.[9] utilise an SMT solver with the Unity 3D game engine to generate and visualise virtual roads for large-scale au-

tonomous car testing; Ardan [11] is a 3D CPS co-simulation framework for distributed CPS testing, utilising deployable application code, phenomena-on-demand and crowds, to create controllable, dynamic and life-like environments; 3D simulation has also been used for design and testing of visual algorithms for robotic interaction, traversal and navigation of virtual environments [1, 10, 3].

Tools such as Gazebo [1], Ardan [11] and others [14, 10, 3], provide 3D environment simulation, but don't consider the difficult task of analysing these 3D simulations, relying on users to attempt observing everything at once, as if looking for a needle in a haystack, or requiring them to later review and manipulate logs of non-visual meta-data without the visual and environmental context. Utilising visual diffing, DejaVu enables visual analysis of visual and non-visual data in real-time within a live simulation; coupled with full simulation replays, it allows for developers to perform more complete analysis across an entire environment from within the simulator.

Testbeds provide an alternative to simulation, mixing the real-world hardware with a semi-realistic environment. Chatzigiannakis et al. focus on overcoming the difficulties in building and managing large and varied testbeds, by creating federated test-beds across many institutions [4]. Dhoutaut et al. utilise sensor memory checkpointing to enable repeatable and replayable testing by saving and resuming a sensors state using a wired backbone [13]. Whilst these approaches maintain the hardware accuracy and improve upon testing efficiency, they are typically deployed in lab environments, thus, don't provide realistic environmental context. In contrast, Kartakis et al. built Waterbox [8], a table-top-scale version water network to test water network based CPS deployments, providing realistic context and control of real valves and water flow.

3 Visual Diffing of Cyber Physical Systems

Inspired by its use in other domains, such as live sports and video games, visual diffing enables viewers to see the differences between two or more instances of an event visually incorporated directly into the visual medium itself, instead of just presenting a simple data metric alongside, such as time. A common technique is visual ghosting[1], e.g., two individual skiers racing, visually overlaid as ghosts onto the same race track, provides a visually appealing and intuitive experience as the race progresses. Similarly in DejaVu, instead of viewing simulations in isolation (1a), developers can visual diff two simulations using ghosts, as shown in figure 1b, to show how differently crowds of people move.

In the rest of this section we introduce DejaVu and explain in more detail the high-level visual diffing features which make DejaVu unique from the current state-of-the-art simulators, before describing the design and implementation of these features.

[1]The name ghosts comes from the ability of entities to pass through objects and each-other unaffected, not necessarily their ghastly colour, which is added to aid in disambiguation.

3.1 DejaVu - A Visual Diffing, 3D Virtual World CPS Simulator

DejaVu is a 3D co-simulator platform for CPS and WSN, designed for running and comparing multiple simulation runs of a system in parallel. DejuVu is built using Cooja, a multi-level WSN simulator for Contiki-based nodes, and Unreal Engine 4 (UE4), a 3D game engine, providing a virtual world to virtually deploy real code and simulate the interactions between a CPS, the environment and virtual crowds. DejaVu can be used in both indoor and outdoor settings.

DejaVu can efficiently record days of simulation data, including data from the virtual environment, sensors and radio network, enabling a full reconstruction of test scenarios for later playback, review and analysis. Unlike video recordings, using this reconstruction, developers are able to move through not only time but also space within recorded simulations, stopping, rewinding and fast-forwarding to review recorded simulations from any point of view at any point in time. To aid with traversing and analysing recordings, developers can select predefined events which DejaVu can watch for; when detected, a checkpoint is logged, enabling reviewers to at a glance see when events of interest have occurred and instantly replay them. DejaVu can also compare recordings in real-time, overlaying and highlighting differences between the current simulation and previously recorded runs, such as battery usage, radio utilisation/interference, movement, as well as other sensor readings.

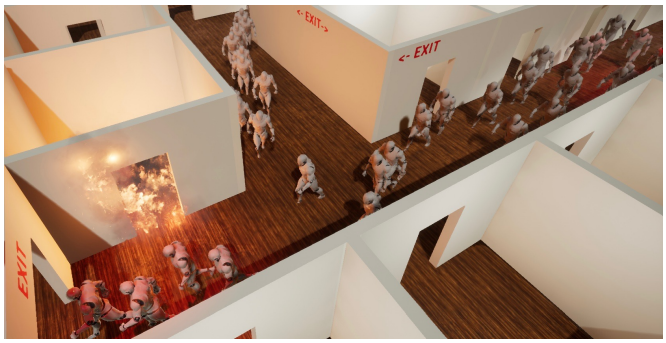
DejaVu simulations contain a number of nodes, which each have a corresponding representation in both Cooja and UE4, virtual and physical, respectively. For each node: Cooja simulates the software and radio components; UE4 simulates the physics for a nodes physical location and movement within the environment, provides visual representations of sensor devices and simulates the sensor interactions with the environment and people e.g., motion detection, fire detection.

Whilst DejaVu is currently configured for use with Contiki's Cooja WSN simulator and UE4, DejaVu's architecture is agnostic and independent from these tools; in fact, it is designed to coordinate and compose multiple different simulators and tools.

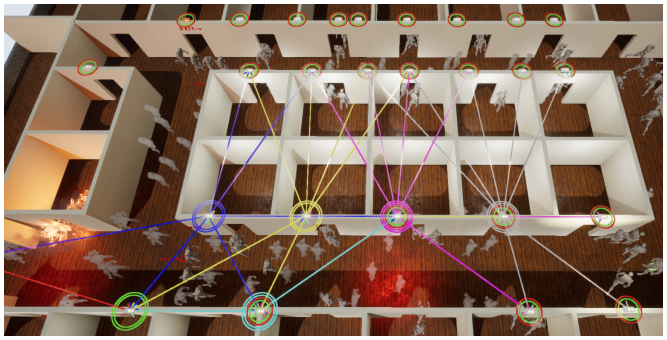
3.2 Time Control

When testing a CPS, time is a key factor: how long will *mote 1* last for, what happened at time *t*, how long did task *x* take to complete, etc. Traditional methods for carrying out tests in the real world or simulations only allow observers to observe an event once live, or record an event from one or more fixed perspectives and then view only exactly what was in view when recording. This has obvious limits when it comes to live and post-test analysis. If events or data are missed, not recorded or out of view, then tests may need to be re-run, data approximated or interpolated. DejaVu supports multiple time control features, including pausing and fast-forwarding live simulations, as well as recording simulations for later playback and review.

During a live simulation time can be paused, giving developers more time to fully observe the current state of the simulation, including moving around the 3D environment,



(a) Non-CPS-enhanced evacuation, with evacuees navigating to the closest exit, possibly past danger zones. Zoom to fullscreen to appreciate visualisation.



(b) A birdseye view of a fire-evacuation using visual diffing. Ghosts represent a previous run, whilst solid grey people represent the live run. Coloured circles and lines represent node status and transmissions, respectively.

Figure 1. A normal evacuation vs a CPS-enhanced evacuation with visual diffing support.

before resuming. Time can also be fast-forwarded, reducing the real-world time for a simulation to be carried out. These two time control features enable developers to become more efficient and effective in carrying out live simulations.

To aid in post-simulation review and analysis, everything within a simulation run can be recorded, saved and played back later, including data on people, objects, sensors, network traffic, etc. Unlike a video recording of a simulation visualisation, which provides only one fixed viewpoint into a simulation, DejaVu, using the recorded data, can completely reconstruct the simulation, enabling developers to review the simulation from any viewpoint at any point in time. This gives developers the power to investigate the simulation for events missed the first time or to simply change perspective to better understand a phenomena from another angle. Similarly, analytical filters and effects can also be retroactively applied, providing new views into recorded simulations.

3.3 Visual Diffing Tools: Ghosts and Colour

Often when testing cyber-physical systems, either deployed in the wild or in simulation, developers are attempting to spot particular events, patterns or phenomena and compare these to previous observations or a baseline.

For real world deployments, we are limited to what we can see through visual interfaces, external physical outputs or textual data logs. Similarly, by simply looking at the cyber-physical system in situ it is very difficult to observe activity and interaction between nodes. On top of this, trying to compare different test runs visually is extremely hard, without resorting to data logging and graphing.

Like in the real world, within the simulated virtual world observing a system as-is can be just as difficult. However, because it is virtual, we can also enhance our view of it, super-imposing or overlaying information that would otherwise hidden from view, such as radio traffic, node stats, or sensor readings.

Taking this one step further, using a recording saved earlier, the simulator can simultaneously run one or more simulations in parallel, enabling observers to visually compare or "diff" them directly. However, the difficulty arises in how to represent the non-visual data in intuitive ways. Depending on the type of data being compared, DejaVu presents diffs in multiple ways, appropriate to the data type.

In a 3D world the simplest data to compare visually is position. When comparing two runs, a live and recorded run, movable physical elements from both can be shown simultaneously. To aid in differentiating elements of each simulation, the recorded run's elements are shown as ghosts, translucent in appearance which don't interact or collide with objects in the live simulation and simply repeat state-by-state what was recorded. This provides immediate visual clues to where simulation runs differ, useful in cases where the mobility of people or objects within it are of interest, such as in a observing crowd movement in an evacuation scenario or navigation decisions for domestic cleaning robots.

To further improve this, additional visual effects can be used to filter out the visual noise. For example, when objects and their ghost stay within a set distance of one another, their colour is desaturated, however, when they breach this limit, they are visibly highlighted in colour, making it clear which objects are of interest.

For meta-data, which is not typically visible on the surface of a node, such as node stats or sensor readings, we need to be more creative and take advantage of the customisation aspects of the 3D virtual world. We have created several techniques to visually signal differences between a live and recorded run: to simply alert the observer we can cause the whole device to periodically illuminate a particular colour, and simultaneously desaturate the others to further enhance the observability of it e.g., a node flashes red to signify its status is different between the two runs at the current time slice; alternatively, the simulator can also illuminate the device using a gradient of colours, signifying the intensity of the difference being observed between the two runs for a particular device e.g., nodes illuminate on a range from yellow to red based on the power difference between two runs, highlighting which nodes are affected the most. Other techniques include changing the size of devices based on the size of change, or shaking nodes.

3.4 Checkpoints: Needles in a Haystack

Utilising faster-than-real-time simulation, developers can run and record simulations up to three times faster than real-time. Thus, developers can run and review a virtual day within 8 hours of real time. However, this poses a problem for reviewing and analysing these long simulated runs, which can be akin to looking for a needle in a haystack when trying to find issues, key events or behaviours to analyse whether or not a system is operating correctly.

To resolve this problem, in DejaVu developers can add an event observer to the simulation, which detects when a particular condition within the simulation is met. For example, a developer may be interested when a node's power level drops below a threshold, or when a sensor is activated. Upon triggering an event observer, the current snapshot is tagged as a checkpoint with the event type. When the simulation is replayed, the developer can then skip directly to any of the event checkpoints that were generated, highlighting snapshots of interest. Of course, developers can then choose to investigate both in time and space around this checkpoint, rewinding or fast-forwarding, as well as moving around the virtual world.

3.5 Design

3.5.1 Co-simulator

DejaVu uses a publish-subscribe event system: in which a plugin for each component publishes or subscribes to events of interest. For example, the Unreal engine publishes sensor positions in the 3D virtual world, to which Cooja subscribes, thus updating it's radio interference model with every new position update. This approach enables the platform to be flexible and modular, allowing multiple components to publish and subscribe to information simultaneously, and for new features, tools or extensions to be added easily.

3.5.2 Recording

In order to perform time-control functions, DejaVu records both observable and unobservable behaviour within a simulation, including physical information about an object's position, trajectory and rotation, as well as metadata and state associated with any devices, sensors and the radio environment e.g., power usage, sensor readings and radio transmissions. Recording both observable and unobservable information ensures replays can be fully reconstructed or later modified to include new views or filters over the information and world.

To optimise the amount of memory used, DejaVu's snapshots only contain data about objects whose state has changed since the last snapshot, thus ensuring memory is not wasted recording duplicate data. However, even when using this method of recording, memory size still becomes an issue when recording a large number of objects over an extended period of time, e.g., 60 people + 20-30 sensors over a 24hr period consumes 36GB, exceeding the RAM available on even high-end machines.

Thus, to further reduce the memory used, DejaVu utilises a segmented compression scheme, seen in figure 2. The continuous stream of snapshots are divided into segments of a fixed size. Segments which are not currently being played are compressed and written to storage. Segments are lazily loaded back into memory and decompressed when necessary. Using this scheme DejaVu saves a considerable amount when storing these recordings, up to 30x improvement as shown in figure 1. Using this technique, DejaVu can record arbitrarily long simulations.

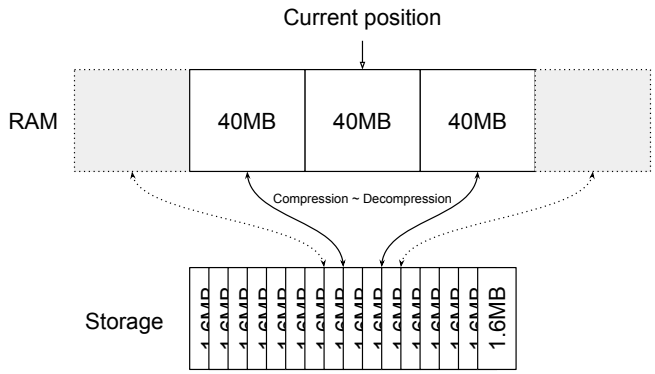


Figure 2. Segmented compression scheme employed by DejaVu to reduce in-memory usage.

To prevent this scheme from negatively impacting the performance when skipping back and forth through a recording, segments before and after the current segment are pre-emptively loaded. The pre-loading distance can be adjusted to balance the performance and storage needs. For long-distance scrubbing, i.e., skipping hours or days, each segment is time-stamped with the start- and end-times of the snapshots held within, providing improved performance when indexing into the correct segment.

Using this scheme, DejaVu is not only able to record week-long simulations which would far surpass the available memory on even a high-end workstation, but can also load multiple recording streams simultaneously, opening up the possibility to replay, diff and compare multiple runs in real-time.

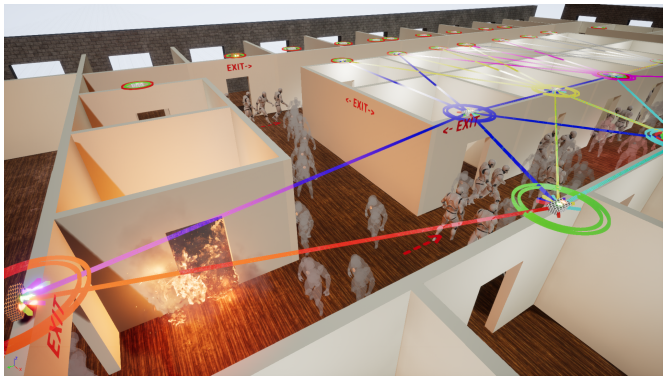
Table 1. Size comparison between raw and compressed snapshots saved to disk, varying in length.

	1min	1hr	24hr
raw	40MB	1.5GB	36GB
compressed	1.6MB	45.1MB	1.1GB

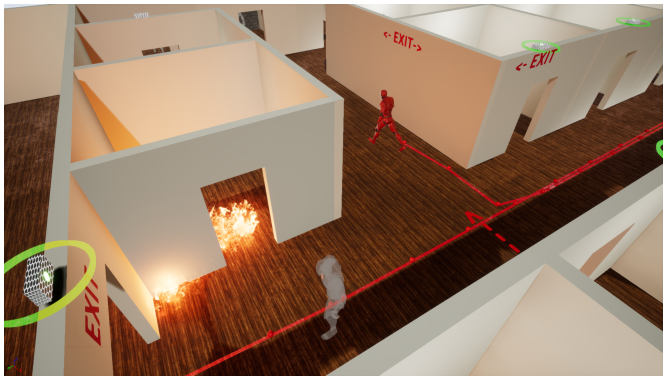
3.5.3 Visual Diffing

DejaVu supports visual diffing a recorded simulation against another recording or a live simulation. Visual diffing utilises the simulation recording and replaying methods described in the section 3.5.2, allowing more than one simulation stream to be played back in memory simultaneously.

When replaying and diffing two simulations, each recordable object (device, person or moveable object) in DejaVu keeps track, within its log of snapshots, of a pointer to the snapshot closest to the current replay time, updated at each tick in the game engine. DejaVu runs the selected diff on each relevant corresponding object in both simulation



(a) CPS-enhanced evacuation diffed with non-enhanced ghosts. Evacuees are being directed to exits via safe routes.



(b) A person evacuating chooses two different paths in different runs based on CPS assistance, the ghost shows the same person taking a different route in a previous run.

Figure 3. Different visual diffing techniques for differentiating different aspects of simulation runs.

streams, which compares the current snapshot of each object to determine the differences. Developers can also define the difference threshold, specifying the closeness of the objects' values before they are considered different. Currently, DejaVu supports a library of built in diffing functions, including location, device sensor state (motion-, fire-detection), LED state and radio usage.

The results from each diff function are then linked to different visual diffing features, e.g., instantaneous location differences are shown through ghosting (figure 3a), long-term location differences through visual tracks (figure 3b), discrete differences through a colour change of the object and continuous differences by a colour gradient. Multiple visual features can also be applied to a single diff function, such as using ghosting for showing location differences, but also colouring the corresponding objects should they differ by a developer-defined threshold, visually alerting the developer to points of interest. New visual features can be added and swapped in to experiment with visual techniques for representing differences. Similarly, we also envision the use of sound alerts or notification overlays.

4 Case study: Fire Evacuation

To demonstrate DejaVu's features we developed a case study which focuses on the use of a cyber-physical enhanced fire detection and evacuation system for large buildings with multiple corridors and exit routes, such as offices or shopping malls. Previous work [6, 2] has demonstrated the use and benefit of utilising a 2D evacuation simulator to develop new cyber-physical systems to enhance detection and evacuation, ensuring people are directed away from high-risk areas, reducing congestion at critical paths and providing detailed information for emergency crew to locate areas of high-interest. Our case study utilises the algorithm from this approach, to demonstrate and evaluate the unique simulation features of DejaVu, including 3D analysis, playback, diffing and checkpointing.

The algorithm, shown in pseudo-code form in figure 4, runs on each node within the network. Each node first broadcasts its distance to an exit; upon receiving a distance message, a node recalculates its shortest path to an exit; upon receiving a fire message, a node removes the senders path and recalculates the shortest distance to an exit taking into consideration the distance to the closest exit via a path and that path's hazard risk based on its distance to the nearest hazard; lastly, if a node receives a fire message from its own sensor, it broadcasts the fire alert to the rest of the network. Each node knows the direction of its nearest neighbours at deployment time, thus, using light indicators, observers can be directed along the calculated safe routes.

```
1  broadcast msg {ID, dist}
2  select:
3    waiton msg {ID, dist}:
4      routes[ID] = dist + 1
5      effectiveDist = min(dist+1, effectiveDist)
6      if changed:
7        broadcast msg {ID, effectiveDist}
8    waiton msg {ID, FIRE}:
9      routes[ID] = FIRE
10     effectiveDist = min_hazard(routes)
11     broadcast msg {ID, effectiveDist}
12   waiton sensor {FIRE}:
13     broadcast msg {ID, FIRE}
```

Figure 4. Pseudo-code for distributed evacuation CPS

Using DejaVu, we interactively programmed the evacuation system, testing and debugging our initial distributed navigation algorithm using virtual agents to react to virtual emergency fire evacuations. Using the replay and diffing features, we were able to observe the differences that occur between runs of the evacuation. For our analysis we focused on observing evacuation navigation between simulation runs, i.e., agent movement in reaction to the algorithm direction.

Upon adapting the fire evacuation algorithm, we are able to use DejaVu to visually observe how it affects evacuation navigation. Before using the algorithm, evacuees would simply navigate to the exit closest to them, signposted by static exit signs, as shown in figure 1a, unless they encounter a fire.

When the CPS is deployed, shown in figure 3a, evacuees are guided to the nearest safe exit, which may be longer than the shortest route, because of a fire hazard en route. Using DejaVu, the difference between these two scenarios is clear, in which the unaided scenario can be seen played out by the ghosts, whilst the current CPS-enhanced scenario avoids directing people towards the fire, using dynamic floor signs - pointing the people towards the safe route.

Sensors are represented as white cuboids mounted along the walls in figure 3a, detecting fire and relaying information between themselves according to the algorithm. The double coloured circles and lines signify radio traffic from one node to another, whilst the smaller coloured rings reflect the status LEDs of the node; these visual overlays expose more information typically hidden from sight in the real-world.

5 Performance

Along with providing visual and diffing benefits, DejaVu also needs to be able to run and record at real-time or faster, with a realistic number of devices, people and activity within the environment, in order for development time to be significantly reduced when compared to traditional methods, such as testbeds, simulations or pre-deployments.

For device simulation, Cooja was loaded with Cooja native motes, Contiki OS-based applications compiled to run natively on the host machine, which offer a significant speed boost of several orders of magnitude when compared to emulated motes at the cost of hardware accurate simulation.

Within the game engine, maintaining a high frames-per-second (FPS), above 30FPS, ensures a smooth and consistent physics simulation within the virtual world. On top of this when recording the simulation at faster than real-time, the FPS in the simulated world is reduced proportionally, to 1/x, where x is the speed up in the real-world. To ensure smooth playback and physics in the replay, a high FPS is needed e.g., 60FPS for x2 speed.

Performance tests were carried out with both UE4 and Cooja co-located on the same machine of the following spec: Xeon E5 1650 6Core with HT, 16GB RAM, 256GB SSD and a sufficiently powerful (MSI GeForce GTX 970) graphics card to support the game engine.

The performance test utilised the evacuation scenario described previously, with 30 sensors deployed and 60 evacuees. When run the simulation maintained an average of 75FPS with Cooja maintaining a constant x1.0 real-time factor; which demonstrates DejaVu can support realistically sized simulations with the use of large numbers of agents and devices simultaneously, whilst maintaining a consistently high FPS, ensuring fluid and efficient simulations.

To further improve simulation performance, simulations can be recorded at a lower visual fidelity, choosing a lower resolution, then increased when played back, improving the visualisation without impacting the recorded simulation.

6 Conclusion and Future Work

In this paper we presented DejaVu, a 3D co-simulator supporting full 3D reconstruction and visual diffing of live and recorded CPS deployment simulations. We demonstrated the significance and capabilities that visual diffing features provide through an evacuation case study. DejaVu enables developers to quickly analyse simulated runs, highlighting points of interest and differences between runs with full time-control, seamlessly within the simulation itself.

In future we envision extending the support for filtering and diffing, utilising event-stream processing to detect events and complex patterns of events. With the recent developments in virtual reality (VR) and mixed-reality (MR) technologies, VR and MR will provide new and exciting ways to support virtual testing and deployment, both in terms of the human-in-the-loop scenarios for virtual experiment participants, as well as for developers experimenting with deployment layout and live analytics superimposed onto objects within the real environment.

7 References

[1] C. E. Agero, N. Koenig, I. Chen, H. Boyer, and et al. Inside the virtual robotics challenge: Simulating real-time robotic disaster response. *IEEE Trans. on Automation Science and Engineering*, 12(2):494–506, April 2015.

[2] O. J. Akinwande, H. Bi, and E. Gelenbe. Managing crowds in hazards with dynamic grouping. *IEEE Access*, 3:1060–1070, 2015.

[3] S. Carpin, M. Lewis, J. Wang, S. Balakirsky, and C. Scrapper. Usarsim: a robot simulator for research and education. In *Robotics and Automation, IEEE International Conf.*, pages 1400–1405, April 2007.

[4] I. Chatzigiannakis, S. Fischer, C. Koninis, G. Mylonas, and D. Pfisterer. Wisebed: An open large-scale wireless sensor network testbed. In *Sensor Applications, Experimentation, and Logistics*, volume 29, pages 68–87. Springer Berlin Heidelberg, 2010.

[5] S. Dawson-Haggerty, A. Krioukov, J. Taneja, S. Karandikar, G. Fierro, N. Kitaev, and D. E. Culler. Boss: Building operating system services. In *NSDI*, volume 13, pages 443–458, 2013.

[6] N. Dimakis, A. Filippoupolitis, and E. Gelenbe. Distributed building evacuation simulator for smart emergency management. *The Computer Journal*, 53(9):1384, 2010.

[7] S. Fekete, A. Kroller, S. Fischer, and D. Pfisterer. Shawn: The fast, highly customizable sensor network simulator. In *Fourth International Conf. on Networked Sensing Systems*, 2007.

[8] S. Kartakis, E. Abraham, and J. A. McCann. Waterbox: A testbed for monitoring and controlling smart water networks. In *Proceedings of the 1st ACM International Workshop on Cyber-Physical Systems for Smart Water Networks*, CySWater'15, pages 8:1–8:6, New York, NY, USA, 2015. ACM.

[9] B. Kim, A. Jarandikar, J. Shum, S. Shiraishi, and M. Yamaura. The smt-based automatic road network generation in vehicle simulation environment. In *2016 International Conf. on Embedded Software (EMSOFT)*, pages 1–10, Oct 2016.

[10] N. Koenig and A. Howard. Design and use paradigms for gazebo, an open-source multi-robot simulator. In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, volume 3, pages 2149–2154 vol.3, Sept 2004.

[11] F. Leahy and N. Dulay. Ardán: Using 3d game engines in cyber-physical simulations (tool paper). In *Cyber Physical Systems. Design, Modeling, and Evaluation: 6th International Workshop, CyPhy, October 6, 2016*, pages 61–70. Springer International Publishing, 2017.

[12] P. Levis, N. Lee, M. Welsh, and D. Culler. Tossim: Accurate and scalable simulation of entire tinyos applications. In *Proceedings of the 1st International Conference on Embedded Networked Sensor Systems*, SenSys '03, pages 126–137, New York, NY, USA, 2003. ACM.

[13] A. Lscher, N. Tsiftes, T. Voigt, and V. Handziski. Efficient and flexible sensornet checkpointing. In B. Krishnamachari, A. Murphy, and N. Trigoni, editors, *Wireless Sensor Networks*, volume 8354, pages 50–65. Springer International Publishing, 2014.

[14] W. Mueller, M. Becker, A. Elfeky, and A. DiPasquale. Virtual prototyping of cyber-physical systems. In *Design Automation Conf. (ASP-DAC), 2012 17th Asia and South Pacific*, pages 219–226, Jan 2012.

[15] F. Österlind, A. Dunkels, J. Eriksson, N. Finne, and T. Voigt. Cross-level sensor network simulation with cooja. In *Local Computer Networks, Proceedings 2006 31st IEEE Conference on*, pages 641–648, Nov 2006.