ML4CREST: Machine Learning for CPS Models*

Stefan Klikovits
University of Geneva
Geneva, Switzerland
stefan.klikovits@unige.ch

Aurélien Coet
University of Geneva
Geneva, Switzerland
aurelien.coet@etu.unige.ch

Didier Buchs
University of Geneva
Geneva, Switzerland
didier.buchs@unige.ch

ABSTRACT

Models of small CPS and IoT applications often use approximated values that describe physical system behaviour. Physical resources, such as electricity consumption and heating power, have to be estimated, since many off-the-shelf components lack the required descriptions. Controllers which are based on these approximations can hence use imprecise models, perform misleading simulation, and cause damaged systems and financial loss. In this paper we present ML4CREST, a machine learning approach to automatically calibrate models using sensor measurements. We show that our approach is well-suited for the calibration of the flow rates within an automated watering system, which allows precise simulation and prevents spillage.

CCS CONCEPTS

• **Computing methodologies** → **Supervised learning**; *Continuous models*; • **Computer systems organization** → **Sensors and actuators**; • **Software and its engineering** → **Domain specific languages**;

KEYWORDS

cyber-physical systems, machine learning, physical system modeling, regression, resource flow

ACM Reference Format:
Stefan Klikovits, Aurélien Coet, and Didier Buchs. 2018. ML4CREST: Machine Learning for CPS Models. In *Proceedings of 2nd International Workshop on Model-Driven Engineering for the Internet-of-Things (MDE4IoT 2018).* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

The continuous advances of the Internet-of-Things (IoT) and cyber-physical systems (CPS) have large influences on every-day life. From "smart" devices such as media systems and CPS gadgets, to self-driving cars and automated health-monitoring applications, the positive impacts are undeniable.

Obviously, as we increasingly rely on these systems, asserting their correct functionality has become essential. The safety-critical systems domain has successfully developed and proved the efficacy of formal modelling techniques [5], model-driven engineering approaches [19] alongside with rigorous development and testing best-practices. These methods help preventing system failures, damage to wealth, health and human life. Their impact increases the trust in transport applications (e.g. trains, airplanes), medical devices and other applications.

The caveats of these powerful but complex approaches are their high financial and temporal requirements. As a result, less critical systems, such as small home or office automation systems, automated gardening applications and similar, are often not verified or only unsatisfyingly tested. Despite the low risk in health, these applications can still have an enormous impact on individual's life. For example, a misconfigured smart home might experience power outages when too many devices are started at the same time. Automated plant hydration systems can damage wooden floors, electrical appliances, etc. if they spill water or cause flowers and plants to die. To prevent negative financial and qualitative influences, it is necessary to prevent these kinds of hardships.

This means that non-expert CPS builders need simple and affordable modelling and simulation applications. But maybe even more importantly, they require support during the model creation process itself. Expressing physical influences between components is often non-trivial. For example, the illumination of an object (measured in lux) depends on the light emitted from the source (in lumen), the size of the illuminated surface, the distance between them, the source's illumination angle, possible reflections of other surfaces, etc. Such a calculation is highly complex and error prone, even to expert engineers and physicists. Comparatively, it is easy to install a lux meter and measure the value. Since in many small CPS systems approximated values are satisfactory, regressions over a set of such measurements can substitute the precise calculations. In home automation, these approximations are also often necessary to model off-the-shelf components, since many of them lack sufficient documentation and precise data-sheet descriptions.

This paper introduces a simple means to perform measurements and deduce resource transfers between components. Our methodology allows CPS creators to automatically learn about the influences between components within their system, so as to create a representative system model. The approach is based on an automated observation of the system using sensor readings and image recognition. This information is subsequently analysed using machine learning techniques that allow the determination of CPS component behaviour.

We validate our approach on an automated plant-growing system that we use as a case study. The system includes a small water tank that is used for distribution of water throughout the system. It is filled by a water pump, whose flow rate is unknown and which has to be discovered to prevent spillage and damages and also assert that the plants receive enough water.

*This project is supported by: FNRS STRATOS : Strategy based Term Rewriting for Analysis and Testing Of Software, the Hasler Foundation, 1604 CPS-Move, and COST IC1404: MPM4CPS

MDE4IoT 2018, October 2018, Copenhagen, Denmark

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM.
https://doi.org/10.1145/nnnnnnn.nnnnnnn

The acquired information is automatically added into a model of the system that we created with CREST [13], a domain-specific language (DSL) that offers a simple yet powerful means to simulate and verify systems.

The rest of this paper is organised as follows: Section 1 briefly introduces machine learning and outlines ML applications in the IoT/CPS domain. Section 2 provides fundamental information about the water tank case study and image recognition system we use. Section 3 discusses the data taking and machine learning within ML4CREST in detail. Section 4 elaborates on the integration of the regression results into the system model and provides simulation results. Section 5 concludes and gives an outlook of our next tasks.

1 RELATED WORKS

Machine learning (ML) [3] is the domain of automatically performing tasks such as regression and classification, by using statistical algorithms. To execute these actions, an initial set of data (a.k.a. the training set) is analysed and, based on the gathered knowledge, predictions about new, previously unknown data points are made. The "machine" aspect refers to the computer-driven calibration to a specific training set. The separation into different ML tasks depends on the individual applications. Classification is the process of assigning distinct labels to new data, e.g. the labelling of shapes as "circle", "triangle", "square", etc. Regression on the other hand aims to discover relationships between individual data features. The regression itself relies on well-known mathematical concepts such as interpolation [8], splines [6] and piecewise function fitting [9] to discover functions that best express variable relationships.

The CPS/IoT community experiences a strong push towards integrating machine learning in their systems. One of the main reasons is the large quantity of analysable data, stemming from low-cost sensors such as RFID tags, as well as the democratisation of increasingly powerful micro-computation nodes (e.g. Arduino [1], Raspberry Pi [21], etc.). System builders and process analysts must analyse and combine highly heterogeneous data sources [12] to extract summaries and trends, while still allowing for the action upon specific individual values. This means data sources are often spread across large geographic areas, producing data at different rates (several MHz to once per hour/day) and also different qualities. For some applications all individual data points are important, while for others some periodic value aggregations might suffice. Microsoft's Azure system [2] is a popular tool for this purpose. It allows the collection and aggregation of vast amounts of data and supports system analysts in their analysis by offering various machine learning tools.

On the other hand, even small farming applications and similar systems use ML to estimate the soil moisture and presence of minerals through soil measurements. Wine producers create wide-area sensor networks using IoT devices [17] and environment data, so as to create models of their environment and to improve the quality of their product [20]. Recently, automated discovery of a plant health through image recognition [18] and similar approaches aim to help farmers maximise their harvest.

Machine learning has also been successfully applied to home automation sensors. Devices that perform "classic" ML tasks such as speech recognition [11] and temperature regulation [16] are widely available. Recent research however aims at combining multi-sensor measurements for more complex analyses. A highly challenging application of this technique is the human activity recognition [22], where various sensors are used to detect whether a person is walking, cycling, drinking coffee or watching TV. The Synthetic Sensors project [15] on the other hand combines a dozen sensors and applies ML techniques to identify burning stoves, running water faucets and distinguish various electrical appliances within a home.

So far, most ML applications in the CPS field aim either at the analysis of large data quantities or the discovery of low-level systems. They however omit one important aspect in CPS, namely the calibration of custom systems. While the discovery of a running water faucet is useful, there is no information about the tap's flow rate. As each system is different, a calibration of influences is highly important, in order to analyse system behaviour and predict its evolution.

2 CASE STUDY - PRELIMINARIES

ML4CREST is a machine learning approach that integrates the automatic discovery of physical systems into CREST[1], a domain-specific modelling language. In this section we elaborate on the foundations upon which we built ML4CREST.

2.1 Water tank case study

In our research we use a scenario taken from an automated plant growing system. This system is composed of two equivalent plant watering subsystems, which each feature a small water reservoir (a two-litre soda bottle) that slowly distributes water to several plants using pipes. The water flow rate can be adjusted for each plant individually, the total outflow of the tank is the sum of the individual flow rates. Figure 1 shows a schematic representation of one such plant watering system.

At regular intervals a relay activates a small water pump that fills the reservoir. The flow rate of the water pump varies depending on the pumping height, and is generally also unknown. Additionally, the filling of the water tank can be slightly delayed due to the length of the inflow hose.

The purpose of our research is to create a model of the two watering subsystems and assert that there is no overflow of the water reservoirs. Hence, it is necessary to calibrate the model according to the actual in- and outflow rates. As the values are unknown, we will apply machine learning techniques based on sample measurements to discover these parameters.

2.2 Image Recognition

In order to learn about the inflow and outflow of the water tanks, we observe the changes in volume over time when pumping water in and out of the system. To that end, we installed a camera to take photos of the water tanks in small time intervals (e.g. every second) and used it to determine the current water level. The detailed methodology of the learning approach is described in the next section. The rest of this section outlines our method to discover the water level within the tanks.

The observation of the current water level within a transparent container is non-trivial and such image recognition approaches

[1] https://github.com/stklik/CREST

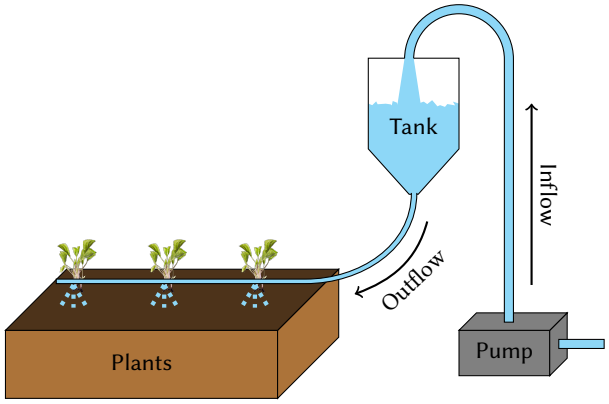


Figure 1: The schema of the plant watering system: A pump fills the water tank from which small pipes are used to transport the water to the individual plants.

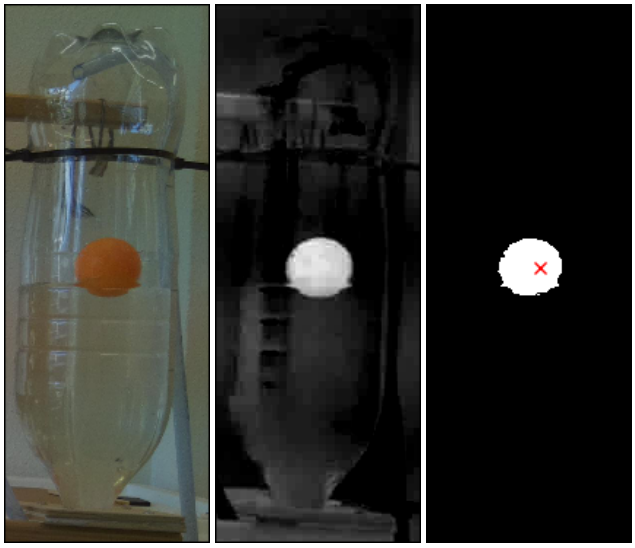
require careful calibration [7]. Instead of trying to discover the water level directly, we opted to place orange ping-pong balls into the tanks. The balls float on the surface and thereby indicate the water level within the reservoir. Our tool automatically takes photos of the entire plant growing system. These photos are cropped, so only the water tank itself is visible (see Figure 2a).

To detect the ball position, an image segmentation algorithm is applied. We use the knowledge that in the RGB spectrum, orange colour shades possess high red values, and low green ones. This means that orange pixels can be discovered by subtracting each pixel's green value from its red value. The resulting image is greyscale, with light colours where red and orange shades were found and darker ones for other colours, as shown in Figure 2b. This greyscale image is then binarised to create a black & white mask. It means that, given a (predefined) darkness-threshold, all pixels that are darker than the threshold are turned black, while lighter ones are coloured in white. One such image mask is shown in Figure 2c. Finally, we calculate the median of the white pixels to get the ball's centre position. Our algorithm uses the median rather than the mean as the binarisation process might set other pixels (not belonging to the ball) to white, thereby creating noise. This occurs occasionally due to light reflections or noisy camera images. Calculation of mean values would create significantly tilted measurements, especially if the noise is distant from the actual ball position. The median value proved to be more robust measure in this sense.

Once the position of the centre of the ball is known, its vertical coordinate can be used as an indicator of the water level in the cropped image.

The image recognition algorithm, including example photos of the system, is provided online in the `ImageRecognition.ipynb`[2] Jupyter[3] notebook. The notebook can be used to reproduce our results.

[2]https://mybinder.org/v2/gh/stklik/CREST/MDE4IOT?filepath=ImageRecognition.ipynb
[3]https://jupyter.org/



(a) Cropped (b) Filtered (c) Binarised

Figure 2: Image recognition process: (a) the base image (not shown) is cropped; (b) subtract of green-filter from red-filter; (c) Binarise image and search median

3 ML4CREST

The goal in our case study is to automatically learn about incoming and outgoing water flow rates of two water tanks. The volumetric flow rate is a measure of the volume of fluid that passes through a surface (e.g. a cross section of a pipe) per time unit. It is defined by the limit

$$Q = \lim_{\Delta t \to 0} \frac{\Delta V}{\Delta t} = \frac{dV}{dt}$$

where V is the volume of some fluid, and t is the time. In our case, we are interested in the amount of water entering and leaving the tank, i.e. the volume pumped through the hose and the volume exiting through the hose to the plants.

As shown in the formula above, the flow rate is the derivative of the volume over time. To compute it, it is necessary to measure the volume of water and its change over time when the tanks are filled by the pumps and then emptied on the plants. The pump activation automatically triggers a script which takes pictures of the system (one picture per second for 15 minutes). This way, it is possible to observe the tank's water level at regular times during the filling and emptying phases.

3.1 Volume measurement

In order to measure the water volume in the tank at a given time, it is necessary to discover the function that maps the ball's y-coordinates (as obtained by the image recognition) to the corresponding volume and then interpolate the behaviour. This calibration was manually performed by taking photos at known volume levels (0 to 1.45 litres in 0.05 litre steps) and extracting the ball's y-coordinates (in pixels) from these photos. We took three photos for each volume, asserting that the ball would be at different positions within the

tank, and calculated the mean values. The reason for the multiple measurements is that, as the camera is not always on level with the ball, distortions by the water surface can occur when the ball is higher than the camera, leading to minor measurement errors.

Figure 3 shows the measurement averages per volume as points. Based on these mean measurements a *polynomial* regression was performed, resulting in the function traced in the figure. This function can be used to approximate the volume of water within the system.

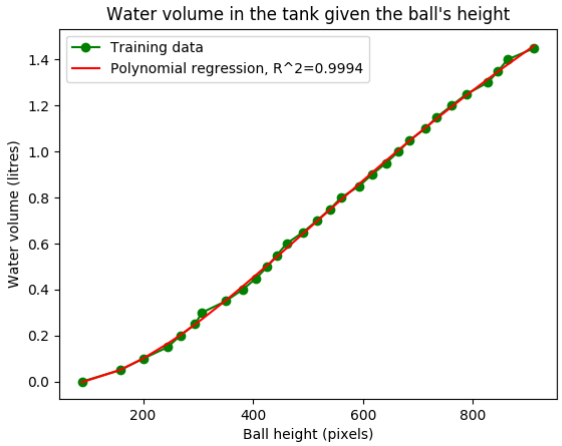


Figure 3: Regression over water volume measurements. Note, how in the beginning the ball height grows quicker than the volume due to the non-cylindrical shape of the soda-bottle.

3.2 Calculation of Inflow and Outflow

Using the calibrated volume function, it is possible to measure the volume in regular intervals during the filling and emptying phases. We chose to analyse the volume every second during these phases to obtain very precise measurements. This granularity was precise enough for our purposes. For systems with slower change rates however, less frequent measurements could be used.

We use the obtained data to deduce the filling and outflow rates as follows: Once the photos have been taken, the images are analysed as described in Section 2.2 to extract the water level at every time step. The resulting pixel measurements are mapped onto the corresponding water volume using the volume regression function described above. Figure 4 shows a graph of the measurements at their corresponding times.

It is clearly visible that during the filling phase, the volume rises continuously, and that it starts dropping when the pump is turned off after 58 seconds. Based on this information, the measurements are split into two parts: filling and emptying. As one can see in Figure 4, the inflow measurement grows almost linearly. We use the *scikit-learn* machine learning library to perform a linear regression on the inflow data.

The linear regression on the measurements of the emptying phase resulted in unsatisfactory results, especially towards the

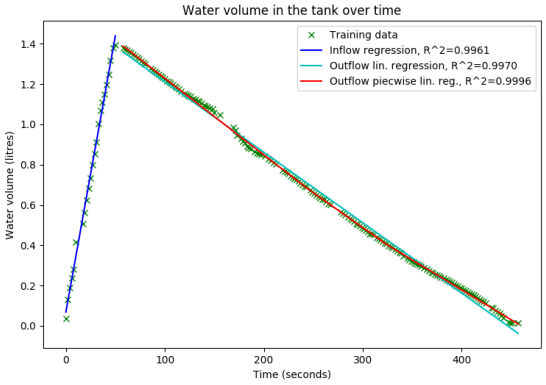


Figure 4: Water volume measurements at regular points in time during the filling and emptying phases of the tank with higher outflow.

beginning and end of the measurement phase. This is due to the fact that the outflow rate is slowing down over time as the water pressure decreases. We use `pwlf`[4], a piecewise linear regression library, to obtain a better approximation. Figure 4 displays the linear and piecewise linear regressions, and that the latter performs significantly better.

Finally, the incoming and outgoing volumetric flow rates of the water tank are obtained by reading the slopes of the regressions. So as to generalise this result, the flow rate will be given for volume levels, rather than for time spans. This is achieved by mapping the time-ranges of the piecewise approximation to the according volume level. Table 1 displays the flowrates within the water tank system.

Inflow rate	27.432 ml/s
Outflow rate (1.389 - 0.678 L)	3.801 ml/s
Outflow rate (0.678 - 0.405 L)	3.467 ml/s
Outflow rate (0.405 - 0.000 L)	2.972 ml/s

Table 1: The inflow and outflow rates of the water tank. The outflow rates vary with changing volume, which is provided in parenthesis.

The raw data as well as the regression approaches are provided as open data in the `FlowComputation.ipynb` Jupyter notebook. It can be accessed and executed online[5].

3.3 Second watering system

The plant growing system hosts another watering system, with fewer but larger plants. Here, the flow rate of the hoses leading from the tank to the plants can be higher, as the higher root density allows for plants to absorb water more quickly.

Figure 5 shows the measurements and regressions for this second system. Note that, due to the longer hose between the pump and

[4]https://github.com/cjekel/piecewise_linear_fit_py
[5]https://mybinder.org/v2/gh/stklik/CREST/MDE4IOT

the water tank, there is a delay of around one second until the linear filling starts, which has to be taken into account when modelling the system. It is also evident that the timing of this tank is very different to the previous one. Due to the higher outflow rate, the filling phase is slightly longer (64 seconds instead of 58) and the emptying phase is quicker.

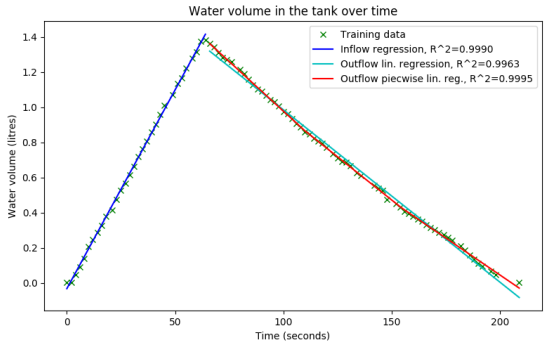


Figure 5: Water volume measurements at regular points and regression of the second watering tank.

We again chose a linear regression for the filling phase and a piecewise linear regression for the emptying phase. The calculations resulted in the flow rates shown in Table 2.

Inflow rate	22.628 ml/s
Outflow rate (1.359 - 0.718 L)	11.077 ml/s
Outflow rate (0.718 - 0.021 L)	9.045 ml/s
Outflow rate (0.021 - 0.000 L)	2.07 ml/s

Table 2: The inflow and outflow rates of the second water tank. The outflow rates vary with changing volume, which is provided in parenthesis.

4 SIMULATION AND DISCUSSION

The obtained measurements and regressions are verified by using the values in a simulation and comparing the simulation data to the actual system behaviour. For the simulation we chose to create a model of the water tank system in the CREST language [13]. CREST is an *internal* domain-specific language (DSL) that uses Python as host language. It focuses on the modelling of resource flow (e.g. water, electricity) throughout a CPS and has a formal basis [14]. An introduction to CREST, it's syntax, semantics and simulation capabilities can be found in the project repository[6].

The pump and water tank were modelled as finite-state machines, which express continuous behaviour that is associated with particular states (similar to hybrid automata [10]). The watering system integrates both the tank and pump as subcomponents and connects the pump's output to the tank's inflow port. The system has only one input port, namely the pump's on-off switch. The water tank itself has five states: empty, full and one for each volume level in between to model its associated outflow rate. For spatial reasons we will not further discuss the model itself, but refer the reader to the online repository[7] for a more detailed description and visualisation of the CREST model and the simulation.

The model is calibrated using the values that were calculated in the previous section. Using this model, we simulated the watering system during several phases where the pump was repeatedly turned on and off. The simulator ran an experiment of 120 seconds where the pump was alternately turned on for ten seconds, then turned off for 30 seconds. In parallel, we executed the same process on the real watering system using an automated script. Our image recognition tools took photos every two seconds to trace the evolution of the water level within the tank. Figure 6 shows the resulting simulation and the measurement points within the test period.

As we can see, the simulation and measurements are very close for the first 120 seconds. It is however also visible that, during the outflow phase, the measurements and simulation predictions begin to diverge. We attribute this offset to the combination of small measurement errors and offsets during the image recognition phase. For our purposes, namely the calculation of flow rates to assert that plants receive the right amount of water, this poses only a minor nuisance. For other, more sensitive applications however, a more precise image recognition setup should be used.

Despite that, our results are promising and show that simple, home-made systems using off-the-shelf components (in our case a Raspberry Pi, Pi Camera board and a USB-connected relay) can be used to drive an automated plant growing setup. The available tools and software libraries allow for easy integration of image recognition and machine learning into the our system model.

5 FUTURE WORKS AND CONCLUSION

This paper presents ML4CREST, a novel approach to calibrate models of small cyber-physical systems such as home- and office automation applications. ML4CREST employs machine learning algorithms on top of automatically gathered data to deduce influences between CPS components. An automated plant watering system within a "smart" gardening application is used to describe our approach. In this system the (previously unknown) inflow and outflow rates of a water tank are deduced using open-source image recognition and machine learning software libraries. We present the implementation of the data analysis workflow and the connection with CREST, a modelling language for CPS.

Given the initial success of our approach, we plan to extend ML4CREST in several directions. The automated plant growing system we set up provides many areas in which automated learning and calibration might be used. For example, we plan to model the soil of our system to learn about the amount of water the soil can hold and the evaporation rate at different temperature and light conditions. A more ambitious goal is to create plant growth models by measuring parameters such as illumination, temperature, soil moisture and plant age.

To support these modelling tasks, we intend to further automate the data analysis and machine learning part and provide support

[6] https://github.com/stklik/CREST

[7] https://mybinder.org/v2/gh/stklik/CREST/MDE4IOT?filepath=Watertank.ipynb

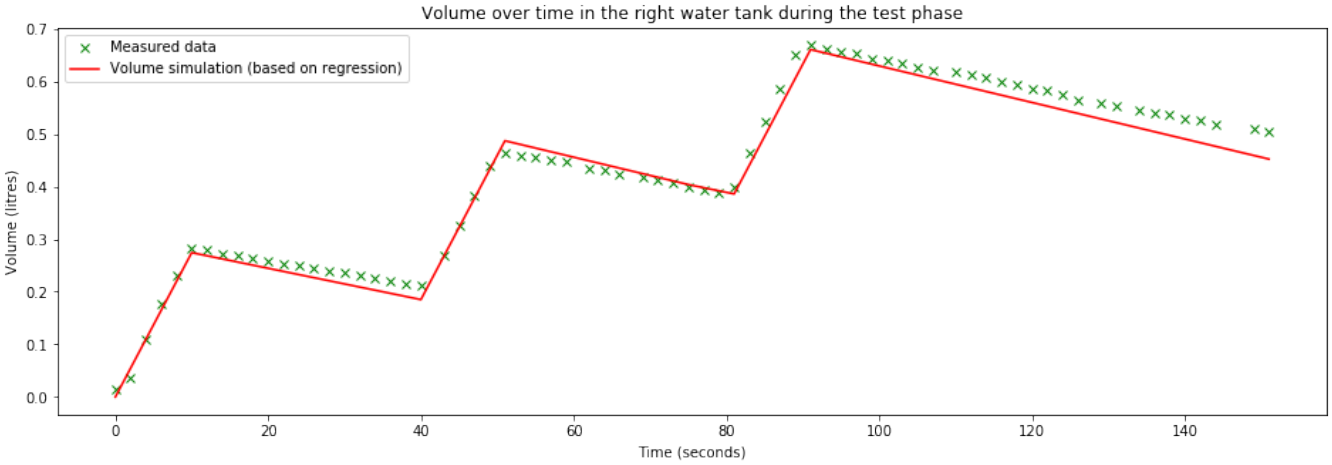


Figure 6: Comparison of simulation and actual measurements of the water tank system. The values correlate with $R^2 = 0.9815$

for more ML-algorithms and automated selection of the best performing one.

Additionally, we want to improve usability by replacing the static volume detection system where the tanks' image recognition parameters are manually defined with an automated system that identifies the tank by itself using automated object detection. We further hope to replace our fixed camera setup with e.g. a smartphone camera and advanced image recognition, to make the calibration task as user-friendly as possible.

Lastly, we plan to use the machine learning techniques in another CPS domain. We observed that in complex CPS modelling, behaviour is sometimes expressed using highly complex code constructs, proprietary libraries or hidden code (e.g. using Functional Mockup Interface [4]). Hence, it is often infeasible to efficiently simulate systems that use such non-analysable functions. By executing these implementations using equivalence classes or symbolic execution, it is possible to gather analysable data and to apply machine learning techniques in order to approximate the actual implementations.

REFERENCES

[1] Massimo Banzi. 2008. *Getting Started with Arduino.* Make Books - Imprint of: O'Reilly Media, Sebastopol, CA.

[2] Roger Barga, Valentine Fontama, and Wee Hyong Tok. 2015. Introducing Microsoft Azure Machine Learning. In *Predictive Analytics with Microsoft Azure Machine Learning*. Springer, 21–43.

[3] Christopher M. Bishop. 2006. *Pattern Recognition and Machine Learning (Information Science and Statistics).* Springer-Verlag, Berlin, Heidelberg.

[4] Torsten Blochwitz, Martin Otter, Johan ÃĚkesson, Martin Arnold, Christoph Clauss, Hilding Elmqvist, Markus Friedrich, Andreas Junghanns, Jakob Mauss, Dietmar Neumerkel, Hans Olsson, and Antoine Viel. 2012. Functional Mockup Interface 2.0: The Standard for Tool independent Exchange of Simulation Models. In *Proceedings of the 9th International Modelica Conference*. The Modelica Association, 173–184. http://dx.doi.org/10.3384/ecp12076173

[5] Edmund M. Clarke and Jeannette M. Wing. 1996. Formal Methods: State of the Art and Future Directions. *ACM Comput. Surv.* 28, 4 (Dec. 1996), 626–643. https://doi.org/10.1145/242223.242257

[6] Carl De Boor. 2001. *A practical guide to splines.* Springer, Berlin.

[7] Sagi Eppel and Tal Kachman. 2014. Computer vision-based recognition of liquid surfaces and phase boundaries in transparent vessels, with emphasis on chemistry applications. *CoRR* abs/1404.7174 (2014). arXiv:1404.7174

[8] Mariano Gasca and Thomas Sauer. 2000. On the History of Multivariate Polynomial Interpolation. *J. Comput. Appl. Math.* 122, 1-2 (2000), 13. https://doi.org/10.1016/S0377-0427(00)00353-8

[9] Bernd Hamann and Jiann-Liang Chen. 1994. Data point selection for piecewise linear curve approximation. *Computer Aided Geometric Design* 11, 3 (1994), 289 – 301. https://doi.org/10.1016/0167-8396(94)90004-3

[10] T. A. Henzinger. 1996. The Theory of Hybrid Automata. In *Proceedings of the 11th Annual IEEE Symposium on Logic in Computer Science (LICS '96).* IEEE Computer Society, Washington, DC, USA, 278–. http://dl.acm.org/citation.cfm?id=788018.788803

[11] Julia Hirschberg and Christopher D. Manning. 2015. Advances in Natural Language Processing. *Science* 349, 6245 (2015), 261–266. https://doi.org/10/f7kfrk arXiv:http://science.sciencemag.org/content/349/6245/261.full.pdf

[12] Aqeel Kazmi, Zeeshan Jan, Achille Zappa, and Martin Serrano. 2017. Overcoming the Heterogeneity in the Internet of Things for Smart Cities. In *Interoperability and Open-Source Solutions for the Internet of Things*, Ivana Podnar Žarko, Arne Broering, Sergios Soursos, and Martin Serrano (Eds.). Vol. 10218. Springer International Publishing, Cham, 20–35. https://doi.org/10.1007/978-3-319-56877-5_2

[13] Stefan Klikovits, Alban Linard, and Didier Buchs. 2017. CREST - A Continuous, REactive SysTems DSL. In *Proceedings of MODELS 2017 Satellite Event: Workshops (ModComp, ME, EXE, COMMitMDE, MRT, MULTI, GEMOC, MoDeVVa, MDETools, FlexMDE, MDEbug), Posters, Doctoral Symposium, Educator Symposium, ACM Student Research Competition, and Tools and Demonstrations co-located with ACM/IEEE 20th International Conference on Model Driven Engineering Languages and Systems (MODELS 2017), Austin, TX, USA, September, 17, 2017.* 286–291. http://ceur-ws.org/Vol-2019/gemoc_2.pdf

[14] Stefan Klikovits, Alban Linard, and Didier Buchs. 2018. *CREST Formalization.* Technical Report. Software Modeling and Verification Group, University of Geneva. https://doi.org/10.5281/zenodo.1284561

[15] Gierad Laput, Yang Zhang, and Chris Harrison. 2017. Synthetic Sensors: Towards General-Purpose Sensing. ACM Press, 3986–3999. https://doi.org/10.1145/3025453.3025773

[16] Qiang Liu, Yujun Ma, Musaed Alhussein, Yin Zhang, and Limei Peng. 2016. Green Data Center with IoT Sensing and Cloud-Assisted Smart Temperature Control System. *Computer Networks* 101 (June 2016), 104–112. https://doi.org/10/f8p6jr

[17] Michal Lagan. 2018. SMAVIN – the IoT sensors network for winers and thought leaders (product presentation). http://www.smavin.com/docs/smavin_smart_sensors_for_vineyards_0607.pdf.

[18] Hanno Scharr, Tony P. Pridmore, and Sotirios A. Tsaftaris. 2017. Editorial: Computer Vision Problems in Plant Phenotyping, CVPPP 2017 – Introduction to the CVPPP 2017 Workshop Papers. In *The IEEE International Conference on Computer Vision (ICCV) Workshops.*

[19] Douglas C. Schmidt. 2006. Model-Driven Engineering. *IEEE Computer* 39, 2 (February 2006). http://www.truststc.org/pubs/30.html

[20] Anna Solana. 2018. Internet of Wines: How this vineyard's smart sensors improve the vintage in your glass. *ZDNet* (2018). https://www.zdnet.com/article/internet-of-wines-how-this-vineyards-iot-improves-vintage-in-your-glass/

[21] Eben Upton and Gareth Halfacree. 2012. *Raspberry Pi User Guide* (1st ed.). Wiley Publishing.

[22] Jindong Wang, Yiqiang Chen, Shuji Hao, Xiaohui Peng, and Lisha Hu. 2018. Deep learning for sensor-based activity recognition: A Survey. *Pattern Recognition Letters* (2018). https://doi.org/10.1016/j.patrec.2018.02.010