Bringing A Robot Simulator to the SCAMP Vision System

Yanan Liu
Bristol Robotics Laboratory
University of Bristol
Bristol, UK
yanan.liu@bristol.ac.uk

Jianing Chen
School of Electrical & Electronic Engineering
University of Manchester
Manchester, UK
jianing.chen@manchester.ac.uk

Laurie Bose
Visual Information Labratory
University of Bristol
Bristol, UK
lauriebose@gmail.com

Piotr Dudek
School of Electrical & Electronic Engineering
University of Manchester
Manchester, UK
p.dudek@manchester.ac.uk

Walterio Mayol-Cuevas
Visual Information Laboratory
University of Bristol
Bristol, UK
walterio.mayol-cuevas@bristol.ac.uk

***Abstract*—This work develops and demonstrates the integration of the SCAMP-5d vision system into the CoppeliaSim robot simulator, creating a semi-simulated environment. By configuring a camera in the simulator and setting up communication with the SCAMP Python host through remote API, sensor images from the simulator can be transferred to the SCAMP vision sensor, where on sensor image processing such as CNN inference can be performed. SCAMP output is then fed back into CoppeliaSim. This proposed platform integration enables rapid prototyping validations of SCAMP algorithms for robotic systems. We demonstrate a rover localisation and tracking task using this proposed semi-simulated platform, with a CNN inference on SCAMP to command the motion of a robot. We made this platform available online.**

***Index Terms*—CoppeliaSim, Semi-simulation, Pixel Processor Array, CNN inference, in-sensor computing**

I. Introduction

The SCAMP vision system [1] is a smart camera device supporting in-sensor processing. The mixed-signal (analogue and digital) general-purpose processing circuits, integrated with image sensors in a pixel processor array (PPA), enable low-power consumption, and efficient parallel computing without external hardware. With these features, it is increasingly being integrated with robots for various applications [2]–[6]. However, it is often time-consuming and difficult to prototype ideas using real robotic platforms. To improve experimental flexibility, we integrate a comprehensive robot simulator CoppeliaSim [7] with the SCAMP hardware system, to test and validate ideas rapidly (Fig. 1). CoppeliaSim is a robot environment simulator where each agent can be controlled via remote API [8]. Its simulated sensor readings can be transferred to other independent platforms written in Python, C/C++, or Matlab through several communication protocols.

This work is accepted by ICRA2021 workshop On and Near-sensor Vision Processing, from Photons to Applications (ONSVP). This work was supported by UK EPSRC EP/M019454/1, EP/M019284/1, EPSRC Centre for Doctoral Training in Future Autonomous and Robotic Systems: FARSCOPE and China Scholarship Council (No. 201700260083).

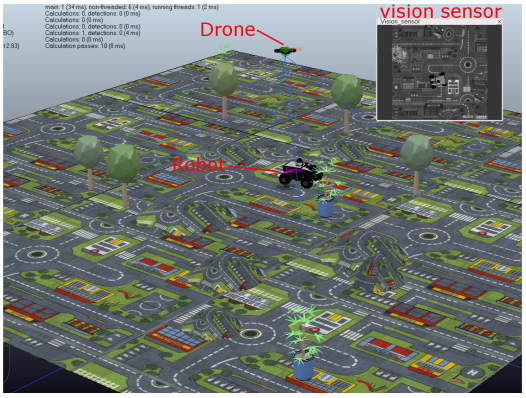

Fig. 1. Example robot simulation environment. The virtual SCAMP camera sensor is 'mounted' under the drone facing the ground. Real-time image can be obtained from the sensor with a resolution of 256×256 which is set the same as that of the SCAMP hardware. Note that the current version of SCAMP-5d only supports gray-scale images; hence CNN inference on SCAMP only relies on gray-level features from the scene.

We developed a Python based interface between CoppeliaSim and the SCAMP vision system. Based on the proposed semi-simulation platform, we implemented a convolutional neural network (CNN) [9], [10] on the SCAMP, processing the imported camera images for localisation purpose from the robot simulator where the camera is mounted under a drone.

II. SCAMP Python Host and the Semi-Simulated Platform

The SCAMP vision system is connected to a computer via USB through scamp5d_interface library[1]. SCAMP Host is a GUI executed on the computer to interact with the vision system and visualise the data sent back from the device. This work develops a scamp_python_module for the Python GUI based on previous C/C++ host libraries, to simplify the

[1] https://scamp.gitlab.io/scamp5d_doc/

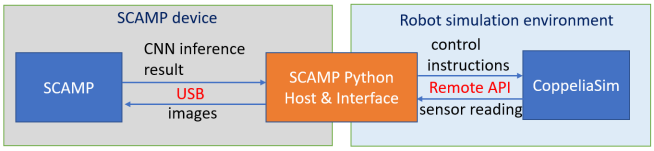

Fig. 2. Semi-simulated platform by integrating the SCAMP with CoppeliaSim Robot Simulator. This platform takes advantage of the SCAMP parallel computation ability and the rich simulation scenes in the simulator, where applications can be exploited and validated virtually.

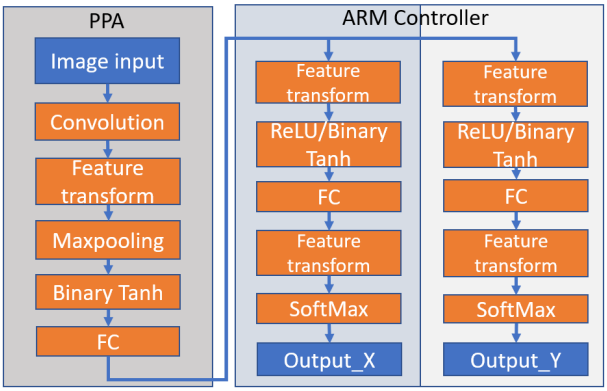

Fig. 3. The CNN architecture for localisation. A shared convolutional layer is used for object 2D localisation on the SCAMP, where networks for labels indicating x and y are using identical convolutional layers but different fully-connected layers.

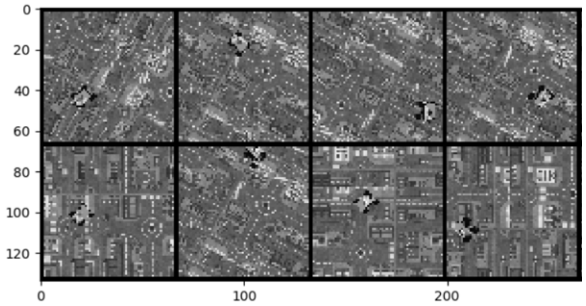

Fig. 4. Selected images for training. The inputs for SCAMP are gray-scale images with a low resolution of 64×64 considering the network architecture for parallel computation purpose, resulting in a challenging localisation task.

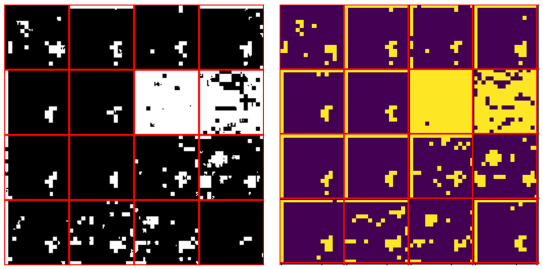

Fig. 5. Binary activations comparison after image convolution on SCAMP and using PyTorch simulation. White and yellow dots represent '1's while the dark area is '-1's. This shows the similarity of binary activations after the first convolutional layer, but with differences introduced due to non-idealities of analogue computing in hardware

connection of the SCAMP host to third party software. With this method, the host visualisation and remote API can be co-designed on the scamp_python_module. The remote API is supported by the CoppeliaSim[2].

We demonstrate an application of a CNN-based vision task performed in our environment. In the semi-simulated platform, a real (hardware) SCAMP vision system collaborates with CoppeliaSim robotics simulation environment through the remote API. The environment setup and sensor image collection is performed in the simulator while the SCAMP hardware is in charge of CNN inference with sensor images from the simulator and outputting useful information to the simulator (Fig. 2).

III. Experiments on Platform

The SCAMP vision system is suitable for mobile robot platforms due to its combination of high performance, low power consumption and light weight. In the experiments presented in this paper, the SCAMP system is mounted on an aerial vehicle (drone), and used to localise a mobile ground vehicle (rover) moving in the 2D simulated environment. The location information is used to guide the drone to track the rover. We implement the localisation task using a neural network.

1) Localisation Dataset: The localisation training dataset is collected from the simulation environment, by placing the rover at a series of positions within the map, with different orientations under the view-field of a camera. An image is recorded once there is a change in the rover pose or camera pose. With this method, a dataset of 104,000 training images and 19,200 testing images was collected (Fig. 4). To simulate the vibration and tilting of a flying drone, random noise is introduced into the camera pose, and this can also be regarded as a type of data augmentation that benefits CNN training.

2) CNN inference for rover localisation: The localisation task is cast as a classification problem by splitting the x and the y axes into eight labels, giving 64 possible positions to localise the rover. During the CNN training, the training loss for backpropagation is the loss summation of x and y as shown in Fig. 3. We trained a binarized CNN using batch normalisation and using both binary weights and activations to reduce the error caused by continuous analogue electronic current computing. The final testing accuracy for localisation is around 93%, which conservatively, only counts the correct predictions of the CNN. In a practical situation, a close prediction to the ground truth should still allow tracking to proceed without significant difficulty. Fig. 7 visualises a sequence of 8 frames with CNN inference on SCAMP where the prediction possibility distribution can be seen plotted as light blue curves along x and y axes, the final prediction is obtained with two highest possibilities from two curves along axes. The complete localisation and tracking are processed frame by frame, and the instructions to pilot the drone are

[2] https://www.coppeliarobotics.com/helpFiles/en/b0RemoteApiOverview.htm

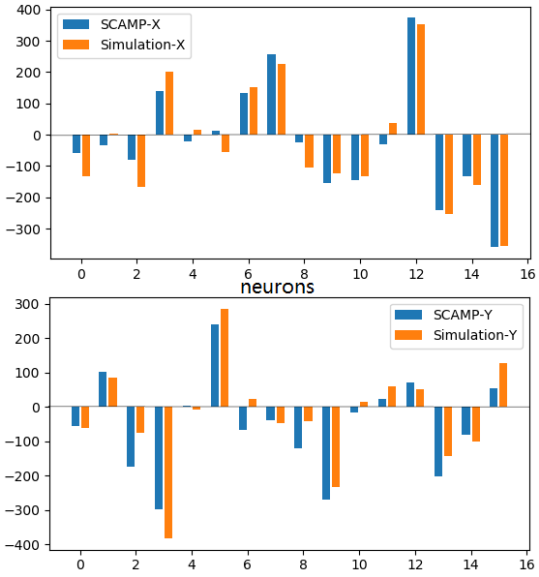

Fig. 6. Neuron activations after the first fully connected computed using SCAMP hardware and PyTorch simulation. Binary activations and bit counting strategies are used to mitigate against the inherent noise of the analogue processing on SCAMP, resulting in similar activation values in fully-simulated and in hardware implementations.

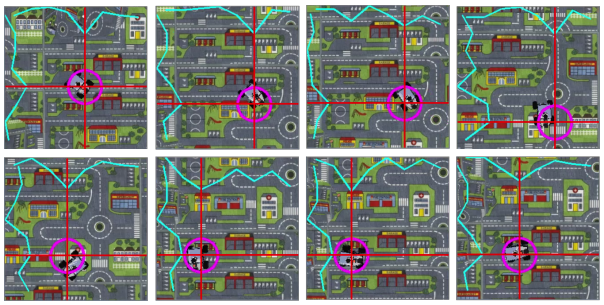

Fig. 7. Rover localisation result visualisation, showing SCAMP CNN inference results for several consecutive frames. The network outputs are plotted as light blue curves along x and y axes, with the largest values for each axis (red straight line) indicating the final 2D localisation prediction (pink circle). The full experimental video for rover localisation and tracking can be seen from *https://youtu.be/semthdfXH5A*.

generated using the PID control to minimise the distance between the drone position and the predicted rover position.

To demonstrate the CNN inference on SCAMP in terms of accuracy, Fig. 5 and Fig. 6 compare the binary activation layer and first fully-connected layer between PyTorch software implementation and SCAMP hardware implementation, which shows a high degree of similarity between PyTorch and SCAMP results, but with differences resulting from analogue signal processing on SCAMP. To further validate the performance of CNN localisation on SCAMP, a chaotic trajectory is pre-set in the simulator for the rover to move along. The drone trajectory is plotted with guidance from SCAMP CNN inference. Fig. 8 shows the comparison among the ground truth rover course, CNN on PyTorch guided drone course and CNN on SCAMP guided drone course. Again,

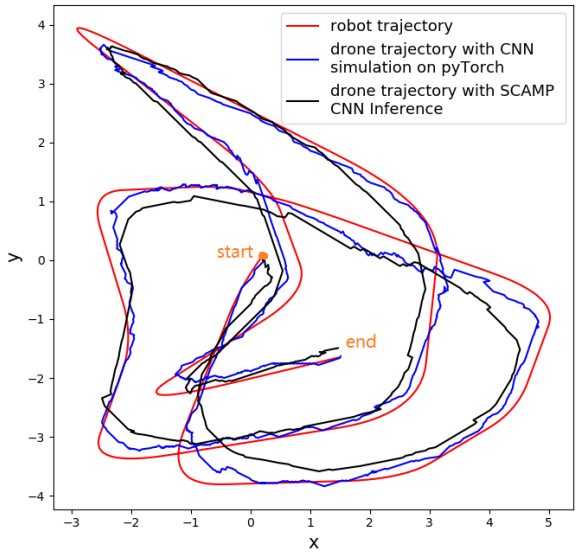

Fig. 8. Tracking trajectories with a drone. There are three paths: the pre-set rover trajectory as the groundtruth, drone tracking trajectory with CNN on PyTorch guidance, and drone tracking trajectory with guidance from SCAMP CNN inference.

the fully simulated results are similar, but not identical to the semi-simulated results using SCAMP hardware. These results indicate that the methodology based on training neural networks in software, using established frameworks such a PyTorch, and then implementing them on SCAMP analogue hardware, can produce useful results. However, software and hardware performance are not identical since the analogue hardware effects can not always be accurately modelled. It is useful to be able to perform hardware-in-the-loop simulations combining the virtual environments and the SCAMP hardware, as presented in this paper. We make our SCAMP Python host, CoppeliaSim model and its configuration available online: *https://github.com/yananliusdu/scamp5d_interface*.

IV. Conclusion

In this work, we proposed a semi-simulated platform where a SCAMP hardware interacts with the robot simulator via remote API for a rapid prototype validation. The SCAMP CNN inference results with the simulated sensor readings can instruct the motion of an agent in the proposed platform. Further applications related to the SCAMP and robots integration can be easily explored based on the developed platform.

References

[1] J. Chen, S. J. Carey, and P. Dudek, "Scamp5d vision system and development framework," in *Proceedings of the 12th International Conference on Distributed Smart Cameras*, 2018, pp. 1–2.

[2] J. Chen, Y. Liu, S. J. Carey, and P. Dudek, "Proximity estimation using vision features computed on sensor," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 2689–2695.

[3] Y. Liu, L. Bose, C. Greatwood, J. Chen, R. Fan, T. Richardson, S. J. Carey, P. Dudek, and W. Mayol-Cuevas, "Agile reactive navigation for a non-holonomic mobile robot using a pixel processor array," *IET Image Processing*, 2021. [Online]. Available: https://ietresearch.onlinelibrary.wiley.com/doi/abs/10.1049/ipr2.12158

[4] C. Greatwood, L. Bose, T. Richardson, W. Mayol-Cuevas, J. Chen, S. J. Carey, and P. Dudek, "Tracking control of a uav with a parallel visual processor," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017, pp. 4248–4254.

[5] A. McConville, L. Bose, R. Clarke, W. Mayol-Cuevas, J. Chen, C. Greatwood, S. Carey, P. Dudek, and T. Richardson, "Visual odometry using pixel processor arrays for unmanned aerial systems in gps denied environments," *Frontiers in Robotics and AI*, vol. 7, 2020.

[6] H. Castillo-Elizalde, Y. Liu, L. Bose, and W. Mayol-Cuevas, "Weighted node mapping and localisation on a pixel processor array," in *IEEE International Conference on Robotics and Automation (ICRA), Xi'an, China*, May, 2021.

[7] E. Rohmer, S. P. Singh, and M. Freese, "V-rep: A versatile and scalable robot simulation framework," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2013, pp. 1321–1326.

[8] S. James, M. Freese, and A. J. Davison, "Pyrep: Bringing v-rep to deep robot learning," *arXiv preprint arXiv:1906.11176*, 2019.

[9] Y. Liu, L. Bose, J. Chen, S. J. Carey, P. Dudek, and W. Mayol-Cuevas, "High-speed light-weight cnn inference via strided convolutions on a pixel processor array," in *The 31st British Machine Vision Conference (BMVC 2020)*, 2020.

[10] L. Bose, P. Dudek, J. Chen, S. J. Carey, and W. W. Mayol-Cuevas, "Fully embedding fast convolutional networks on pixel processor arrays," in *European Conference on Computer Vision – ECCV 2020*.