Università della Svizzera italiana

Faculty of Informatics

Learning angular velocity with a CNN regressor

Noli Manzoni, Michael Denzler

May 26, 2020

Source code: github.com/raikilon/cnn-angular-velocity
Models and Datasets: mega.nz/folder/FhQjVQhB#WGYx3LL-L5fwcznx5PM3tw
Video presentation: mega.nz/file/JoISzQIA#vulVnJqwZRYiRiPgmU5gzZGqhVLqzB6h03MiDh1NYhU

1 Introduction

The Myghty Thymio [Guzzi et al., 2018] is an expanded version of the Thymio: a small, inexpensive, mass-produced mobile robot. This robot has a front camera and 9 infrared proximity sensors (5 in the front, 2 in the rear, 2 on the ground) which can detect objects at a distance of maximum 12 centimetres. For a moving robot this relatively short range of the proximity sensors is not enough to smoothly avoid obstacles. Even more challenging is the case of pitfalls, which only the ground sensors as the sole vertical sensors can detect. Because of the ground sensors' positions, pitfalls are detected only when at least parts of the robot already tumble over the abyss.

To solve these problems, we use the built-in camera and a deep learning architecture to predict the angular velocity such that the Mighty Thymio can detected obstacles and pitfalls at appropriate distance and avoid them in a smooth manner. Additionally, we use this prediction system to create a semi-automatic assistant that can help a teleoperated robot to avoid crashing.

2 Method

2.1 Data collection

For data collection we started with a simple world filled with cardboard boxes (Figure 4 in Appendix A) created with the Gazebo [Koenig and Howard, 2004] built-in building editor. Then, we moved to a more complicated world filled with boxes and pitfalls which are represented by staircase (Figure 5 in Appendix A).

2.1.1 Longer ranges

Because this is a simulated world, we changed the Thymio IR sensor ranges to detect objects up to a distance of 2 meters. In reality the Thymio IR sensor have a range of 12 cen-

timetres but for a real-life application they could be swapped with another kind of sensors or with a different data collection technique (Section 2.1.2 and 2.1.3). To collect data, the Thymio wanders randomly around the world and it saves a camera frame and the IR sensors values every second. Once the Thymio is close to an object (less than 20 centimetres distance) it stops and it rotates away from it and once it has a free path it starts again its forward movement.

2.1.2 Save & Flag

The longer sensors ranges do not help with pitfalls detection because they are parallel to the ground. To solve this problem we implemented a save and flag system. Here, the Thymio wanders randomly around the world and if it senses an object, it saves the sensors' values and it flags the last saved image as an object. If it senses a pitfall with the ground sensors, it flags the last saved image as a pitfall.

2.1.3 Teleoperation

The ground sensors do not allow to fully detect pitfalls because if the Thymio approaches a hole with a small angle, one wheel falls into the hole before the sensor can sense something. This problem was solved in the simulation by creating a reset function which resets the Thymio position once it falls. For real-life scenarios, this would be problematic when a robot gets damaged on pitfall drops. One solution would be to extend the floor with glass panels around pitfall edges. So, before falling into it, the Thymio could detect the pitfall using its ground sensors that see through the transparent glass extension. Another option is teleoperation. For this reason, we developed a teleoperated data collection system which every second stores the camera image and the Thymio angular velocity. To move the Thymio we relied on the open source key-based teleoperation [Magyar, 2020]. Because of the small amount of time available for this project and the big amount of time needed to collect data via teleperation we did not train a model with this data collection system.

2.2 Architecture

Because of the small amount of time to collect data we relied on the concept of transfer learning to be able to predict the angular velocity of the Thymio based on an input picture. More precisely, we use the VGG16 model [Simonyan and Zisserman, 2015] pretrained on ImageNet [Deng et al., 2009] and we replace the fully connected layers at the end of the network with similar layers with the right output size. The network's two output values are T and C. T should be 1 if the object is on the left, -1 if it is on the right and 0 if the object(s) are symmetric or there are no objects. C is 0 when there are no objects and 1 if an object is centred or there is a pitfall. To find the best learning rate value, optimisation algorithm (SGD or ADAM [Kingma and Ba, 2014]), weight decay and fine-tuning vs transfer-learning (train all network or only the final fully connected layer) we performed a grid search [1] using the Wandb sweep [Biewald, 2020] (Figure 1) trying to minimise the test/validation loss. The training was implemented with early stopping and a patience of 10 epochs.

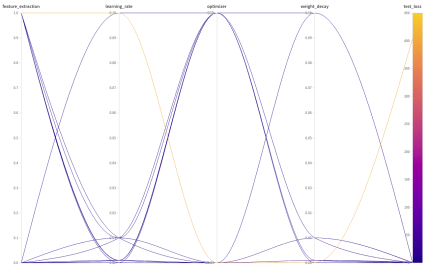

Figure 1: Wandb hyper-parameters search

[1] app.wandb.ai/raikilon/cnn-angular-velocity

Unfortunately, after some tests in Gazebo we realised that a small loss did not always end up in a good model probably because the network was overfitting on the training/validation world (tests were done on the test world). For this reason, we picked some trained models and we create a scatter plot with the predictions and targets (Figure 2). Then, we chose the best hyper-parameters combination which resulted in a good division in the scatter plot (which meant also a small lost but not the smallest) and good Gazebo results. At the end, we selected the following hyper-parameter values for training: learning rate = 0.0001, weight decay = 0.0001, ADAM and fine-tuning.

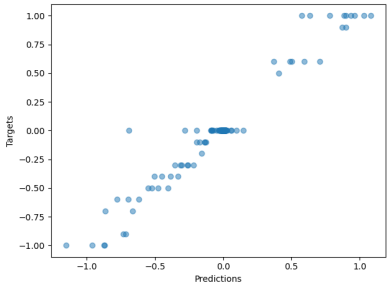

Figure 2: Angular velocity prediction

2.2.1 Post-processing

In the data collection step we saved three types of target values namely IR sensors ranges (Section 2.1.1), objects and pitfall position (Section 2.1.2) and angular velocity (Section 2.1.3). Apart from the third case, the collected information needed some post-processing. When dealing with IR sensors we used only the centre-left, centre and centre-right sensors. The reason is the limited frame of the camera: the left-most the camera sees corresponds to the view ray of centre-left IR sensor, to the right the frame limit is the centre-right IR sensor. This means that the sensors at the very right and very left are invisible to the camera frame and feeding proximity data that the camera cannot see would result in a more difficult training process. In the case of longer ranges we simply normalise the values between 0 and 1 (1 is given when the object is half a meter away). Then, we compute T as $dot([centre\,left, centre\,right], [1, -1])$ and C as $centre$. The post-processing is similar for the Save & Flag system. We compute T and C and assign them to the flagged image as well as the 4 images preceding the flagged image. Again we normalise the range values between 0 and 1 (1 is given when the object is 5 centimetres away). Moreover, if an image is flagged as pitfall, we update the C values to 1 for the flagged image and its 7 preceding images.

3 Results

To be able to have consistent tests between models, we design a test world with six different zones (Figure 6 in Appendix A). The first zone is a reproduction of the simple world (Figure 4 in Appendix A) with moved boxes. The second zone is a reproduction of the pitfalls world (Figure 5 in Appendix A) with moved boxes and pitfalls. Sections 3 to 5 are a small reproduction of the simple world with other kind of objects (small shelf, big shelf and standing person) to test how well the model has generalised and how it behaves in unknown situations. The last zone is a combination of zone 1, 3, 4 and 5 with an incremental level of difficulty (from the known boxes to the complex human body shape).
In addition to the test world, we develop an automatic controller that takes a picture every second and uses the model to get a prediction about the objects/pitfalls position. The controller uses T to decide in which direction to move (e.g. $T > 0$ the object is on the left and the controller moves right proportional to T). Moreover, if $C > abs(T)$ it means that there is an object/pitfall in front of the Thymio and therefore, the controller moves left proportional to C.

3.1 Object avoidance

For this task, we trained the model using the data from the longer ranges system (section 2.1.1) and the simple world. More precisely, we collect data for 30 minutes which resulted in around 2000 samples. The result of this model are pretty good. In fact it is able to detect objects from far distance and avoid them in a smooth manner in the test world zone 1. When dealing with pitfalls (test world zone 2) the system fails and the Thymio falls. An interesting fact is that it is able to detect something even if it never saw a pitfall because when it is in front of a pitfall it starts to rotate away really slowly. Regarding more complex object (zone 3 to 5) the trained model generalises extremely well. It is in fact able to avoid all the three more complex objects with a little bit more difficulty with the standing person. This means that the network learned to avoid objects even if they look like the floor (i.e. wooden shelves).

3.2 Pitfall avoidance

For this task, we trained the model using the data from the Save & Flag system and the pitfalls world. We collected around 30 minutes of data to get 2000 picture with 100 objects and 20 pitfall encounters. The results of this model are quite good both for object and pitfalls avoidance (Figure 3 shows the scatter plot with prediction and targets where 1 means an object/pitfalls is in front).

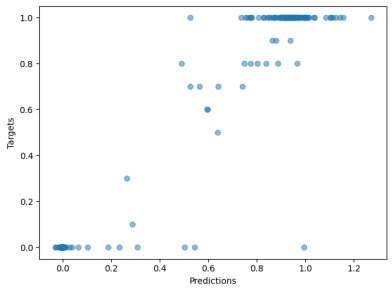

Figure 3: Detecting front obstacles

The main difference from the previous model is that the detection happens at a shorter distance from objects/pitfalls and therefore the Thymio, at times, gets really close to crashing. This problem could probably be solved by applying a post-processing to more than only the 4 images preceding an image flag (section 2.2.1).

3.3 Illumination stress test

To build in an extra stress test, we lowered the illumination of the test image (Figure 7 in Appendix A). We were delighted to see that the model did not show any shortcomings in generalising to darker ambiance light. The robot identified and avoided all objects and pitfalls as if it was under normal illumination. This gives us evidence that the model is applicable to various lighting conditions and did not overfit on a limited range of incoming camera light intensity.

3.4 Semi-automatic assistance

This model is the same as used for Pitfall avoidance (Save & Flag system, pitfalls world). The extra here is that the robot is manually controlled via teleoperation. The trained model only assists in extreme cases to avoid crashed such as collisions with objects or drops into pitfall. As long as the output values C,T are small, teleoperation is in full control. Once either C or T exceed the threshold of 0.5, the assistant takes over and steers the robot out of the dangerous situation. As soon as there is no more threat, the assistant turns off and the control is given back to the teleoperating user. The results are very promising as the assistant can avoid almost all crashes. Only when approaching a pitfall from the right with a very small angle the robot crashed. The reason is that we decided to always go left when $C > abs(T)$, in this case meaning a pitfall is

detected. Now if we are approaching a pitfall from the right with a small angle and turn left, we drive right into the pitfall. A simple solution to this problem would be to stop, turn on the spot and continue going forward once the robot turned away from the pitfall. This is however not a practical solution. The correct way would be to detect that the pitfall is on the left of the robot, and then turn right going away from the pitfall. In order to train such a behaviour, after the robot falls during training, the robot needs to be reset twice to a position just before the edge: once looking 45 degrees to the left, once 45 degrees to the right. If the robot also detects a pitfall looking 45 degrees more to the right, we know the pitfall was on the right and the target behaviour should be to turn left. In the opposite case where the robot turned 45 degrees to the left still sees the pitfall, the respective target behaviour should be to turn right. Given the fact that such an extensive training process would well exceed the scope of this project, we decided to leave this fine-tuning up for future work.

4 Conclusion

In this project we have shown that by collecting training data for a time frame as short as just 30 minutes we were able to overcome the shortcomings of the Mighty Thymio when it comes to obstacle detection and avoidance. We were able to create autonomous robots that randomly wander in realistically coloured worlds without crashing into neither objects nor pitfalls. Positive was how well the object detection generalised to complex object such as persons, also under darker lighting conditions. Further, we also successfully implemented the model into a semi-automatic assistance for teleoperated robots. From a practical perspective this is very interesting as it demonstrates the potential of such a autonomous agent in the field of AI-assisted robot control. Future work would however be require to fine-tune the system, especially regarding pitfall avoidance. One way could be to extend the Save & Flag process to more than just the preceding 4 images. Another approach could be to make the pitfall detection direction aware, so that it always picks the shortest way out of dangerous situations.

References

[Biewald, 2020] Biewald, L. (2020). Experiment tracking with weights and biases. Software available from wandb.com.

[Deng et al., 2009] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*.

[Guzzi et al., 2018] Guzzi, J., Giusti, A., Di Caro, G. A., and Gambardella, L. M. (2018). Mighty thymio for higher-level robotics education. In *AAAI*.

[Kingma and Ba, 2014] Kingma, D. and Ba, J. (2014). Adam: A method for stochastic optimization. *International Conference on Learning Representations*.

[Koenig and Howard, 2004] Koenig, N. and Howard, A. (2004). Design and use paradigms for gazebo, an open-source multi-robot simulator. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2149–2154, Sendai, Japan.

[Magyar, 2020] Magyar, B. (2020). Packages especially for teleoperating ros robots.

[Simonyan and Zisserman, 2015] Simonyan, K. and Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*.

A Gazebo worlds

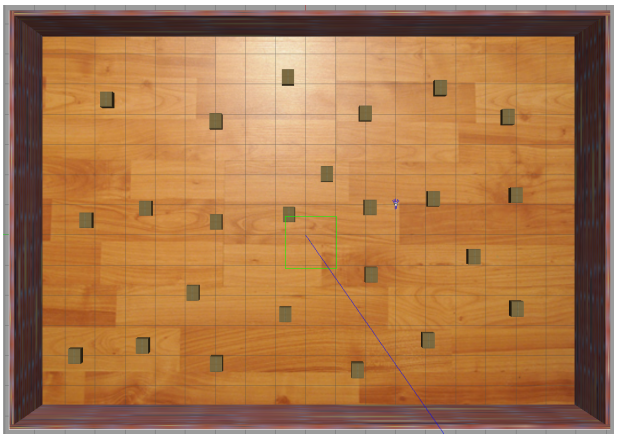

Figure 4: Simple training world

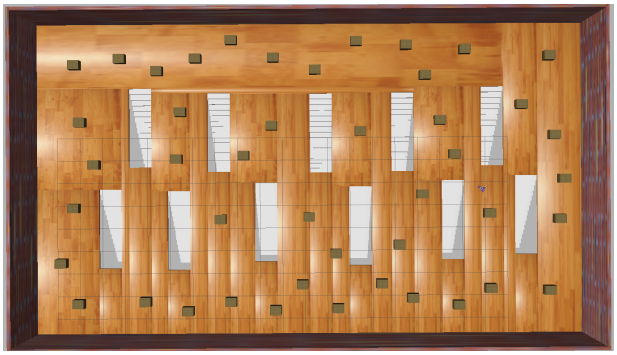

Figure 5: Pitfalls training world

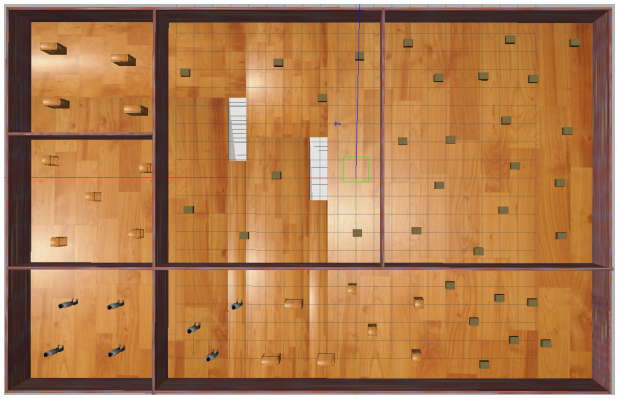

Figure 6: Test world

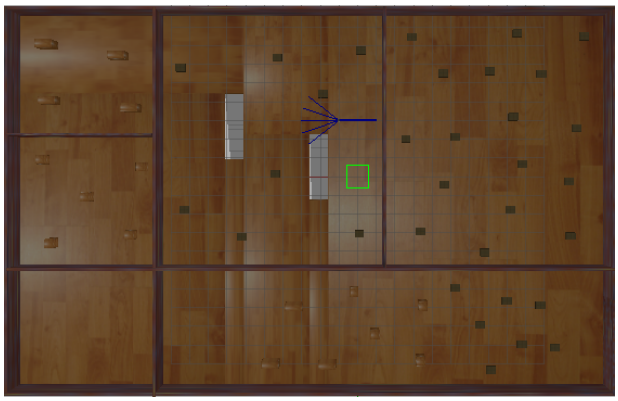

Figure 7: Low illuminated test world