Trashy - Your Friendly Neighbourhood Trashcan

1st Mathias Emil Slettemark-Nielsen†
The Maersk Mc-Kinney Moller Institute
University of Southern Denmark
Odense, Denmark
masle16@student.sdu.dk

2nd Mikkel Larsen †
The Maersk Mc-Kinney Moller Institute
University of Southern Denmark
Odense, Denmark
milar16@student.sdu.dk

3th Danish Shaikh*
The Maersk Mc-Kinney Moller Institute
University of Southern Denmark
Odense, Denmark
danish@mmmi.sdu.dk

† these authors contributed equally, elaboration on contributions can be found in Appendix A.
* these authors were supervisors.

***Index Terms*—Artificial intelligence, Correlation based learning, Semantic segmentation, Mobile Robot.**

***Abstract*— Programming a robot the traditional way to solve problems is a difficult task. Making the robot learn to solve a problems is a more robust way as the robot can adapt to its environment. In this paper, a system for avoiding obstacle and following a target for differential drive mobile robot is developed. The system is based on input correlation learning which is an unsupervised machine learning method, making the system capable of learning its environment it is deployed in. The system was deployed on a small differential drive mobile robot containing a mono camera, a Raspberry Pi 4, a Matrix Voice and an Edge TPU.**

The mobile robot learned to navigate in a indoor environment at the Technical Faculty at Southern University of Denmark, here it was able to locate a person, i.e. a target, by the use of computer vision. Furthermore, it learned to avoid static obstacles and drive around them in a smoothly manner.

This paper shows, how a simple system can learn to smoothly navigate in an indoor environment. The system could be used create a more interaction trash disposal option, thus, instead of having a static trashcan a mobile robot with a trashcan could be setup. Further applications could also be added, e.g. a robot arm to collect forgotten trash.

The code for this project can be found here[1], and the workload of each contributor can be found in Appendix A.

I. Introduction

Humans consumption is still high [1], which leads to more trash getting thrown out. Even though there has been a focus on setting up more trashcans there is still trash in nature, streets, and in-door environments, which then has to be cleaned manually. This paper focuses on developing a prototype platform for a self-learning trashcan robot, which learns through artificial intelligence to adapt to its environment.

[1]https://github.com/mikkellars/Obstacle-Avoidance-with-ICO

The trashcan robot prototype is based on a differential drive mobile robot consisting of a mono camera in the front, a Raspberry Pi 4 [13] as processing unit, a Matrix Voice [11] is mounted on the top of the robot, and an Edge TPU [3] to speed up image processing. The robot can be seen in Figure 1.

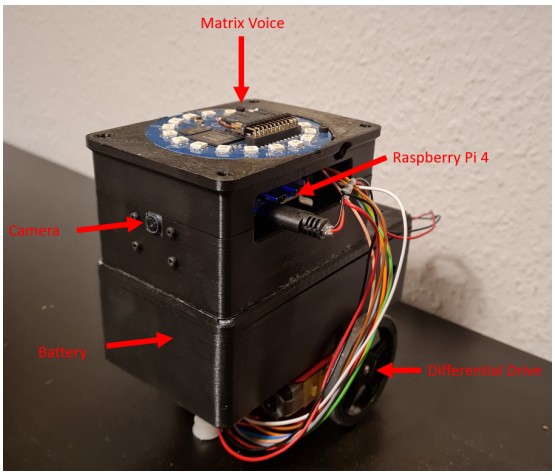


Fig. 1: The mobile robot prototype platform.

To make the image processing system robust, a neural network semantic segmentation and a neural network object detector are trained and implemented. Thus, the framework Tensorflow Lite [5] and an Edge TPU is used to keep processing time down to a minimum.

To make the navigation smooth, an online learning approach is used in this paper. More precise does the mobile robot learn to navigate with Input Correlation, ICO, learning [12]. Thus, the image processing is used as input to a ICO controller.

The final system is capable of driving towards a human while avoiding driving into obstacles, e.g. walls and boxes. All

of the experiments of this project took place at the Technical Faculty at the Southern University of Denmark.

II. Related Work

A previous study made by Shaikh and Manoonpong [15], showed it was possible to learn, by the use of input correlation learning, to navigate to a virtual sound source. Furthermore, the system was capable of making smooth path planning while avoiding obstacles. Their system was also implemented on a simulated differential drive robot as ours. Our system has taken the basic structure of using the input correlation learning output to control the motor values of the differential drive robot. However, compared to their solution which is in simulation, our solution is made in real-life. Furthermore, our solution utilizes visual input from a camera, instead of audio inputs. Lastly, our final combined solution has two overall input correlation learning nodes, one responsible for obstacle avoidance and one for target following, whereas in [15] solution combines target finding and obstacle avoidance into one input correlation learning node.

In [16], a semantic segmentation neural network is used to develop an indoor navigation system for a mobile robot. The state-of-the-art segmentation network DeepLabV3+ [2] is used to segment, the wall as an obstacle and the floor as free space. The segmented image is then used to mathematically calculate a collision-free path in the image. The main difference between their system and ours is that our system learns to drive in the middle of a hallway by the use of input correlation learning and learns to follow a target. Furthermore, the use of the state of the art network DeepLabV3+ is computational heavy so [16] have to use a laptop instead of a small embedded computer, like the Raspberry Pi, whereas our system uses a smaller network PSP-Net [18] along with the TensorFlow lite framework to keep the computations as low as possible.

III. Methods

To make the mobile robot successful in finding and navigating to a human, three main components have been made: learning to locate obstacles with semantic segmentation, learning to locate a human with object detection and learning to navigate with input correlation learning. Thus, the learning components are the localization with semantic segmentation and object detection, which is supervised learning, and the navigation with input correlation learning, which is unsupervised learning. The overall structure of the system is illustrated in Figure 2.

A single mono camera, as illustrated in Figure 2, is used to navigate in an unknown environment. Two assumptions are used when learning to avoid obstacles and follow a target. Both utilize mono camera theory. Firstly, the further toward the bottom of the image, the closer is the obstacle to the robot. Secondly, the further the target is toward the edges of the image the closer it is to get out of the frame. These two assumptions are utilized in section III-A.

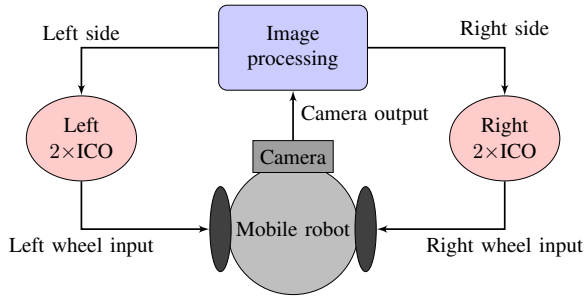


Fig. 2: Overview of the structure of the system.

A. Image Processing

The image processing system illustrated in Figure 2 consists of three parts: a semantic segmentation to segment objects in the input image, an object detector to detect the target in the input image, and localization to find the nearest object. The flow of the image processing is illustrated in Figure 3.

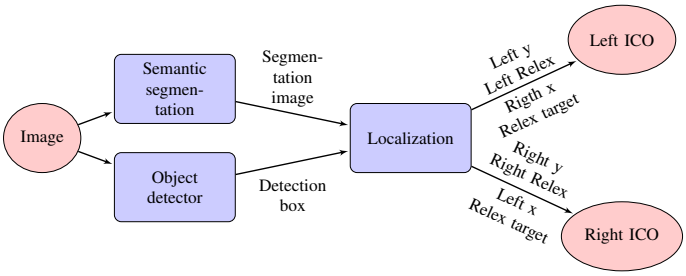


Fig. 3: Flow of the image processing.

1) Semantic segmentation: To make the robot able to detect objects in an unknown environment, a semantic segmentation method is applied. A semantic segmentation method gives each pixel a category label based on an input image. The categories chosen in this paper are human, floor, wall, box, and others. The categories wall, box, and others are categorized as obstacles and the category human is categorized as the target.

Two deep neural network methods are evaluated for semantic segmentation, U-Net [14] and PSP-Net [18]. Deep learning methods are chosen as these achieve much higher accuracy compared to traditional vision methods. This is mainly due to neural networks learning through supervision rather than traditional methods which are the programmer's optimal way to solve the problem. It was chosen for both U-Net and PSP-Net to utilize a pre-trained feature extractor in form of ResNet50 [8], which is trained on the dataset ImageNet [4], to achieve higher accuracy.

2) Object Detector: To get better and more consistent results on the target position, a pre-trained object detector made for running of the Edge TPU [7] was used as the human detector. The model used was a MobileDet Single State Detector [17].

3) Localization: The localization method takes the semantic segmentation image and the bounding box of the target as input. The method then finds the obstacle that has the highest

y image coordinate in the image, i.e. the obstacle which is closest to the bottom of the image. This is done both for the right and left side of the image. The y coordinates are also normalized by dividing with the height of the image to make them compatible with the ICOs. These coordinate is used as the predictive signal to the ICOs. Furthermore, a reflex signal for detecting if a obstacle is within a threshold of the mobile robot is calculated as

$$\text{Reflex} = \begin{cases} 1 & \text{if (left y} > \text{right y) and (left y} > \text{col}_{\text{thresh}}) \\ -1 & \text{if (left y} < \text{right y) and (right y} > \text{col}_{\text{thresh}}) \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

where $\text{col}_{\text{thresh}}$ is the collision threshold. The output from Equation 1 are then converted to the following output of the localization method to the input of the right and the left ICO:

$$\text{Right Reflex} = \begin{cases} 1 & \text{if Reflex} = 1 \\ 0 & \text{otherwise} \end{cases}$$

$$\text{Left Reflex} = \begin{cases} 1 & \text{if Reflex} = -1 \\ 0 & \text{otherwise} \end{cases}$$

The same is done for the output to the target following ICOs, except the image coordinate is the x coordinate instead of the y coordinate in Equation 1.

B. Navigation with Input Correlation Learning

For the mobile robot to navigate in an unknown environment, Input Correlation, ICO, learning is used as illustrated in Figure 2. The ICO is a unsupervised learning algorithm, and its purpose is to learn to give an output so the reflex signal will never occur based on the predictive signal. The mathematical structure of one ICO is illustrated in Figure 4.

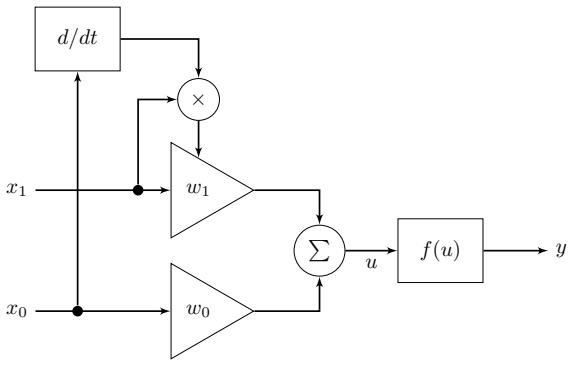


Fig. 4: Mathematical structure of an single layer feedforward ICO. x_1 and x_0 are the predictive and reflective signal, respectively, and their corresponding weights are w_1 and w_0. u indicates the output before the activation function, $f()$, and y is the final output.

The mathematical equations of Figure 4 are

$$u = w_0 \cdot x_0 + w_1 \cdot x_1 \quad (2)$$

$$y = f(u) \quad (3)$$

$$w_1 = w_1 + \mu \cdot x_1 \cdot d/dt \cdot x_0 \quad (4)$$

where μ in Equation 4 is the learning rate. Note that in Equation 4, the weights of the predictive signal are only updated when a change in the reflex signal is present.

In this paper four ICOs were implemented, two for obstacle avoidance, and two for target following. The input, processing, and output of the left ICO is illustrated in Figure 5.

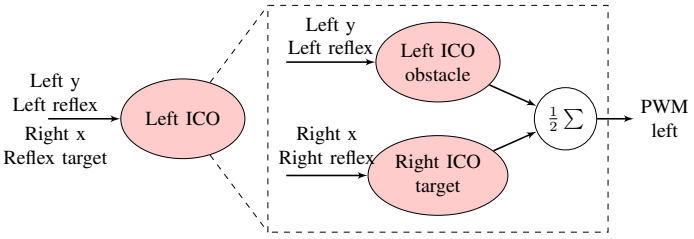


Fig. 5: Flow the left ICO system containing both obstacle avoidance and target following.

1) Obstacle Avoidance: An ICO for each wheel is implemented, as Figure 2 illustrates. Thus, each wheel can act independently from the other wheel. The output from the localization node *Left y* and *Right y* is the predictive signals for the left and right ICO, respectively. Furthermore, the *Left reflex* and *Right reflex* are the reflex signal for the left and right ICO, respectively.

In each ICO the activation function $sigmoid$ is used to be able to always give a PWM ≥ 0 when the predictive signal is 0. Thus, the robot will drive with equal PWM on both wheels if no obstacle is present. As the output of the $sigmoid \geq 0$, the output of the left ICO is added to the left wheel to drive right around the obstacle and the output of the right ICO is added to the right wheel to drive left around the obstacle.

As the ICO learns the optimal weights through the reflex signal, a collision avoidance behavior is implemented to ensure that there is not a collision with an obstacle no matter the weights. This behavior overwrites the values from the obstacle avoidance ICOs and it activated as long the reflex $\neq 0$. The behavior sets the PWM $= 20$ on either the right or left motor depending on which side the obstacle is detected.

2) Target Following: As in section III-B1, an ICO for each wheel is implemented. The objective for the target following is to keep the target in the image, thus, the predictive signal is the x coordinate of the target center. Furthermore, the reflex signal, *Reflex target*, trigger on the x coordinate of the target. The two ICOs also have the activation function $sigmoid$ to gain the same benefits mentioned in subsubsection III-B1. However, the output of the left ICO is added to the right wheel to turn towards the target and vice versa, since the objective here is to follow the target and not avoid it, as in section III-B1.

3) Final learning system: In the final learning system both the obstacle avoidance ICOs and the target following ICOs are present. The output PWM of the right and left wheel are

calculated as

$$\text{right}_{\text{PWM}} = \frac{\text{right}_{\text{obs}} + \text{left}_{\text{targ}}}{2} \quad (5)$$
$$\text{left}_{\text{PWM}} = \frac{\text{left}_{\text{obs}} + \text{right}_{\text{targ}}}{2}$$

where *obs* refers to the output of the ICO responsible for the obstacle avoidance, and *targ* refers to the output of the ICO responsible for the target following. This is illustrated in Figure 5.

IV. Experiments

This section empirically validates the system on the mobile robot. Firstly, the training and accuracy of the semantic segmentation and object detection methods are elaborated. Secondly, the obstacle avoidance and the target following of the full system are examined.

A. Image processing

The dataset used to train the two semantic segmentation networks consists of 152 training images and 39 validation images. The images are collected at the University of Southern Denmark inside the TEK building and hand-label using the tool hasty.ai. The distribution of the objects appearing on the images and example of the label images can be seen in Table I and Figure 6.

Floor	Wall	Box	Person
196	245	117	38

TABLE I: The distribution of the labels in the dataset: floor (blue), wall (purple), box (yellow), and person (red).

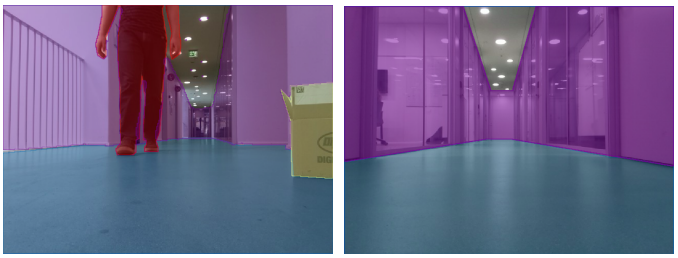


Fig. 6: Examples of images from the dataset. The different overlays indicates the labels of the pixel: floor (blue), wall (purple), box (yellow), and person (red).

1) Semantic segmentation: The images used to train the semantic segmentation algorithms are resized to $96 \times 96 \times 3$ to keep the processing at a reasonable level on the raspberry pi. The data augmentation applied to the images can be seen in Table II.

The U-Net and PSP-Net are then trained for 20 epochs, i.e. iterations through the dataset, with the encoder frozen and afterward trained for 80 epochs unfrozen. The training is done with a batch size of 4 using a categorical cross-entropy-jaccard loss, and the default settings of the optimizer Adam from the neural network framework Keras [10]. The graph for the training- and validation- loss as well as the training- and validation- intersection over union, IOU, score can be seen Appendix D. The best IOU scores are the highest validation IOU and the results for these can be seen in Table III.

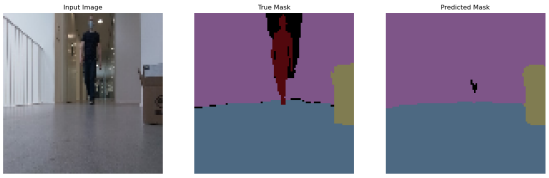
Augmentations type	Value
Random flip left-right	–
Random Crop	0.25 · image res
Random image saturation	[0.1, 3.0]
Random image brightness	[0.1, 1.0]
Random image contrast	[0.1, 0.5]
Random image HUE	0.5

TABLE II: Data augmentations with their setting used for training the semantic segmentation networks. Each augmentation has 50 % probability of getting applied to the data. Image res indicates the image resolution.

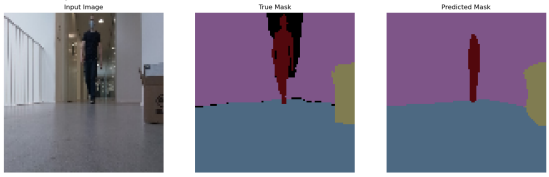
Network type	Validation IOU	Validation Loss	Training IOU	Training Loss	Epoch
U-Net	0.75	0.40	0.54	0.70	93
PSP-Net	0.71	0.41	0.55	0.69	86

TABLE III: The training and validation results for best IOU score.

From the Table III the U-Net should in theory be the best network to run on the robot. However, looking at an example image from the validation set containing a human in Figure 7, can it be seen that the U-Net prediction mask does not detect the human, but the PSP-Net prediction mask does.



(a) U-Net run on image with a human and box in the hallway.



(b) PSP-Net run on image with a human and box in the hallway.

Fig. 7: Example of a validation image run through the U-Net and the PSP-Net.

Based on these results the PSP-net was chosen to run on the robot as the segmentation algorithm. Thus, the results seen in the rest of the paper is with the PSP-Net running as the segmentation algorithm.

2) Object detector: The MobileDet used was pre-trained on 90 types of objects on the COCO dataset. All of the 90 classes was discarded except the human class. Furthermore,

a prediction threshold of $p > 0.5$ on the human class was implemented.

Another image from the validation set containing a human further away from the robot is run through the MobileDet and PSP-Net. The result can be seen in Figure 8.

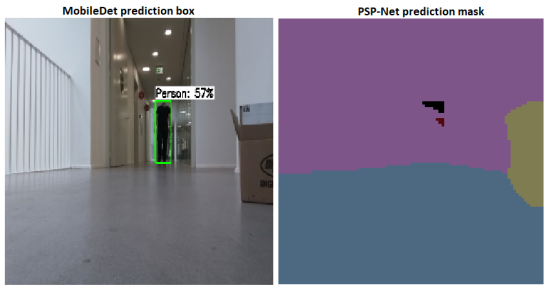


Fig. 8: Comparison image run through the MobileDet and the PSP-Net.

From the result can it be seen that the object detector detects the human better than the PSP-Net. This was also the case during evaluation of the target learning ICO, were it was observed that with the external human detector the robot was able to drive towards humans further away. Thus, in the rest of this paper, the MobileDet is used as a target detector and the PSP-Net is used for segmenting obstacles.

In Table IV, the inference time of MobileDet is shown. From this is can be seen that the Edge TPU gives a significant performance boost in terms of inference time of the model. The Edge TPU is 90 % faster than the CPU on the Raspberry Pi.

Processing unit	**Inference time [s]**
Edge TPU (INT 8)	0.0598 ($\pm$ 0.0072)
CPU, ARMv7 1.4 GHz, (FP 32)	0.6378 ($\pm$ 0.0324)

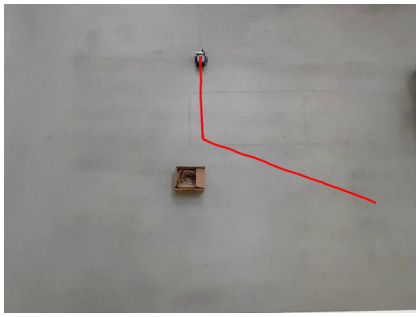
TABLE IV: Inference time with different processing units calculated over 100 samples. INT 8 indicates that the model is unsigned integer 8 bit and FP 32 indicates that the model is floating point 32 bit.

B. Navigation with Input Correlation Learning

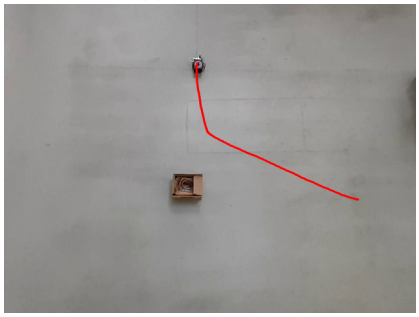
Using the PSP-Net and MobileDet trained in section IV-A, the systems ability to avoid obstacles and following a target is explored here. Note that all navigation experiments in this section is done with a learning rate of 0.1 and the weights are random initialized within [0.0, 0.1]. Furthermore, the $\text{collision}_{\text{threshold}}$ of the reflex signals are set to 0.8 of the predictive signals.

Since the experiments are recorded as videos a simple neural network, Yolov5 [9], is implemented to track the robot through the frames. The training of Yolov5 is elaborated in Appendix E. In summary, the dataset consists of 143 images and after 100 epochs, i.e. iterations through the dataset, it achieves a mAP@0.5 of 0.995.

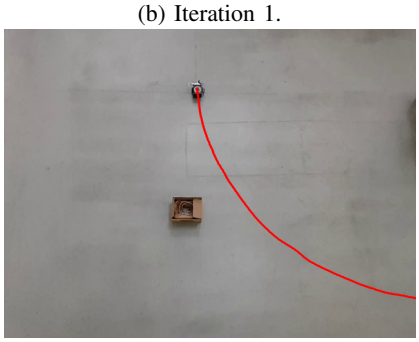
1) Obstacle Avoidance: To verify that the mobile robot can learn to avoid obstacles, the system is tested with a box and wall as an obstacle. Figure 9 shows the system avoiding a box and more examples of obstacle avoidance can be found in Appendix B.



(a) Iteration 0.



(b) Iteration 1.



(c) Iteration 2.

Fig. 9: Illustration of trajectory (red) when obstacle avoiding a box. Videos of the iterations can be found here: iteration 0, iteration 1, and iteration 2.

Figure 9 shows the trajectory of the mobile robot at different iterations. It can be seen that it gets to close to the obstacle and must perform a collision avoidance behavior in iteration 0 and 1. However, in iteration 2 the robot is capable of avoiding the obstacle smoothly. The trajectories from Figure 9 can be seen in Figure 10.

From Figure 10, it can be seen that iteration 2 avoids the box in a smoothly manner. Thus, after three iterations the system has learned to avoid a static obstacles in a smoothly manner. The reflex signal and input, weight, and output of the ICOs in Figure 9 and Figure 10 can be seen in Figure 11 and Figure 12, respectively.

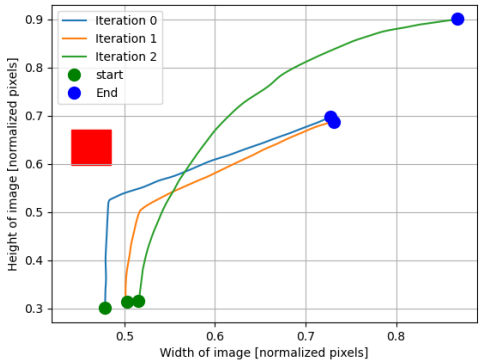


Fig. 10: Illustration of trajectories when obstacle avoiding a box. The box is visualised by the red square. The start and end point is visualised with a green and blue circle, respectively.

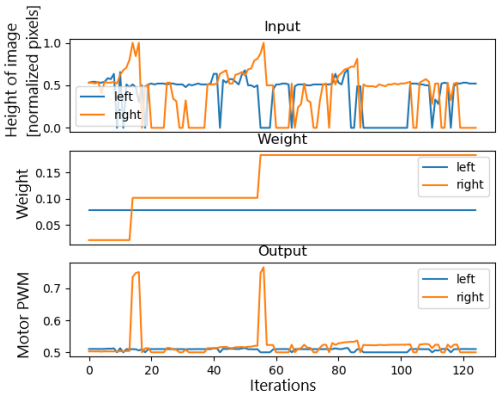


Fig. 12: Input, weight, and output values for both the right and left ICO when doing obstacle avoidance in Figure 9 and Figure 10. Note that Iterations in this figure determines the iterations of the systems super loop.

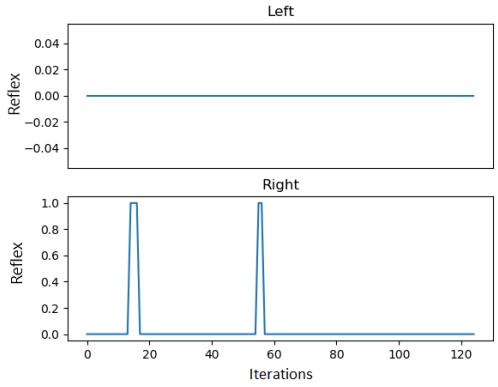


Fig. 11: Reflex signal for left and right ICO during the experiment in Figure 9 and Figure 10.

Figure 11 and Figure 12 shows that the reflex signal for the right ICO goes high two times, thus, the right weight is updated two times. Furthermore, the output to the motor shows that after the reflex signal has been high two times, the system is capable of avoiding the spikes that occurred before. Even though the system can smoothly do obstacle avoidance when looking at the trajectory in Figure 10, the output given to the motors is not smooth.

2) Human Following: To verify that the mobile robot can learn to follow a target, the system is tested with a human. Figure 13 shows the system driving towards the target. Further examples of target following can be found in Appendix C.

In Figure 13 can it be seen that the mobile robot is able to drive smoothly towards the target after just one iteration. This is further supported by looking at Figure 15 and Figure 16, where it can be seen that the reflex signal only is high once through the whole experiment. The trajectories in Figure 13 can be seen in Figure 14.

Figure 14 shows the mobile robot is capable of navigating smoothly towards the target after one reflex signal. The reflex signal from this experiment can be seen in Figure 15, and the input, weight, and output of the left and right ICO is shown in Figure 16.

From Figure 16 can it been seen that the system avoids reaching a reflex behaviour, after the system has been in one reflex behaviour. Thus, is able to perform smooth navigation as shown in Figure 13 and Figure 14. However, the change in motor PWM in Figure 16 indicates that the motor values does not change smoothly.

3) Combination of Obstacle Avoidance and Target Following: As of now a combination of the obstacle avoidance and target following does not work perfectly together, this is verified here. To do so an experiment with both an obstacle and target was done, the obstacle being a box and target a human. Illustrations of the trajectory of each iteration can be seen in Figure 17.

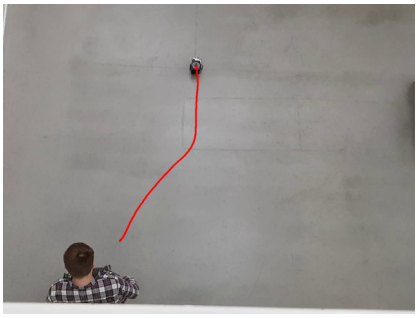
Figure 17 shows that the mobile robot drives towards the human, i.e. the target. However, it does not drive smoothly around the obstacle, thus, it does not avoid the box. As seen in Figure 17, the mobile robot drives towards the target and when the box is within the collision threshold it performs a collision avoidance behavior. Depending on this behavior, the mobile robot might still be able to see the target, as in Figure 17b and will steer towards it. In Figure 17a and Figure 17c, the camera is not capable to see the target and therefore the mobile robot does not steer towards the target. The trajectories in Figure 17 can be seen together in Figure 18.

From Figure 18 can it be seen that, the mobile robot steer towards the target, i.e. the blue rectangle, in each iteration. However, when colliding with the obstacle, i.e. the red rectangle, the mobile robot steers of in different directions. Thus, how the robot approaches the obstacle has an influence on its ability to reach the target.

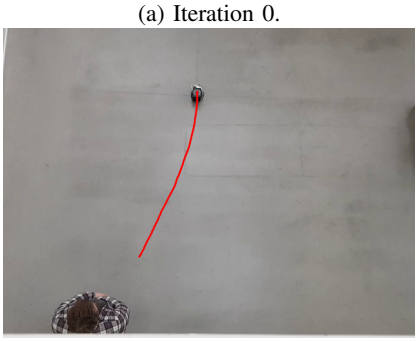
V. Discussion

Dataset representation

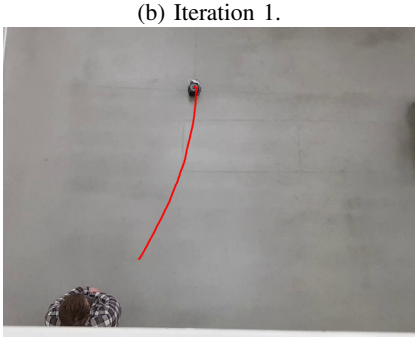
As seen in Figure 7 the U-Net could not detect a human, but the PSP-Net could. However, the U-Net scored a better IOU at



(a) Iteration 0.



(b) Iteration 1.



(c) Iteration 2.

Fig. 13: Illustration of trajectory (red) when following a human. Videos of the iterations can be found here: iteration 0, iteration 1, and iteration 2.

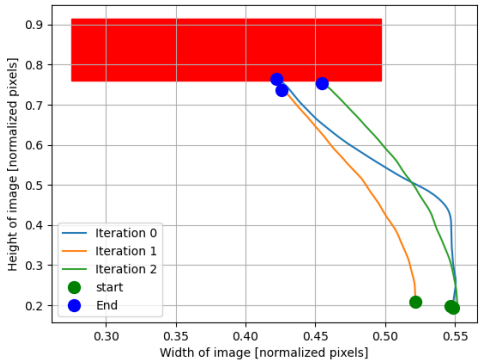


Fig. 14: Illustration of trajectories when following a human. The rectangle illustrates the position of the target. The start and end point are visualised with green and blue circles, respectively.

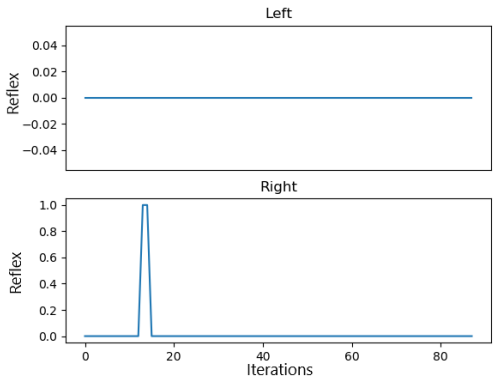


Fig. 15: Reflex signal for left and right ICO during the experiment in Figure 13 and Figure 14.

the validation set compared to the PSP-Net. One explanation of this is the class human is represented a low number of times in the dataset compared to the other objects. This leads to the U-Net focuses on fitting to the over-represented classes. This is confirmed by looking at the other examples of validation images not containing a human in Appendix D, where the U-Net network is generally more precise regarding the true mask compared to the PSP-Net. For further optimization in the future, a more representative dataset should be collected or a class weighting term should be added to the loss function to counteract over-fitting to specific classes.

Tuning segmentation network

In section IV-A2 the pre-trained object detection algorithm, MobileDet, detected the human better than the PSP-Net. This is mainly due to the MobileDet being trained on a much larger dataset, COCO, compared to our dataset consisting of 152 training images. Furthermore, the PSP-Net did not go through any major hyperparameter optimization. In the future, the PSP-Net should be trained on a larger dataset and hyperparameter optimization should be done, so only one network is running on the platform.

Edge TPU

For a model to be compatible with the Coral Edge TPU Accelerator, the neural network model has to comply with some requirements [6], e.g. being quantized to an 8-bit integer. Thus, a special conversion method is required to transform a normal TensorFlow model to a quantized TensorFlow lite model. This conversion did not succeed regarding the trained PSP-Net or U-Net model because of unsupported operations. The semantic segmentation algorithm should be further investigated in the future for the conversion as this increases the inference speed significantly.

Combination of obstacle avoidance and target following

As described in section III-B3 the ICOs output for the obstacle avoidance and the target following is added, thus,

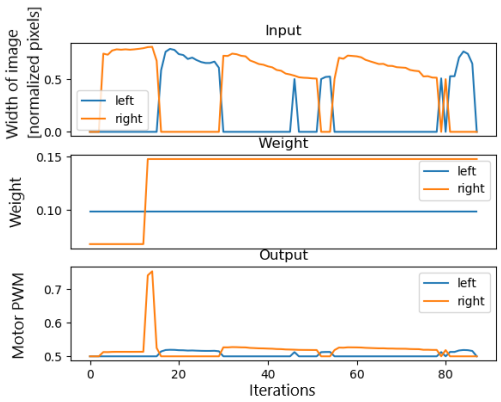


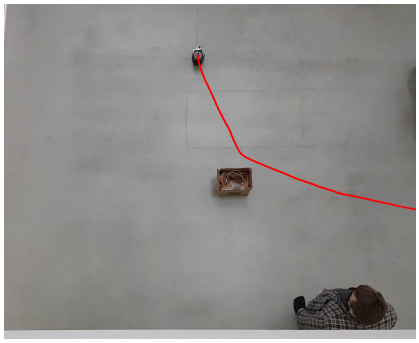
Fig. 16: Input, weight, and output values for both the right and left ICO when doing obstacle avoidance in Figure 13 and Figure 14. Note that Iterations in this figure determines the iterations of the systems super loop.

resulting in the two ICO behaviors counteracting each other. This was especially a problem when a human was right behind an obstacle where the robot could see both. Here it was seen that the robot stayed on its path toward the obstacle until the reflex behavior triggered. For the next iteration of the system, a better way than adding the two ICO behaviors output should be found so it would be possible for the robot to first avoid an obstacle and then afterward find its target and then move towards it.

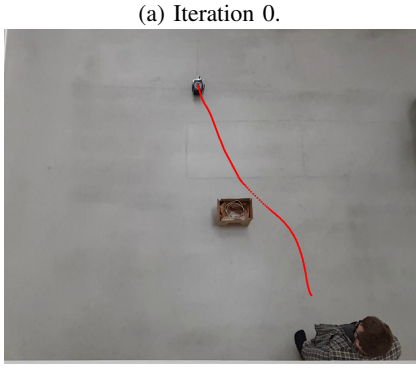
VI. Conclusion

In this paper a successful prototype of a self-learning robot trashcan were made. Using correlation based learning the trashcan robot is able to firstly learn to avoid obstacles and secondly learn to follow a human as a target. These results is obtainable because of the neural networks used to segment obstacles and detect humans. However, problems occur when obstacle avoidance and target following are combined where the robot will steer towards the target without smoothly avoiding the obstacles. This problem should be investigated further in the future.

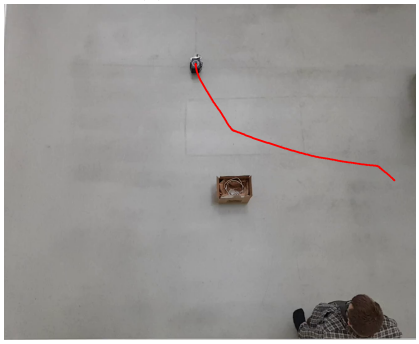
For future work, should a sound algorithm be implemented so the robot will be able to turn in the direction of the sound and locate the target if the target is not visible at first. Furthermore, should the system be implemented on a mobile robot capable of driving around with a trashcan, thus, enabling the robots ability for collecting trash.



(a) Iteration 0.



(b) Iteration 1.



(c) Iteration 2.

Fig. 17: Illustration of trajectory (red) when following a human and avoiding a box. Videos of the iterations can be found here: iteration 0, iteration 1, and iteration 2.

References

[1] United States Environmental Protection Agency. *National Overview: Facts and Figures on Materials, Wastes and Recycling*. Sept. 17, 2020. URL: https://www.epa.gov/facts-and-figures-about-materials-waste-and-recycling/national-overview-facts-and-figures-materials.

[2] Liang-Chieh Chen et al. *Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation*. 2018. arXiv: 1802.02611 [cs.CV].

[3] Coral. *USB Accelerator datasheet*. Dec. 28, 2020. URL: https://coral.ai/docs/accelerator/datasheet/.

[4] J. Deng et al. "ImageNet: A Large-Scale Hierarchical Image Database". In: *CVPR09*. 2009.

[5] Google. *Deploy machine learning models on mobile and IoT devices*. Jan. 5, 2021. URL: https://www.tensorflow.org/lite.

[6] google-coral. *Model Requirements*. Jan. 5, 2021. URL: https://coral.ai/docs/edgetpu/models-intro/#model-requirements.

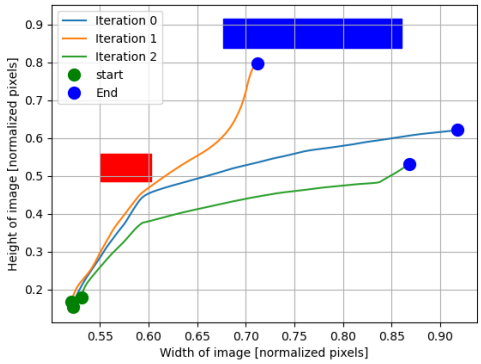


Fig. 18: Illustration of trajectories when following a human and avoiding a box. The rectangles illustrates the position of the target (blue) and obstacle (red). The start and end point are visualised with green and blue circles, respectively.

[7] google-coral. *Models*. Jan. 5, 2021. URL: https://github.com/google-coral/test_data/raw/master/ssdlite_mobiledet_coco_qat_postprocess_edgetpu.tflite.

[8] Kaiming He et al. "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[9] Glenn Jocher et al. "Yolov5". In: *Code repository https://github. com/ultralytics/yolov5* (2020).

[10] Keras. *Adam*. Jan. 5, 2021. URL: https://keras.io/api/optimizers/adam/.

[11] MATRIX Labs. *Voice Development Board For Everyone*. Jan. 4, 2021. URL: https://www.matrix.one/products/voice.

[12] Bernd Porr and Florentin Wörgötter. "Strongly improved stability and faster convergence of temporal sequence learning by utilising input correlations only". In: *Neural Computation* 18.6 (2006), pp. 1380–1412.

[13] RaspberryPi.dk. *Raspberry Pi 4 Model B – 4 GB*. Jan. 5, 2021. URL: https://raspberrypi.dk/produkt/raspberry-pi-4-model-b-4-gb/.

[14] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. "U-Net: Convolutional Networks for Biomedical Image Segmentation". In: *CoRR* abs/1505.04597 (2015). arXiv: 1505.04597. URL: http://arxiv.org/abs/1505.04597.

[15] Danish Shaikh and Poramate Manoonpong. "A neuroplasticity-inspired neural circuit for acoustic navigation with obstacle avoidance that learns smooth motion paths". In: *Neural Computing and Applications* 31.6 (2019), pp. 1765–1781.

[16] Daniel Teso-Fz-Betoño et al. "Semantic Segmentation to Develop an Indoor Navigation System for an Autonomous Mobile Robot". In: *Mathematics* 8.5 (2020), p. 855.

[17] Yunyang Xiong et al. *MobileDets: Searching for Object Detection Architectures for Mobile Accelerators*. 2020. arXiv: 2004.14525 [cs.CV].

[18] Hengshuang Zhao et al. "Pyramid Scene Parsing Network". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. July 2017.

APPENDIX A
WORKLOAD FOR EACH CONTRIBUTOR

In this section the workload of each contributor is stated. Firstly, the workload of implementing the different methods are shown in Table V and lastly, the workload of written the different sections are shown in Table VI.

Contributor	Mathias	Mikkel
Semantic segmentation		×
Labeling of dataset	×	×
Setup of Matrix Lite on Raspberry Pi	×	
Localization	×	
ICO implementing		×
Testing of ICO	×	×
Yolov5 for data analysis	×	

TABLE V: The workload for each contributor in regards of the project.

Contributor	Mathias	Mikkel
Abstract		×
Introduction	×	×
Related Work		×
Methods	×	×
Methods, image processing		×
Methods, navigation with input correlation learning	×	
Experiments	×	×
Experiments, image processing		×
Experiments, navigation with input correlation learning	×	
Discussion		×
Conclusion	×	×

TABLE VI: The workload for each contributor in regards of the report.

APPENDIX B
OBSTACLE AVOIDANCE WITH INPUT CORRELATION LEARNING

In this section are the experiments which are not shown in section IV-B1 shown. Firstly, is the obstacle a box and then a wall. The system is force to steer either left or right around the obstacle, this is done by adding a small offset to the start position, to show that the system is capable of avoiding obstacle in both directions.

Right obstacle avoidance with box: Figure 19 shows the trajectory of the mobile robot when avoiding the box.

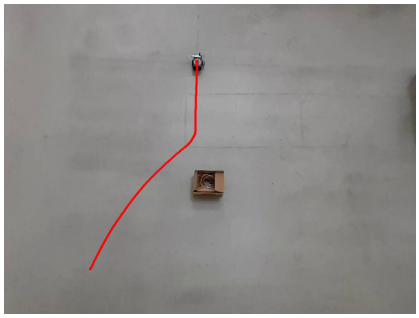
The trajectories from Figure 19 are shown in Figure 20. The reflex signal from the whole experiment is shown in Figure 21. The input, weight, and output of the left and right ICO are shown in Figure 22.

Left obstacle avoidance with wall: Figure 23 shows the trajectory of the mobile robot when avoiding the wall.

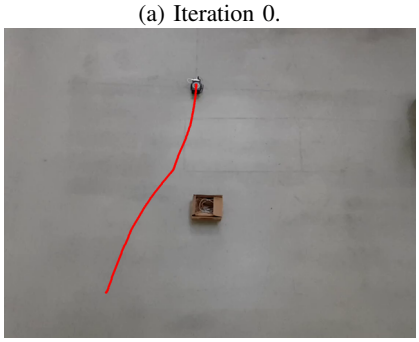
The trajectories from Figure 9 are shown together in Figure 10. The reflex signal from the trajectories in Figure 23 and Figure 24 is shown in Figure 25. The input, weight, and output of the left and right ICO are shown in Figure 26.

Right obstacle avoidance with wall: Figure 27 shows the trajectory of the mobile robot when avoiding the wall.

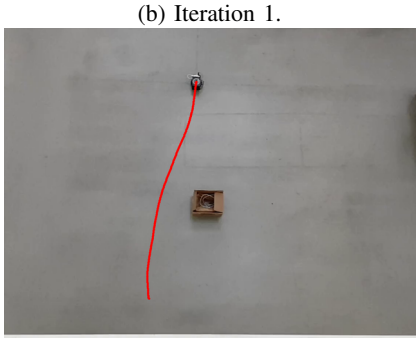
The trajectories from Figure 27 are shown together in Figure 28. The reflex signal from the trajectories in Figure 27 and Figure 28 are shown in Figure 25. The input, weight, and output of the left and right ICO from the trajectories Figure 27 and Figure 28 are shown in Figure 30.



(a) Iteration 0.



(b) Iteration 1.



(c) Iteration 2.

Fig. 19: Illustration of the trajectory (red) when obstacle avoiding a box. Videos of the iterations can be found here: iteration 0, iteration 1, and iteration 2.

APPENDIX C
TARGET FOLLOWING WITH INPUT CORRELATION LEARNING

Left target following with human:

APPENDIX D
TRAINING OF SEMANTIC SEGMENTATION NETWORKS

The training and validation curve for both U-Net and PSP-Net is shown in Figure 35. Each network is first trained for 20 epochs with the encoder layer frozen and then afterwards for 80 epochs with all layers unfrozen.

Examples of the output of U-Net and PSP-Net can be seen in Figure 36.

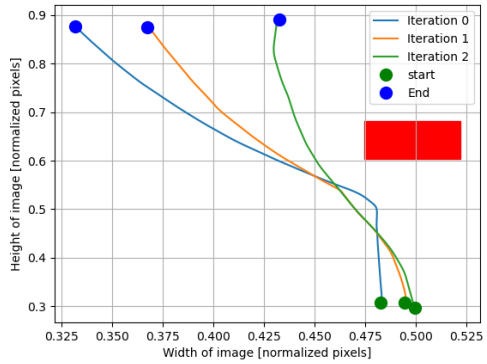


Fig. 20: Illustration of trajectories when obstacle avoiding a box. The red square illustrates the obstacle. The start and end point is visualised with a green and blue circle, respectively.

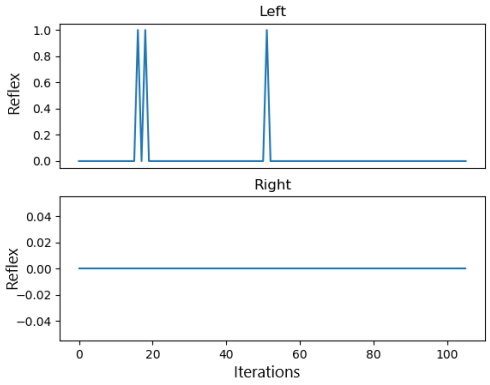


Fig. 21: Reflex signal for left and right ICO under the experiment in Figure 19 and Figure 20.

APPENDIX E
TRAINING OF YOLOV5

The dataset used to train the object detector Yolov5 consists of 143 images which is split into 114 training images and 29 validation images. Examples of the images are shown in Figure 37.

The model is trained for 100 epochs and achieves a mAP@0.5 of 0.995, which deemed acceptable. The results can be seen in Figure 38. Examples of the prediction made with Yolov5 are shown in Figure 39.

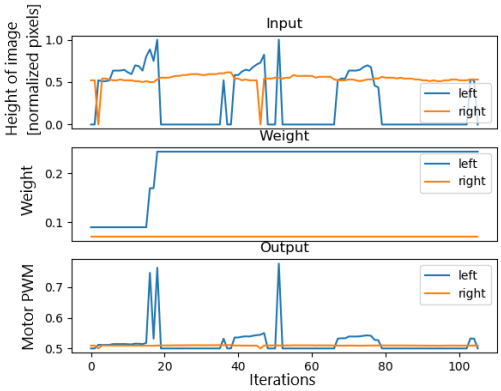
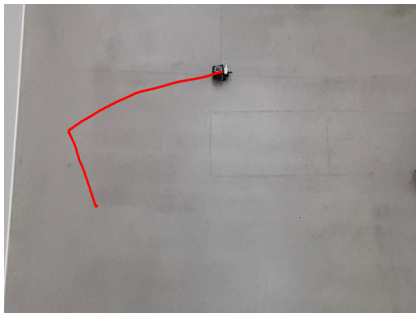
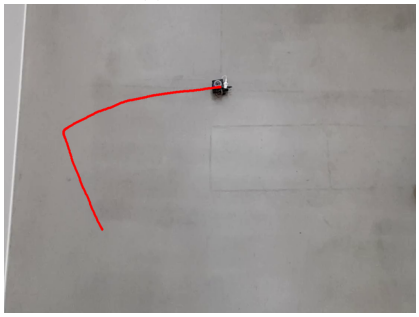


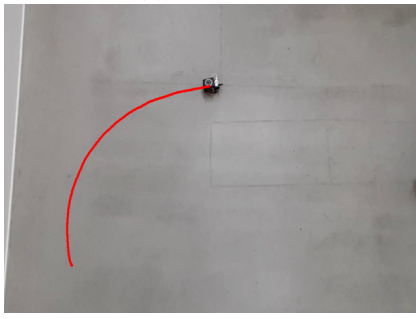
Fig. 22: Input, weight, and output values for both the left and right ICO when obstacle doing avoiding in Figure 19 and Figure 20.



(a) Iteration 0.



(b) Iteration 1.



(c) Iteration 2.

Fig. 23: Illustration of the trajectory (red) when obstacle avoiding a wall. Videos of the iterations can be found here: iteration 0, iteration 1, and iteration 2.

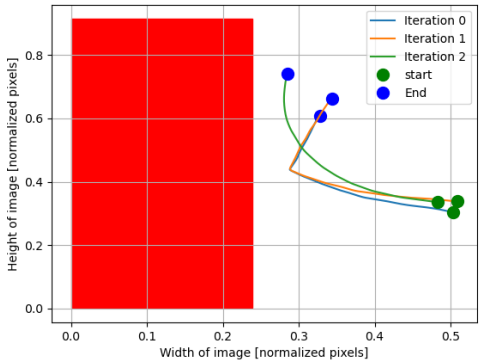


Fig. 24: Illustration of trajectories when obstacle avoiding a wall. The red square illustrates the obstacle. The start and end point is visualised with a green and blue circle, respectively.

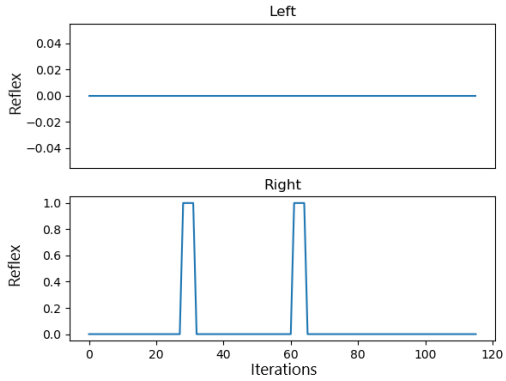


Fig. 25: Reflex signal from the trajectories in Figure 23 and Figure 24

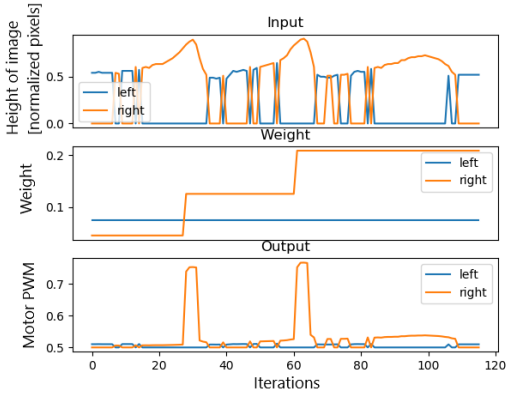
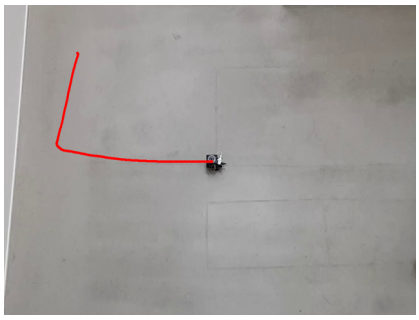
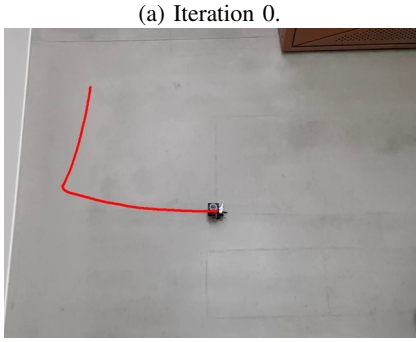


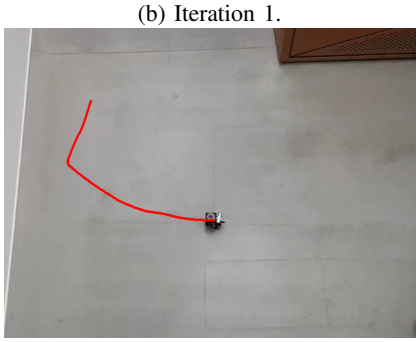
Fig. 26: Input, weight, and output values for both left and right ICO from the trajectories Figure 23 and Figure 24.



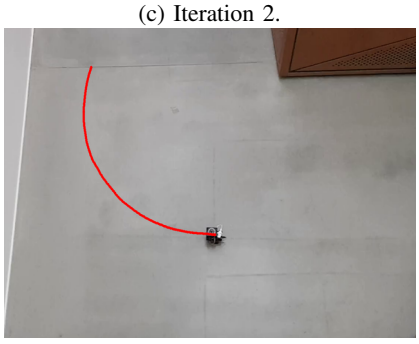
(a) Iteration 0.



(b) Iteration 1.



(c) Iteration 2.



(d) Iteration 3.

Fig. 27: Illustration of the trajectory (red) when obstacle avoiding a wall. Videos of the iterations can be found here: iteration 0, iteration 1, iteration 2, and iteration 3.

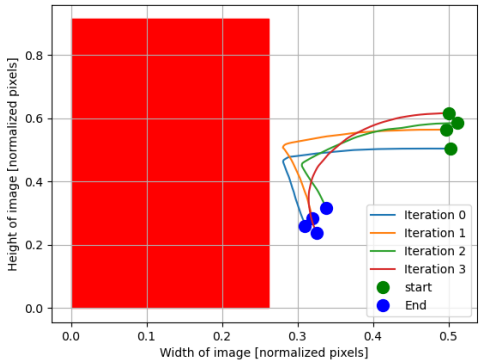


Fig. 28: Illustration of trajectories when obstacle avoiding a wall. The red square illustrates the obstacle. The start and end point is visualised with a green and blue circle, respectively.

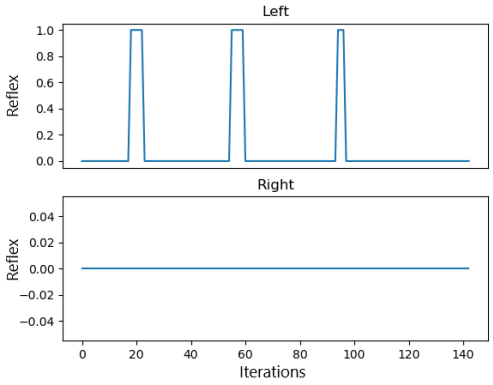


Fig. 29: Reflex signal from the trajectories in Figure 27 and Figure 28.

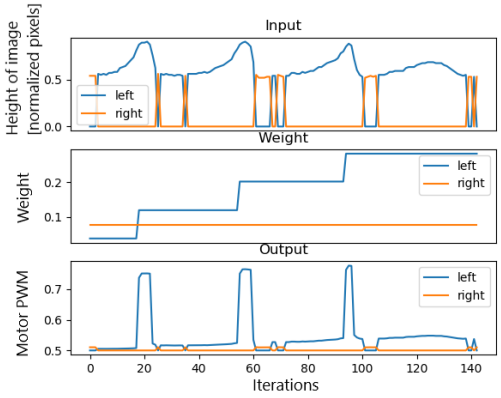
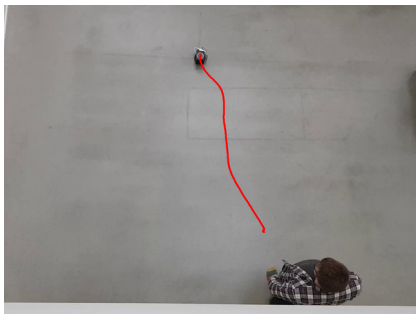
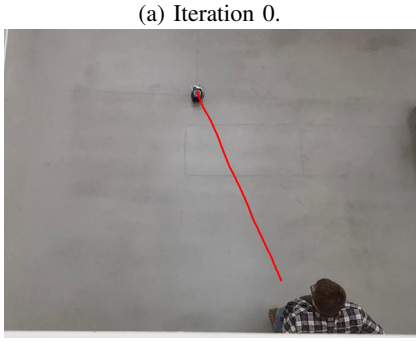


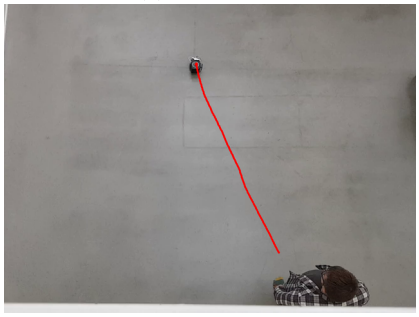
Fig. 30: Input, weight, and output values for both left and right ICO from the trajectories Figure 27 and Figure 28.



(a) Iteration 0.



(b) Iteration 1.



(c) Iteration 2.

Fig. 31

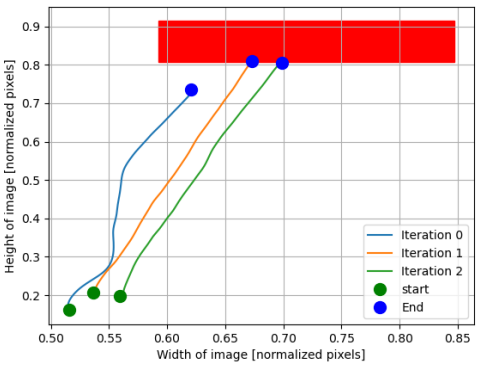


Fig. 32

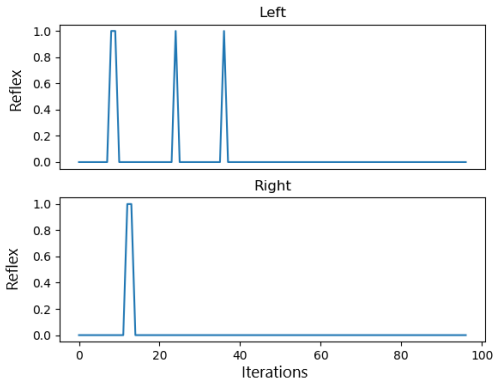


Fig. 33

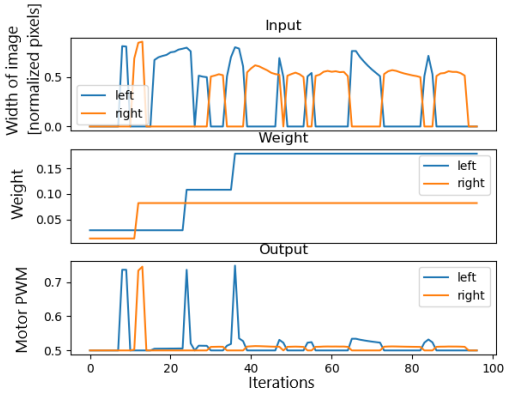
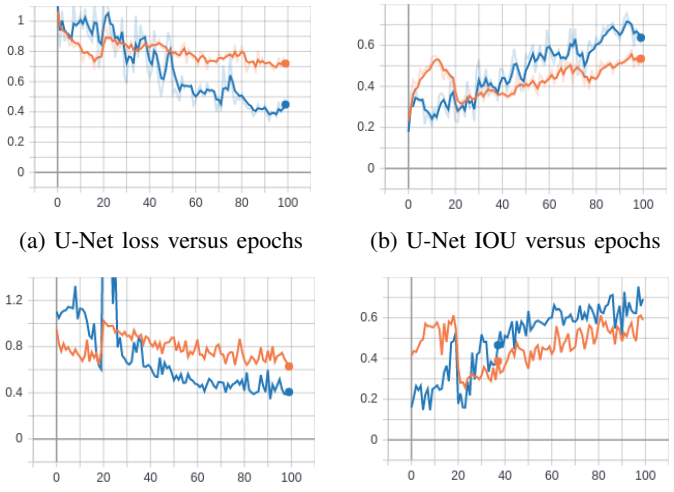


Fig. 34



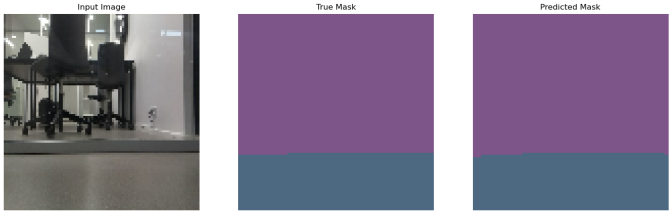
(a) U-Net loss versus epochs

(b) U-Net IOU versus epochs

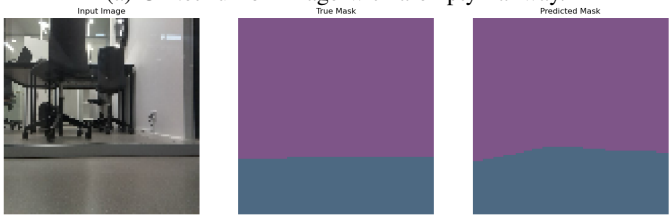
(c) PSP-Net loss versus epochs.

(d) PSP-Net IOU versus epochs.

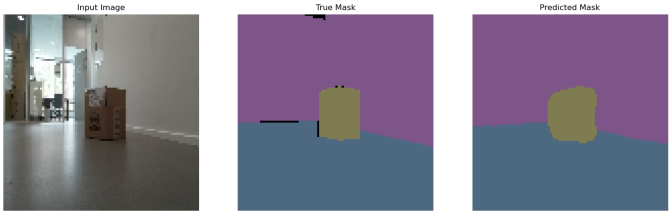
Fig. 35: Training (orange curve) loss and IOU and validation (blue curve) loss and IOU for U-Net and PSP-net.



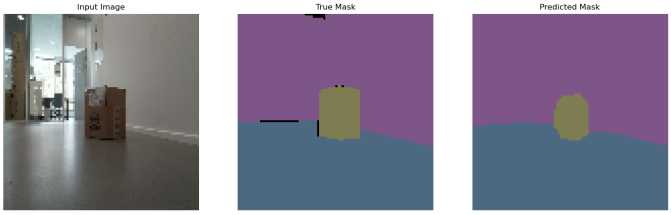
(a) U-Net run on image with a empty hallway.



(b) PSP-Net run on image with a empty hallway.



(c) U-Net run on image with a box in the hallway.



(d) PSP-Net run on image with a box in the hallway.

Fig. 36: Example of validations images run on U-Net and PSP-Net.

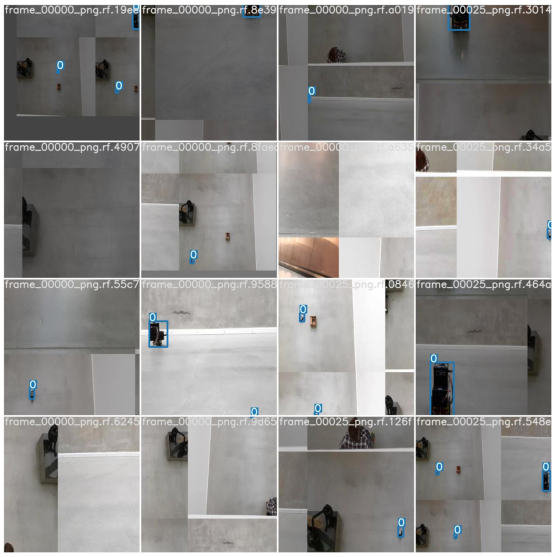


Fig. 37: Example of images used for training yolov5.

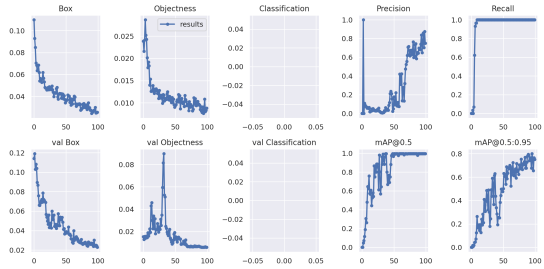


Fig. 38: Results of Yolov5 on the dataset illustrated in Figure 37

Fig. 39: Examples of prediction with Yolov5.