U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

MadDuck Avoidance

Intelligent Robotics - M.EIC - FEUP - 2022-2023

João Pires 201806079
Julián Herrero 202202340
Lisa Sonck 202202272

Agenda

- MadDuck Avoidance
- DuckieRobot and DuckieTown Simulator
- DuckieRobot Basic Movement
- Object Detection
- Object Avoidance
- Conclusions
- Acknowledgements
- Demonstration Video
- References

MadDuck Avoidance

This project focus on two main parts:

- Object detection
- Object avoidance

The purpose is to use DuckieTown and its DuckieRobot to recognize other duckies during its movement and avoid them, adapting the movement.

DuckieRobot and DuckieTown Simulator

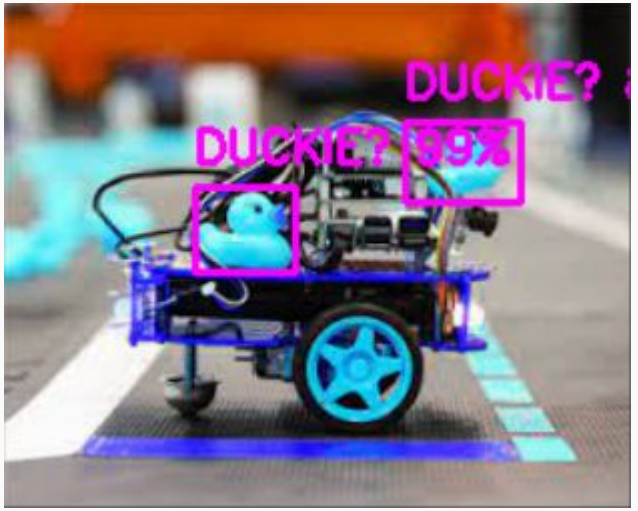
- Physical DuckieRobot not working
- Alternative: use the DuckieTown Simulator
  - Gym-Duckietown (Python/OpenGL - Pyglet)
- Custom map built to be used in the implementation
  - Usage of ‘straight/N’ and ‘floor’ tiles
  - Duckies as the only obstacle, positioned in different places to assess DuckieRobot’s behaviour

DuckieRobot Basic Movement

- Manual movement using the arrow keys
- Automatic movement by pressing the ‘space’ key
  - Straight line movement if no duckie found
  - Curvature around the identified duckie
- In both cases, the linear and angular velocities are being changed to reflect on the DuckieRobot movement

Object Detection

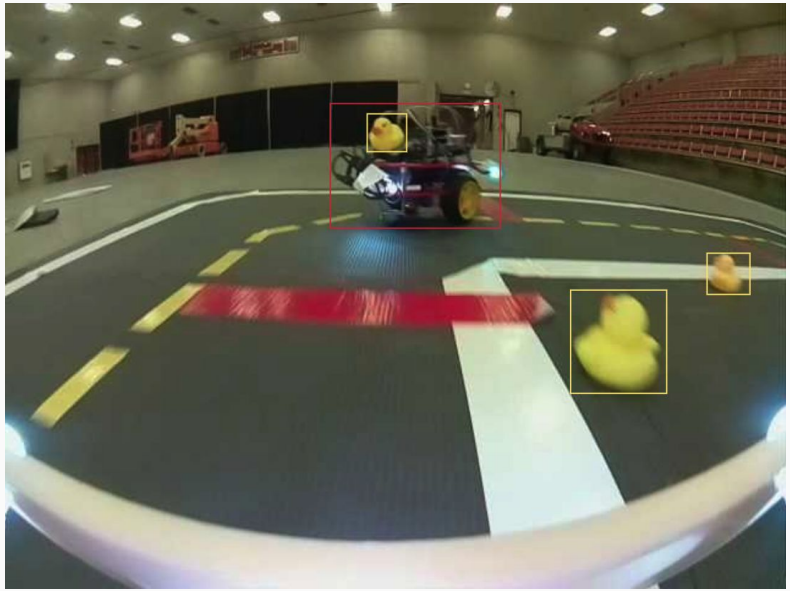
- Dataset
- Yolov3



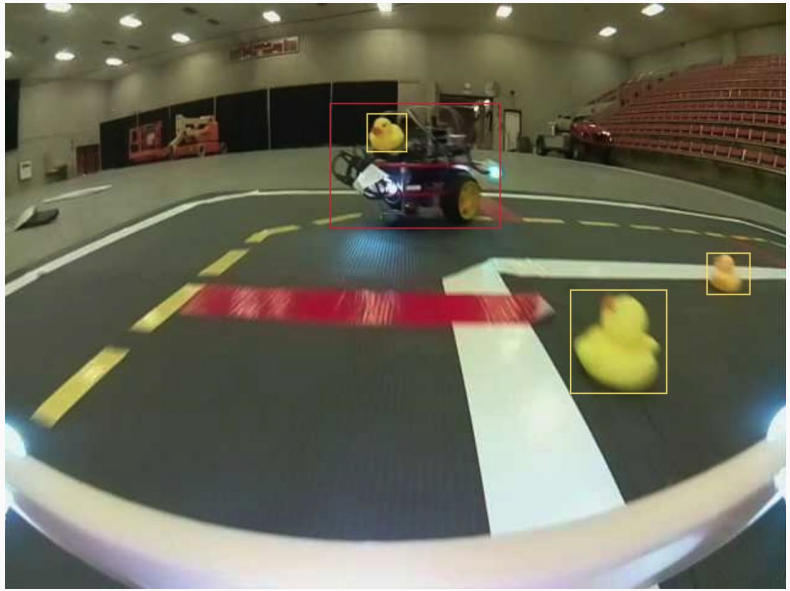
Object Detection

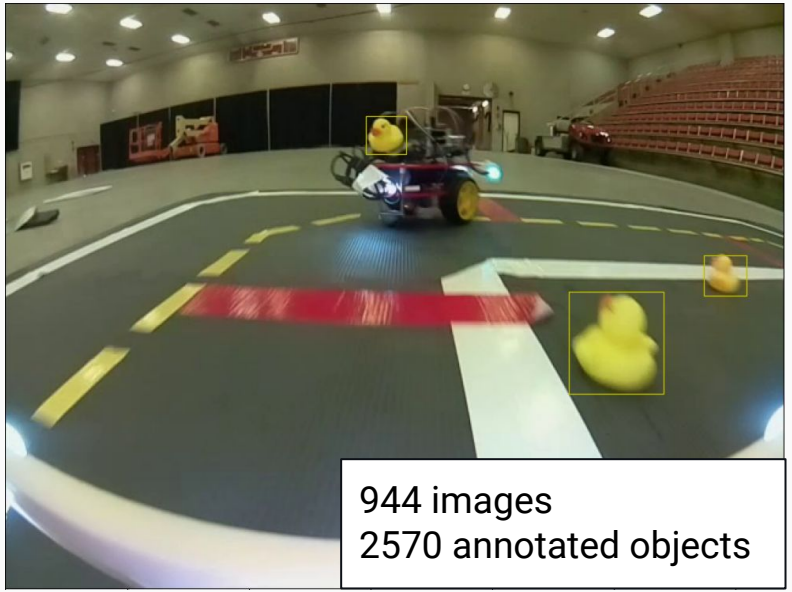
Dataset

- The Duckietown Object Detection Dataset
- 1956 images
- 5068 annotated objects



Object Detection





Object Detection

Yolov3

Darknet
Basic Yolov3 model

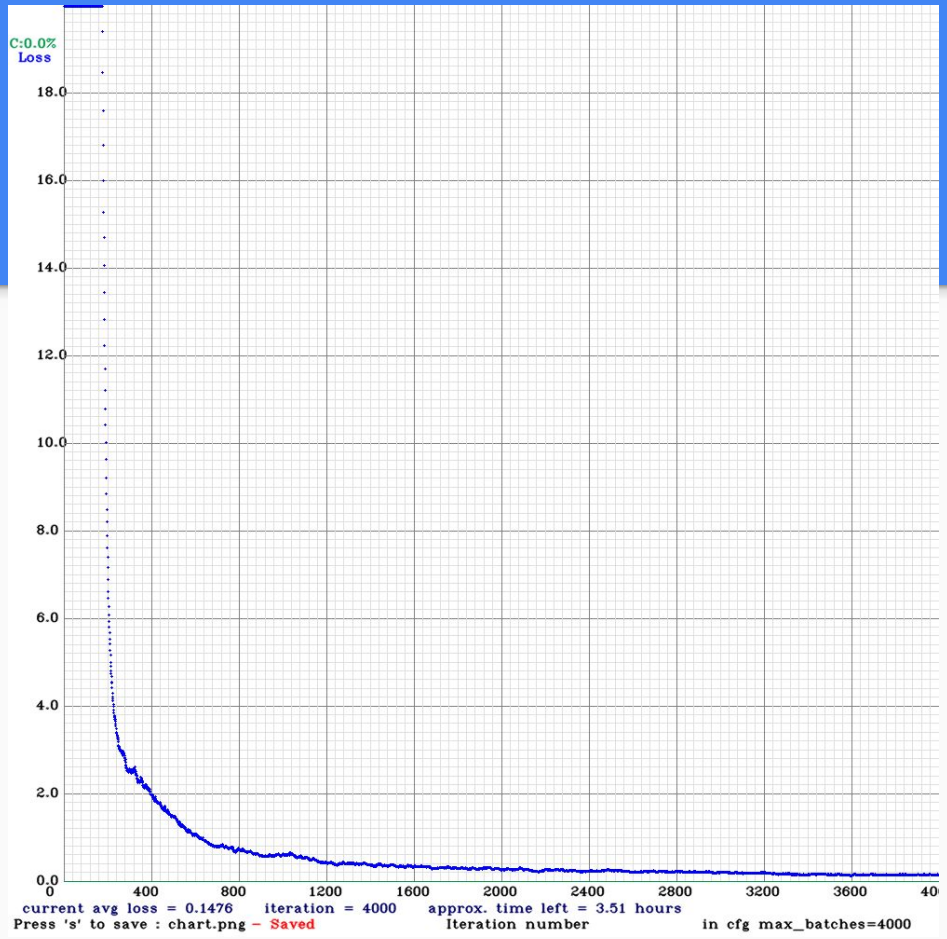
1. Preparation
    a. Change format: <class> <x> <y> <width> <height>
2. Implementation
    a. Modify the Yolov3 model

Object Detection

Yolov3

3. Results

Train set: 630 images
Duration: 5 days, 4000 iterations
Test set: 315 images
Average Precision = 90%



Object Detection

Yolov3

3. Results

   Train set: 630 images
   Duration: 5 days, 4000 iterations
   Test set: 315 images
   Average Precision = 90%

Object Avoidance

- Every 10 frames, the current frame is analyzed to check for duckies
- If a duckie is detected:
  - The first verification is to check how close the duckie is. If not too close, keep the straight line movement.
  - If the duckie is close to the DuckieRobot, then it checks if is out of reach, i.e. on one of the sides, out of the straight line movement.
  - If it's in reach, depending whether the duckie is more to the left or more to the right of the DuckieRobot, it will do a curvature in the opposite direction, keeping the straight line movement afterwards.

Conclusions

- Good results obtained in the object detection and avoidance parts
- Proven that it’s possible to make the DuckieRobot detect and avoid obstacles
- Faster development cycle by using the DuckieTown Simulator rather than the physical DuckieRobot

Acknowledgements

- Application of Yolov3 with an annotated dataset to the object detection part
- Implementation of the logic to avoid a detected duckie
- Main limitations:
  - Built DuckieRobot can only move forward and avoid obstacles
    - Future work: make it move freely inside any map, detecting the road
  - Multiple side-by-side duckies avoidance not implemented
  - Not able to test on the physical robot

Demonstration Video

References

[1] S. Saryazdi, D.Bhatt. "The Duckietown Object Detection Dataset". Github.com. https://github.com/saryazdi/Duckietown-Object-Detection-LFV/blob/master/DuckietownObjectDetectionDataset.md.

[2] V. Meel. "YOLOv3: Real-Time Object Detection Algorithm (Guide)". viso.ai. https://viso.ai/deep-learning/yolov3-overview/

[3] J. Redmon. "YOLO: Real-Time Object Detection". pjreddie.com. https://pjreddie.com/darknet/yolo/.

[4] DuckieTown. "DuckieTown Simulator Maps". github.com. https://github.com/duckietown/gym-duckietown/tree/master/gym_duckietown/maps.

[5] DuckieTown. "DuckieTown Simulator". docs.duckietown.org. https://docs.duckietown.org/daffy/AIDO/draft/dt_simulator.html.