Autonomous Car Chasing

Pavel Jahoda[2][0000−0002−3363−6406], Jan Cech[1][0000−0002−2181−5917], and Jiri Matas[1][0000−0003−0863−4844]

[1] Visual Recognition Group, Czech Technical University in Prague
[2] Faculty of Information Technology, Czech Technical University in Prague

Abstract. We developed an autonomous driving system that can chase another vehicle using only images from a single RGB camera. At the core of the system is a novel dual-task convolutional neural network simultaneously performing object detection as well as coarse semantic segmentation. The system was firstly tested in CARLA simulations. We created a new challenging publicly available CARLA Car Chasing Dataset collected by manually driving the chased car. Using the dataset, we showed that the system that uses the semantic segmentation was able to chase the pursued car on average 16% longer than other versions of the system. Finally, we integrated the system into a sub-scale vehicle platform built on a high-speed RC car and demonstrated its capabilities by autonomously chasing another RC car.

Keywords: autonomous driving, chasing, multi-task convolutional neural network, RC car, deep learning, CARLA, simulation

1 Introduction

An outgoing effort is made to enable autonomous systems to drive in a safe manner. To be safe, such a system needs to quickly react to unexpected situations in a dynamic environment. Especially at high speed. A car chase is a scenario, which requires dynamic maneuvers and tests the limits of an autonomous driving system. The key aspects of driving autonomously include an understanding of the surrounding environment, detecting other cars, and reacting to them on the road. In this paper, we present a novel neural architecture that gives information about other cars as well as the surrounding drivable surface. We then test an autonomous driving system that we developed, in the car chasing scenario.

To the extent of our knowledge, a car chasing scenario has not been presented. On the other hand, car following has been studied for more than half a century. Car following models how vehicles should follow each other in a traffic stream. Most of the theoretical models assume having precise continuous information about speed, distance, and acceleration of each car [2]. With the progress of machine learning and computer vision, more practical car-following models have been developed and tested. In 1991, Kehtarnavaz et al. developed the first autonomous car-following system called BART [11]. The system was able to follow the leading car at 20 km/h speed. However, car-following models have

typically not been tested in non-cooperative scenarios. Not to mention scenarios involving high speeds and sudden fast acceleration and deceleration encountered in car chases.

A car chase is a vehicular pursuit that typically involves high speeds and therefore dynamic driving maneuvers are necessary. In the car chasing scenario, a vehicle being pursued is driven by a person, while the vehicle that is chasing it is controlled by an autonomous system. Addressing an autonomous car chasing problem typically requires a complex methodology including modules of computer vision, planning, and control theory. Our vision-based system has an architecture that is similar in spirit to the DARPA Urban Challenge vehicles [25, 4, 5] and consists of three parts: perception and localization, trajectory planning, and a trajectory controller.

In the perception part, we introduce a novel neural network that detects objects and segments an image at the same time, i.e. in a single pass of the image through the network. It is used to find a 2D bounding box of the pursued object and segments an image into a coarse grid of $S \times S$ cells. In our case, each cell is either drivable or not. Based on the 2D bounding box detection, the angle and the distance between the two cars is estimated. After localizing the chased car, the trajectory is planned. The planning algorithm takes the segmentation of the drivable surface into account. Finally, we calculate the steer and throttle to drive the vehicle. A Pure Pursuit [23] inspired algorithm is developed to control the steering and a PID controller [24] is used to control the throttle.

The contributions of the paper are as follows: (1) A novel dual-task CNN performing object detection and semantic segmentation simultaneously, (2) an algorithm for autonomous car chasing was proposed and both extensively evaluated on a simulated dataset and tested on a real sub-scale platform, and (3) the CARLA Car Chasing dataset together with a test protocol and error statistics.

2 Related work

To the best of our knowledge, no system for a non-cooperative car chasing scenario has been described in the literature. However, related problems have been extensively researched. Many of the early autonomous vehicles were able to drive by driving along a pre-specified path [19]. In 1985, Carnegie Mellon University demonstrated the first autonomous path-following vehicle [23]. A year later, their Navlab vehicle was able to drive on a road at the top speed of 28 km/h [22].

Many attempts to address the car following problem focused on different parts of an autonomous system independently. In the 1990s, the first attempts of localization of the "lead" chased car were made [18, 17, 20]. Schwarzinger et al. focused on estimating dimensions of the "lead" car [18] while Gohring et al. focused on estimating the velocity of the chased car [8]. Our solution circumvents these non-trivial problems and works without the need of knowing the velocity of the chased car.

The problem of autonomously chasing an object has been investigated for unmanned aerial vehicles. Target chasing using UAVs is different to the car

chasing problem only in two aspects. Firstly, UAVs have very different kinematics and dynamics and move in more directions, which also means that altitude control is necessary. Secondly, frequently there is an assumption that no obstacles would prevent the UAV to move in all directions [15, 21, 13]. This is in contrast with autonomous chasing using road vehicles, where obstacle avoidance is a crucial part of the problem. At a high level, the architecture of an autonomous UAV is similar to the architectures used in autonomous vehicles [15, 21]. As an example of an autonomous UAV target chasing, Telière et al. developed an autonomous quad-rotor UAV that chases a moving RC car [21].

3 Proposed Method

3.1 Perception

Dual-task Neural Network simultaneously detects 2D bounding box of the pursued vehicle and predicts coarse semantic segmentation, i.e., segments an image into a coarse grid of $S \times S$ cells. The network shares the same backbone for both tasks – a 53 layer feature extractor called Darknet-53 [16]. Attached to the feature extractor are two sets of heads – for the object detection and for the image segmentation. The architecture of the neural network is depicted in Fig. 1.

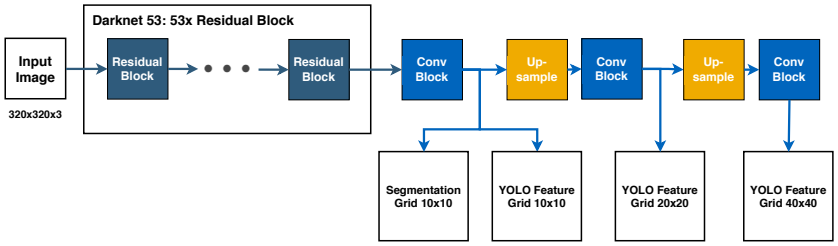


Fig. 1: Architecture of the novel dual-task convolutional neural network. The three YOLO Feature Grids are used to predict bounding boxes at 3 different scales [16]. The Segmentation Grid is the output of the coarse semantic segmentation.

The network was trained by alternating optimization – in every second batch, the network is optimized only for detection, while the segmentation is optimized in the remaining batches. The neural network uses different loss functions depending on the batch.

During inference, an image is passed through the network just once. The network provides an object detection as well as the semantic segmentation outputs. While the training is slightly slower than training a single-task neural network, the extra cost is negligible during inference since the backbone features

are shared. Fast execution and a smaller memory footprint of the dual-task network are especially important for real-time tests on embedded hardware with limited resources.

The YOLOv3 backbone [16] was pre-trained on the COCO dataset consisting of more than 330k images with objects from 80 different classes [14]. For the detection problem, the network was fine-tuned on a collected dataset of a few hundred images of the chased car called "Car Chasing Dataset". Only the final layers that follow the Darknet-53 feature extractor were updated.

The segmentation output provides a semantic map of the input image consisting of $S \times S$ cells, see Fig. 2. We set $S = 10$ and two classes: a drivable surface and a background. The logistic function is used to train the network.

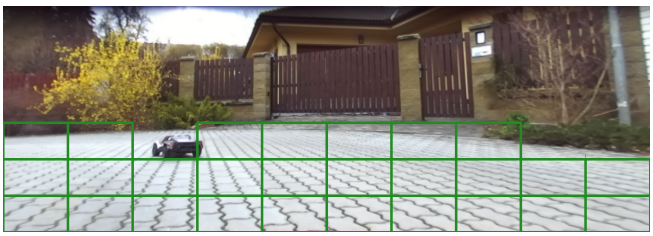


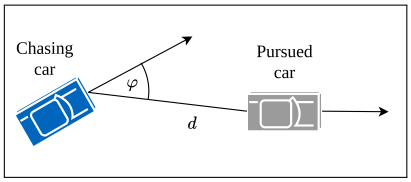
Fig. 2: Segmented image from the RC car camera. The image is segmented into 10×10 grid of cells. The green cells represent a drivable surface. The image is taken from the test set of the Car Chasing Dataset.

For coarse segmentation, we annotated the images from the Car Chasing Dataset. The images were first annotated by pixel-level precision, then 10×10 maps were generated by summing drivable pixels belonging to a cell. If the sum exceeds 50%, the cell was set as drivable, and as the background otherwise.

3.2 Chased Car Localization

Angle and Distance Estimation A relative angle and distance to the pursued car are sketched in Fig. 3.

Fig. 3: Angle and distance definition: Chasing car in blue and pursued car in gray. Distance d angle φ are estimated using a camera mounted on the chasing car.



To estimate the distance and the angle to the chased car from a camera image, we employ a PnP (Perspective-n-Point) problem [12] solver. Having a 3D

model – image correspondences and the intrinsic camera calibration matrix $\mathbf{K}$, a relative camera–model pose $\mathbf{R}, \mathbf{t}$ is estimated.

Known dimensions of the chased car (obtained from an RC manufacturer documentation), namely a tight virtual 3D bounding-box gives 3D model coordinates, and the image bounding box of the detection gives the corresponding image projections. We assume that the bounding box encloses only the corresponding image points, i.e., the back of the RC car. We acknowledge that the bounding box oftentimes encloses more than just the back of the car, but we found the resulting inaccuracies negligible. The camera calibration is known. An OpenCV [3] solver is used to estimate the model pose. Then, the distance d is calculated as Euclidean distance between the translation vector $\mathbf{t}$ and the origin. Similarly, we extract angle φ.

Angle and Distance Extrapolation The image-based detector may fail, especially due to poor lighting, motion blur, or because the pursued vehicle is only partially visible or fully occluded. Therefore, we extrapolate the angle and the distance of the chased car based on the previous estimates. The extrapolation algorithm uses an exponential moving average. Six variables store information about the whole history of estimated angles and distances. The idea is explained by the process of extrapolating distances.

For every frame, we update the exponential moving average variable using the following equation:

$$\hat{e_i} = \begin{cases} \alpha \cdot d_i + (1-\alpha) \cdot \hat{e}_{i-1}, & \text{if bbox found,} \\ \alpha \cdot d_{ex} + (1-\alpha) \cdot \hat{e}_{i-1}, & \text{otherwise.} \end{cases} \tag{1}$$

Variable $\hat{e_i}$ is the exponential moving average at frame i, parameter α is in range [0,1] and sets the weight of the last predicted distance. We set $\alpha = 0.5$ in all experiments.

If the chased car is detected at frame i, distance d_i is set to the predicted estimate received from the PnP solver. Then $\hat{e_i}$ is updated using d_i. In case no bounding box was detected, $\hat{e_i}$ is updated by

$$d_{ex} = 2 \cdot d_{i-1} - d_{i-2}. \tag{2}$$

Estimate $\hat{e_i}$ is the output for frame i in case no bounding box was detected. We set $d_i = d_{ex}$ for future computations.

The angle extrapolation is done similarly. Finally, each extrapolated angle is processed by a saturation block that limits the output values. The limits are -175 and 175 degrees. This method allows to extrapolate the angle and the distance very fast in a $\mathcal{O}(1)$ time and space complexity with almost no overhead.

3.3 Planning and Control

Pure Pursuit [23] Inspired Algorithm implements the lateral control of the vehicle. The Pure pursuit is a path tracking algorithm that moves a vehicle from

its current position to a look-ahead position on the path. The algorithm was invented in the 1980s as it was used in the first demonstration of an autonomous vehicle capable of following a road using imagery from a black and white camera [23]. The goal of the demonstration was to keep the vehicle centered on the road while it was driving at a constant speed.

We were inspired by the pure pursuit algorithm, however, a chasing car does not move at a constant speed and we are not following a path, but rather chasing a moving target, a car. Therefore, for each frame, the target position is updated with respect to the current position of the car. Unlike the pure pursuit algorithm, we do not calculate the curvature to control the steering. The steering is updated at every frame, such that the steering angle is set as the estimated angle φ described in Sec. 3.2. The steering angle is then mapped to the steering input of a test vehicle, see Sec. 3.4.

The lateral and longitudinal controllers are completely independent. The velocity of the car is not considered for turning, no limit of the steering angle based on current speed is set. In this basic algorithm, we simply assume that the dynamics of the chasing car is better or equal to the pursued car.

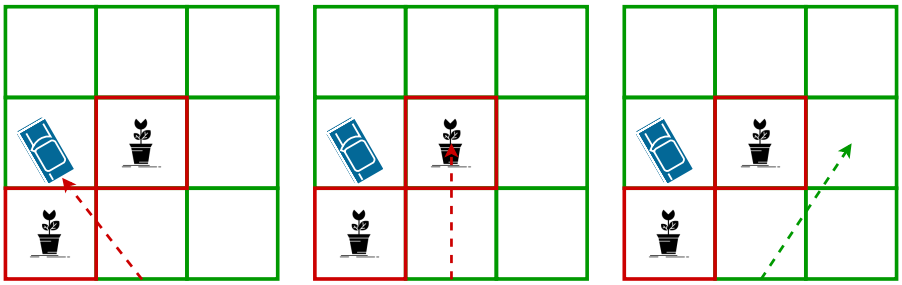
Semantic Segmentation Based Planning The above control algorithm assumes that the entire surface is drivable. If it is not the case, the car may collide with obstacles when pursuing the other car blindly by driving to the latest position of the car. The neural network provides a coarse grid of drivable surface segmentation, so we propose a simple algorithm that resolves the issue.

The idea is illustrated in Fig. 4. First, we construct a line segment from the center of the bottom side of the image to the bounding box of the detected car. If this line segment goes only through cells that are drivable, the trajectory planning phase stays unchanged and we try to follow the car directly as described above. Otherwise, a new target is found. A line segment going to other cells on the same row as the bounding box of the car. If the line segment is all drivable, the endpoint is the new target. If not, the process is repeated until a possible target is found or the algorithm stops if an acceptable target does not exist.

PID Controller We used a PID controller to maintain a desired distance between the two cars. The controller actuates the throttle of the vehicle. The three constants of the controller were set with respect to the longitudinal dynamics of the vehicle. The setting was found by testing over a set of weights and picked the one that resulted in the lowest absolute average error $\varepsilon = d - d_{desired}$ in simulations and adjusted to fit the real sub-scale vehicle dynamics empirically. We set the controller to be rather slow but highly stable in our experiments.

3.4 Implementation details

We tested the proposed algorithm in CARLA simulations and in real sub-scale platform. The steering angle to the steering input was mapped to respective vehicles. Steering input is in range [-1,1], which corresponds to steering from



(a) (b) (c)

Fig. 4: Illustration of the segmentation based planning. First, the algorithm checks if it is possible to drive directly towards the car **(a)**. This is not possible as the constructed line segment goes through a non-drivable grid cell. Then, the remaining cells on the same row as the detected car are checked **(b,c)**. The algorithm finally proposes a direction represented by a line segment that goes only through drivable surfaces and that is closest to the direction of the chased car **(c)**.

maximum left to maximum right. A linear mapping was found for the simulator, while a non-linear mapping $steer \propto (\varphi/180)^2$ was used for controlling the RC car platform. Similarly with the throttle input, a value in range [0,1] is expected. No torque is 0, while full throttle is 1. PID controller setting was: $w_p = 0.1$, $w_i = 0$, and $w_d = 1$ (proportional, integration and derivative component weights) respectively for CARLA simulator, while $w_p = 0.25$, $w_i = 0$, and $w_d = 0.07$ for the real vehicle. The output of the controller was clipped to [0,1] range.

4 Simulation Experiments

4.1 CARLA Environment

All the simulation experiments were performed in CARLA – an open-source simulator for autonomous driving research [6]. The experiments were done in version 0.9.8. CARLA has a wide selection of sensors including camera, segmentation camera, LIDAR, etc. It provides multiple maps with an urban layout. Users can control the weather, as well as many different actors (vehicles, pedestrians) at the same time.

4.2 Experimental Setup

We chose Tesla Model 3 as the chased car and Jeep Wrangler Rubicon as the chasing car. In each experiment, the pursued Tesla Model 3 was placed in front of the autonomously driven Jeep, as shown in Fig. 5. At each frame, we simulated a bounding box detection of the chased car. We did so by projecting the ground truth world coordinates of the chased vehicle bounding box to the camera image

coordinates. Then, we randomly perturbed the bounding box by adding random exponential noise proportional to the size of the bounding box. Furthermore, we simulated detection failure (false negative) in 10% of the frames. The detection was not provided when the chased car was out of the field of view or occluded.

All the experiments were performed in a synchronous mode in which the simulation waits for the sensor data to be ready before sending the measurements as well as halts each frame until a control message is received [6].

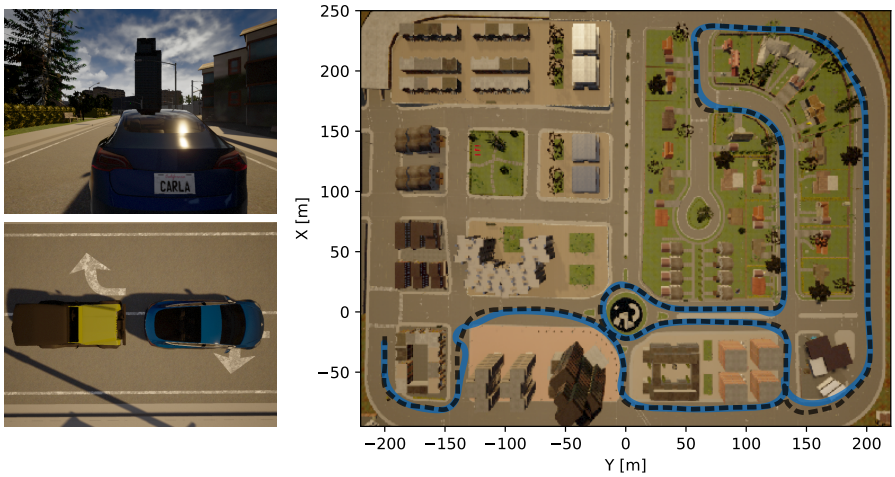


Fig. 5: CARLA experiment. **Top-left image:** view from the chasing car. **Bottom-left image:** chasing and pursued car from above. **Right image:** An example of a difficult drive trajectory. The blue and dashed-black lines are trajectories of the chasing and pursued car respectively.

CARLA Car Chasing Dataset To perform the experiments we first collected a driving dataset (available on our GitHub repository [9]). We used the Tesla Model 3 and manually drove it around the city using a connected steering wheel and pedals Hama uRage GripZ to generate the trajectory of the pursued car. The location and orientation were recorded at every frame. This way, we have collected a database of 20 different drives with varying difficulty: 10 easy, and 10 challenging rides. The rides are about 1 minute long on average.

The easy subset contains simple drives in which the car was driven slowly and without sudden turning and braking. Any changes such as turns were slow. The average speed in these drives is 35.01 km/h. While the difficult subset aims to test the algorithm to its limits. The drives include sudden braking, fast turns as well as cutting corners. Sometimes even drifts were performed. The average speed in these drives is 49.68 km/h with the fastest drive having an average speed of 63.29 km/h.

We used the database to evaluate the autonomous chasing algorithm. At the beginning of the following experiments, a chasing car was placed half of a meter behind the chased car. Then, we updated every frame the location and orientation of the chased car based on the saved coordinates in the dataset. Meanwhile, the chasing vehicle was driven autonomously and trying to maintain a desired distance from the pursued car. Fig. 5 depicts the initial setup as well as an example of a full trajectory of an autonomous chase in CARLA.

4.3 Evaluation for the autonomous chasing algorithm

In the first experiment, we compared different versions of the algorithm. We evaluate the algorithm with its full functionality enabled, and also ablated algorithm without using coarse image segmentation and without the segmentation and distance and angle extrapolation. We compare these versions on both of the easy and difficult sets described in Sec. 4.2.

The evaluation was done using five statistics: (1) An average root-mean-square error (RMSE), and (2) mean absolute error (MAE) of the desired distance and the actual distance between the two cars calculated over frames of all trials. (3) The average number of crashes per drive. A crash can be a collision with the environment (building for example) or with the pursued car. (4) The average drive completion, which is based on how long the autonomous system was chasing the pursued car. It is calculated as follows. The closest point of the pursued car trajectory to the last position of the chasing car is found. The drive completion is the percentage of the length of the trajectory to this point over the entire trajectory length. The statistics are averaged over the test rides. (5) The chase is considered to be successfully finished if the drive was at least 95% completed.

Easy Set	Finished	Avg. Completion	Crashes	MAE	RMSE
Full Algorithm	10	97.48%	0.1	9.28	10.91
W/o Segmentation	10	97.41%	0.1	9.26	10.90
W/o Segm. + Ex.	9	97.03%	0.11	9.34	11.23
Difficult Set	Finished	Avg. Completion	Crashes	MAE	RMSE
Full Algorithm	4	**63.84%**	1.5	14.39	18.30
W/o Segmentation	4	54.76%	1.25	14.30	18.22
W/o Segm. + Ex.	3	53.29%	1	14.49	18.89

Table 1: Results on CARLA Chasing dataset

The results are summarized in Tab. 1. As opposed to the difficult driving set, all three versions of the system performed well on the easy test set with even the simplest version finishing most of the drives. On the difficult subset, we can see, that the full algorithm performed significantly better than the remaining two versions. It achieved almost 10 percentage points higher drive completion on average than the next best-evaluated version.

Qualitative summary While performing the experiments, we saw that the use of coarse semantic segmentation allowed the system to stay longer on the road in situations when the pursued car was not detected for a prolonged period. For example, when the pursued car drove behind a building, both versions of the algorithm that use extrapolation correctly predicted the position of the pursued car. However, the algorithm that did not use segmentation drove straight into the building and collided as it did not have any information about the surrounding environment. Only the full version of the algorithm drove around the building and continued the chase. For demonstrations, see [9].

4.4 Impact of the Detector Quality on Chasing Success

We measured the effect of the detector accuracy on the chasing results. We simulated the algorithm with a detector of different miss rates. Failed detection have a uniform distribution over the frames of the sequence. All three versions of the algorithm were tested. Results are shown in Fig. 6.

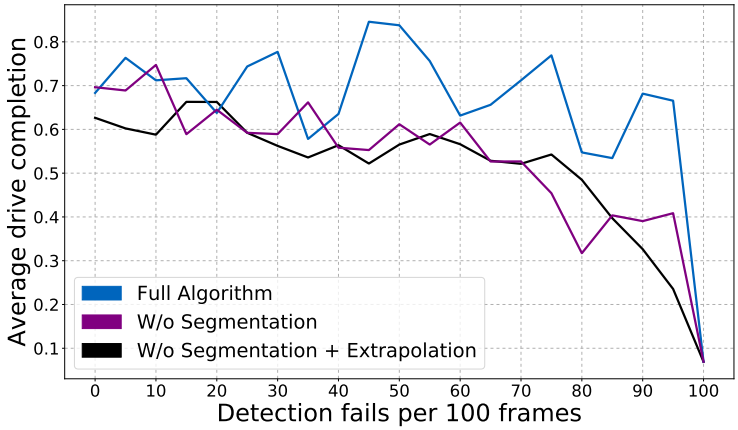


Fig. 6: Average drive completion plotted against detection recall. Only difficult drives were included.

We can see, that for the two versions that do not use semantic segmentation, the average drive completion steadily declines with the detection failure rate. The full version consistently maintains the highest drive completion for all detectors. When the recall is at 60% or lower, the average drive completion of the full version is almost 20% higher than the average drive completion of the remaining two versions.

Note that non-monotonic behaviour is caused by the randomness of the detection failure events. Averaging over several random trials is computationally very expensive since the simulations for all test rides need to be computed. Nevertheless, the trend is apparent.

5 Real Experiments

Chasing RC car For the deployment of the autonomous chasing algorithm, a sub-scale vehicle platform called ToMi was used [1]. The platform is built around a large 1:5 scale RC car "Losi Desert Buggy XL-E 4WD". This electrically powered car has 0.9×0.5 meters length and width dimensions with reported maximum speed of up to 80 km/h. The platform has a Raspberry Pi generating pulse width modulation (PWM) signals for the throttle and to the steering servomotor. It is equipped with ZED stereo camera (used as a monocular camera) for taking color HD images at 30 FPS and NVIDIA Jetson AGX Xavier with GPU for image processing and neural network inference.

The whole simplified process flow is as follows. First, a single image of the stereopair is taken by the camera, which then goes to the Jetson and the GPU where it is analyzed by the neural network. Then, the planning and control algorithm computes the steer and throttle commands, that are transmitted to the Raspberry Pi which finally generates signals to the motor controller and servomotor that actuates the vehicle.

Chased RC car The RC car used as the pursued vehicle is a 1:10 scale "Losi XXX-SCT Brushless RTR". This electrically powered car has 0.55×0.29 meters length and width dimensions with reported maximum speed of up to 55 km/h. Both vehicles are shown in Fig. 8.

5.1 Vehicle Detection

Dataset In order to train and evaluate an object detector capable of predicting a 2D bounding box around the chased RC car in an image, the Car Chasing Dataset was collected. The dataset consists of 460 manually annotated images. All image annotations are in the PASCAL VOC data format introduced in the PASCAL Visual Object Classes Challenge [7]. Each image has its annotation that contains pixel coordinates of the bounding box corners for each object in the image. A single object per image is present – the chased RC car. The data were collected on various surfaces, and under different lighting conditions. The majority of images were collected by the platform camera. Around 5% of the collected images were taken with a smartphone.

Detection Results The collected dataset was randomly split into three sets: Training (80%), test (10%), and validation (10%) sets. To evaluate the model, we calculated several statistics – recall, precision, XY loss, and WH loss.

Following Pascal VOC, the prediction is true positive (tp) when IoU (Intersection over Union) between the predicted bounding box and the ground truth bounding box is greater than 0.5. XY loss is a mean squared error (MSE) between the center of the predicted bounding box and the center of the ground truth bounding box. WH loss is the difference between width and height of the predicted bounding box and the ground truth bounding box. When calculating

both of these statistics, coordinates and image dimensions were scaled to the [0, 1] range.

During training, the model with the smallest sum of the WH and XY losses on the validation subset was selected. The evaluation of the detector on the test set is shown in Table 2. Results indicate that the model detects the chased car accurately. This is achieved with only a few hundred training examples. Examples of the detection, together with angle and distance estimates are shown in Fig. 7.

	Precision	Recall	XY loss	WH loss
Test set	99.8%	97.7%	0.066	0.227

Table 2: Evaluation of the detector on a test set.

Fig. 7: Detection bounding boxes, distance, and angle estimation performed on test images acquired by autonomous car camera.

5.2 Segmentation Results

The segmentation accuracy of the dual-task network was evaluated on the independent test set of 46 images, the same instances as for the detection test. The neural network achieved 94.4% classification accuracy. False positive error (non-drivable cell predicted as drivable) is 3.4% while false negative rate 2.2%. The distribution of the drivable/non-drivable cells was 32/68 percent respectively. See Fig. 2 for an example of the segmentation result.

5.3 Test Drive Using RC Cars

We performed several live tests under different weather and lighting conditions. The system was tested on an empty roundabout as well as in a residential area

as depicted in Fig. 8. The chasing trajectories driven on a road leading to the roundabout included straight and curvy segments and a turn on the intersection. On the roundabout, the trajectories consisted of arc segments and full circles.

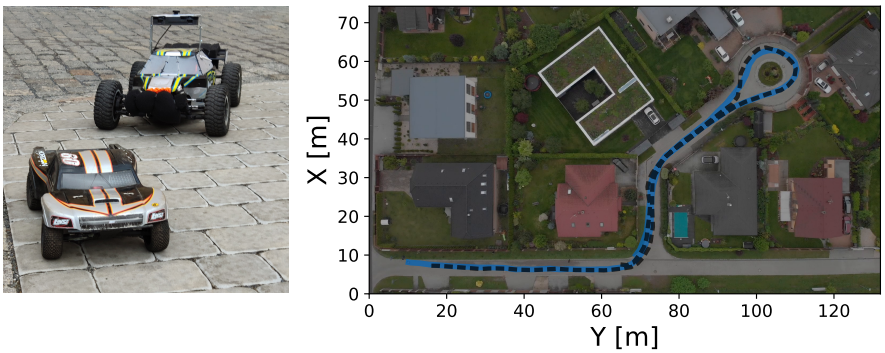
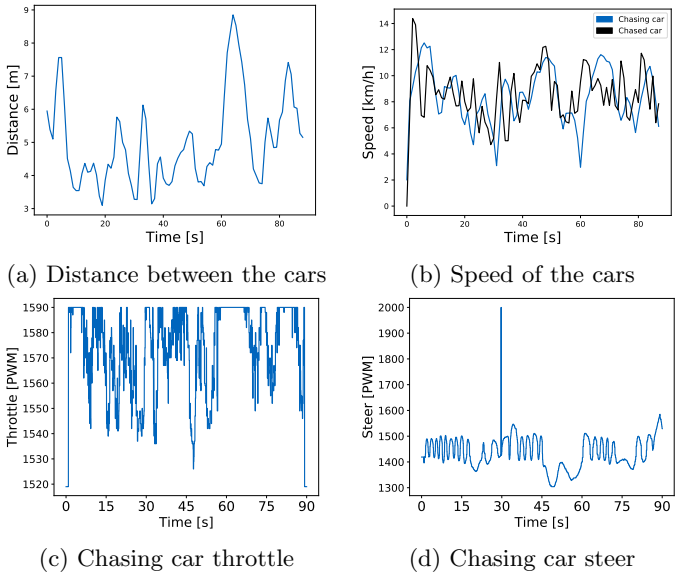


Fig. 8: **Left image:** RC cars used for testing. The autonomous car with the camera is farther away. **Right image:** an example of a real-world test captured by a drone flying 103 meters above the ground. Trajectories of both vehicles are shown: blue of the chasing and dashed black of the pursued car.

The autonomous system followed the other car smoothly without jerky movements. For the most part, it was able to successfully chase the other vehicle. It was maintaining the desired distance when the pursued car was driving in a straight line. If the chased car stopped, so did the autonomous system. A limitation of the system comes from its current reactive nature, which in certain rides affected the ability to make a U-turn on a narrow road.

For better insight, we provide several variables recorded over time for an autonomous chasing instance, see Fig. 9. The trajectory with timestamps of both cars is obtained by tracking their images in stabilized drone recordings. Thus we calculate the distance between the two cars and their velocities. Besides that, we show the throttle and steering commands. Note that the throttle was limited for safety reasons. The small fluctuation of the steering input is probably caused by improperly trimmed mechanics of the steering. The outlier of the steering at around 30 seconds was caused by a false positive detection.

All autonomous chasing drives were documented and a highlight video is available online [10] for illustration.



(a) Distance between the cars (b) Speed of the cars

(c) Chasing car throttle (d) Chasing car steer

Fig. 9: Trajectory-related and control variables in time for a ride shown in Fig. 8. The sudden spike in the graph **(d)** is caused by a false positive detection in a single frame

6 Conclusion

We have developed a system, capable of autonomously chasing another vehicle, using the novel dual-task network that simultaneously detects objects and predicts coarse semantic segmentation. The proposed system was extensively tested in CARLA simulator and on a real sub-scale vehicle platform [10].

A new challenging publicly available [9] chasing dataset was collected by manually driving the pursued car. We proposed evaluation statistics to measure success of chasing algorithms. Different versions of the system were compared. We have shown that the system benefits from using the coarse drivable surface segmentation and the trajectory extrapolation. Especially when the detector has a low detection recall. We believe that the CARLA Car Chasing dataset will serve as a benchmark for novel algorithms that could compare with ours.

Despite the simplicity of the proposed system, it shows robust chasing capabilities by using only information from a single RGB camera. One of the system limitations is its reactive nature. We believe that the system could improve by using a more sophisticated trajectory planning algorithm that would include predictive modeling of the chased car. In the future, we will evaluate the system more extensively with the RC cars.

Acknowledgement The research was supported by Toyota Motor Europe.

References

1. Bahnik, M., Filyo, D., Pekarek, D., Vlasimsky, M., Cech, J., Hanis, T., Hromcik, M.: Visually assisted anti-lock braking system. In: IEEE Intelligent Vehicles Symposium. IEEE Intelligent Vehicles Symposium (2020)
2. Brackstone, M., McDonald, M.: Car-following: a historical review. Transportation Research Part F: Traffic Psychology and Behaviour **2**(4), 181–196 (Dec 1999). https://doi.org/10.1016/s1369-8478(00)00005-x
3. Bradski, G.: The OpenCV Library. Dr. Dobb's Journal of Software Tools (2000)
4. Broggi, A., Cerri, P., Debattisti, S., Laghi, M.C., Medici, P., Panciroli, M., Prioletti, A.: PROUD-public road urban driverless test: Architecture and results. In: 2014 IEEE Intelligent Vehicles Symposium Proceedings. IEEE (Jun 2014). https://doi.org/10.1109/ivs.2014.6856478, https://doi.org/10.1109/ivs.2014.6856478
5. Buehler, M., Iagnemma, K., Singh, S.: The DARPA Urban Challenge: Autonomous Vehicles in City Traffic. Springer (2009)
6. Dosovitskiy, A., Ros, G., Codevilla, F., Lopez, A., Koltun, V.: CARLA: An open urban driving simulator. In: Proceedings of the 1st Annual Conference on Robot Learning. pp. 1–16 (2017)
7. Everingham, M., Gool, L., Williams, C.K., Winn, J., Zisserman, A.: The Pascal Visual Object Classes (VOC) Challenge. Int. J. Comput. Vision **88**(2), 303–338 (Jun 2010). https://doi.org/10.1007/s11263-009-0275-4
8. Gohring, D., Wang, M., Schnurmacher, M., Ganjineh, T.: Radar/lidar sensor fusion for car-following on highways. In: The 5th International Conference on Automation, Robotics and Applications. IEEE (Dec 2011). https://doi.org/10.1109/icara.2011.6144918, https://doi.org/10.1109/icara.2011.6144918
9. Jahoda, P.: Autonomous car chase. https://github.com/JahodaPaul/autonomous-car-chase (2020)
10. Jahoda, P.: Autonomous car chasing — czech technical university in prague. https://www.youtube.com/watch?v=SxDJZUTOygA (2020)
11. Kehtarnavaz, N., Griswold, N., Lee, J.: Visual control of an autonomous vehicle (BART)-the vehicle-following problem. IEEE Transactions on Vehicular Technology **40**(3), 654–662 (1991). https://doi.org/10.1109/25.97520
12. Lepetit, V., Moreno-Noguer, F., Fua, P.: EPnP: An accurate o(n) solution to the PnP problem. International Journal of Computer Vision **81**(2), 155–166 (Jul 2008). https://doi.org/10.1007/s11263-008-0152-6
13. Lin, F., Dong, X., Chen, B.M., Lum, K.Y., Lee, T.H.: A robust real-time embedded vision system on an unmanned rotorcraft for ground target following. IEEE Transactions on Industrial Electronics **59**(2), 1038–1049 (Feb 2012). https://doi.org/10.1109/tie.2011.2161248
14. Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollar, P., Zitnick, C.L.: Microsoft COCO: Common objects in context. In: Computer Vision ECCV 2014, pp. 740–755. Springer International Publishing (2014). https://doi.org/10.1007/978-3-319-10602-1_48
15. Rafi, F., Khan, S., Shafiq, K., Shah, M.: Autonomous target following by unmanned aerial vehicles. In: Gerhart, G.R., Shoemaker, C.M., Gage, D.W. (eds.) Unmanned Systems Technology VIII. SPIE (May 2006). https://doi.org/10.1117/12.667356
16. Redmon, J., Farhadi, A.: YOLOv3: An incremental improvement. ArXiv preprint (2018), http://arxiv.org/abs/1804.02767

17. Schneiderman, H., Nashman, M., Lumia, R.: Model-based vision for car following. In: Schenker, P.S. (ed.) Sensor Fusion VI. SPIE (Aug 1993). https://doi.org/10.1117/12.150245
18. Schwarzinger, M., Zielke, T., Noll, D., Brauckmann, M., von Seelen, W.: Vision-based car-following: detection, tracking, and identification. In: Proceedings of the Intelligent Vehicles 92 Symposium. IEEE (1992). https://doi.org/10.1109/ivs.1992.252228
19. Snider, J.M.: Automatic steering methods for autonomous automobile path tracking. Tech. Rep. CMU-RI-TR-09-08, Carnegie Mellon University, Pittsburgh, PA (February 2009)
20. Sukthankar, R.: RACCOON: A real-time autonomous car chaser operating optimally at night. In: Proceedings of the Intelligent Vehicles 93 Symposium. pp. 37–42. IEEE (08 1993). https://doi.org/10.1109/ivs.1993.697294
21. Teuliere, C., Eck, L., Marchand, E.: Chasing a moving target from a flying UAV. In: 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems. IEEE (Sep 2011). https://doi.org/10.1109/iros.2011.6094404
22. Thorpe, C., Herbert, M., Kanade, T., Shafer, S.: Toward autonomous driving: the cmu navlab. i. perception. IEEE Expert **6**(4), 31–42 (1991)
23. Wallace, R., Stentz, A., Thorpe, C., Maravec, H., Whittaker, W., Kanade, T.: First results in robot road-following. In: Proceedings of the 9th International Joint Conference on Artificial Intelligence - Volume 2. p. 1089–1095. IJCAI'85, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (1985)
24. Ziegler, J.G., Nichols, N.B.: Optimum settings for automatic controllers. Journal of Dynamic Systems, Measurement, and Control **115**(2B), 220–222 (Jun 1993). https://doi.org/10.1115/1.2899060, https://doi.org/10.1115/1.2899060
25. Ziegler, J., Bender, P., Schreiber, M., Lategahn, H., Strauss, T., Stiller, C., Dang, T., Franke, U., Appenrodt, N., Keller, C.G., Kaus, E., Herrtwich, R.G., Rabe, C., Pfeiffer, D., Lindner, F., Stein, F., Erbs, F., Enzweiler, M., Knoppel, C., Hipp, J., Haueis, M., Trepte, M., Brenk, C., Tamke, A., Ghanaat, M., Braun, M., Joos, A., Fritz, H., Mock, H., Hein, M., Zeeb, E.: Making bertha drive—an autonomous journey on a historic route. IEEE Intelligent Transportation Systems Magazine **6**(2), 8–20 (2014). https://doi.org/10.1109/mits.2014.2306552