

Learning to Map Vehicles into Bird’s Eye View

Andrea Palazzi, Guido Borghi, Davide Abati,
Simone Calderara and Rita Cucchiara

University of Modena and Reggio Emilia, Italy
`{name.surname}@unimore.it`

Abstract. Awareness of the road scene is an essential component for both autonomous vehicles and Advances Driver Assistance Systems and is gaining importance both for the academia and car companies. This paper presents a way to learn a semantic-aware transformation which maps detections from a dashboard camera view onto a broader bird’s eye occupancy map of the scene. To this end, a huge synthetic dataset featuring 1M couples of frames, taken from both car dashboard and bird’s eye view, has been collected and automatically annotated. A deep-network is then trained to warp detections from the first to the second view. We demonstrate the effectiveness of our model against several baselines and observe that it is able to generalize on real-world data despite having been trained solely on synthetic ones.

1 Introduction

Vision-based algorithms and models have massively been adopted in current generation ADAS solutions. Moreover, recent research achievements on scene semantic segmentation [9,14], road obstacle detection [3,12] and driver’s gaze, pose and attention prediction [7,22] are likely to play a major role in the rise of autonomous driving.

As suggested in [5], three major paradigms can be individuated for vision-based autonomous driving systems: *mediated perception* approaches, based on the total understanding of the scene around the car, *behavior reflex* methods, in which driving action is regressed directly from the sensory input, and *direct perception* techniques, that fuse elements of previous approaches and learn a mapping between the input image and a set of interpretable indicators which summarize the driving situation.

Following this last line of work, in this paper we develop a model for mapping vehicles across different views. In particular, our aim is to warp vehicles detected from a dashboard camera view into a bird’s eye occupancy map of the surroundings, which is an easily interpretable proxy of the road state. Being almost impossible to collect a dataset with this kind of information in real-world, we exclusively rely on synthetic data for learning this projection.

We aim to create a system close to surround vision monitoring ones, also called around view cameras that can be useful tools for assisting drivers during maneuvers by, for example, performing trajectory analysis of vehicles out from own visual field.

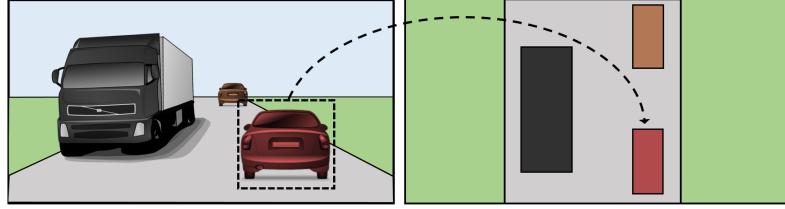


Fig. 1: Simple outline of our task. Vehicle detections in the frontal view (left) are mapped onto a bird-eye view (right), accounting for the positions and size.

In this framework, our contribution is twofold:

- We make available a huge synthetic dataset (> 1 million of examples) which consists of couple of frames corresponding to the same driving scene captured by two different views. Besides the vehicle location, auxiliary information such as the distance and yaw of each vehicle at each frame are also present.
- We propose a deep learning architecture for generating bird's eye occupancy maps of the surround in the context of autonomous and assisted driving. Our approach does not require a stereo camera, nor more sophisticated sensors like radar and lidar. Conversely, we learn how to project detections from the dashboard camera view onto a broader bird's eye view of the scene (see Fig.1). To this aim we combine learned geometric transformation and visual cues that preserve objects size and orientation in the warping procedure.

Dataset, code and pre-trained model are publicly available and can be found at <http://imagelab.ing.unimore.it/scene-awareness>.

2 Related work

Surround view Few works in literature tackle the problem of the vehicle's surround view. Most of these approaches are vision and geometry based and are specifically tailored for helping drivers during parking manoeuvres. In particular, in [13] a perspective projection image is transformed into its corresponding bird's eye view, through a fitting parameters searching algorithm. In [16] exploited the calibration of six fish eye cameras to integrate six images into a single one, by a dynamic programming approach. In [17] were described algorithms for creating, storing and viewing surround images, thanks to synchronized and aligned different cameras. Sung *et al.* [20] proposed a camera model based algorithm to reconstruct and view multi-camera images. In [21], an homography matrix is used to perform a coordinate transformation: visible markers are required in input images during the camera calibration process.

Recently, Zhang *et al.* [24] proposed a surround view camera solution designed

for embedded systems, based on a geometric alignment, to correct lens distortions, a photometric alignment, to correct brightness and color mismatch and a composite view synthesis.

Videogames for collecting data The use of synthetic data has recently gained considerable importance in the computer vision community for several reasons. First, modern open-world games exhibit constantly increasing realism - which does not only mean that they feature photorealistic lights/textures etc, but also show plausible game dynamics and lifelike autonomous entity AI [18,19]. Furthermore, most research fields in computer vision are now tackled by means of deep networks, which are notoriously data hungry in order to be properly trained. Particularly in the context of assisted and autonomous driving, the opportunity to exploit virtual yet realistic worlds for developing new techniques has been embraced widely: indeed, this makes possible to postpone the (very expensive) validation in real world to the moment in which a new algorithm already performs reasonably well in the simulated environment [23,8]. Building upon this tendency, [5] relies on TORCS simulator to learn an interpretable representation of the scene useful for the task of autonomous driving. However, while TORCS [23] is a powerful simulation tool, it's still severely limited by the fact that both its graphics and its game variety and dynamics are far from being realistic.

Many elements mark as original our approach. In principle, we want our surround view to include not only nearby elements, like commercial geometry-based systems, but also most of the elements detected into the acquired dashboard camera frame. Additionally, no specific initialization or alignment procedures are necessary: in particular, no camera calibration and no visible alignment points are required. Eventually, we aim to preserve the correct dimensions of detected objects, which shape is mapped onto the surround view consistently with their semantic class.

3 Proposed Dataset

In order to collect data, we exploit *Script Hook V* library [4], which allows to use Grand Theft Auto V (GTAV) video game native functions [1]. We develop a framework in which the game camera automatically toggle between frontal and bird-eye view at each game time step: in this way we are able to gather information about the spatial occupancy of the vehicles in the scene from both views (*i.e.* bounding boxes, distances, yaw rotations). We associate vehicles information across the two views by querying the game engine for entity IDs. More formally, for each frame t , we compute the set of entities which appear in both views as

$$E(t) = E_{frontal}(t) \cap E_{birdseye}(t) \quad (1)$$

where $E_{frontal}(t)$ and $E_{birdseye}(t)$ are the sets of entities that appear at time t in frontal and bird's eye view, respectively. Entities $e(t) \in E(t)$ constitute the

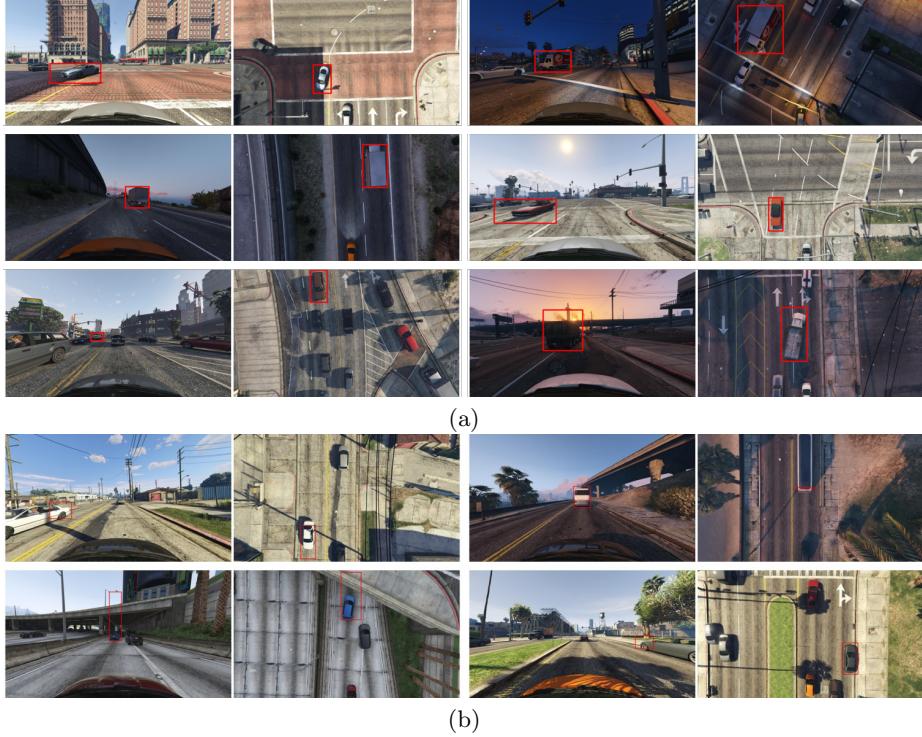


Fig. 2: (a) Randomly sampled couples from our GTAV dataset, which highlight the huge variety in terms of landscape, traffic condition, vehicle models etc. Each detection is treated as a separate training example (see Sec. 3 for details). (b) Random examples rejected during the post-processing phase.

candidate set for frame t $C(t)$; other entities are discarded. Unfortunately, we found that raw data coming from the game engine are not always accurate (Fig. 2). To deal with this problem, we implement a post-processing pipeline in order to discard noisy data from the candidate set $C(t)$. We define a discriminator function

$$f(e(t)) : C \mapsto \{0, 1\} \quad (2)$$

which is positive when information on dumped data $e(t)$ are reliable and zero otherwise. Thus we can define the final filtered dataset as

$$\bigcup_{t=0}^T D(t) \quad \text{where } D(t) = \{c_i(i) \mid f(c_i(t)) > 0\} \quad (3)$$

being T the total number of frames recorded. From an implementation standpoint, we employ a rule-based ontology which leverage on entity information (*e.g.* vehicle model, distance etc.) to decide if the bounding box of that entity can be considered reasonable. This implementation has two main values: first it's

	Total
Number of runs	300
Number of bounding boxes	1125187
Unique entity IDs	56454
Unique entity models	198

Table 1: Overview of the statistics on the collected dataset. See text for details.

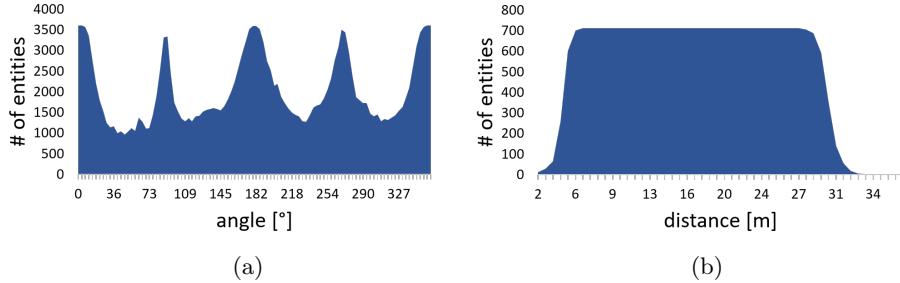


Fig. 3: Unnormalized distribution of vehicle orientation (a) and distances (b) present in the collected dataset. Distribution of angles conversely presents two prominent modes around $0^\circ/360^\circ$ and 180° respectively, due to the fact that the major part of vehicles encountered travel in parallel to the player’s car, on the same ($0/360^\circ$) or the opposite (180°) direction. Conversely, distance is almost uniformly distributed between 5 and 30 meters.

lightweight and very fast in filtering massive amounts of data. Furthermore, rule parameters can be tuned to eventually generate different dataset distribution (*e.g.* removing all trucks, keeping only cars closer than 10 meters, etc.).

Each entry of the dataset is a tuple containing:

- $frame_f, frame_b$: 1920×1080 frames from the frontal and bird’s eye camera view, respectively;
- $ID_e, model_e$: identifiers of the entity (e) in the scene and of the vehicle’s type;
- $frontal_coords_e, birdeye_coords_e$: the coordinates of the bounding box that encloses the entity;
- $distance_e, yaw_e$: distance and rotation of the entity w.r.t. the player.

Fig. 3 shows the distributions of entity rotation and distance across the collected data.

4 Model

At a first glance, the problem we address could be mistaken with a bare geometric warping between different views. Indeed, this is not the case since targets are not completely visible from the dashboard camera view and their dimensions in

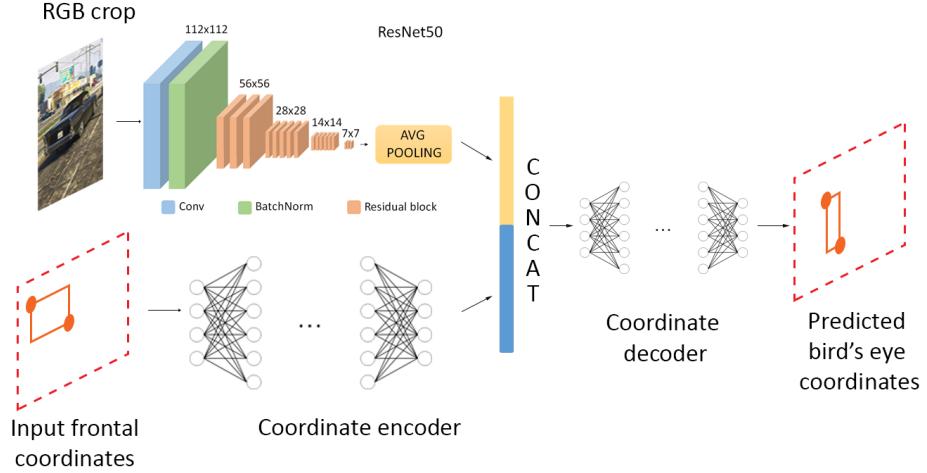


Fig. 4: A graphical representation of the proposed SDPN (see Sec. 4). All layers contain *ReLU* units, except for the top decoder layer which employs *tanh* activation. The number of fully connected units is (256, 256, 256) and (1024, 1024, 512, 256, 128, 4) for the coordinate encoder and decoder respectively.

the bird’s eye map depend on both the object visual appearance and semantic category (*e.g.* a truck is longer than a car). Additionally, it cannot be cast as a correspondence problem, since no bird’s eye view information are available at test time. Conversely, we tackle the problem from a deep learning perspective: dashboard camera information are employed to learn a spatial occupancy map of the scene seen from above.

Our proposed architecture composes of two main branches, as depicted in Fig. 4. The first branch takes as input image crops of vehicles detected in the dashboard camera view. We extract deep representations by means of *ResNet50* deep network [10], taking advantage of pre-training for image recognition on ImageNet [6]. To this end we discard the top fully-connected dense layer which is tailored for the original classification task. This part of the model is able to extract semantic features from input images, even though it is unaware of the location of the bounding box in the scene.

Conversely, the second branch consists of a deep *Multi Layer Perceptron* (MLP), composed by 4 fully-connected layers, which is fed with bounding boxes coordinates (4 for each detection), learning to encode the input into a 256 dimensional feature space. Due to its input domain, this segment of the model is not aware of objects’ semantic, and can only learn a spatial transformation between the two planes.

Both appearance features and encodings of bounding box coordinates are then

merged through concatenation and undergo a further fully-connected decoder which predicts vehicles’ locations in the bird’s eye view. Since our model combines information about object’s location with semantic hints on the content of the bounding box, we refer to it as *Semantic-aware Dense Projection Network* (SDPN in short).

Training Details: ImageNet [6] mean pixel value is subtracted from input crops, which are then resized to 224×224 before being fed to the network. During training, we freeze *ResNet50* parameters. Ground truth coordinates in the bird’s eye view are normalized in range $[-1, 1]$. Dropout is applied after each fully-connected layer with drop probability 0.25. The whole model is trained end-to-end using *Mean Squared Error* as objective function and exploiting *Adam* [11] optimizer with the following parameters: $lr = 0.001, \beta_1 = 0.9, \beta_2 = 0.999$.

5 Experimental results

We now assess our proposal comparing its performance against some baselines. Due to the peculiar nature of the task, the choice of competitor models is not trivial.

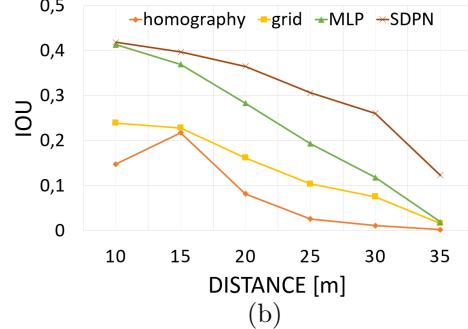
To validate the choice of a learning perspective against a geometrical one, we introduce a first baseline model that employs a projective transformation to estimate a mapping between corresponding points in the two views. Such correspondences are collected from bottom corners of both source and target boxes in the training set, then used to estimate an homography matrix in a least-squares fashion (*e.g.* minimizing reprojection error). Since correspondences mostly belong to the street, which is a planar region, the choice of the projective transformation seems reasonable. The height of the target box, however, cannot be recovered from the projection, thus it is cast as the average height among training examples. We refer to this model as *homography model*.

Additionally, we design second baseline by quantizing spatial locations in both views in a regular grid, and learn point mappings in a probabilistic fashion. For each cell G_i^f in the frontal view grid, a probability distribution is estimated over bird’s eye grid cells G_j^b , encoding the probability of a pixel belonging to G_i^f to fall in the cell G_j^b . During training, top-left and bottom-right bounding box corners in both views are used to update such densities. At prediction stage, given a test point p_k which lies in cell G_i^f we predict destination point by sampling from the corresponding cell distribution. We fix grid resolution to 108x192, meaning a 10x quantization along both axes, and refer to this baseline as *grid model*.

It could be questioned if the appearance of the bounding box content in the frontal view is needed at all in estimating the target coordinates, given sufficient training data and an enough powerful model. In order to determine the importance of the visual input in the process of estimating the bird’s eye occupancy map, we also train an additional model with approximately the same number of trainable parameters of our proposed model SDPN, but fully connected from input to output coordinates. We refer to this last baseline as *MLP*.

	IoU	CD	hE	wE	arE
homo	0.13	191.8	0.28	0.34	0.38
grid	0.18	154.3	0.74	0.70	1.30
MLP	0.32	96.5	0.25	0.25	0.29
SDPN	0.37	78.0	0.21	0.24	0.29

(a)



(b)

Fig. 5: (a) Table summarizing results of proposed SDPN model against the baselines; (b) Degradation of IoU performance as the distance to the detected vehicle increases.

For comparison, we rely on three metrics:

- *Intersection over Union* (IoU): measure of the quality of the predicted bounding box BB_p with respect to the target BB_t :

$$IoU(BB_p, BB_t) = \frac{A(BB_p \cap BB_t)}{A(BB_p \cup BB_t)}$$

where $A(R)$ refers to the area of the rectangle R ;

- *Centroid Distance* (CD): distance in pixels between box centers, as an indicator of localization quality¹;
- *Height, Width Error* (hE,wE): average error on bounding box height and width respectively, expressed in percentage w.r.t. the ground truth BB_t size;
- *Aspect ratio mean Error* (arE): absolute difference in aspect ratio between BB_p and BB_t :

$$arE = \left| \frac{BB_p.w}{BB_p.h} - \frac{BB_t.w}{BB_t.h} \right| \quad (4)$$

The evaluation of baselines and proposed model is reported in Fig. 5 (a). Results suggest that both *homography* and *grid* are too naive to capture the complexity of the task and fail in properly warping vehicles into the bird’s eye view. In particular, *grid* baseline performs poorly as it only models a point-wise transformation between bounding box corners, disregarding information about the overall input bounding box size. On the contrary, MLP processes the bounding box in its whole and provides a reasonable estimation. However, it still misses the chance to properly recover the length of the bounding box in the bird’s eye view, being unaware of entity’s visual appearance. Instead, SDPN is able to capture the object’s semantic, which is a primary cue for correctly inferring vehicle’s location and shape in the target view.

¹ Please recall that images are 1920x1080 pixel size.



Fig. 6: Qualitative comparison between different models. Baselines often predict reasonable locations for the bounding boxes. SDPN is also able to learn the orientation and type of the vehicle (*e.g.* a truck is bigger than a car etc.).

A second experiment investigates how vehicle’s distance affects the warping accuracy. Fig. 5 (b) highlights that all the models’ performance degrades as the distance of target vehicles increases. Indeed, closer examples exhibit lower variance (*e.g.* are mostly related to the car ahead and the ones approaching from the opposite direction) and thus are easier to model. However, it can be noticed that moving forward along distance axis the gap between the SDPN and MLP gets wider. This suggests that the additional visual input adds robustness in these challenging situations. We refer the reader to Fig. 6 for a qualitative comparison.

A real-world case study In order to judge the capability of our model to generalize on real-world data, we test it using authentic driving videos taken

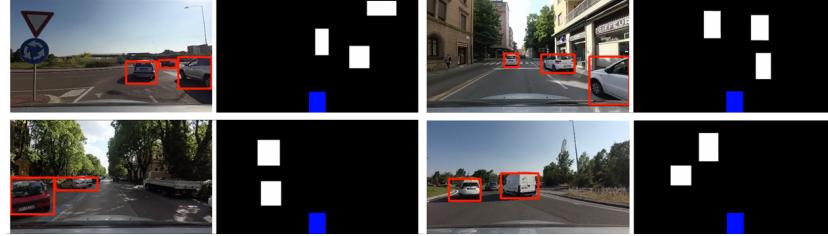


Fig. 7: Qualitative results on real-world examples. Predictions look reasonable even if the whole training was conducted on synthetic data.

from a roof-mounted camera [2]. We rely on state-of-the-art detector [15] to get the bounding boxes of vehicles in the frontal view. As the ground truth is not available for these sequences, performance is difficult to quantify precisely. Nonetheless, we show qualitative results in Fig. 7: it can be appreciated how the network is able to correctly localize other vehicles’ positions, despite having been trained exclusively on synthetic data.

SDPN can perform inference at approximately 100Hz on a NVIDIA TitanX GPU, which demonstrates the suitability of our model for being integrated in an actual assisted or autonomous driving pipeline.

6 Conclusions

In this paper we presented two main contributions. A new high-quality synthetic dataset, featuring a huge amount of dashboard camera and bird’s eye frames, in which the spatial occupancy of a variety of vehicles (i.e. bounding boxes, distance, yaw) is annotated. Furthermore, we presented a deep learning based model to tackle the problem of mapping detections onto a different view of the scene. We argue that these maps could be useful in an assisted driving context, in order to facilitate driver’s decisions by making available in one place a concise representation of the road state. Furthermore, in an autonomous driving scenario, inferred vehicle positions could be integrated with other sensory data such as radar or lidar by means of *e.g.* a Kalman filter to reduce overall uncertainty.

References

1. DeepGTAV. Software available at <https://github.com/ai-tor/DeepGTAV> (2017)
2. Alletto, S., Palazzi, A., Solera, F., Calderara, S., Cucchiara, R.: Dr (eye) ve: A dataset for attention-based tasks with applications to autonomous and assisted driving. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops. pp. 54–60 (2016)
3. Bernini, N., Bertozzi, M., Castangia, L., Patander, M., Sabbatelli, M.: Real-time obstacle detection using stereo vision for autonomous ground vehicles: A survey. In: Intelligent Transportation Systems (ITSC), 2014 IEEE 17th International Conference on. pp. 873–878. IEEE (2014)
4. Blade, A.: Script Hook V. Software available at <http://www.devc.com/gtav/scripthookv/> (2017)
5. Chen, C., Seff, A., Kornhauser, A., Xiao, J.: Deepdriving: Learning affordance for direct perception in autonomous driving. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 2722–2730 (2015)
6. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on. pp. 248–255. IEEE (2009)
7. Dong, Y., Hu, Z., Uchimura, K., Murayama, N.: Driver inattention monitoring system for intelligent vehicles: A review. IEEE transactions on intelligent transportation systems 12(2), 596–614 (2011)

8. Gaidon, A., Wang, Q., Cabon, Y., Vig, E.: Virtual worlds as proxy for multi-object tracking analysis. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 4340–4349 (2016)
9. Ghiasi, G., Fowlkes, C.C.: Laplacian pyramid reconstruction and refinement for semantic segmentation. In: European Conference on Computer Vision. pp. 519–534. Springer (2016)
10. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 770–778 (2016)
11. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. CoRR abs/1412.6980 (2014), <http://arxiv.org/abs/1412.6980>
12. Levi, D., Garnett, N., Fetaya, E., Herzlyia, I.: Stixelnet: A deep convolutional network for obstacle detection and road segmentation. In: BMVC. pp. 109–1 (2015)
13. Lin, C.C., Wang, M.S.: A vision based top-view transformation model for a vehicle parking assistant. Sensors 12(4), 4431–4446 (2012)
14. Lin, G., Shen, C., van den Hengel, A., Reid, I.: Efficient piecewise training of deep structured models for semantic segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 3194–3203 (2016)
15. Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.Y., Berg, A.C.: Ssd: Single shot multibox detector. In: European Conference on Computer Vision. pp. 21–37. Springer (2016)
16. Liu, Y.C., Lin, K.Y., Chen, Y.S.: Birds-eye view vision system for vehicle surrounding monitoring. In: International Workshop on Robot Vision. pp. 207–218. Springer (2008)
17. Nielsen, F.: Surround video: a multihead camera approach. The visual computer 21(1), 92–103 (2005)
18. Richter, S.R., Vineet, V., Roth, S., Koltun, V.: Playing for data: Ground truth from computer games. In: European Conference on Computer Vision. pp. 102–118. Springer (2016)
19. Ros, G., Sellart, L., Materzynska, J., Vazquez, D., Lopez, A.M.: The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 3234–3243 (2016)
20. Sung, K., Lee, J., An, J., Chang, E.: Development of image synthesis algorithm with multi-camera. In: Vehicular Technology Conference (VTC Spring), 2012 IEEE 75th. pp. 1–5. IEEE (2012)
21. Tseng, D.C., Chao, T.W., Chang, J.W.: Image-based parking guiding using ackermann steering geometry. In: Applied Mechanics and Materials. vol. 437, pp. 823–826. Trans Tech Publ (2013)
22. Venturelli, M., Borghi, G., Vezzani, R., Cucchiara, R.: Deep head pose estimation from depth data for in-car automotive applications. In: Proceedings of the 2nd International Workshop on Understanding Human Activities through 3D Sensors (2016)
23. Wymann, B., Espié, E., Guionneau, C., Dimitrakakis, C., Coulom, R., Sumner, A.: Torcs, the open racing car simulator. Software available at <http://torcs.sourceforge.net> (2000)
24. Zhang, B., Appia, V., Pekkucuksen, I., Liu, Y., Umit Batur, A., Shastry, P., Liu, S., Sivasankaran, S., Chitnis, K.: A surround view camera solution for embedded systems. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops. pp. 662–667 (2014)