Detecting Road Obstacles by Erasing Them

Krzysztof Lis, Member, IEEE, Sina Honari, Pascal Fua, Fellow, IEEE, and Mathieu Salzmann

Abstract—Vehicles can encounter a myriad of obstacles on the road, and it is impossible to record them all beforehand to train a detector. Instead, we select image patches and inpaint them with the surrounding road texture, which tends to remove obstacles from those patches. We then use a network trained to recognize discrepancies between the original patch and the inpainted one, which signals an erased obstacle.

Index Terms—Computer Vision, convolutional neural networks, intelligent vehicles, semantic segmentation, vehicle safety.

I. INTRODUCTION

MODERN methods provide a nearly complete toolkit for vision-based autonomous driving. That includes road segmentation [1], [2], lane-finding [3], [4], vehicle and pedestrian detection [5], [6], [7], and multi-class semantic [8], [9], [10], instance [11] and panoptic [12], [13] segmentation. Moreover, vision-based assistance is now widely accepted in the market [14] and enables vehicles to plan a path within the predicted drivable space while avoiding other traffic participants. However, the risk of missing strange and unexpected obstacles lying on the road remains. Such obstacles are as rare as they are diverse, which prevents direct application of the now standard approach of training deep networks by showing them an exhaustive set of annotated samples.

In practice, detecting such unexpected obstacles often requires LiDAR sensors [15] or multiple cameras [16]. Here, we propose instead a method that only needs a single RGB image to detect obstacles in the drivable area, under the assumption that objects outside that area are irrelevant because a self-driving car will detect the road before planning to drive and will not leave the drivable area of its own accord. To demonstrate this to be a viable assumption, we will show results given either the ground-truth location of the road edges or only an imperfect road segmentation produced by an off-the-shelf segmentation algorithm.

Our approach relies on the fact that obstacles look different from the surrounding road surface. We thus detect them by inpainting image-patches using their surroundings and then checking how similar the inpainted patch is to the original one. While a similar intuition has been used to detect anomalies in several application scenarios, such as detecting manufacturing defects [17], [18] or anomalous faces [19], the very constrained nature of these tasks made it possible to rely on simple comparisons of handcrafted features. By contrast, on roads, this would yield many false positives due to road markings, diversity in road texture, and obstacles extending beyond the inpainted patch.

Our solution is to introduce a discrepancy network trained to recognize which differences between the inpainted patch and the original one are significant. It returns a per-pixel heatmap denoting the presence of obstacles, as shown in Fig. 1. To train it to handle objects that are not part of the training database, we generate samples featuring synthetic obstacles by moving existing training objects, such as road signs and people, onto the road.

Our experiments show that our discrepancy network trained solely on Cityscapes [20] objects successfully detects obstacles on images depicting significantly different road scenes, without requiring any annotated data nor any re-training for these new scenes. In other words, our method generalizes well to previously unseen real obstacles and new road surfaces. It outperforms earlier monocular road anomaly detectors [21], [22], [23], [24] on the Lost & Found [16] data featured in the Fishyscapes benchmark [22], as well as on our own newly collected dataset featuring additional unusual objects and road surfaces.
Our contribution is therefore a simple but effective approach to detecting obstacles that never appeared in any training database, given only a single RGB image. We also contribute a new dataset for evaluating anomaly detection models. We will make our obstacle detection code, our semi-synthetic training dataset, and our new benchmark dataset publicly available.

II. RELATED WORK

In this section, we briefly review methods that, as ours, rely on a single RGB image for obstacle detection and do not use explicit obstacle training sets. For a more complete survey of obstacle detection algorithms, we refer the reader to [16], [25], [26], [27].

A. Image Reconstruction for Anomaly Detection

If an image is reconstructed so as to preserve the appearance in normal regions and discard anomalous ones, those anomalies can be detected by comparing the input to the reconstruction. This has been achieved in several ways we now discuss.

In [28], the distribution of inlier images is modeled by training an autoencoder, followed by a feature comparison between the input and reconstruction. Auto-encoding has been used to find highway obstacles in [29], [30], where input patches are passed through a Restricted Boltzmann Machine. Its limited expressive power is supposed to preserve the smooth road surface while altering obstacles. This approach, however, does not address textured road surfaces. As opposed to encoding the input image, the method of [31] trains a generator to capture the training distribution and searches for a latent vector producing an image most similar to the input. However, this method operates on microscopic scans of tissue samples, and, to our knowledge, it has not been applied to data with distributions as diverse as outdoor road scenes.

Other methods [21], [32] explicitly restrict the intermediate representation of the scene to a dense semantic map, and synthesize a plausible matching image using conditional GANs for image translation [33], [34]. Since the anomalous regions are not represented by the typical semantic classes, their appearance will be altered by this process. In [21], the input and synthesized images are compared using a learned discrepancy module, while [32] uses a feature distance measure. JSRNet [35] takes a similar approach, but uses the activations of the last layer as the bottleneck instead of a semantic map. A reconstruction network then recreates the image that is compared to the original using a SSIM measure.

Rather than encoding the input, one can remove parts of the image and inpaint them based on the surrounding context. In [18] square patches are inpainted and compared with an L1 metric to detect material defects; the method of [17] combines the reconstructions obtained with a set of random inpainting masks and uses a multi-scale gradient magnitude similarity metric for comparison. In the context of road scenes, [36] proposes to compare the road appearance to similar images memorized from previous video frames; however this would lead to false positives when entering an area with a new road texture. Furthermore, these methods assume high fidelity of the reconstruction, and detect every visible difference as an anomaly. In outdoor scenes with road markings and diverse surface textures, the inpainting is bound to be imperfect. We address this by training a discrepancy network to focus on the relevant differences.

Methods specifically designed for removing dynamic objects from traffic scenes have been presented in [37], [38], but they rely on object masks being known a-priori without having to detect them first.

B. Anomaly Detection in Semantic Segmentation

The problem of detecting anomalies can be posed as one of open-set semantic segmentation. With standard, fully-supervised semantic segmentation networks, all pixels, including the anomalous ones, will be classified into one of the training semantic categories. Open-set semantic segmentation then aims to find the outliers in the resulting semantic maps. The method of [24] introduces an outlier detection head sharing backbone features with the semantic segmentation one. It is trained using extensive out-of-distribution data, injecting outlier patches drawn from ImageNet-Ik [39] into the Cityscapes and Vistas [40] scenes.

The work of [22] proposes to learn the inlier distribution of features extracted from a layer of the semantic segmentation network. It uses a normalizing flow to bijectively map the features to latent vectors following a Gaussian distribution. The mapping is trained to maximize the likelihood of the features observed in inlier samples. Another approach involves estimating the uncertainty of label prediction based on the assumption that outlier regions should yield low classification confidence. Bayesian deep learning treats the network weights and outputs as probability distributions. In practice, dropout [41] is used to approximate the distribution over model weights, applied to semantic segmentation in [42], [43]. Alternatively an ensemble of networks can be used [44], [45]. In [23], the Dirichlet differential entropy is used as a measure of uncertainty. We find that the performance of these statistical methods as obstacle detectors degrades significantly when faced with road surfaces differing from the training set, as the novel textures are treated as anomalies. We will show that our approach based on comparing the original image with an inpainted one with a discrepancy detector does not suffer from this drawback.

In [46], generalization is achieved by training the segmentation network to maximize the output entropy on explicit out-of-distribution samples. This is done by injecting COCO [47] objects into Cityscapes frames. According to our experiments this is effective for some types of obstacles, presumably ones more similar to COCO objects, but not all of them. The additional samples can also be created by a generative model jointly optimized with the detector to output points on the edge of the training distribution [48]. A normalizing flow is used to generate synthetic training anomalies in a similar manner in [49]. We also use synthetic data to train our discrepancy network but our object-pasting scheme is much simpler. Hence, these approaches are complementary to ours, which focuses on the inference mechanism. Hence it, would make sense to combine them in future work.
Finally, the SynBoost [50] method fuses resynthesis dissimilarity with semantic segmentation uncertainty estimates. This improves upon previous resynthesis-based approaches. However, we will show that our method outperforms SynBoost while requiring neither detailed semantic segmentation nor associated uncertainty; we only need to segment the road area to guide our inpainter.

Another way to detect semantic anomalies is to analyze the distribution of the predicted semantic logits. A low value of the maximum logit implies an anomaly as none of the classes is confidently predicted [51]. However the logit values differ strongly between each class, making a single threshold work poorly. The method of [52] standardizes the logit values to align distributions between classes. It also removes spurious detections on object edges by post-processing. In our approach we do not reason about the semantic predictions - we only use the semantic network as a convenient way to segment the road but a simpler road or lane detector could be used instead.

III. APPROACH

Our goal is to identify obstacles that are objects on the road the vehicle could collide with. This means that they are within the space deemed drivable by a previous stage in the self-driving perception pipeline. As such, they are the most relevant obstacles as a competent driving system will only plan trajectories within that space.

In other words, our chosen task is to identify all pixels, within that estimated road area, that denote obstacles. This is difficult because obstacles can take many forms. Furthermore, because they are unexpected, there is no guarantee that they were present in the database used to train a network to recognize them. Hence, the network must be made to respond to objects that does not belong to the road without any clear description, or even exemplars, of those objects.

To this end, given a binary mask denoting the drivable area in the image, we propose the following two-step approach:
1) Erase the obstacles by removing road patches and inpainting them in a sliding-window manner;
2) Use a discrepancy network to compare the original image to the inpainted one and decide if they are similar enough.

The intuition behind this scheme is that, if there is an obstacle, the inpainted area will look very different from the original image. However, even if there is no obstacle, the inpainted area will be similar to the original one, but not strictly equal. Hence, the discrepancy network is needed to assess if they are dissimilar enough to flag a potential obstacle. It yields a heatmap denoting the likelihood for each pixel in the drivable area of belonging to an obstacle. In the remainder of this section, we discuss these two steps in more detail.

A. Drivable Area

A self-driving system must determine the road area within which it can move. Our method detects obstacles within the area identified as drivable, as these are the only ones that might be on the vehicle’s planned path and can endanger it. Of course, a part of the road may be mistakenly marked as non-drivable, causing obstacles it contains to be ignored as shown Fig. 8. However, this does not compromise safety because the vehicle will never attempt to go there.

Our approach can exploit any method that delivers the required drivable area information. In our experiments, we use the PSP-Net semantic segmentation network of [8] trained on the Cityscapes dataset [20], as implemented in the framework of [53]. We take the road area to be all pixels classified as either road or sidewalk, since the many road textures we are targeting can be classified as either. Note that standard categories, such as car and pedestrian, are inherently accounted for by PSP-Net. Hence, we focus on the unusual obstacles for which no training data, either supervised not unsupervised, is available. Nevertheless, since such unusual obstacles could be partially classified as non-road, we include the regions containing other classes that are fully enclosed within the road area. Fig. 2 demonstrates this process. As a limit case, we also evaluate the case of perfect road detection by using the ground truth road mask.

Our approach needs only a coarse mask of the drivable area, so in a practical deployment the semantic segmentation can be replaced by computationally more efficient system, for instance predicting just the road edges and filling the space between them.

B. Inpainting

To erase the obstacles while preserving the surrounding road appearance, we use a general-purpose inpainter [54] that relies on an adversarial approach to ensure that the inpainted image looks realistic. We use a version of this model trained on the scene recognition dataset Places2 [55], and do not train it further.

The inpainter is given an image in which a part has been replaced by black pixels and the pixel mask of the removed

![Fig. 2. Drivable space from semantic segmentation. Top: Input images. Middle: Semantic segmentation performed by PSP-Net [8], the class colors follow Cityscapes convention. Bottom: We take the drivable space to be the union of road (purple) and sidewalk (magenta) pixels. The parts of obstacles can sometimes be classified as non-road, so we include the regions of other classes fully enclosed within the road area. In Lost & Found, the known ego-vehicle mask is excluded.](image-url)
area. It outputs an image with the missing part filled in so as to best maintain the continuity and content of the scene.

A naive way to use it would be to inpaint the entire road area at once. This, however, would provide no indication to the inpainter of the road appearance, leading to inpainted images that differ from the original ones in the whole road region, thus precluding subsequent obstacle detection. Instead, we inpaint road patches to provide sufficient context for the network to reconstruct the road surface, as shown in Fig. 3. The patches nonetheless need to be large enough to encompass obstacles whose size we do not know a priori. We therefore follow a sliding-window approach, inpainting patches of 200 × 200 pixels of drivable area within 400 × 400 pixel regions to provide context.

While an obstacle is usually nicely erased when the area to inpaint encloses it completely, the inpainter is able to re-create the obstacles that are only partially contained in the inpainted region. To resolve this, we use consecutive patches with a relative overlap of 0.7, increasing the likelihood of having at least one patch that covers the entire obstacle. This means that each image pixel is inpainted multiple times. We then fuse the multiple inpaintings of each pixel by weighted averaging, where the weight of each inpainting is computed based on the Manhattan distance between the corresponding patch center and the pixel location of interest. Formally, a patch centered at location \( c_j = [u_j, v_j]^\top \) contributes to the inpainting of a pixel at location \( p = [u, v]^\top \) with a weight

\[
w = \frac{1 - \frac{2}{s} \max(|u - u_j|, |v - v_j|)}{\sum_{[u_i, v_i] \in \Pi(u, v)} 1 - \frac{2}{s} \max(|u - u_i|, |v - v_i|)},
\]

(1)

where \( s \) is the patch width or height and \( \Pi(u, v) \) is the set of patches overlapping the \( [u, v]^\top \) pixel, with patch \( i \) centered at \( [u_i, v_i] \).

Our inpainting strategy enables us to generalize to new road surfaces, such as those of our Road Obstacles dataset. By contrast, the GAN synthesizers of [21], [32], [50] always produce images resembling the Cityscapes training set.

C. Discrepancy Network

While our inpainting strategy preserves the general appearance of the road surface, it still yields unavoidable imperfections due to road markings, texture details, and the non-zero contributions of obstacles located close to the patch edges. Thus, simply comparing the original image with the inpainted one via pixel difference would yield many false positive detections. To handle this, we introduce a discrepancy network that we train to distinguish significant differences from inpainting artifacts.

We implement our discrepancy network using a two-stream architecture, shown in Fig. 4, that takes as input the original image and the inpainted one. Both inputs are first processed by a ResNeXt101 [56] feature extractor, pretrained for ImageNet [39] and frozen at training time. The resulting features are then concatenated and fused through 1 × 1 convolutions. Furthermore, we compute a point-wise correlation map: At each location in the feature map, we calculate the dot product between the image feature vector and the corresponding inpainting feature vector. We append these dot products as an additional channel to the output of the 1 × 1 convolutions. The concatenated features are then passed to an upconvolution pyramid, and we obtain the desired heatmap via a softmax.

1) Training the Discrepancy Network: Recall that we target unusual road obstacles that may have never been seen at training time. Therefore, we need the discrepancy network to generalize to previously-unseen objects.

To tackle this challenge, we built a synthetic training set from only the Cityscapes [20] dataset, which contains no unusual traffic obstacles. We extracted instances of people and vehicles using the instance annotations, together with traffic signs and traffic lights, which lack instance labels, but can be extracted as connected components within their pixel-wise semantic label mask. Since many road obstacles are small and difficult to detect, to simulate small obstacles seen from a far distance, we selected from the whole dataset instances of size ranging from 10 to 150 pixels, and area between 100 and 5000 pixels. We then sampled random objects from this database and overlaid them.
onto the drivable area to mimic obstacles. Fig. 5 features several images synthesized in this way.

2) High Frequency Textures: The Cityscapes training set features mostly smooth road surfaces. Hence, test images featuring rougher road surfaces could produce false false positives. To prevent this, we eliminate high frequencies by means of a simple $5 \times 5$ Gaussian blur. In the ablation study of Section IV-E we show this to effective against domain shift.

IV. EXPERIMENTS

In this section, we first present the baselines, evaluation metrics, and datasets used in our experiments. We then compare our approach to the baselines and finally perform an ablation study.

A. Baselines

We compare against the following approaches.

Resynthesis [21] performs semantic segmentation of the image and then synthesizes an image solely from the resulting semantic map using a conditional GAN [34]. The differences between the original and reconstructed images are detected using a discrepancy network similar in purpose to the one we use, but trained by synthetic anomalies; those are produced by selecting object instances and swapping their semantic class to a random different one prior to resynthesis.

JSRNet [35] likewise performs image reconstruction but the bottleneck is the semantic segmenter’s last layer activation. The reconstructed image is then compared to the original one using an SSIM measure.

SynBoost [50] expands the resynthesis approach by providing uncertainty estimates of the semantic segmentation as an additional input to its dissimilarity network, which predicts the anomaly score.

Entropy Maximization [46] trains the segmentation network to maximize the output entropy on explicit OOD samples obtained by adding COCO [47] objects into Cityscapes frames. It also performs post-processing on connected components of obstacle pixels, but this part is not applicable in our per-pixel evaluation.

Learned Embedding Density [22] learns the inlier distribution of features extracted from a DeepLab [57] layer. It then maps the features to latent, Gaussian-distributed vectors via a normalizing flow. The mapping is trained to maximize the likelihood of the features observed in inlier samples.

Void Classifier [22] uses the Cityscapes void areas as examples of outliers. It can then either explicitly add void to the set of predicted classes, or learn to maximize the softmax entropy in the void regions.

Bayesian DeepLab [58] introduces dropout layers into the DeepLab network. At inference time, it draws samples by randomizing the dropout. The uncertainty is measured as the mutual information between the resulting distribution and the network weights.

Maximum Softmax [51] measures the predicted softmax probability of the most probable class (max softmax) - with a low value indicating uncertain prediction and a likely semantic anomaly.

B. Evaluation Metrics

We follow the evaluation protocol of the Segment Me If You Can [59] benchmark and specifically the obstacle track. Since we focus on detecting obstacles on the road, we take the Region of Interest (ROI) for evaluation purposes to be the ground-truth road area as shown in Fig. 6. This restriction of ROI to drivable space follows the evaluation protocol of the original Lost and Found [16] road obstacle benchmark and matches the area relevant to self-driving, that is, the road area where the vehicle is going to move and can encounter obstacles. Our method, like the baselines we compare with, outputs a heatmap in the [0,1] range denoting the likelihood for each pixel within the ROI of belonging to an obstacle. The benchmark measures performance in terms of obstacle pixel classification and obstacle instance detection.

- The primary pixel metrics is Average Precision (AP), that is, the area under the precision-recall curve. This metric is more meaningful than metrics based on the receiver operating curve (ROC) due to strong class imbalance, as obstacles typically cover less than 2% of the total road surface.
- A secondary pixel metric is false positive rate (FPR) at a 95% true positive rate (TPR), which we denote as $\text{FPR}_{95}$. To compute it, the binary classification threshold is lowered until 95% of the obstacle pixels are detected and we then measure how many false positives are introduced.
- The main instance metric is the average $F_1$, which summarizes the numbers of true positives, false positives, and false negatives, averaged over several IoU thresholds.

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
TABLE I  
OBSTACLE DETECTION SCORES

|                      | Lost and Found - test no known | RoadObstacles21 - test |
|----------------------|--------------------------------|------------------------|
|                      | Pixel-level                    | Segment-level          | Pixel-level | Segment-level |
|                      | AP ↑ | FPR₃₅ ↓ | sIoU ↑ | PPV ↑ | F₁ ↑ | AP ↑ | FPR₃₅ ↓ | sIoU ↑ | PPV ↑ | F₁ ↑ |
| Ours - road ground truth | 87.1 | 2.3    | 55.9  | 58.7  | 61.1 | 67.8 | 30.1  | 58.9  | 52.5  | 51.1 |
| Ours - road segmentation  | 82.3 | 68.5   | 49.3  | 55.6  | 54.0 | 67.5 | 32.8  | 59.3  | 52.4  | 50.8 |
| SynBoost [50]          | 81.7 | 4.6    | 36.8  | 72.3  | 48.7 | 71.3 | 3.2   | 44.3  | 41.8  | 37.6 |
| Entropy Maximization [46] | 77.9 | 9.7    | 45.9  | 63.1  | 49.9 | 85.1 | 0.8   | 47.9  | 62.6  | 48.5 |
| JSRNet [35]           | 74.2 | 6.6    | 34.3  | 45.9  | 36.0 | 28.1 | 28.9  | 18.6  | 24.5  | 11.0 |
| Embedding Density [22] | 61.7 | 10.4   | 37.8  | 35.2  | 27.5 | 0.8  | 46.4  | 35.6  | 2.9   | 2.3  |
| Image Resynthesis [21] | 57.1 | 8.8    | 27.2  | 30.7  | 19.2 | 37.7 | 4.7   | 16.6  | 20.5  | 8.4  |
| Bayesian DeepLab [58]  | 36.6 | 35.5   | 17.4  | 34.7  | 13.0 | 4.9  | 50.3  | 5.5   | 5.8   | 1.0  |
| Max Softmax [51]       | 30.1 | 33.2   | 14.2  | 62.2  | 10.3 | 15.7 | 16.6  | 19.7  | 15.9  | 6.3  |
| Void Classifier [22]  | 4.8  | 47.0   | 1.8   | 35.1  | 1.9  | 10.4 | 41.5  | 6.3   | 20.3  | 5.4  |

The primary metric is average precision of detecting obstacle pixels and average $F_1$ of instance detection.

- The sIoU is the average of the pixel intersection-over-union calculated for each obstacle instance, adjusted to account for predictions covering multiple instances, as described in [59].
- PPV is the positive predictive value (precision) calculated separately for each instance and then averaged.

C. Datasets

We report empirical results on two datasets, the well-established Fishyscapes Lost & Found [16] benchmark and a new RoadObstacles21 dataset that we acquired in collaboration with colleagues at the University of Wuppertal as part of the Segment Me If You Can benchmark [59].

Lost and Found Dataset [16]: It contains 1030 frames from image sequences captured by a vehicle approaching lost cargo items placed on parking lots and streets. We use the Test No Known subset of the test split where the training Cityscapes objects such as pedestrians and vehicles have been removed. By ensuring that all obstacles are semantic anomalies, this allows the evaluation of anomaly-detection methods in the role of obstacle detectors.

RoadObstacles21 Dataset [59]: Lost and Found features urban environments similar to those in the Cityscapes training data. To evaluate our obstacle detector on a wider variety of road surfaces and objects, such as those depicted by Fig. 7, we collected our own Road Obstacles dataset.

It features 8 different road textures and comprises a total of 327 labeled $1920 \times 1080$ frames. The labels include pixel masks for individual obstacle instances along with approximate outlines of the drivable area. We take the evaluation ROI to be the area within these outlines. If an object has a hole, such as a basket with a handle, we label the hole as outside of the ROI and ignore it in our evaluations, as we do for the background.

Fig. 7. Road Obstacles 19 dataset. Top: Example frames representing each of the 7 scenes. Bottom: Some of the objects featured in the dataset.

Fig. 8. Obstacle outside of predicted drivable area failure mode. Top-left: An obstacle is located near a vehicle shadow. Top-right: The semantic segmenter classifies it as part of the background. Bottom-left: Consequently it is not included in the drivable space where our obstacle detector is applied. Bottom-right: Ground truth road label used as the region-of-interest for evaluation purposes. This obstacle will be counted as missed.
TABLE II
Ablation Study Results

|                  | Lost and Found - test no known | RoadObstacles21 - validation |
|------------------|-------------------------------|------------------------------|
|                  | Pixel-level                   | Segment-level                |
|                  | AP ↑  | FPR<sub>95</sub> ↓ | IoU ↑  | PPV ↑  | AP ↑  | FPR<sub>95</sub> ↓ | IoU ↑  | PPV ↑  | F<sub>1</sub> ↑ |
| Ours             | 82.3  | 68.5    | 49.3  | 55.6  | 54.0  | 91.0    | 100.0  | 59.4  | 57.0  | 60.0  |
| Resynthesis      | 65.0  | 100.0   | 44.2  | 35.9  | 35.6  | 83.7    | 100.0  | 55.4  | 39.6  | 45.2  |
| Blur             | 78.5  | 50.7    | 49.7  | 51.0  | 50.3  | 89.3    | 100.0  | 58.4  | 51.6  | 55.3  |
| No Inpainting    | 81.9  | 36.2    | 49.2  | 56.5  | 54.6  | 89.9    | 100.0  | 62.2  | 46.9  | 54.7  |
| No Discrepancy   | 19.6  | 83.7    | 26.7  | 16.7  | 7.8   | 13.1    | 92.9   | 13.5  | 45.5  | 11.1  |
| Segmentation Alone| 36.5  | 100.0   | 14.4  | 91.3  | 23.1  | 28.3    | 100.0  | 22.5  | 48.8  | 26.1  |
| No input blur    | 82.4  | 100.0   | 51.5  | 56.9  | 56.2  | 87.7    | 100.0  | 63.3  | 54.3  | 59.7  |

Top result achieved without ground-truth road segmentation shown in bold.

Fig. 9. Qualitative comparison of method outputs. The darkened area corresponds to the ground-truth drivable space.

D. Comparative Results

We tested two versions of our method, one that operates on the road area given in the ground truth and the other on the drivable area segmented by the network of [8]. They are referred to as “Ours road ground truth” and “Ours road segmentation”, respectively, in Table I. A qualitative comparison and further examples of our method’s outputs are shown in Figs. 9 and 10, respectively.

Our method outperforms all others in terms average instance F<sub>1</sub>, the primary instance metric in [59], on both datasets. As for pixel Average Precision (AP), it outperforms baselines on the Lost and Found dataset. On RoadObstacles21 only two methods, SynBoost and Entropy Maximization, achieve a better AP. The ‘Ours road segmentation” variant achieves an abysmal score of the FPR<sub>95</sub> secondary pixel metric on Lost and Found but the score using the ground truth is excellent. This points to the cause of the problem: The segmentation algorithm we use only found a part of the drivable area and, since we only look for obstacles there, we missed all those that were elsewhere, making it impossible to reach a 95% TPR in some of the images. Fig. 8 illustrates this problem. Nevertheless, even in this situation, safety is maintained because the vehicle controller will only drive within the predicted road space. Hence, obstacles outside of it do not pose a risk of collision. Furthermore, as evidenced by the good results of “Ours road ground truth”, this problem will gradually fade away as road boundary detection algorithms improve.

While most methods including ours use solely the Cityscapes dataset for training, some others [46] sample the objects injected as synthetic obstacles from other datasets, such as COCO or ImageNet-1k. This strategy works well for some obstacle types, particularly the distinct and colorful ones in the RoadObstacles21 set, but falls short in the Fishyscapes set. In other words, extending the original dataset with others is effective if the test obstacles are similar to the ones in the additional dataset but does not necessarily promote generalization.

E. Ablation Study

In Table II, we report the results of an ablation study during which the discrepancy network was retrained with selected components altered or disabled.

In the Resynthesis variant, the inpainting described in Section III-B is replaced by an image synthesizer [34] from predicted semantic labels as in [21]. While the inpainter can

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
reconstruct novel road textures based on the visual context, the generator produces a texture similar to the training roads, and this is reflected by degraded performance on Road Obstacles.

The Blur variant replaces inpainting by a Gaussian blur operation. We took the kernel size to be 200 to match the size of the inpainting window. The No Inpainting variant does away with inpainting. We keep the architecture unchanged, but pass two copies of the image into both input streams and retrain the network. This also degrades performance, thus confirming the importance of inpainting. We can also do the reverse, that is, remove the discrepancy network, and compute the $L_1$ distance between the RGB values of the input image and the inpainted result. The results of the corresponding No Discrepancy variant are even worse.
The *Segmentation Alone* entry corresponds to detections made by finding groups of non-road pixels enclosed within the road region detected by the segmentation algorithm [8]. This by itself clearly fails, thereby justifying the extra step we propose in this paper.

### F. Limitations

**Erasing Safe Regions:** The out-of-distribution regions detected by our approach can include both genuine obstacles and distinctive but safe road regions such as manholes, road cracks, and leaves. The discrepancy network of Section III-C alleviates this problem by learning what altered regions can be safely ignored, but that relies on examples being present in the training set. For further improvements, an approach would be to look into methods that expand the training set, such as [48], [49].

**Perspective Effects:** Our approach does not explicitly account for further obstacles appearing smaller, due to perspective effects. Correcting for this, as we did in [60], should also help reduce false-positives, especially in regions close to the camera.

**Computational Cost:** The sliding window inpainting algorithm is computationally expensive, requiring multiple executions of the inpainter. The current naive implementation takes 6.7 s (inpainting) plus 0.2 s (discrepancy network) per frame on a single A100 GPU, as measured on the *RoadObstacles21* validation dataset. As this work is intended as a proof-of-concept, we used off-the-shelf inpainting code without production optimization. One way to remedy this would be to switch to 16-bit precision, which is much more efficient on modern GPUs. Another would be to parallelize the inpainting. Furthermore, even the slow version of our approach can be used effectively: Since our algorithm detects previously unseen obstacles, it can be used to extract obstacles from unlabeled videos for additional supervised training, even without real-time inference.

### V. CONCLUSION

We have introduced a pipeline capable of detecting road obstacles in driving scenarios, given only single monocular images as input. We perform inpainting of the drivable area, erasing the localized obstacles while preserving the road surface. Our discrepancy network learns to accurately detect the removed obstacles and ignore irrelevant artifacts of reconstruction. This detector, trained only with synthetically altered Cityscapes data, is capable of generalizing to a variety of real-world obstacles and road surfaces. We have demonstrated this on the *Fishyscapes Lost & Found* benchmark, as well as on our newly collected dataset. We will release our dataset to the community to help researchers in this field.

In the future, we will work on coupling more tightly the algorithms that detect the drivable area and ours for improved results and potential integration into a full self-driving system. We also plan to exploit adaptive inpainting patch sizes to account for perspective distortion.

### ACKNOWLEDGMENT

The authors would like to thank Hermann Blum for adapting the Fishyscapes benchmark and baseline methods to our data as well as help with the evaluations. We thank Mirosław Lis for assistance in the collection of the *Road Obstacles* dataset.

### REFERENCES

[1] Z. Chen and Z. Chen, “RNET: A deep neural network for unified road and road boundary detection,” in *Proc. Int. Conf. Neural Inf. Process.*, 2017, pp. 677–687.

[2] J. Munoz-Bulnes, C. Fernandez, I. Parra, D. Fernández-Llorca, and M. A. Sotelo, “Deep fully convolutional networks with random data augmentation for enhanced generalization in road detection,” in *Proc. IEEE Int. Conf. Intell. Transp. Syst.*, 2017, pp. 366–371.

[3] L. Liu, Z. Yuan, T. Liu, and Z. Xiong, “End-to-end lane shape prediction with transformers,” in *Proc. IEEE Winter Conf. Appl. Comput. Vis.*, 2021, pp. 3694–3702.

[4] Z. Qin, H. Wang, and X. Li, “Ultra fast structure-aware deep lane detection,” in *Proc. Eur. Conf. Comput. Vis.*, 2020, pp. 276–291.

[5] S. Ren, K. He, R. Girshick, and J. Sun, “Fast R-CNN: Towards real-time object detection with region proposal networks,” in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2015, pp. 91–99.

[6] P. Dollár, C. Wojek, B. Schiele, and P. Perona, “Pedestrian detection: An evaluation of the state of the art,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 4, pp. 743–761, Apr. 2012.

[7] F. Chabot, M. Chasouch, J. Rabarisoa, C. Teuliere, and T. Chateau, “Deep MANTA: A coarse-to-fine many-task network for joint 2D and 3D vehicle analysis from monocular image,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 1827–1836.

[8] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, “Pyramid scene parsing network,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 6230–6239.

[9] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam, “Rethinking atrous convolution for semantic image segmentation,” 2017, arXiv: 1706.05587.

[10] E. Romero, J. M. Alvarez, L. M. Bergasa, and R. Arroyo, “ERFNet: Efficient residual factorized ConvNet for real-time semantic segmentation,” *IEEE Trans. Intell. Transp. Syst.*, vol. 19, no. 1, pp. 263–272, Jan. 2018.

[11] K. He, G. Gkioxari, P. Dollár, and R. B. Girshick, “Mask R-CNN,” in *Proc. Int. Conf. Comput. Vis.*, 2017, pp. 2980–2988.

[12] A. Kirillov, K. He, R. Girshick, C. Rother, and P. Dollár, “Panoptic segmentation,” in *Proc. Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 9396–9405.

[13] Y. Xiong et al., “UPSNNet: A unified panoptic segmentation network,” in *Proc. Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 8810–8818.

[14] AutoNation drive editors’ guide to lane assist systems, 2021. [Online]. Available: https://www.autonationdrive.com/research/best-cars-with-lane-assist.htm

[15] A. Singh, A. Kamireddypalli, V. Gandhi, and K. M. Krishna, “LiDAR guided small obstacle segmentation,” in *Proc. Int. Conf. Intell. Robots Syst.*, 2020, pp. 8513–8520.

[16] P. Pinggera, S. Ramos, S. Gehrig, U. Franke, C. Rother, and R. Mester, “Lost and found: Detecting small road hazards for self-driving vehicles,” in *Proc. Int. Conf. Intell. Robots Syst.*, 2016, pp. 1099–1106.

[17] V. Zavrtanik, M. Kristan, and D. Skočaj, “Reconstruction by inpainting for visual anomaly detection,” *Pattern Recognit.*, vol. 112, 2021, Art. no. 107706.

[18] M. Haselmann, D. P. Gruber, and P. Tabatabai, “Anomaly detection using deep learning based image completion,” in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 1237–1242.

[19] A. Bhattachar, J. Rock, and D. Forsyth, “Detecting anomalous faces with ‘No Peeking’ autoencoders,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018.

[20] M. Cords et al., “The Cityscapes dataset for semantic urban scene understanding,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 3213–3223.

[21] K. Lis, K. Nakka, M. Salzmann, and P. Fua, “Detecting the unexpected via image resynthesis,” in *Proc. Int. Conf. Comput. Vis.*, 2019, pp. 2152–2161.

[22] H. Blum, P.-E. Sarlin, J. Nieto, R. Siegwart, and C. Cadena, “Fishyscapes: A benchmark for safe semantic segmentation in autonomous driving,” in *Proc. Int. Conf. Comput. Vis.*, 2019, pp. 2403–2412.
Pascal Fua (Fellow, IEEE) is a professor of computer science with EPFL, Switzerland. His research interests include shape and motion reconstruction from images, analysis of microscopy images, and augmented reality. He has been an associate editor of the IEEE Journal Transactions for Pattern Analysis and Machine Intelligence.

Mathieu Salzmann received the PhD degree from EPFL, in 2009. He is a senior researcher with EPFL and an artificial intelligence engineer with ClearSpace. Previously, he was a senior researcher and research leader with NICTA’s Computer Vision Research Group, a research assistant professor with TTI-Chicago, and a postdoctoral fellow with ICSI and EECS, UC Berkeley. His research interests lie at the intersection of machine learning and computer vision.