Friedrich Fraundorfer
Peter M. Roth
Fabian Schenk (eds.)

Graz University of Technology

Proceedings of the

24th Computer Vision Winter Workshop

Stift Vorau, Austria
February 6–8, 2019

Friedrich Fraundorfer, Peter M. Roth, Fabian Schenk (eds.)

Proceedings of the 24th Computer Vision Winter Workshop

February 6–8, 2019

Stift Vorau, Austria

Graz University of Technology
Institute of Computer Graphics and Vision

OESTERREICHISCHE COMPUTER GESELLSCHAFT ®
AUSTRIAN COMPUTER SOCIETY

Editors

Friedrich Fraundorfer, Peter M. Roth, and Fabian Schenk

Layout

Graz University of Technology
Institute of Computer Graphics and Vision

Cover

Graz University of Technology
Institute of Computer Graphics and Vision

Cover image: © Stift Vorau

ISBN e-book 978-3-85125-652-9
DOI 10.3217/978-3-85125-652-9

Contents

Preface

The *24th Computer Vision Winter Workshop (CVWW 2019)*, taking place at Stift Vorau, Austria, was organized by the Institute of Computer Graphics and Vision at Graz University of Technology. The Computer Vision Winter Workshop is the annual meeting of computer vision research groups located in Graz, Ljubljana, Prague, and Vienna. The main goal of this workshop is to communicate fresh scientific ideas within these four groups and to provide conference experience to PhD students. However, the workshop is open to everyone, which can be seen from many international contributions and attendees.

After a double-blind full paper review process by an international programme committee, finally, seven original works have been accepted for publication. These have been presented at the workshop as oral presentations. In addition, we were happy that *Gabriel J. Brostow* (University College London) accepted our invitation and gave an invited talk on *Self-supervision for 3D Shape and Appearance Modeling*. The workshop programme was completed by 13 further oral presentations.

Finally, we are happy, that excellent work could be highlighted by an award sponsored by the *Austrian Computer Society (OCG)*.

Friedrich Fraundorfer, Peter M. Roth, and Fabian Schenk
Vorau, February 2019

Workshop Chairs

Workshop Administration

Program Committee

Index of authors

Keynote Talk

Self-supervision for 3D Shape and Appearance Modeling

Gabriel Brostow
University College London
G.Brostow@cs.ucl.ac.uk

Abstract

A single glimpse is hardly enough to triangulate the 3D shapes of a scene. But many glimpses taken together, can give enough supervision to accomplish interesting tasks, such as depth from a single photo, volume from a single depth, and appearance of objects and scenes from novel viewing angles. In this talk, I will distill the main lessons we have learned recently, in attempting to a) design networks that understand "a bit" about 3D, and to b) train networks to predict depth, or volumes, or appearance, for several application domains. Some details matter, and the data itself is a key ingredient. There is still more exciting work to be done! This talk will cover equivariance, consistency losses, and some personal views on diversity in predictions.

Original Contributions

24th Computer Vision Winter Workshop
Friedrich Fraundorfer, Peter M. Roth, Fabian Schenk (eds.)
Stift Vorau, Austria, February 6–8, 2019

Situation-Aware Pedestrian Trajectory Prediction with Spatio-Temporal Attention Model

Sirin Haddad Meiqing Wu He Wei
Siew Kei Lam
Nanyang Technological University (NTU)
50 Nanyang Ave, Singapore
{siri0005,wei005}@e.ntu.edu.sg {meiqingwu,assklam}@ntu.edu.sg

Abstract.
Pedestrian trajectory prediction is essential for collision avoidance in autonomous driving and robot navigation. However, predicting a pedestrian's trajectory in crowded environments is non-trivial as it is influenced by other pedestrians' motion and static structures that are present in the scene. Such human-human and human-space interactions lead to non-linearities in the trajectories. In this paper, we present a new spatio-temporal graph based Long Short-Term Memory (LSTM) network for predicting pedestrian trajectory in crowded environments, which takes into account the interaction with static (physical objects) and dynamic (other pedestrians) elements in the scene. Our results are based on two widely-used datasets to demonstrate that the proposed method outperforms the state-of-the-art approaches in human trajectory prediction. In particular, our method leads to a reduction in Average Displacement Error (ADE) and Final Displacement Error (FDE) of up to 55% and 61% respectively over state-of-the-art approaches.

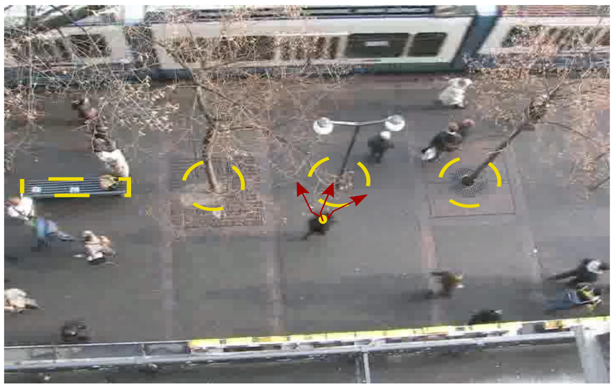


Figure 1: Estimating pedestrian trajectory given the surrounding environment physical structure in a real-life scenario. In the figure above, a lamp post lies in the direction of traversal of the pedestrian of interest. Thus it is essential to capture the existence of static obstacle and understand how they will navigate around it.

1. Introduction

The provision to estimate future trajectories of pedestrians and predicting the possibility of collisions can prevent accidents in autonomous driving and robot navigation. However, pedestrian trajectory prediction in crowded environments is a challenging task as human navigation decisions are influenced by their interactions with other traffic participants and the static physical objects. In particular, humans navigate in a situation-aware manner by avoiding collisions with static objects and other pedestrians in the space surrounding them, based on common social rules. As such, prediction models must take into account the interactions of both static and dynamic elements in the environment in order to accurately predict the pedestrians' motion paths. Figure 1 shows a real scenario that requires awareness of the lamp post presence in order to make realistic prediction about the pedestrian trajectory who will avoid walking into paths leading to the obstacles area.

Previous works that addressed human motion prediction focused on modeling human-human and human-space interaction separately. [1, 2, 3, 4, 5] account for scene static configuration such as obstacles and scene structures for improving human trajectory predictions in the presence of dynamic objects. However, these works mainly target constrained environments with low crowd density.

Recently, the work in [6] presented a deep convolutional network that models the impact of scene static elements on the pedestrian motion. However, they relied on complex tools comprising convolution layers and multiple feature maps for modeling knowledge about the scene. Recurrent neural networks in [7, 8, 9] tackled pedestrian trajectory prediction on challenging datasets of outdoor scenes [10]. Nevertheless, these approaches only modeled the social interaction among pedestrians without taking into account the surrounding static context. Social Attention [7] encapsulated the social interactions along the spatial and temporal domains by adopting spatio-temporal graph architecture. Their model considered the social interaction as a global event occurring between each and every pedestrian using their velocity to state their influence on each other. In contrast, Social LSTM [8] only accounted for the influences within a fixed-size local neighborhood.

In our work, we propose an enhancement to the models in [7, 11] and improve the modeling of multiple trajectories correlations over space-time dimensions using the 2D locations of the the static and dynamic elements. In particular, the proposed model overcomes the limitation of Social LSTM [8] which only accounted for the influence of other pedestrians within a local neighborhood, while at the same time being cognizant of the static obstacles at close proximity. This concern was not present in Social Attention [7]. Our intuition for this model is that while a pedestrian's trajectory can be affected by the dynamic motions of other pedestrians at a distance, the decision to avoid static objects is usually made when the pedestrian is close to the object. Thus, we manage to reduce the graph complexity and achieve more stable predictions by dynamically incorporating the static elements in the graph structure only when they potentially pose an impact on pedestrian trajectory.

Our main contributions are as follows: (1) we present a spatio-temporal graph that explicitly captures the global interaction of all the pedestrians in the scene and the local interaction with the static objects, and (2) we propose a new spatio-temporal attention mechanism for each pedestrian trajectory. This mechanism takes into account the local interaction among pedestrians and objects. Our spatio-temporal mechanism is inspired by the work of [12] which casts the attention methods [13] for sequence learning tasks on graphs. Experimental results on two widely-used datasets demonstrate that our method achieves significant quantitative and qualitative improvements over state-of-the-art methods for pedestrian trajectory prediction.

2. Related Work

In this section, we present a summary of research on pedestrians trajectory prediction. The literature branches into two main trends regarding context inclusion: local context and global context. Additionally, the existing works unfolds into two other branches in terms of distinguishing multiple objects influence: attention-based and uniformly-based approaches.

Local context Versus Global context. It is obvious from the previous introduction that the modern trajectory prediction approaches [14, 15, 8, 3, 16, 17] resorted to a limited spatial extent of the surrounding context as they observed the interactions occurring within short distance from the pedestrian included, while [7, 9] were globally-based as they considered all the pedestrians in the scene even those who are far away from each other.

According to local context methods, observing the interaction for a short duration once pedestrians are close enough to each other, gives limited understanding of the social interaction. While including the social interactions on a global scene scale, enables the model to better understand how the interaction evolves between a pair of pedestrians based on the velocity effect that the model inherently grasps upon capturing the change in the spatial distances along time.

Attention-based Versus Uniformly-based approaches. Pedestrians navigating in urban environments influence each other and very often are influenced by the obstacles around them, thus it is essential for predicting multiple pedestrians trajectories to recognize the importance of various sources impact on a pedestrian and pay attention to the more influential ones. Applying attention in sequence learning tasks has proved its effectiveness in the overall algorithm performance and in pedestrian trajectory prediction methods it helped drawing more plausible trajectories.

The variational encoder-decoder methods, such as, Social GAN [9] took the global neighborhood around pedestrian but it evaluates all pedestrians in a uniform manner, by assigning equal importance values to them. Existing RNN approaches [7] applied soft attention mechanism to assign different importance weights to multiple pedestrians based on their velocities. While [16] applied hard attention to assign weights based on pedestrians distance, they also introduced additional soft attention to evaluate the interaction salience in a scene region. So, their trajectory prediction drew conclusions about which region was more likely for a pedestrian to navigate through. In our work, we are rather interested in microscopic prediction of the interaction between pedestrians and a specific fixed obstacle, hence, we use the soft attention mechanism [12] to evaluate the social interactions only.

Graph-Structured Networks. Real-life applications generate complicated forms of information in which they are best represented through graph structures compared to other rigid hierarchical and end-to-end organizations. Variational Encoder-Decoder methods[9, 18, 19], have the advantage of generating a variety of results, however, they are not capable of providing a factorized and explicit high-level representation of the environment components. Graph Neural Network [20] advanced the application of graph-structured data in neural networks in environments that naturally contain highly interrelated behaviors, such as: social media, molecular biology, etc. Outdoor pedestrians navigation typically induces a spatio-temporal nature due to alterations that happen in pedestrian motion trajectory and the complex interactions with different objects. Therefore, modeling a rich interactive context requires a scalable graph-based structuring of the elements and factorize their relationships in a principled way. Neural relational networks [14, 15], attempted to predict the interactions among multiple moving object using physical motion semantics, however, they did not account for realistic scenarios such as urban environments, which makes these networks better fitting for object linear motion in free space.

Recurrent Neural Networks. Recently, Recurrent Neural Networks (RNN) have shown notable success in modeling data sequences and time-varying patterns. They organize in a recursively unfolded structure, which makes them a perfect choice for temporal analysis and sequence learning tasks, such as machine translation and human motion forecasting [21, 22, 23, 24, 25]. Tree-structured RNN [24], illustrated spatio-temporal network organization analogous to [11]. However, their spatio-temporal architecture was designed around a skeletal-based human activity prediction such that, all the units had fixed dependencies and belong to one cohesive movement. This prior assumption does not fit with highly dynamic contexts such as crowd motion.

Few models [11, 26, 27] structured RNN units based on graph topology that explicitly represented elements and their interactions semantics. In our paper, we extend the generic spatio-temporal graph used in [11] in a hybrid manner, by combining globally-based human-human interaction with locality-based human-space interaction, in addition to using attention mechanism to distinctively model social interactions.

3. Approach

3.1. Problem Definition

Given a set of static objects O, and a set of pedestrians V and their trajectories $X^t_{v_i}$ observed at time-steps t = 1,...,T_{obs}, our model predicts the future locations $\hat{X}^t_{v_i}$ at t = $T_{obs}+1, ..., T_{pred}$ time-steps, with regards to potential influence of any obstacles presented in the scene, such that T_{obs} = 8, $v_i \in V$, T_{pred} = 12.

3.2. Model Architecture

The spatio-temporal graph is a dynamic structure that evolves temporally and spatially, due to the motion state of the pedestrians and changes in the scene (e.g. as the elements in the scene increase/decrease). Figure 2 shows the corresponding representation of crowd subjects in spatio-temporal graphs $G = (V, \varSigma_S, \varSigma_T)$, comprising three key components: nodes set $V*$, spatial edges set Σ_S and temporal edges set Σ_T, where nodes represent the dynamic and the static element (e.g. pedestrians and static objects), spatial edges represent the relationship between two nodes to indicate the interaction between them. Temporal edges link the same pedestrian node over successive time-steps and thus connect the graph when it is unrolled over time.

Figure 2a illustrates the dynamic structure with an arbitrary crowd at two consecutive time-steps. At (t=1), there are four pedestrians. At (t=2), a new pedestrian (5) enters the scene. Notice that by (t=2), pedestrian (2) enters the vicinity of the red obstacle, where they appear to pass through the dashed circular boundary. Figure 2b shows the corresponding spatio-temporal graph representation, which evolves dynamically over the spatial and temporal domain. This is evident when the graph unfolds at (t=2), where a new node is introduced for pedestrian (5) and all pedestrian nodes are connected by undirected edges to model the mutual interaction. This creates 2(N-1) spatial edges between pedestrian nodes at every time-step, where N is the number of pedestrians. In contrast, only a single directed edge is pointing from the obstacle node to the corresponding pedestrian node to depict the influence posed by the static obstacle on pedestrian (2).

The components of graph G are replaced with the corresponding LSTM components, *temporal edgeLSTM, spatial edgeLSTM, nodeLSTM*. The relationship between two nodes is characterized by their relative coordinates, where $x_{v_2v_3}$ is the spatial distance between nodes v_2 and v_3, and $x_{v_2v_2}$ is location of node v that changes over time.

Eq. (1) defines *spatial edgeLSTM* embedding function ϕ that takes as input: $x^t_{v_2.}$, all the relative spatial distances between node x_{v_2} and its neighbors (e.g. including $x_{v_2v_3}$), embedding weight matrix W_s .

$$e^t_{v_2.} = \phi(x^t_{v_2.}; W_s) \tag{1}$$

The *spatial edgeLSTM*s take the embedded input feature along with previous spatial hidden states from all related nodes $h^{t-1}_{v_2.}$ and transform them using normally initialized weight matrix W^{lstm}_s. The output hidden states vector $h^t_{v_2.}$ is shown in Eq. (2).

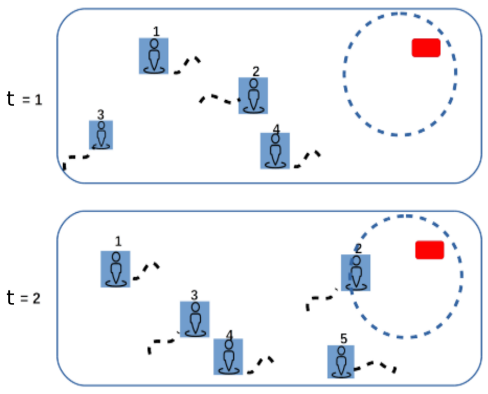
$$h^t_{v_2.} = LSTM(h^{t-1}_{v_2.}, e^t_{v_2.}, W^{lstm}_s) \tag{2}$$

Eq. (3) defines *temporal edgeLSTM* embedding function ϕ that takes as input: the temporal location of pedestrian node $x^t_{v_2v_2}$, embedding weight matrix W_t.

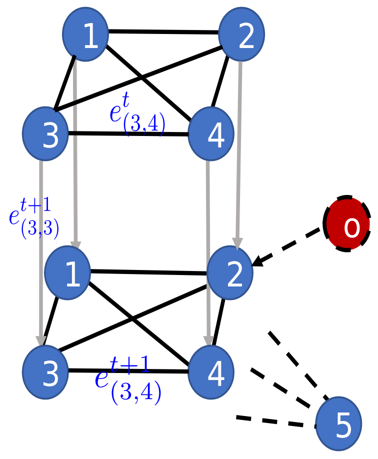
$$e^t_{v_2v_2} = \phi(x^t_{v_2v_2}; W_t) \tag{3}$$

Eq. (4) defines the LSTM cell and its inputs: previous temporal hidden state $h^{t-1}_{v_2v_2}$, embedded input feature $e^t_{v_2v_2}$ from Eq. (3) and normally initialized weight matrix W^{lstm}_t for transforming these inputs into the current hidden state $h^t_{v_2v_2}$.

$$h^t_{v_2v_2} = LSTM(h^{t-1}_{v_2v_2}, e^t_{v_2v_2}, W^{lstm}_t) \tag{4}$$



(a) Crowded environment displayed over 2 time-steps.



(b) Crowd mapping to abstract spatio-temporal graph unrolled through two time-steps

Figure 2: Crowd mapping to Spatio-temporal Graph. (a) A static obstacle is drawn as red rectangle surrounded by a virtual circle which indicates its neighborhood boundaries. (b) The Blue nodes represent pedestrians 1,2,3,4,5 and the red dashed node represents obstacle o such that $o \in O$. Directed downward lines indicate temporal edges linking the same node over time-steps and undirected lines are two-way spatial edges connecting pedestrian nodes. A directed edge is pointing from Obstacle node to pedestrian node to indicate obstacle influence on pedestrian. For the sake of clarity, we use dashed links from node (5) to indicate the remaining spatial edges. (Best viewed in color).

3.3. Spatio-Temporal Attention Module

Given the success of attentional mechanisms in sequence-based prediction of natural language processing applications, this work adopts the concept of attention-based generative algorithms [13]. We propose a variation of Multi-Head method, a soft attention based on two simple operations, i.e. concatenation and averaging across all edge feature vectors for

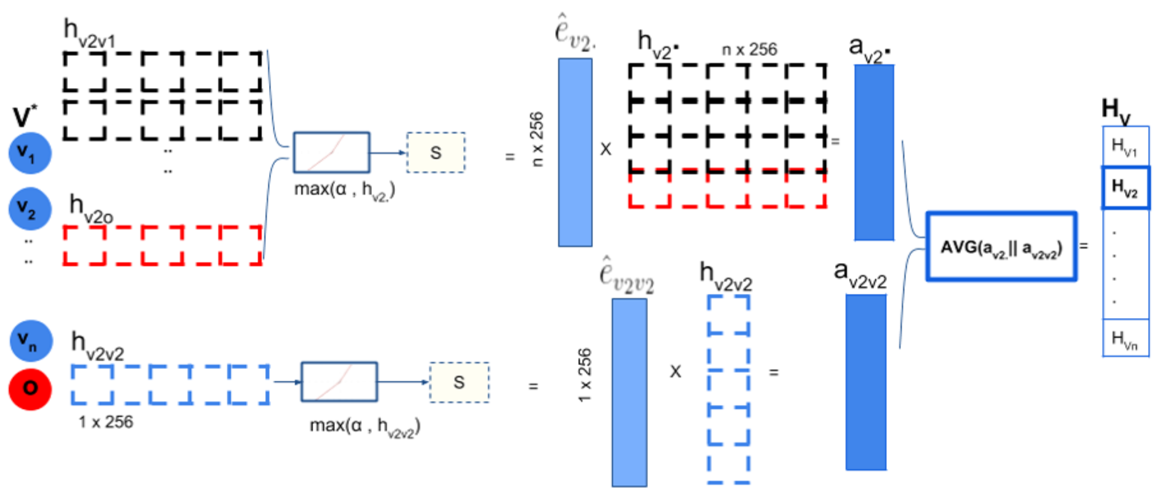


Figure 3: Multi-node attention mechanism pipeline for pedestrian node v_2 at time-step $t = 2$. On the left-most side, it shows nodes set $V^* = \{V, O\}$. The black dashed vectors store $h^{t+1}_{v2.}$, the hidden states of the spatial edges related to node v_2. The red dashed vector stores h^{t+1}_{v2o}, the hidden state of spatial edge between node v_2 and obstacle o. The blue dashed vector stores h^{t+1}_{v2v2}, temporal edge hidden state of node $v2$. These hidden states are then passed into PReLU and Softmax (S) activations to generate new embeddings $\hat{e}$. The concrete blue vectors store spatial hidden states $\hat{e}^{t+1}_{v2.}$ and temporal hidden state $\hat{e}^{t+1}_{v2v2}$. Multiplying the new embeddings vector by their hidden states array results in attention coefficients vectors a, where $a^{t+1}_{v2.}$ is the spatial attention coefficients vector and a^{t+1}_{v2v2} is the temporal attention coefficients vector.

each node. In [13], the input comprises fixed number of words with fixed positions, and the Multi-Head attention works by stacking multiple attention layers (heads) in which each layer makes mappings between words in two sentences. We use a simple attention mechanism, i.e. Multi-Node attention, which only has a single layer that jointly pays attention to the features from spatial and temporal domains and store the attention coefficients into single vector for node v_2 trajectory at each time-step. To illustrate this, Figure 3 exemplifies attention on pedestrian (2) and its neighbors at time (t=2). Neighboring *edgeLSTMs* states are transformed before concatenation using the embedding function in Eq. (5) and Eq. (6), which is a composite of Parametric ReLU and softmax. This combined activation ensures that hidden states remain within a small range of [-1,1] which will be mapped once again at the sampling stage to a range of normalized outputs range of [0,1].

$$\hat{e}^t_{v_2.} = softmax(PReLU(h^t_{v_2.})) \tag{5}$$

$$\hat{e}^t_{v_2v_2} = softmax(PReLU(h^t_{v_2v_2})) \tag{6}$$

The Parametric ReLU as illustrated in Eq (7), is the generalized ReLU function as it ties the leak parameter α as a network learnable parameter. Employing such activation function with an adaptive leak parameters, allows a slightly different span of the negative hidden states along training batches. This has proved its benefit for the model prediction performance.

$$PReLU(h) = max(0, h) + \alpha * min(0, h); \quad \alpha = 0.2 \tag{7}$$

The product of embedding vectors $\hat{e}$ with the original hidden states results in attention weights (also called coefficients). Eq. (8) and Eq. (9) shows the spatial attention coefficients $a^t_{v_2.}$ and temporal attention coefficients $a^t_{v_2v_2}$, respectively.

$$a^t_{v_2.} = \hat{e}^t_{v_2.}.h^t_{v_2.} \tag{8}$$

$$a^t_{v_2v_2} = \hat{e}^t_{v_2v_2}h^t_{v_2v_2} \tag{9}$$

Eventually, these coefficients will be concatenated and averaged to generate the final weighted hidden states vector ${H_{v_2}}^t$ as shown by Eq. (10):

$$H^t_{v_2} = \frac{\sum_v^N (a^t_{v_2v_2} || a^t_{v_2.})}{N}; \; N = |\, a^t_{v_2v_2} ||\, a^t_{v_2.}\, | \tag{10}$$

Comparing the Multi-Head attention with the single head multiplicative attention (scaled dot-Product), it turns out that the scaled dot-Product gives a compact representation of all incoming hidden states and it serves a similar objective to the linear pooling mechanism in [8] due to the highly variable-sized environment. However, it diminishes the expressive power lost upon compressing feature vectors size.

While Multi-Head attention averages across the spatial and temporal attention coefficients without compressing their depth. Hence, we realized that retaining the vectors depth provides sufficient feature representation for learning the influence of pedestrians on each other.

The pedestrian location coordinates $x^t_{v_2}$ are passed through an embedding layer ϕ as in Eq. (11) before its taken as input into *nodeLSTM*:

$$e^t_{v_2} = \phi(x^t_{v_2}; W_{embed}) \tag{11}$$

Finally, the output vector ${H_{v_2}}^t$ is concatenated with previous hidden state $h^{t-1}_{v_2}$, and which are then passed to *nodeLSTM* v_2, along with transformation weight matrix W^{lstm} to generate current hidden state $h^t_{v_2}$.

$$h^t_{v_2} = LSTM(e^t_{v_2}, concat(h^{t-1}_{v_2}, {H_{v_2}}^t, e^t_{v_2}), W^{lstm}) \tag{12}$$

The future location of pedestrian is sampled from a bivariate normal distribution *N* as in Eq. (14). For estimating the Mean μ, variance σ and correlation ρ we apply a linear transformation layer in Eq. (13) W_{out} to transform $h^t_{v_2}$ into the estimated parameters.

$$(\mu^{t+1}_{v_2}, \sigma^{t+1}_{v_2}, \rho^{t+1}_{v_2}) = W_{out} h^t_{v_2} \tag{13}$$

$$(x^{t+1}_{v_2}, y^{t+1}_{v_2}) \sim N\,(\mu^{t+1}_{v_2}, \sigma^{t+1}_{v_2}, \rho^{t+1}_{v_2}) \tag{14}$$

4. Experimental Results

4.1. Datasets and Metrics

Our evaluation is based on two widely-used datasets, ETH Walking Pedestrians (EWAP) [28], UCY Students and Zara [29]. In total, the datasets consist of five videos taken from outdoor surveillance cameras. The datasets contain 2206 human trajectories, exhibiting different traits that range between straight linear and curvilinear motion splines. From our observations, ETH scenes consist of more straight trajectories with few social interactions as the video captures people motion at the university entrance, while UCY scenes display more scenarios pertaining to human-space interactions. For example, the UCY-ZARA datasets include pedestrians bending at the shop entrance, while UCY-University scenes have more social interactions among standing groups. Furthermore, these cases in particular, increase the unpredictability of an individual path unless social and spatial contexts are taken into account. In our experiments, two benchmark metrics are used, i.e. Averaged Displacement Error (*ADE*) and Final Displacement Error (*FDE*) of the TrajNet challenge [10], for measuring Euclidean deviations (*in meters*) between predicted trajectory and actual trajectory.

Averaged Displacement Error: The mean average *l2 distances* between predicted trajectory coordinates $(\hat{x}, \hat{y})$ and true trajectory (x, y) for all time-steps $i = (1, .., n)$ over N pedestrian trajectories in the scene.

$$ADE = \frac{\sum_{j=1}^{N} \frac{\sum_{i=1}^{n} \sqrt{(\hat{x}^j_i - x^j_i)^2 + (\hat{y}^j_i - y^j_i)^2}}{n}}{N} \tag{15}$$

Final Displacement Error: The average *l2 distance* between the final predict step $(\hat{x}_n, \hat{y}_n)$ and the true step (x_n, y_n) over j pedestrians trajectory, where $j = (1, ..., N)$.

$$FDE = \frac{\sum_{j=1}^{N} \sqrt{(\hat{x}^j_n - x^j_n)^2 + (\hat{y}^j_n - y^j_n)^2}}{N} \tag{16}$$

4.2. Ablation Study

We have performed an ablation study by dropping the scaled-dot attention module from Social Attention and restoring back original settings of Structural-RNN, to study the usefulness of dot-Product attention model. The comparison between the quantitative results of both baselines with our method, shows that the scaled dot-Product performance is lower than the Multi-Node mechanism performance for the 5 datasets in Table 1. On the other hand, the optimal choice of the human-obstacle connectivity threshold $\lambda = 0.5$ parameter, was determined empirically, based on the objective of lowering the Euclidean errors for both evaluation metrics.

4.3. Training Setup

We accumulated trajectory data for every pedestrian with *skip_rate* = 10 frames to avoid overfitting the minimal changes in pedestrian trajectory. Each LSTM cell is of 256 depth. We transform data into normalized interpolated pixel coordinates within range [0,1]. In batch processing, we fixate the batch size *batch_size* = 24, observation length T_{obs} = 8 time-steps (3.2 seconds), prediction length T_{pred} = 12 time-steps (4.8 seconds) and epochs *epoch_num* = 100. After several hyper-parameter tunings, learning rate is set as *lr* = 0.001 and optimizer algorithm is Adam. Activation function in attention layer is Parametric ReLU, initialized to negative slope α = 0.20 and fractionally degraded throughout the training process. The training objective is to minimize the negative log-likelihood loss of the i^{th} trajectory from time-step T_{obs+1} to T_{pred}:

$$L_i = \quad - \sum_{t=T_{obs}+1}^{T_{pred}} log(P(x_i^t, y_i^t | \sigma_i^t, \mu_i^t, \rho_i^t)) \quad (17)$$

4.4. Quantitative Results

As illustrated in Table 1, we set up experiments to evaluate our proposed models, H-H and H-H-O, which stand for Human-Human and Human-Human-Obstacle respectively. The table has two segments, the first 4 rows evaluate our model with graph-based baselines: Social Attention and Structural-RNN, while the next 3 rows evaluate our model with state-of-the-art models: Social-LSTM and Social GAN (SGAN). Our attention mechanism for graphs improved prediction for human-human interaction and human-obstacle interaction over the other graph-based baselines: Social Attention and Structural-RNN. This is observed from the average errors under column (AVG) in Table 1. Comparing H-H-O with Social Attention, H-H-O achieves 55% in the average of ADE and 61% in the average of FDE in all datasets. As Social GAN and Social-LSTM display the best trajectories produced by their models, we extracted the average of minimum errors pertaining only to the best predicted trajectories in H-H-O model. It can be observed that the minimum FDE is considerably lower than minimum FDE generated by SGAN model and Social LSTM, due to our model awareness of surrounding context. This has made predictions to be plausible and compliant with the environmental constraint. The Social GAN work shows several versions of their model, so we selected their best model version which is SGAN-20V for our comparison. The most significant improvement is realized when comparing our model with Social GAN model, under the Hotel set with 93% reduction in FDE. Furthermore, the Hotel scene contains more static elements such as trees and lamp posts as indicated in Figure 2. The second best improvement is realized when comparing our model with Social-LSTM model under the ETH set with 89% reduction in FDE. The ETH dataset consists of a set of tightly coupled trajectories due to the crowd at the university entrance. This is a busy contextual point where pedestrians are mostly concerned about avoiding collisions with each others at the entrance site. Additionally, our model performance yields 69% reduction on FDE metric in Ucy-University, which proves that embedding information about physical structure of the scene and busy interaction points, refines the model understanding of pedestrian navigation in crowded sites and reduced the prediction errors in FDE, as our model was more capable of predicting the final step on a pedestrian trajectory. From the previous table, it is noticeable that the ADE and FDE exhibit small discrepancies due to the accumulative nature in prediction errors. If the predicted path was entirely approximate to the ground-truth, the final predicted point will not have large error, but if the prediction was increasingly deviating along the ground-truth, this can impact the final point errors. This supports our quantitative results as being consistent and realistic.

4.5. Qualitative Results

In this section, we qualitatively evaluate model predictions in Hotel and ZARA sets. Figure 4 displays predicted paths from our models. We have spotted interesting cases for pedestrian moving near static objects, and compared both of our models outputs, Social Attention and Social LSTM with ground-truth trajectory. Notice the Human-Human model prediction for pedestrian walking near the bench in Figure 4a. The ground-truth shows that pedestrian is avoiding the bench, while Human-Human model spline achieves lower displacement than the baseline splines, those fail at evading the bench area. This case is correctly predicted in our Human-Human-Obstacle model as illustrated in Figure 4b. Additionally, Figure 4d shows that Social Attention and Social LSTM predicts plausible paths that pedestrian might have chosen, however, it is not compliant with pedestrian surrounding objects. Thus, with the aid of obstacle awareness, our model understands pattern of collision avoidance with any static subject in their way.

Figure 4c plots trajectories from H-H-O model where pedestrians are bending toward the shop en-

Table 1: Prediction errors ADE/FDE (in meters). Our results are averaged over 30 sampled sequences of 12-steps length for every set under our method. For baselines errors, Social Attention results are obtained upon re-training their model, while Structural-RNN results are obtained upon manual implementation of their architecture in PyTorch.

Method	ETH	HOTEL	ZARA1	ZARA2	UNIV	AVG
Structural-RNN	2.72/4.60	0.85/1.35	1.05/2.20	1.60/3.50	1.45/3.00	1.53/2.93
Social Attention	3.60/4.70	0.79/1.44	1.30/2.66	0.95/2.05	1.00/2.14	1.53/3.52
H-H	**1.19/2.00**	**0.39**/0.96	0.55/1.56	0.58/1.50	0.74/1.89	**0.69**/1.58
H-H-O	1.24/2.35	0.48/**0.80**	**0.51/1.15**	**0.56/1.13**	**0.69/1.45**	0.70/**1.38**
Social LSTM	1.09/2.35	0.79/1.76	0.47/1.00	0.56/1.17	0.67/1.40	0.72/1.54
SGAN-20V	**0.81**/1.52	0.72/1.61	**0.34**/0.69	**0.42**/0.84	0.60/1.26	**0.58**/1.18
Minimum H-H-O	0.96/**0.16**	**0.35/0.11**	0.57/**0.30**	0.58/**0.33**	**0.53/0.38**	0.60/**0.26**

trance, and our model generates splines that approximate the curvy ground-truth trajectory, as the model learns the motion pattern at the entrance point.

In some situations, the predictions do not perfectly match the ground-truth path, although the deviations are quite small. This situation also applies for the baseline models. Upon extensive visual comparisons for all frames in all datasets, we confirmed that the erroneous results and deviations of the proposed method are much fewer than those found in the baselines plots. Quantitatively, Euclidean deviations at the path endings have been reduced by up to 61%, which identifies the improvements that we highlighted earlier.

5. Conclusion

In this paper we have presented a new spatio-temporal graph that operates on the local and global contexts around pedestrian, for predicting their trajectory in outdoor environments. For an accurate modeling of human-human interactions and human-space interactions, we employ a simplified version of Multi-Head attention mechanism for accumulating the influence from spatial and temporal subspaces. Our attention mechanism consistently demonstrated improved prediction results over baseline methods, for groups as well as individual non-linear trajectories.

References

[1] H. Kretzschmar, M. Kuderer, and W. Burgard, "Learning to predict trajectories of co-operatively navigating agents," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*. IEEE, 2014, pp. 4015–4020. 1

[2] H. S. Koppula and A. Saxena, "Anticipating human activities using object affordances for reactive robotic response," *IEEE transactions on pattern analysis and machine intelligence*, vol. 38, no. 1, pp. 14–29, 2016. 1

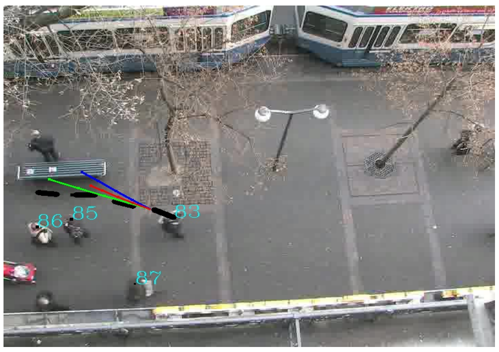
[3] F. Bartoli, G. Lisanti, L. Ballan, and A. Del Bimbo, "Context-aware trajectory prediction," *arXiv preprint arXiv:1705.02503*, 2017. 1, 2

[4] D. Ellis, E. Sommerlade, and I. Reid, "Modelling pedestrian trajectory patterns with gaussian processes," in *Computer Vision Workshops (ICCV Workshops), 2009 IEEE 12th International Conference on*. IEEE, 2009, pp. 1229–1234. 1

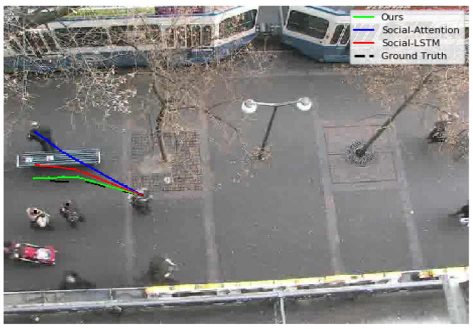
[5] K. Kim, D. Lee, and I. Essa, "Gaussian process regression flow for analysis of motion trajectories," in *Computer vision (ICCV), 2011 IEEE international conference on*. IEEE, 2011, pp. 1164–1171. 1

[6] A. Robicquet, A. Sadeghian, A. Alahi, and S. Savarese, "Learning social etiquette: Human trajectory understanding in crowded scenes," in *European conference on computer vision*. Springer, 2016, pp. 549–565. 2

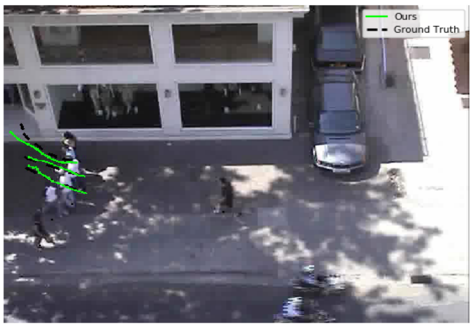
[7] A. Vemula, K. Muelling, and J. Oh, "Social attention: Modeling attention in human crowds,"



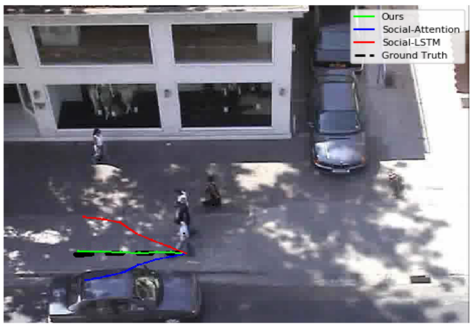
(a) H-H model - Hotel scene



(b) H-H-O model - Hotel scene



(c) H-H-O model - ZARA scene



(d) H-H-O model - ZARA scene

Figure 4: Visualization results for Hotel and Zara sets.

in *Proceedings of the International Conference on Robotics and Automation (ICRA) 2018*, May 2018. 2

[8] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese, “Social lstm: Human trajectory prediction in crowded spaces,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 961–971. 2, 6

[9] A. Gupta, J. Johnson, L. Fei-Fei, S. Savarese, and A. Alahi, “Social gan: Socially acceptable trajectories with generative adversarial networks,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, no. CONF, 2018. 2, 3

[10] A. Sadeghian, V. Kosaraju, A. Gupta, S. Savarese, and A. Alahi, “Trajnet: Towards a benchmark for human trajectory prediction,” *arXiv preprint*, 2018. 2, 6

[11] A. Jain, A. R. Zamir, S. Savarese, and A. Saxena, “Structural-rnn: Deep learning on spatio-temporal graphs,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 5308–5317. 2, 3

[12] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, “Graph attention networks,” *stat*, vol. 1050, p. 20, 2017. 2, 3

[13] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, “Attention is all you need,” in *Advances in Neural Information Processing Systems*, 2017, pp. 6000–6010. 2, 4, 5

[14] S. van Steenkiste, M. Chang, K. Greff, and J. Schmidhuber, “Relational neural expectation maximization: Unsupervised discovery of objects and their interactions,” *arXiv preprint arXiv:1802.10353*, 2018. 2, 3

[15] P. Battaglia, R. Pascanu, M. Lai, D. J. Rezende *et al.*, “Interaction networks for learning about objects, relations and physics,” in *Advances in neural information processing systems*, 2016, pp. 4502–4510. 2, 3

[16] T. Fernando, S. Denman, S. Sridharan, and C. Fookes, "Soft+ hardwired attention: An lstm framework for human trajectory prediction and abnormal event detection," *Neural networks*, vol. 108, pp. 466–478, 2018. 2, 3

[17] D. Helbing and P. Molnar, "Social force model for pedestrian dynamics," *Physical review E*, vol. 51, no. 5, p. 4282, 1995. 2

[18] D. Varshneya and G. Srinivasaraghavan, "Human trajectory prediction using spatially aware deep attention models," *arXiv preprint arXiv:1705.09436*, 2017. 3

[19] N. Lee, W. Choi, P. Vernaza, C. B. Choy, P. H. Torr, and M. Chandraker, "Desire: Distant future prediction in dynamic scenes with interacting agents," 2017. 3

[20] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE Transactions on Neural Networks*, vol. 20, no. 1, pp. 61–80, 2009. 3

[21] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014. 3

[22] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014. 3

[23] S. Song, C. Lan, J. Xing, W. Zeng, and J. Liu, "An end-to-end spatio-temporal attention model for human action recognition from skeleton data." in *AAAI*, vol. 1, no. 2, 2017, p. 7. 3

[24] J. Liu, A. Shahroudy, D. Xu, and G. Wang, "Spatio-temporal lstm with trust gates for 3d human action recognition," in *European Conference on Computer Vision*. Springer, 2016, pp. 816–833. 3

[25] H. Xue, D. Q. Huynh, and M. Reynolds, "Ss-lstm: A hierarchical lstm model for pedestrian trajectory prediction," in *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, 2018, pp. 1186–1194. 3

[26] X. Liang, X. Shen, J. Feng, L. Lin, and S. Yan, "Semantic object parsing with graph lstm," in *European Conference on Computer Vision*. Springer, 2016, pp. 125–143. 3

[27] Y. Yuan, X. Liang, X. Wang, D. Y. Yeung, and A. Gupta, "Temporal dynamic graph lstm for action-driven video object detection," *arXiv preprint arXiv:1708.00666*, 2017. 3

[28] S. Pellegrini, A. Ess, K. Schindler, and L. Van Gool, "You'll never walk alone: Modeling social behavior for multi-target tracking," in *Computer Vision, 2009 IEEE 12th International Conference on*. IEEE, 2009, pp. 261–268. 6

[29] A. Lerner, Y. Chrysanthou, and D. Lischinski, "Crowds by example," in *Computer Graphics Forum*, vol. 26, no. 3. Wiley Online Library, 2007, pp. 655–664. 6

24th *Computer Vision Winter Workshop*
Friedrich Fraundorfer, Peter M. Roth, Fabian Schenk (eds.)
Stift Vorau, Austria, February 6–8, 2019

SyDD: Synthetic Depth Data Randomization for Object Detection using Domain-Relevant Background

Stefan Thalhammer, Kiru Park, Timothy Patten, Markus Vincze
Automation and Control Institute, TU Wien
Gußhausstraße 27-29, 1040 Vienna, Austria
{thalhammer, park, patten, vincze}@acin.tuwien.ac.at

Walter Kropatsch
Institute of Visual Computing and Human-Centered Technology, TU Wien
Favoritenstraße 9, 1040 Vienna, Austria
krw@prip.tuwien.ac.at

Abstract. *In industry CAD-models are readily available while it is expensive to obtain 3D scans of actual objects. Consequently, training object detectors exclusively from CAD-models leads to a considerable decrease of the data creation effort. While this works well for recognition, detection requires better models to distinguish the object of interest from the background and to take the expected sensor properties into account. To tackle this problem we synthetically create depth data with domain-relevant background and apply randomized augmentation to create a superset of the variations of real-world depth images. Results with a state-of-the-art object detector, trained using our synthetic data, show that our approach yields better results than learning from real-world, hand-annotated data with the LineMOD dataset.*

1. Introduction

Assembly systems in manufacturing are subject to increasing number of variants, smaller lot sizes and shorter life cycles. The application of assistance systems will lead to a reduced error rate and increased capacity [4]. The task of visual assistance systems is accurate and robust object detection.

Recently deep learning advanced the state of the art for computer vision tasks such as object detection. While deep networks achieve superior performance, they require a huge amount of training data [8]. Capturing and annotating these data is time and labour consuming and often requires physical in-

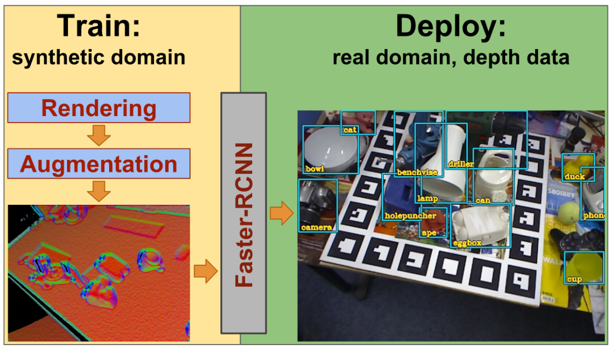


Figure 1. By rendering scenes with domain relevant objects and augmenting the noise model, we create better synthetic training data for object detection.

stances, which is problematic in fast paced manufacturing environments. Industrial applications, however, typically have CAD-data readily available. We propose to take advantage of this by creating synthetic training data directly from CAD-models by rendering depth images from a virtual scene representing the domain of deployment.

Synthetic depth images are rendered using CAD models to create a scene then we apply a randomized noise model. A standard tool to create synthetic data is the freely available, open-source software Blender[1] [1, 2, 11]. When training an object detector it is important to create data of sufficient variability to discriminate the objects of interest from the background. For object recognition, where one object is identified in a cropped image, it is known that a randomized background is sufficient to improve re-

[1]www.blender.org

sults [15, 17]. For object detection (i.e. classification and bounding box regression), where multiple objects are identified in a scene, randomized backgrounds are still insufficient to overcome all the ambiguities. Inspired by Handa *et al.* [2], who create full synthetic scenes for a semantic segmentation task, we propose to create scenes that include the expected object placements for better training with realistic depth images.

Another issue to consider is that training deep networks using synthetic data and deploying these on real-world data leads to reduced performance due to the different domains, the so called reality gap. A common method to close the reality gap is to create data of sufficient variability using domain randomization [16] or using the Perlin noise technique [17]. A major challenge is to capture the expected variations in the actual test images. Hence, we propose to combine Perlin noise [10] with a randomized sensor model in order to improve object detection in real-world depth images.

In summary, we propose a domain-related rendering step with an improved noise modelling step referred to as augmentation. Figure 1 outlines the approach. The contributions are the following:

- Rendering synthetic scenes with domain relevant objects to create a realistic background for object detection in depth images.
- Introducing randomized augmentation of synthetic depth images to better capture the expected variations in real-world data.
- Showing advance by evaluating object detection on a standard dataset, the LineMOD dataset [5], since bounding-box targets and class labels are available.

The remaining paper is structured as follows. Section 2 summarizes related work. The rendering and augmentation method is described in Section 3. Section 4 presents the results and evaluation. Section 5 concludes with a short discussion.

2. Related Work

This section discusses synthetic data creation and domain randomization for object detection.

2.1. Synthetic Data Creation

Carlucci *et al.* [1] use *Blender* to create a synthetic depth image dataset for object recognition. They use 3D CAD models downloaded from different databases to create object categories. Views are rendered from a configuration space consisting of object distance, camera position, focal length and random object warping minimizing the amount of identical rendered images. Planche *et al.* [11] present a pipeline to render realistic depth images for object recognition. They simulate the image appearance for a wide range of sensors. Their pipeline consists of a pattern projection mechanism, an intermediate step impinging sensor noise followed by stereo matching and post-processing to reproduce the spatial sensitivity of the sensors and to simulate the impact of surface materials. Backgrounds such as primitive shapes and captured real-world scans can be added. Rozantsev *et al.* [12] project the object geometry, taken from CAD models, into RGB images. A texture filling algorithm varies the object appearance with respect to blur, noise and material properties. Su *et al.* [15] render multiple views of 3D objects to generate a single compact descriptor of that object using a CNN. Handa *et al.* [2] create annotated synthetic indoor scenes using an automatic furniture arrangement mechanism. They use a simulated Kinect noise model to include noise in the synthetic depth scans.

In order to enable object detection in synthetic images it is important to create background information with sufficient variability to separate the objects of interest from the insignificant scene parts. Previous work has only addressed the randomization and augmentation of the generated data for the object of interest, which is mainly due to the focus on the task of object recognition. We instead consider object detection, and thus augment full scenes, including the background, to generate high quality training data.

2.2. Domain Randomization

Since we use an off-the-shelf architecture as detector, trained on synthetic data, it is necessary to transfer the domain to match the real-world image statistics. Domain randomization is a common strategy to create data of sufficient variability to include the variations of a desired domain [14, 16, 17].

Sadeghi *et al.* [14] learn collision avoidance for autonomous flight from simulation. They render RGB-images from synthetic 3D hallways. Parameters such as wall textures, furniture position, illumination and camera pose are randomized. Tobin *et al.* [16] use domain randomization to produce sufficient variability at training time to enable robot

Table 1. Background objects in the virtual scenes.

simple	no additional background information
limited	Apple IMac, bin, keyboard, lamp, laptop, two types of screens, mouse, pot plant, speakers
realistic	All from the *limited* objects, Apple Iphone, ball, BeatsAudio, two types of cans, bottle, Buick model, bulb, DualShock 4 controller, pc fan, knife, Nintendo Gameboy, Nvidia GeForce GTX 1080, plier, spacer, stapler, tablet

grasping. Their approach randomizes shape, position, orientation and texture of the objects involved. The characteristics of lights and the camera extrinsics are also randomized. Zakharov *et al.* [17] use domain randomization to augment depth images. Fractal Perlin noise, Voronoi texturing and white noise is used as background for rendered 3D object models. Perlin noise is an inexpensive way to simulate sensor noise. Randomized patterns are used to simulate occlusion.

A remaining challenge for domain randomization is to randomize the data in the source domain in such a way that the variations of the target domain are captured. We address this by augmenting synthetic depth scans using a combination of Perlin noise and a randomized sensor noise model. Variations of the background information and the occlusion patterns are achieved by randomizing the placement of domain-relevant objects.

3. SyDD: Closing the Reality Gap

We present a method to create and subsequently augment synthetic depth images. The pipeline, named *SyDD*, is presented in Figure 2.

The first step is the creation of synthetic depth images from a virtual scene. The second step is augmenting the synthetic depth images by adding randomly sampled variation to the pixel's depth values. This variation of the augmented domain X^{a} (depth noise, lateral noise, occluded image regions, errors due to the limited depth resolution) creates a superset $X^{\mathrm{a}} \supseteq X^{\mathrm{r}}$ of the variations in real-world scans, i.e., the target domain X^{r}. However, we take care that X^{a} does not diverge too much from X^{r} by choosing variations in a way not to violate the sampling theory.

3.1. Rendering: Synthetic Data Creation

We create synthetic data with diverse scene setups and background information in order to produce data with high variation to train object detectors. Three different approaches to create synthetic scenes are chosen in order to evaluate the importance of the background information:

- *simple*: Objects are arranged on a table, without further background information.
- *limited*: Objects are arranged on a table with static domain-relevant background objects.
- *realistic*: Objects are arranged on a table with static domain-relevant background objects and randomly placed domain-relevant objects.

Table 1 presents a list of background objects used for rendering. The additional objects are downloaded from GrabCAD[2].

For every scene five to eight objects of interest are randomly placed with repetition. These objects are annotated with a bounding box and with pixel class correspondences if fully visible. If not fully visible the bounding box is reduced accordingly and occluded pixels are not annotated. The camera pose is randomly chosen from valid views described in the dataset used for validation. The output of the synthetic data creation step is a depth image, a binary mask indicating visible image regions and a mask indicating pixel level class correspondences. The binary mask provides information about image regions with valid depth values due to the baseline distance of the infrared projector and the sensor. Figure 3 shows an example of the synthetically rendered depth images and visibility masks.

3.2. Augmentation: Randomized Depth Image Variations

We apply an augmentation loosely based on a sensor model and Perlin noise-based pixel warping to the rendered depth images. In order to create a superset of the variations of real-world depth images the parameters of our augmentation are randomly chosen for each image.

Various works evaluate and quantify the errors of the depth scans from infra-red based structured light cameras such as the *Microsoft Kinect V1*. The most common sources of error are the depth sensor itself, the measurement setup and properties of the object surface. Missing depth values are typically caused by infrared occlusion, specular surface reflection and

[2]https://grabcad.com

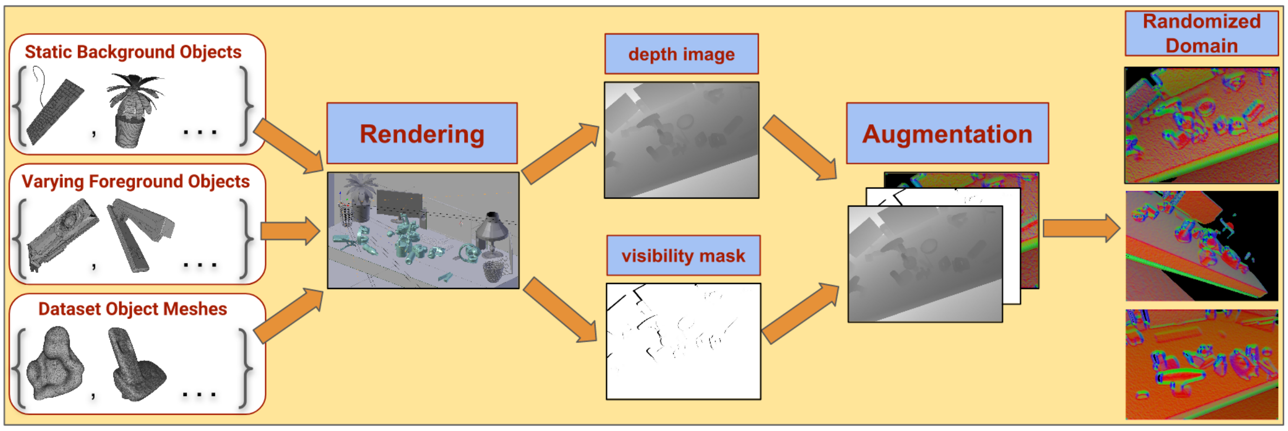


Figure 2. Synthetic depth image creation and augmentation pipeline.

gaps in the depth images due to strong light [7]. Our approach is designed for objects of interest with surface materials that diffusely scatter incoming light, hence omitting the simulation of specular reflections. We propose to randomize the parameters of our augmentation to account for the intractable number of variations and combinations of the influences in the depth image capturing process.

Based on the imaging geometry, parts of the scene are occluded, these occlusions are affected by strong light illuminating the scene. In order to simulate that influence morphological opening and subsequent median filtering is applied to the mask image that is created by the rendering script. The binary mask is applied to the synthetic depth images to remove the occluded image regions. The kernel sizes are sampled from $\{3, 5, 7\}$. These kernel sizes are also used for blurring.

For further augmentation depth images are resized to 320 by 240 pixels, since that is the resolution of the infra-red based structured light camera, the *Microsoft Kinect V1*. The images are down sampled using area interpolation to avoid aliasing. Blur is added to minimize the discrepancy between depth gradients in the real-world and synthetic images. The standard deviation of the blurring operation is chosen uniformly in a range from 0.25 to 3.5. The synthetic depth values are rounded to the nearest quantization value, based on the hypothesized sensor's depth resolution [7] to obtain synthetic depth values in an eleven bit range.

Additional noise is added to the quantized depth values using an offset chosen randomly from a Gaussian distribution. The depth noise of the sensor increases non-linearly with depth, though since the expected object placement is in a range of 65 centimeters to 115 centimeters we approximate it linearly, similar to [6]. The offset is calculated per pixel using its nearest quantized value, scaled by the parameter n_{sd}. The randomized parameter n_{sd} is drawn uniformly between 0.002 and 0.004. This range is based on the actual depth noise of the *Microsoft Kinect V1*.

Further randomness of the appearance of occluded scene parts, depth and lateral noise is added by warping the depth images through the application of pixel offsets, using the Perlin noise technique. This approach is similar to Zakharov *et al.* [17]. The basic concept is that a 3D vector field is generated to randomly distort synthetic depth images. Pixel locations are warped by applying the sampled vector field to the already augmented depth images. We use their proposed parameter ranges.

An example of the synthetic depth scans is presented in Figure 4.

4. Experiments

Three experiments are conducted to evaluate our approach. First, we compare object detection trained on the same number of real-world and on synthetic depth images. Second, results are presented providing quantitative information about the influence of the background in the synthetic scene. Third, the influence of the different steps of the augmentation method is shown. Finally, we discuss open problems.

All the experiments are conducted on the LineMOD dataset [5], which is taken from the *SIXD Challenge 2017*[3]. This is a standard and well-known baseline for object recognition and pose estimation in RGB-D. The test set of the LineMOD dataset consists of 15 test sets, one for each dataset object, with approximately 1200 captured images per scene. Every set has different object instances visible, although

[3]http://cmp.felk.cvut.cz/sixd/challenge_2017

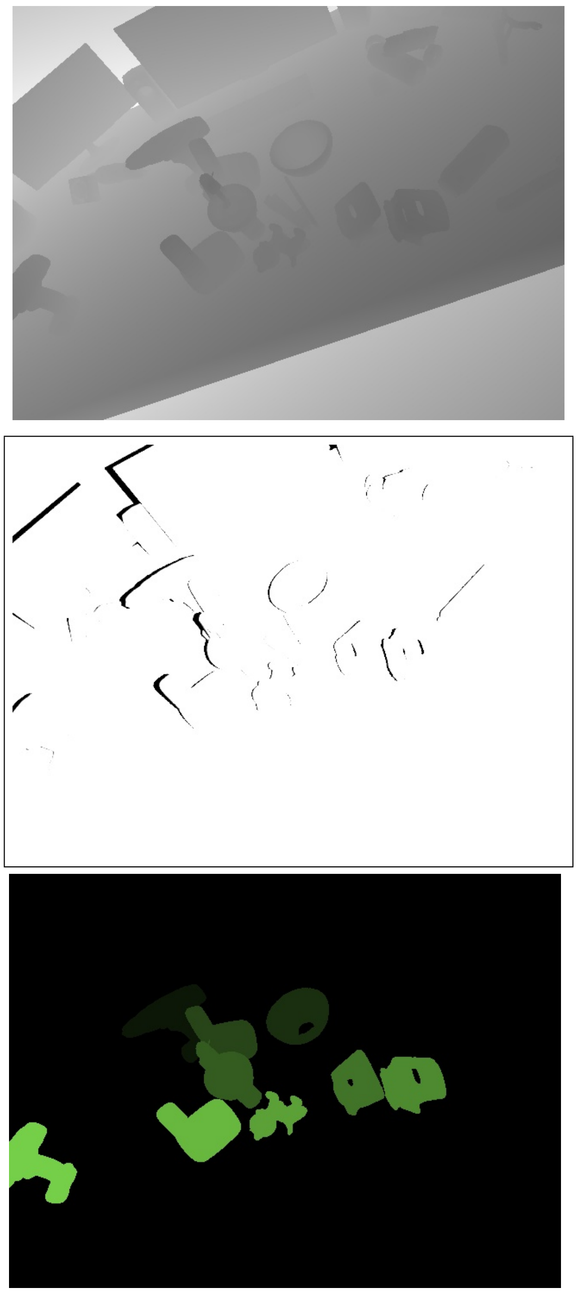


Figure 3. Synthetic depth image (top), visibility mask (middle) and pixel level class correspondence (bottom).

only the object in the center of the image is annotated with a bounding box, class and pose. An exception is the *benchvise* test set that has all dataset objects annotated. Since different object instances without annotation are visible in the test images, only the annotated object is considered for calculating the detection recall. In all experiments we report the percentage of correctly detecting and classifying annotated objects with an Intersection-over-Union (IoU) of 0.5.

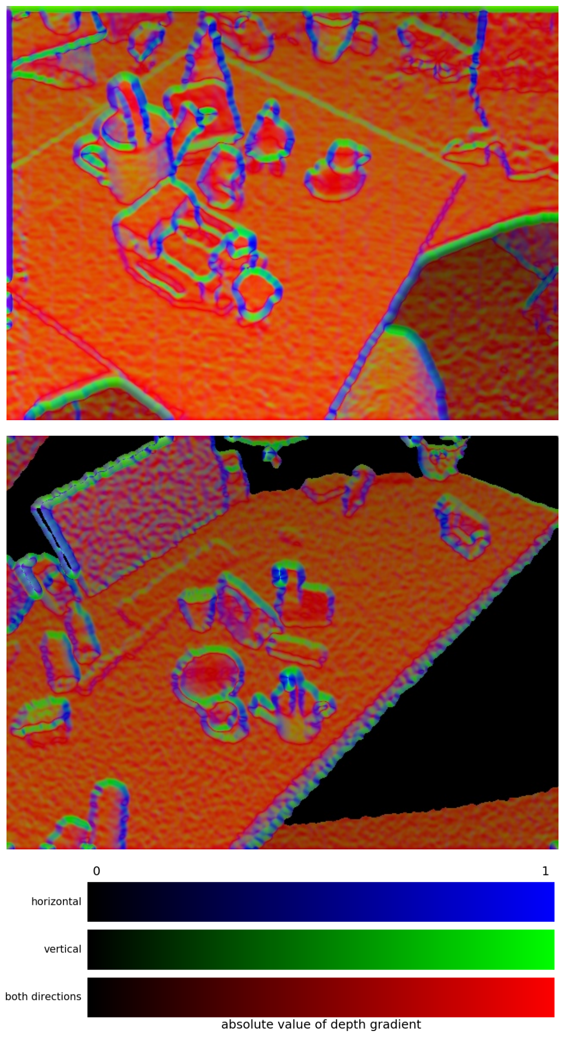


Figure 4. Comparison of a real-world (top) and a synthetic (bottom) depth image, converted to RGB images.

4.1. Experimental Setup

All tests are conducted with the following preprocessing and network configuration. All real-world and synthetic images are converted to three channel RGB images. These are coloured based on the normal direction using the approach of Nakagawa *et al.* [9]. Image regions with missing depth values are inpainted and depth cuts are applied up to 20 centimeters and regions further than 1.8 meters.

We use Faster-RCNN with ResNet-101 [3] backbone, pretrained on ImageNet [13], with the standard optimizer and loss functions. The learning rate starts at 0.01 and decays to 0.0001. We train for 180000 iterations using a batch size of one and a weight decay of 0.0001.

4.2. Performance on real-world data

We compare an object detector trained on real-world images that are taken from the *benchvise* test

Table 2. Detection recall of Faster-RCNN trained on real-world and on synthetic data. Numbers in percent.

Classes	real	*SyDD*
ape	53.56	**76.86**
can	**97.24**	94.15
cat	41.31	**82.70**
driller	**96.21**	92.76
duck	89.39	**93.70**
eggbox	64.8	**81.01**
glue	**81.72**	70.08
holepuncher	**89.89**	77.69
overall	76.77	**83.62**

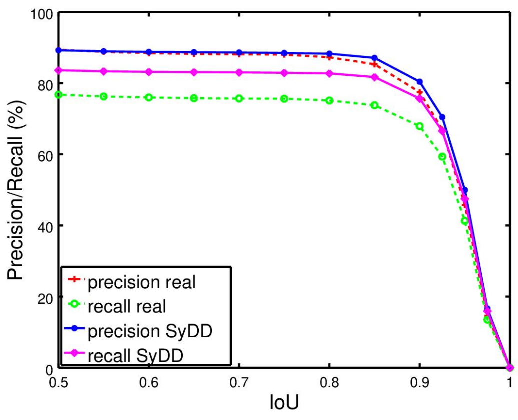


Figure 5. Recall and precision curve comparison of real-world and synthetic data using different IoU scores.

set of the LineMOD dataset against an object detector trained on images created by *SyDD*. Table 2 compares per category detection recall.

The average recall of the detector trained on our synthetic dataset outperforms the recognizer trained on real-world data. The performance margin results from the higher variability in the synthetic dataset. The biggest differences in detection recall are visible for the objects *ape*, *cat* and *eggbox*. This is caused by the scene setup used for capturing the real-world depth scans. The *ape* is placed in different poses in the scene and is either not occluded or completely occluded in most of the images. The *benchvise* test set only includes these extreme cases and does not have many examples for partial occlusion. The *cat* is placed with the same pose in all scans, which again results in very low visibility or full visibility with the addition of low variability of pose in the *benchvise* test images. The *eggbox* is placed in different poses but with a strong similarity of viewpoints. Furthermore, occlusion is caused mostly by the same object. The randomized augmentation covers a wider range of variations influencing the image creation process

Table 3. Detection recall of Faster-RCNN trained on *SyDD*, with different backgrounds in the virtual scenes. Numbers in percent.

Classes	simple	limited	realistic
ape	57.79	59.79	**79.69**
benchvise	65.16	64.09	**96.05**
bowl	91.24	**93.03**	85.81
camera	66.61	74.94	**94.17**
can	57.02	79.10	**91.97**
cat	58.95	80.75	**97.71**
cup	76.13	83.06	**88.06**
driller	65.99	84.76	**96.72**
duck	82.30	81.58	**95.37**
eggbox	83.32	92.42	**93.77**
glue	65.16	79.98	**82.79**
holepuncher	74.54	85.53	**92.97**
iron	42.19	64.58	**89.84**
lamp	50.77	65.69	**96.09**
phone	71.36	68.22	**93.00**
overall	63.03	72.29	**85.88**

as well as the placement of objects in the virtual scene. This increases the variation of occlusions and views in comparison to the real-world images in the *benchvise* test set. Figure 5 indicates that the performance of the Region Proposal Network is not affected by the usage of our synthetic training data.

4.3. Influence of the Background Information

The importance of the background information in the training data is evaluated by comparing three object detectors trained on different background objects, each consisting of 10000 images. Table 3 shows the performance recall.

The results indicate that the usage of additional background information during synthetic data generation improves the detection recall. Results also indicate that is unnecessary to use the same background objects during training as during deployment. Our findings show that domain specific background objects are sufficient for detectors to yield similar performance to detectors trained on real-world, hand-annotated images. The reader is directed to the detection results of the *bowl*. The recall for this object decreases with the usage of more comprehensive background information.

4.4. Evaluation of the Augmentation Method

The influence of the augmentation used for creating training data is evaluated by comparing four object detectors trained on 10000 images. The augmen-

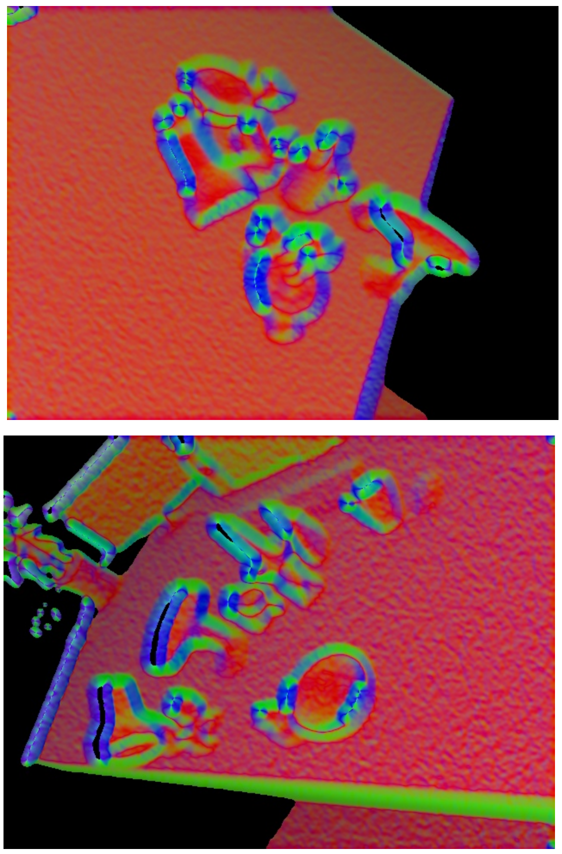


Figure 6. Exemplary images, displaying the synthetic training dataset, with simple background information (top) and limited background information (bottom).

tation methods are:

- *synth*: non augmented synthetic depth data.
- *perlin*: augmenting the synthetic images only using Perlin noise with the parameters from [17], after removing occluded image regions using the randomized visibility mask.
- *auth*: randomized realistic sensor model, where the difference to our proposed method *SyDD* is that the depth noise n_{sd} is added before quantizing these to eleven bit range.
- *SyDD*: our proposed augmentation.

The results presented in Table 4 indicate that strong average detection performance is achieved when adding Perlin noise. However, even better performance is achieved using our augmentation. We conclude that augmenting images with Perlin noise can effectively close the domain gap, but combining with a randomized sensor model leads to even more powerful detectors. Re-sampling the augmented images to the *Kinect's* depth resolution decreases detection and classification results.

Table 4. Detection recall of Faster-RCNN trained using different augmentation methods. Numbers in percent.

Classes	*synth.*	*perlin*	*auth.*	*SyDD*
ape	59.06	71.76	67.96	**79.69**
benchvise	71.99	93.90	91.85	**96.05**
bowl	**91.64**	91.48	91.08	85.81
camera	56.54	84.60	89.84	**94.17**
can	53.51	94.15	**95.32**	91.97
cat	89.91	97.20	91.18	**97.71**
cup	73.23	84.19	81.21	**88.06**
driller	89.31	95.62	95.71	**96.72**
duck	62.04	93.78	89.63	**95.37**
eggbox	45.49	81.80	90.90	**93.77**
glue	44.02	**85.08**	83.44	82.79
holepuncher	59.18	**93.37**	81.33	92.97
iron	39.15	89.41	78.83	**89.84**
lamp	75.31	93.48	**97.96**	96.09
phone	45.78	91.31	90.27	**93.00**
overall	59.76	83.82	82.28	**85.88**

4.5. Open Problems

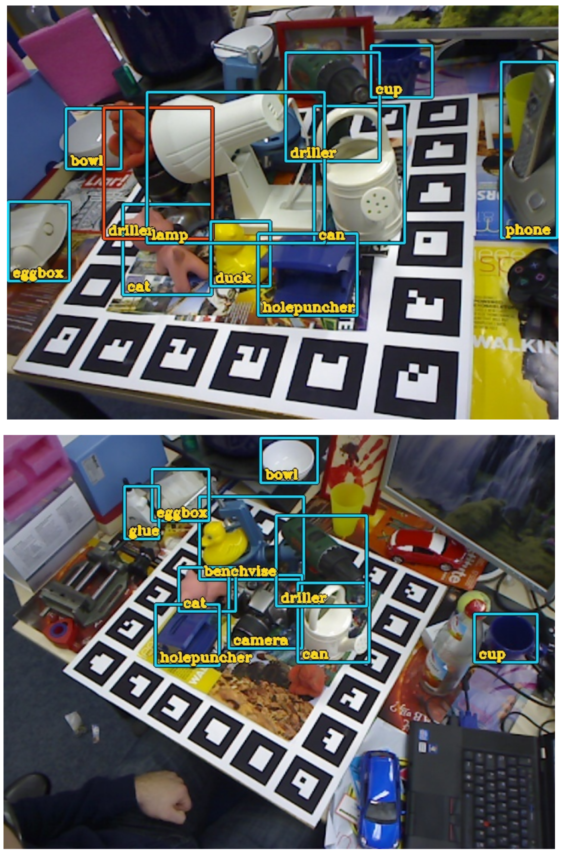


Figure 7. Detection result with incorrect detections of stacked objects.

Qualitative results of object detection using training images from *SyDD* and test images from LineMOD are presented in Figure 7 and Figure 8.

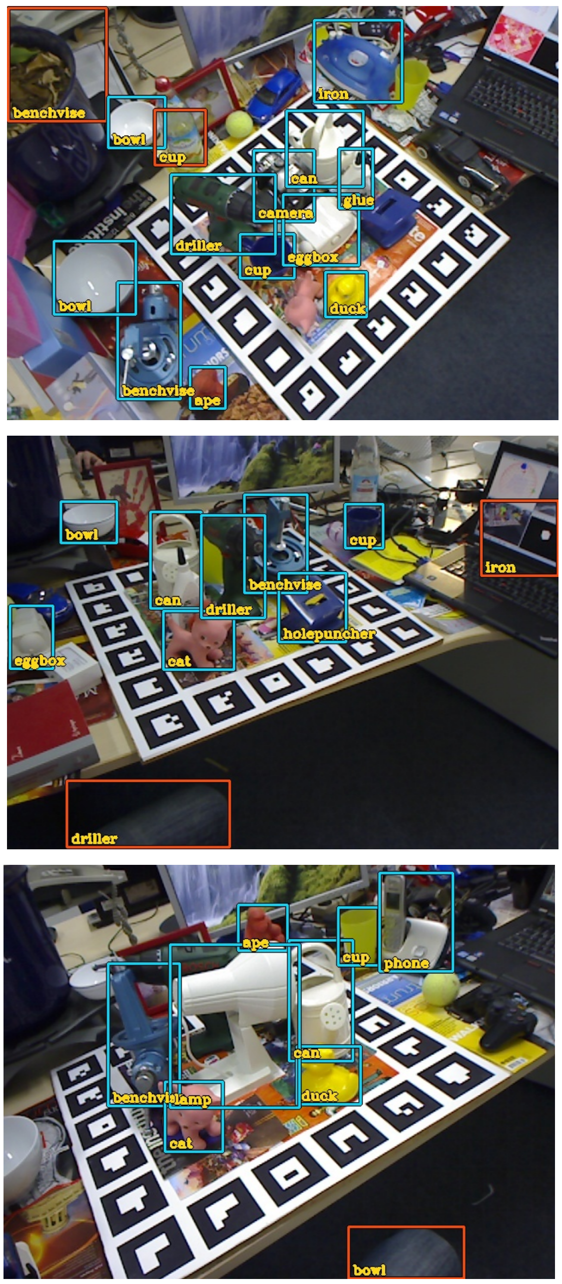


Figure 8. Detection result with incorrect detections on boundary regions of the image or fabric.

The RGB-images are only used for visualization. The top of Figure 7 shows an *ape* placed on top of a *camera*, which is incorrectly classified as *driller*. A similar error is visible in the bottom of Figure 7. The *benchvise* is correctly classified but the *duck* is not detected. This error arises because objects in the virtual scene are enclosed by a convex hull. Consequently, perfectly stacked objects can not be found in the synthetic images. A convex hull is used to represent the collision shape of objects to minimize errors when performing the physics simulation.

Figure 8 shows detection results with objects incorrectly detected on fabric, near the image boundary. The top image shows an incorrect detection of *benchvise* in the upper left corner of the image. Another incorrectly detected instance of *iron* is visible in the middle image on the right edge. These detections result from annotating only partly visible objects that are cropped by the image boundary during training. Another common error is the detection of objects on smoothly curved fabric surfaces as can be seen in the bottom parts of the middle and the bottom image in Figure 8. This error is a combination of annotating boundary regions in the synthetic images and missing background information during the rendering process.

5. Conclusion

We present a pipeline to create and augment synthetic depth data to close the reality gap for object detection. Our experiments demonstrate that deep networks trained using our data outperform detectors trained on available real-world, hand-annotated data. This is promising because we can significantly reduce the time and effort to generate training data for real-world deployment of modern computer vision algorithms.

Our method efficiently closes the domain gap on the LineMOD dataset and hence completely alleviates the need to use real-world training data. The main drawbacks of our method can be easily overcome by fine tuning the rendering script to the desired task, but would compromise the generalization of the approach. We show that the usage of domain-specific objects creates discriminating background information for object detectors trained using synthetic data. Additionally we show that it is preferable to use an augmentation loosely based on a sensor rather than using an authentic sensor model.

Our domain randomization approach omits certain aspects of depth image variations since they are not relevant for the challenges at hand. Future work will address the task of generalizing our domain randomization to other sensors.

Acknowledgements

This work has been supported by the Austrian Research Promotion Agency in the program "Production of the Future" funded project MMAssist_II (FFG No.: 858623) and the Austrian Ministry for Transport, Innovation and Technology (bmvit).

References

[1] F. M. Carlucci, P. Russo, and B. Caputo. A deep representation for depth images from synthetic data. In *Robotics and Automation, IEEE International Conference on*, pages 1362–1369. IEEE, 2017. 1, 2

[2] A. Handa, V. Patraucean, V. Badrinarayanan, S. Stent, and R. Cipolla. Understanding real world indoor scenes with synthetic data. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4077–4085, 2016. 1, 2

[3] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 5

[4] S. Hinrichsen, D. Riediger, and A. Unrau. Assistance systems in manual assembly. In *Proceedings 6th International Conference on Production Engineering and Management*, pages 3–13, 2016. 1

[5] S. Hinterstoisser, V. Lepetit, S. Ilic, S. Holzer, G. Bradski, K. Konolige, and N. Navab. Model based training, detection and pose estimation of texture-less 3d objects in heavily cluttered scenes. In *Asian conference on computer vision*, pages 548–562. Springer, 2012. 2, 4

[6] T. Hodaň, P. Haluza, Š. Obdržálek, J. Matas, M. Lourakis, and X. Zabulis. T-LESS: An RGB-D dataset for 6D pose estimation of texture-less objects. *IEEE Winter Conference on Applications of Computer Vision*, 2017. 4

[7] K. Khoshelham and S. O. Elberink. Accuracy and resolution of kinect depth data for indoor mapping applications. *Sensors*, 12:1437–1454, 2012. 4

[8] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012. 1

[9] Y. Nakagawa, H. Uchiyama, H. Nagahara, and R.-I. Taniguchi. Estimating surface normals with depth image gradients for fast and accurate registration. In *3D Vision, International Conference on*, pages 640–647. IEEE, 2015. 5

[10] K. Perlin. Improving noise. In *ACM Transactions on Graphics*, volume 21, pages 681–682. ACM, 2002. 2

[11] B. Planche, Z. Wu, K. Ma, S. Sun, S. Kluckner, O. Lehmann, T. Chen, A. Hutter, S. Zakharov, H. Kosch, et al. Depthsynth: Real-time realistic synthetic data generation from cad models for 2.5 d recognition. In *3D Vision, International Conference on*, pages 1–10. IEEE, 2017. 1, 2

[12] A. Rozantsev, V. Lepetit, and P. Fua. On rendering synthetic images for training an object detector. *Computer Vision and Image Understanding*, 137:24–37, 2015. 2

[13] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115:211–252, 2015. 5

[14] F. Sadeghi and S. Levine. CAD2RL: Real single-image flight without a single real image. In *Robotics: Science and Systems*, 2017. 2

[15] H. Su, S. Maji, E. Kalogerakis, and E. Learned-Miller. Multi-view convolutional neural networks for 3d shape recognition. In *Proceedings of the IEEE international conference on computer vision*, pages 945–953, 2015. 2

[16] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *Intelligent Robots and Systems, IEEE/RSJ International Conference on*, pages 23–30. IEEE, 2017. 2

[17] S. Zakharov, B. Planche, Z. Wu, A. Hutter, H. Kosch, and S. Ilic. Keep it unreal: Bridging the realism gap for 2.5d recognition with geometry priors only. *2018 International Conference on 3D Vision*, pages 1–11, 2018. 2, 3, 4, 7

24th Computer Vision Winter Workshop
Friedrich Fraundorfer, Peter M. Roth, Fabian Schenk (eds.)
Stift Vorau, Austria, February 6–8, 2019

A Spatiotemporal Generative Adversarial Network to Generate Human Action Videos

Stefan Ainetter, Axel Pinz
Graz Univeirsity of Technology
stefan.ainetter@student.tugraz.at,
axel.pinz@tugraz.at

Abstract. *We propose a method to generate high resolution human action videos, by extending a 2D generator network to the spatiotemporal domain. Our generative model consists of a fully convolutional 3D generator, combined with a domain specific video classifier. By using activation maximization in the spatiotemporal domain, we are able to generate action videos at a spatial resolution of $227 \times 227px$. Our model, evaluated on the UCF-101 dataset, achieves a state-of-the-art Inception Score of 23.44. Additionally, we improve the accuracy of a video classifier using our videos for data augmentation, which proves high quality and variance of our generated videos.*

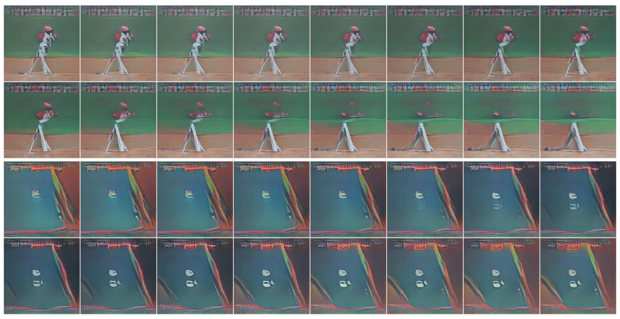


Figure 1. Visualization of generated videos using our $STGAN$. Each video consists of 16 consecutive frames. Results are shown for the UCF-101 classes **Baseball Pitch** (top) and **Billiard** (bottom).

1. Introduction

Introduced in 2014, Generative Adversarial Networks (GANs) [13] have been continuously researched due to their ability to learn perceptual representations of images in an unsupervised manner. Especially in computer vision, GANs can be beneficial for a variety of tasks like clustering, classification, and sample generation. Recent research on GANs for image generation has led to methods which can synthesize images up to a spatial resolution of 2 megapixels [4, 42]. However, video generation lags behind, with current resolutions of, e.g. 64×64 pixels [29], so that further research is needed to extend the use of GANs towards the generation of realistic looking videos.

This work addresses the generation of videos in the domain of action recognition. Training a model to generate videos of human actions will provide further insight into this domain, and therefore might improve the performance and design of current action recognition classifiers. For this purpose, we persent a spatiotemporal GAN ($STGAN$), which can generate videos in the domain of action recognition at a spatial resolution of $227 \times 227px$. Figure 1 shows two examples of our generated video clips with a duration of 16 frames.

We quantitatively evaluate our results using the Inception Score (IS) [32]. In comparison to other approaches for action video generation, we establish a new state-of-the-art both in terms of IS and spatial resolution. Furthermore, we demonstrate that it is possible to fool the Inception Score metric by generating samples that are optimized to achieve a high score, as claimed by [32]. We show how to use IS to obtain valid results, by strictly separating the video generation process and the evaluation. In addition, we use our generated videos for data augmentation, by training an action recognition classifier with both, real and generated data. This increases the accuracy of the classifier and indicates an overall high image quality and variance in our generated videos.

2. Related Work

Generative models for image generation. The idea of GANs was first proposed in [13], outperforming other generative models like autoencoders [20, 40] and Boltzmann machines [23, 31] according to image quality. Since then, GANs have been a popular field of research, and are used in different domains like image in-painting [17], image-to-image translation [24, 43] and super resolution [22]. GAN training using the loss function proposed in [13] is considered as unstable. Therefore, Dosovitskiy and Brox [7] combined the GAN loss with additional losses in image and feature space to make the training more stable. Their loss function proved to be stable during training and generated realistic looking images. Several attempts [25, 11, 26] used this loss function in combination with activation maximization [45, 9, 46, 33] to generate class specific images.

Another popular approach is the Wasserstein-GAN [2], where the loss function is modeled by the Earth-Mover or Wasserstein-1 distance. Several GANs [28, 14, 32, 3, 1] adapted this loss function to improve quality, stability and variation of the generated images.

GANs have also been used in the context of super-resolution [34, 22] to create high resolution output from low resolution input. Denton et al. [5] proposed a framework that generates images in a coarse-to-fine fashion, combining a conditional GAN model with a Laplacian pyramid representation. Karras et al. [19] were able to generate images at a resolution of $1024 \times 1024px$ using GANs and a coarse-to-fine network structure. For this approach, both the generator and discriminator grow progressively during training.

Video Generation using GANs. Video recognition and classification received a lot of attention in recent past. Therefore, generating videos via generative models is also drawing much attention. TGAN [29] generates videos for action recognition using two generators, where one generator learns to generate images, and the other one models temporal coherence. It produces frames with the spatial resolution of $64 \times 64px$. Vondrick et al. [41] proposed a model to generate videos with a spatiotemporal convolutional architecture that separates foreground and background. Tulyakov et al. [39] proposed a video generator which decomposes motion and content of videos. They use an additional recurrent neural network combined with an image generator, to model a sequence of consecutive frames. Recently, TGANv2 [30] was introduced, which uses subsampling layers in the generator and several sub-discriminators for video generation. Fuchs [11] used a similar network architecture compared to our approach to generate videos of arbitrary length, with the goal to visualize deep driving models. Koltun and Chen [4], and Wang et al. [42] are able to synthesize videos with a spatial resolution of $2048 \times 1024px$, also using progressively growing network architectures. However, they use semantic layouts instead of noise as input, which simplifies the process of generating samples.

Evaluation methods for generative models. Directly comparing the images of two different generative models, in terms of image quality, is hard to achieve. Salimans et al. [32] proposed an evaluation metric called the Inception Score (IS). To achieve a high IS, images should contain meaningful objects and the GAN should generate varied images. This metric is widely used to evaluate the quality of generated images [14] and videos [36, 29, 39, 30]. Another way of GAN evaluation is to measure the variance of the generated data. This can be achieved using structural similarity [44], and was used as evaluation metric in [27, 19]. Another popular method is using the Fréchet Inception distance [15], which is based on the Fréchet distance [8].

3. Generative Adversarial Networks for Image Generation

Our spatiotemporal GAN is based on a 2D generative model provided by [7], which is used for image generation. This section briefly summarizes the important aspects about the spatial network architecture and the loss function for image generation. Note that we did not train the 2D generator on our own, but used a pre-trained version provided by [7], before we extended the architecture to generate videos instead of images.

3.1. Spatial Network Architecture

The 2D generative model consists of a generator and a discriminator, with an additional encoder network used for feature extraction. During training, the extracted features (which are provided by the encoder) serve as input code vector $\mathbf{z}$ for the generator and the discriminator. All code vectors $\mathbf{z}$ belong to a low-dimensional input space, which we denote as

latent space.
Generator. The generator network consists of fully-connected layers, followed by convolutional and up-convolutional layers to map a code vector $\mathbf{z}$ to an image. Upconvolutional layers make it possible to up-sample the respective layer input so that the code vector $\mathbf{z}$ with dimension 1×4096 maps to an image of $256 \times 256px$. Rectified Linear Units (ReLUs) are used as nonlinear activation functions in all layers.
Discriminator. The architecture of the discriminator network consists of five convolutional layers, followed by a pooling layer to extract features of the input images. These features are concatenated with a code vector $\mathbf{z}$, which is extracted from real images by the encoder network. Dropout [37] is performed on the resulting feature vector before it is further processed by two fully-connected layers and then classified as real or fake image.
Encoder network. The network architecture of the encoder is AlexNet [21] pre-trained on ImageNet. During training, the encoder network is used as auxiliary network to extract features from real and generated images. These features are then used for several purposes: 1) During training, the generator uses features from real images as input vector, rather than randomly generated ones. This helps to identify compact regions in the latent space which correspond to natural images. 2) The discriminator gets a copy of the same features, to prevent unbalanced training. 3) The extracted features of the encoder are used to calculate the loss function.

3.2. Loss Function for Image Generation

Dosovitskiy and Brox [7] proposed a generator loss function which minimizes distances in image and feature space, additionally to the adversarial loss. This boosts the invariance with respect to irrelevant transformations, and the sensitivity to local image statistics, according to [7]. The composition of these three losses leads to better results in image quality, because the additional feature loss reflects the perceptual similarity of images. The loss in image space helps to stabilize the training process, as adversarial training is known to be unstable and sensitive to hyperparameter selection [13].

The composite loss function $\mathcal{L}$ according to [7] is defined as

$$\mathcal{L} = \lambda_{img}\mathcal{L}_{img} + \lambda_{feat}\mathcal{L}_{feat} + \lambda_{adv}\mathcal{L}_{adv}, \qquad (1)$$

with the Euclidean loss in image space $\mathcal{L}_{img}$, the Euclidean loss in feature space $\mathcal{L}_{feat}$, and the adversarial loss $\mathcal{L}_{adv}$. All three parts are weighted with a specific parameter λ.
The loss in image space $\mathcal{L}_{img}$ is written as

$$\mathcal{L}_{img} = \sum_{h,w} \|G(\mathbf{z}) - \mathbf{x}\|_2^2, \qquad (2)$$

where $G(\mathbf{z})$ and $\mathbf{x}$ are the generated and real images, respectively, with the dimensionality $[H \times W \times 3]$, where H and W define the spatial resolution of the RGB image. The indices h, w are therefore the spatial indices of the images.
The feature loss $\mathcal{L}_{feat}$ is defined as

$$\mathcal{L}_{feat} = \sum_{h,w} \|E(G(\mathbf{z})) - E(\mathbf{x})\|_2^2, \qquad (3)$$

using the encoder network E to extract features form real and generated images. The adversarial loss $\mathcal{L}_{adv}$ is defined as

$$\mathcal{L}_{adv} = -\log D(G(\mathbf{z})), \qquad (4)$$

where $D(\cdot)$ defines the discriminator function.

Furthermore, it is also necessary to optimize the discriminator, to successfully classify real and fake images. The generator and discriminator are trained concurrently, using

$$\mathcal{L}_{discr} = -[\log(D(\mathbf{x})) + \log(1 - D(G(\mathbf{z})))], \qquad (5)$$

as the loss function for the discriminator.

4. Spatiotemporal Network Structure and Parameter Transfer

We extended the network structures described in Subsection 3.1 to generate videos instead of images. The main idea is to convert the filters from spatial to spatiotemporal kernels using the approach proposed in [10]. Then, the parameters of the pre-trained networks provided by [7] are transferred to the spatiotemporal domain. The new spatiotemporal weights $\mathbf{w}_{3D}$ are initialized layerwise, by following these steps:

1. The temporal weights for each layer are defined as $\mathbf{w}_{2D}$ with the dimensions $[W \times H \times C]$, where W and H define the spatial filter size, and C defines the number of channels. In the first step, the dimensions of the spatiotemporal weights $\mathbf{w}_{3D}$ are defined as $[W \times H \times T' \times C]$, where T' describes the temporal filter depth.

Layer	Kernel	Stride	Output Size
G_fc8	/	/	16, 4096
G_fc7	/	/	16, 4096
G_fc6	/	/	16, 4096
reshape layer	/	/	4x4x16, 256
G_upconv5	4x4x3	2x2x1	8x8x16, 256
G_conv5	3x3x3	1x1x1	8x8x16, 512
G_upconv4	4x4x3	2x2x1	16x16x16, 256
G_conv4	3x3x3	1x1x1	16x16x16, 256
G_upconv3	4x4x3	2x2x1	32x32x16, 128
G_conv3	3x3x3	1x1x1	32x32x16, 128
G_upconv2	4x4x3	2x2x1	64x64x16, 64
G_upconv1	4x4x3	2x2x1	128x128x16, 32
G_upconv0	4x4x3	2x2x1	256x256x16, 3

Table 1. Network structure of 3D generator. The spatiotemporal dimensions for kernel, padding and stride are shown in the order [width x height x time]. The output size is written as [width x height x time, # channels]. We applied [1 x 1 x 1] padding at all convolutional and upconvolutional layers. For video generation, the temporal duration was set to $T = 16$ frames.

2. All spatial weights $\mathbf{w}_{2D}$ are then copied to each temporal position t of the weights $\hat{\mathbf{w}}_{3D}$. This step can be written as

$$\hat{\mathbf{w}}_{3D}(t) = \mathbf{w}_{2D}, \forall t \in [1, T']. \tag{6}$$

3. The last step is to divide the spatiotemporal weights $\hat{\mathbf{w}}_{3D}$ by the temporal filter depth T':

$$\mathbf{w}_{3D} = \frac{\hat{\mathbf{w}}_{3D}}{T'}. \tag{7}$$

   This step is used to average the weights across time, and therefore serves as temporal smoothing.

We decided to use a temporal filter depth of $T' = 3$ for all convolutional and upconvolutional layers. All biases and weights from the fully-connected layers are directly copied from the 2D to the 3D architecture.

The temporal stride is kept dense throughout all network layers, to ensure that the full duration of the sequence is preserved through the whole network. Table 1 shows a detailed overview of our 3D generator architecture after parameter transfer.

4.1. Generating Videos using Activation Maximization

To generate videos for a specific action, we use our spatiotemporal generator with Activation Maximization (AM) [9, 46, 33] to maximize the probability that a generated video belongs to a specific class. For this purpose, we use a task specific condition network Φ. This condition network is a pre-trained classifier, and delivers output probabilities for each class in the dataset. The goal of this technique is to find a specific code vector that maximizes the activation of a neuron h. Formally, the objective function can be written as

$$\hat{\mathbf{z}} = \arg\max_{\mathbf{z}}(\Phi_h(G(\mathbf{z})) - \lambda|\mathbf{z}|), \tag{8}$$

with the parameter λ weighing the regularization term. To perform activation maximization in combination with a generative model, we build on the PPGN framework [25], which is an extension of [26]. The experimental setup using our generator in combination with a condition network is shown in Figure 2. In the forward path, our 3D generator produces videos given a random code vector $\mathbf{z}$. These generated videos serve as input for the condition network, which calculates the softmax probability that the video belongs to a specific UCF-101 class. As we want to maximize the activation for a specific output neuron h, we can back-propagate the gradient $\frac{\partial \log \Phi_h(G(\mathbf{z}))}{\partial \mathbf{z}}$ through the condition network and the 3D generator, to update the code vector $\mathbf{z}$. Note that Gaussian noise $\mathcal{N}(0,\ 10^{-17})$ is added to the gradient before it is applied, to boost the variance of the generated videos.
We perform up to 100 iterations of AM, where the learning rate for the update function starts at 1.0 and linearly decays to 0.1. Other AM specific parameters are taken from [25]. Figure 3 shows the iterative process of AM, transforming random code vectors to a specific action video.
The generator serves as a learned natural image prior, which makes it possible to synthesize interpretable videos. Without using the generator as prior, it would not be possible to generate realistic looking samples, because the set of all possible outputs is too vast.

4.2. Condition Networks

A condition network is a trained classifier, which is used to perform activation maximization. Therefore, the generator and the condition network should

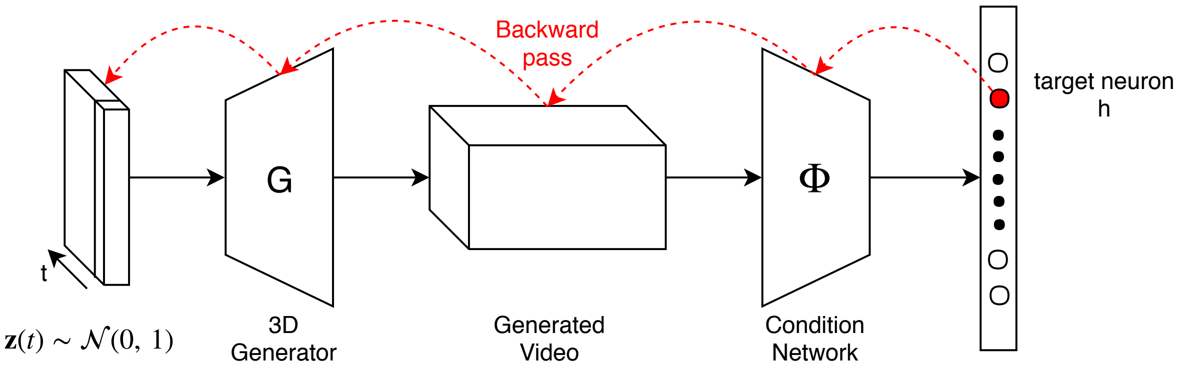


Figure 2. Video generation using activation maximization. The 3D generator G uses a random code vector $\mathbf{z}(t)$, which consists of multiple 1×4096 latent vectors, where each vector corresponds to one frame of the generate video. This video serves as input for a pre-trained condition network Φ. The goal is to maximize the activation for a specific target neuron h, and to back-propagate the gradient through both networks to adjust the code vector $\mathbf{z}(t)$.

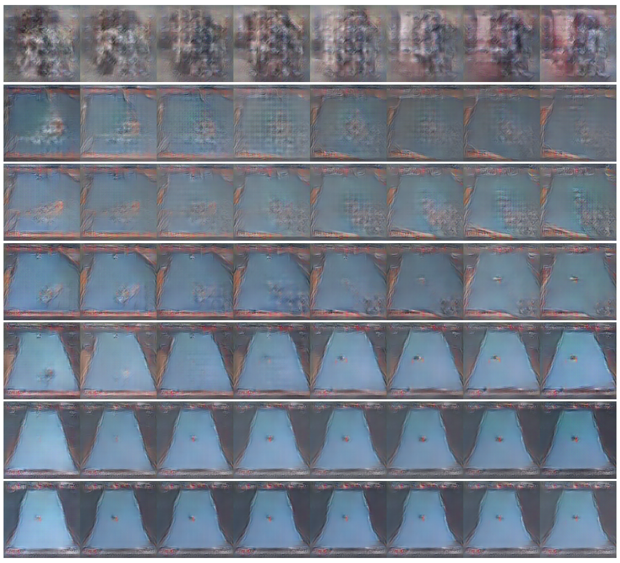


Figure 3. Visualization of activation maximization for class **Billiard**. Top to bottom shows the process of activation maximization applied to 8 frames at different iteration steps. The sequence of images changes from random patterns at top to billiard at bottom. The videos were generated with our spatiotemporal GAN and the LRCN condition network.

operate in a similar domain, using similar datasets for training. As we want to generate videos for action recognition we were looking for classifier networks trained on datasets like UCF-101 [35] and Sports-1M [18]. Therefore, we used LRCN [6] and C3D [38] as condition networks in our experiments. Note that the spatial resolution of our generated videos directly depends on the spatial input dimension of the condition network. Although our 3D generator would be able to generate output videos with $256 \times 256px$, our generated videos after AM are restricted to $227 \times 227px$, which is the spatial resolution for the input of LRCN (which we used in our main experiments).

5. Evaluation

The evaluation of our $STGAN$ is divided into three different parts:

1. Qualitative evaluation of the generated videos with analysis of findings and potential failure cases.
2. Quantitative evaluation using the Inception Score, and comparison to state-of-the-art approaches.
3. Using our generated videos for data augmentation, to increase the performance of an action recognition classifier.

5.1. Qualitative Evaluation

We performed AM for each class in the UCF-101 dataset, where the input for our generator is a sequence of random code vectors, each of them drawn from a Gaussian distribution $\mathcal{N}(0,\,1)$. We decided to sample videos with a length of 16 frames in all main experiments, to be consistent in our evaluations. Figure 4 shows generated image sequences for several classes, using our $STGAN$.
General findings, according to the quality of generated samples are:
1) One can see that our $STGAN$ does have problems to generate faces correctly, shown in the example for the class **Apply Lipstick** (Figure 4 top). This is because our $STGAN$ is not able to learn the correct number of face parts, like eyes and nose. Similar

problems occur for all classes which focus on human faces. The reason for this problem could be the usage of max-pooling at some stage of the convolutional neural network, which makes the representation invariant to small translations of the input. This means, that the network only focuses on a feature being absent or present, but not on how many times it occurs [12].

2) Videos generated for the class **Billiard** (Figure 4 middle) show results with high image quality. This means, that the quality for each individual frame is high, and the whole video shows a reasonable sequence of consecutive frames which represent a specific action. According to temporal coherence, it seems that the combination of weight transfer to 3D and a condition network pre-trained on video action recognition are enough to generate meaningful videos. Our $STGAN$, which was never trained on video data, seems to inherit the information of temporal coherence from the condition network (which was trained on action videos) during AM.

3) Because we use AM, the generated videos indicate what is most important for the condition network to differentiate between different classes. For the class **Clean and Jerk** (Figure 4 bottom), the network clearly focuses on the weights. It seems that the human body is not important for classifying this action, and therefore the frames contain no human body. This insight could possibly be used to further understand how convolutional neural networks work, and could therefore help to improve the quality of action recognition classifiers.

5.2. Quantitative Evaluation

We quantitatively evaluated our $STGAN$ using the Inception Score (IS). We compare our results to other state-of-the-art GANs, which also operate in the domain of video generation for action recognition.

Evaluation procedure. In order to compute the IS, we generated 10000 videos, which were randomly sampled from a uniform distribution over all UCF-101 classes. Afterwards, we used the C3D action recognition classifier provided by [16] to calculate the IS. The resolution and the number of frames for a C3D input video are $112 \times 112px$ and 16, respectively. Therefore, we resized our generated samples to match the spatial dimension. Additionally, we calculate the IS for the whole UCF-101 dataset. This serves as a key score indicating the IS for real videos. Table 2 shows the IS of our approaches compared to other state-of-the-art GANs for video generation. Note that the evaluation procedure of all approaches in Table 2 is the same, which enables us to directly compare the methods with each other. The IS of the other approaches is provided by the authors in their papers.

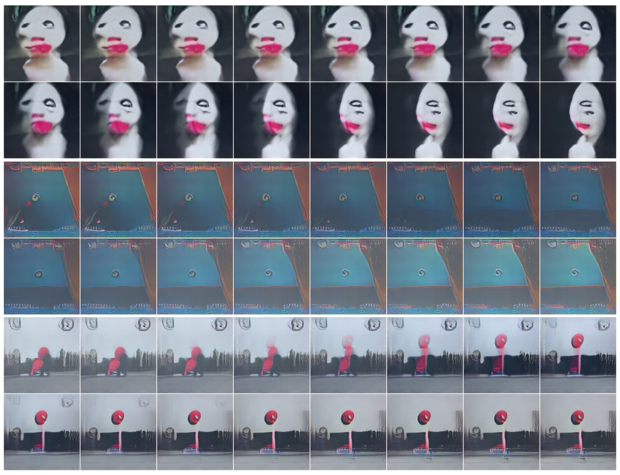


Figure 4. Visualization of generated videos using $STGAN$ in combination with LRCN as condition network. Each video consists of 16 consecutive frames. Results are shown for the classes **Apply Lipstick**, **Billiard** and **Clean and Jerk** (top to bottom).

Method	**Inception Score**
MoCoGAN [39]	12.42
Conditional TGAN [29]	15.83
TGANv2 ($bs = 64$) [30][1]	22.70
TGANv2 ($bs = 256$) [30][1]	24.34
STGAN	**23.44**
STGAN (C3D)	66.33
UCF101 dataset	70.07

Table 2. IS for different models trained on UCF-101. State-of-the-art approaches (MoCoGAN, TGAN and TGANv2) are compared to our $STGAN$. Also, the IS for the UCF-101 dataset is stated. Our IS, achieved by using different networks for conditioning and classifying, is written in **boldface**. Please note that we outperformed all published competitors [39, 29] by a large margin[1].

Our STGAN achieves state-of-the-art Inception Scores. Using LRCN as condition network and the C3D network for evaluation, we achieve an IS of 23.44, which outperforms several other approaches and is comparable to TGANv2 [30][1]. It is also worth mentioning that our $STGAN$ gener-

[1]During the preparation of our paper, we note the appearance of [30] on arXiv. This is, to our knowledge, the only work presenting results comparable to ours.

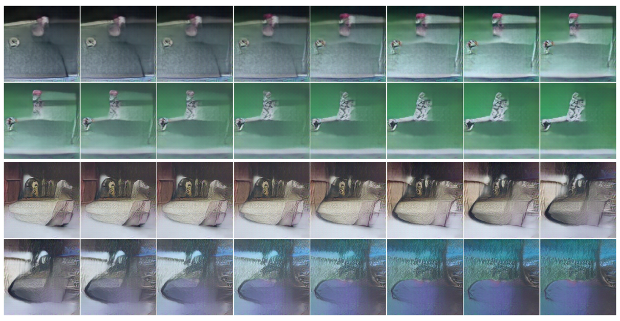


Figure 5. Visualization of generated videos using $STGAN$ and C3D as condition network. Each video consists of 16 consecutive frames. Results are shown for the classes **Baseball Pitch** (top) and **Clean and Jerk** (bottom). Although these generated samples achieve a high IS (see Table 2), the videos to not contain any real action scenarios. Therefore, we label them as adversarial examples, which fooled the classifier used for the IS calculation.

ates videos with the highest spatial resolution, compared to the approaches in Table 2. Note that the high IS of TGANv2 comes with high computational cost. Our $STGAN$ runs on a single 12Gb GPU, whereas TGANv2 needs four and 16 GPUs (12Gb) to run their model with a batch size of 64 and 256, respectively (corresponds to the IS of TGANv2 ($bs = 64$) and TGANv2 ($bs = 256$) in Table 2). Therefore, our model has a clear advantage in terms of computational cost and memory consumption.

Fooling the Inception Score is possible. If the same classifier is used as condition network for AM **and** to calculate the IS, the IS can be fooled, because the generated samples are in fact generated to have a high probability with a certain condition network. We proved this by using the C3D network for sample generation and calculation of the IS (corresponds to $STGAN(C3D)$ in Table 2). The IS for this setup is 66.33, which would indicate samples that are comparable to real data, in terms of image quality and variance. However, Figure 5 shows examples using $STGAN(C3D)$ for video generation. By analyzing these samples, one can see that the videos do not contain any real objects, indicating that these are adversarial examples. Salimans et al. [32] stated that directly optimizing the IS would lead to adversarial examples, but did not provide results to prove this claim. Our experiments prove this claim, and show that it is easily possible to fool the IS.

5.3. Supervised Learning using Generated Videos

Additional to the previous evaluations, we used our generated videos for data augmentation, to train an action recognition classifier. We used a C3D implementation [16] as classifier, and the UCF-101 dataset for training and evaluation.

5.3.1 Dataset Preprocessing

Our $STGAN$ enables us to generate videos for all 101 classes of the UCF-101 dataset. However, not all generated samples have the same image quality, due to the variance in the generated data. Therefore, we analyzed the results of our Inception Score calculation, and chose samples for data augmentation that contributed to a high score. Furthermore, we limited the number of samples per class to 100, to prevent unbalancing the training data too much. Figure 6 shows a comparison of the UCF-101 train split 1 and our generated samples.

5.3.2 Training Schedule

We used the pre-trained C3D model, which was trained on the Sports-1M dataset as initialization for our network parameters. The pre-trained model is available online[2]. Other training parameters were taken from the original code [16]. We fine-tuned the classifier on UCF-101, using our generated videos, the UCF-101 train split 1, and a combination of both. For each experiment, the training data were randomly shuffled at the beginning of training. After 200 iterations of training, using a batch size of 10, we evaluated the performance of the classifier by calculating the random clip accuracy on the UCF-101 test split 1. This helped to determine the optimal point to stop training and avoided overfitting.

5.3.3 Classification Results

Table 3 provides the accuracy of our trained classifiers. If we use our generated samples without any real data for training, we achieve an accuracy of 15.6%. This is significantly lower than the accuracy for train split 1, which is 50.6% in our experiments. However, this experiment indicates that our generated samples provide meaningful information about the UCF-101 dataset. Using both, the real and generated data for training the classifier, we achieve an accuracy of 52.2%. This improvement shows that there is some additional information in the augmented data, although the generated data do not have

[2]https://github.com/hx173149/C3D-tensorflow, last visited: Dec. 2018

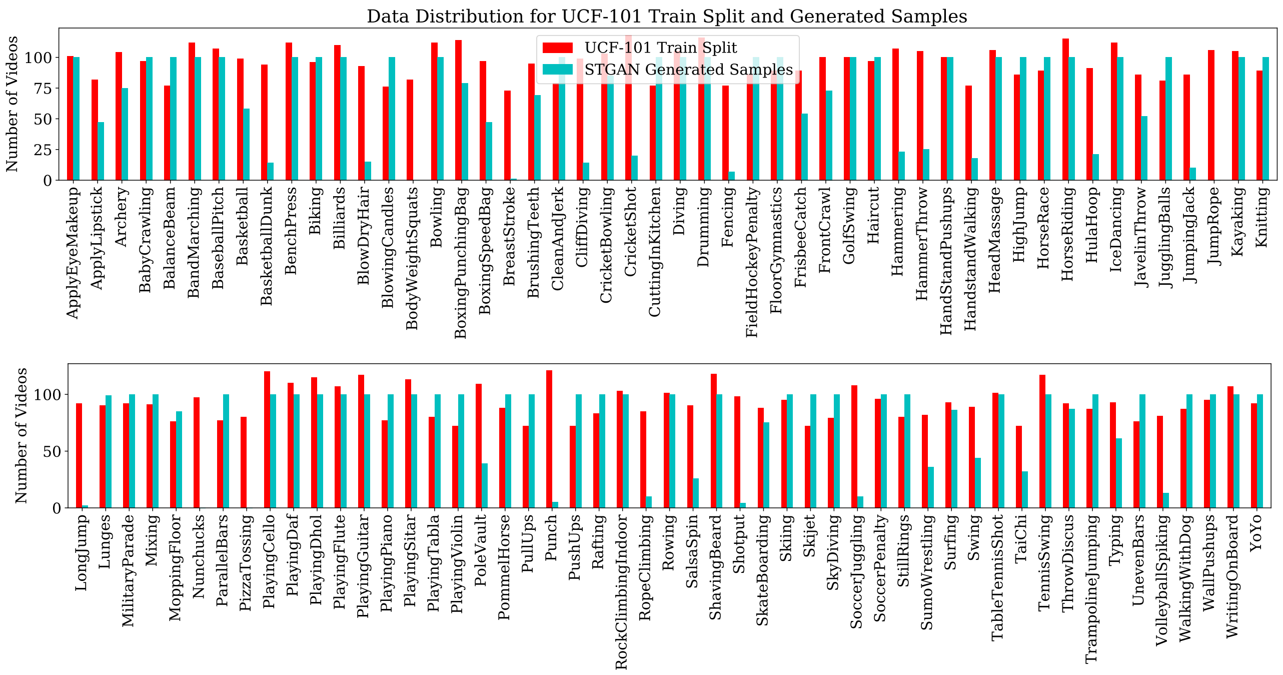


Figure 6. Comparison of data distribution between UCF-101 train split 1 and our generated samples. The generated samples for data augmentation were chosen according to the IS, and the number of samples per class is restricted to at most 100 samples. The data distribution shows that our $STGAN$ is not able to produce high quality samples for certain classes e.g. JumpRope, Nunchucks, Pizza Tossing.

Training Procedure	**Accuracy**
Generated Samples Only	15,6%
Train Split 1	50,6%
Train Split 1 + Generated Samples	**52,2%**

Table 3. Accuracy of C3D action recognition classifier. The accuracy was calculated for the UCF-101 test split 1, where we randomly extracted 16 consecutive frames from each clip for testing. Training with generated samples leads to a lower score, compared to training with real data. However, training with both, real and generated data leads to an increase of accuracy, which indicates that our generated samples contain valuable additional information for action recognition.

the same image quality as the real ones. Note that the goal of this experiment was to show the practical benefit of our $STGAN$, rather than competing for a state-of-the-art accuracy for UCF-101. This means, that another network architecture, and a more careful selection of hyperparameters, combined with an optimized training schedule, would certainly lead to a higher overall accuracy for all experiments.

6. Conclusion

We have presented a method to generate human action videos for 101 action classes at a spatial resolution of $227 \times 227px$, by extending a generative model proposed for image generation to the spatiotemporal domain. A condition network performs activation maximization, where our generator is used as image prior. In terms of image quality measured by the Inception Score, our approach outperforms all published state-of-the-art methods by a large margin, and also improves with respect to spatial resolution of the generated videos.

Our evaluations also demonstrate a major weakness of the Inception Score metric: We were able to generate adversarial examples, which contained no meaningful objects, but achieved an exceptionally high Inception Score. This means that (a) adversarial examples deserve in-depth research in the future, and (b) Inception Score should be reconsidered.

Action videos generated by our approach can readily be used for data augmentation, in addition to real training data. This leads to an increased accuracy of the classifier, which underlines the high quality and variance of our videos, and highlights direct benefits towards improved recognition.

References

[1] M. Arjovsky and L. Bottou. Towards principled methods for training generative adversarial net-

works. In *International Conference on Learning Representations (ICLR)*, 2017. 2

[2] M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein generative adversarial networks. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, pages 214–223, 2017. 2

[3] D. Berthelot, T. Schumm, and L. Metz. Began: boundary equilibrium generative adversarial networks. *arXiv preprint arXiv:1703.10717*, 2017. 2

[4] Q. Chen and V. Koltun. Photographic image synthesis with cascaded refinement networks. In *The IEEE International Conference on Computer Vision (ICCV)*, pages 1520–1529, 2017. 1, 2

[5] E. L. Denton, S. Chintala, R. Fergus, et al. Deep generative image models using a laplacian pyramid of adversarial networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1486–1494, 2015. 2

[6] J. Donahue, L. Anne Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2625–2634, 2015. 5

[7] A. Dosovitskiy and T. Brox. Generating images with perceptual similarity metrics based on deep networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 658–666, 2016. 2, 3

[8] D. Dowson and B. Landau. The fréchet distance between multivariate normal distributions. *Journal of Multivariate Analysis*, 12(3):450–455, 1982. 2

[9] D. Erhan, Y. Bengio, A. Courville, and P. Vincent. Visualizing higher-layer features of a deep network. Technical report, University of Montreal, 2009. 2, 4

[10] C. Feichtenhofer, A. Pinz, and R. Wildes. Spatiotemporal residual networks for video action recognition. In *Advances in Neural Information Processing Systems (NIPS)*, pages 3468–3476, 2016. 3

[11] H. Fuchs. Visualizing and Understanding Deep Driving Models. Masters Thesis at Institute of Electrical Measurement and Measurement Signal Processing, Graz University of Technology, 2017. 2

[12] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016. 6

[13] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2672–2680, 2014. 1, 2, 3

[14] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville. Improved training of wasserstein gans. In *Advances in Neural Information Processing Systems (NIPS)*, pages 5767–5777, 2017. 2

[15] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in Neural Information Processing Systems (NIPS)*, pages 6626–6637, 2017. 2

[16] HouXin. C3D-tensorflow. `https://github.com/hx173149/C3D-tensorflow`, 2018. last visited: Sep 2018. 6, 7

[17] S. Iizuka, E. Simo-Serra, and H. Ishikawa. Globally and Locally Consistent Image Completion. *ACM Transactions on Graphics (TOG)*, 36(4):107:1–107:14, 2017. 2

[18] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei. Large-scale video classification with convolutional neural networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1725–1732, 2014. 5

[19] T. Karras, T. Aila, S. Laine, and J. Lehtinen. Progressive growing of GANs for improved quality, stability, and variation. In *International Conference on Learning Representations (ICLR)*, 2018. 2

[20] D. P. Kingma and M. Welling. Auto-encoding variational bayes. In *International Conference on Learning Representations (ICLR)*, 2014. 2

[21] A. Krizhevsky, I. Sutskever, and G. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1097–1105, 2012. 3

[22] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. P. Aitken, A. Tejani, J. Totz, Z. Wang, and W. Shi. Photo-realistic single image super-resolution using a generative adversarial network. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 105–114, 2017. 2

[23] H. Lee, R. Grosse, R. Ranganath, and A. Y. Ng. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *Proceedings of the 26th International Conference on Machine Learning (ICML)*, pages 609–616, 2009. 2

[24] M.-Y. Liu, T. Breuel, and J. Kautz. Unsupervised image-to-image translation networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 700–708, 2017. 2

[25] A. Nguyen, J. Clune, Y. Bengio, A. Dosovitskiy, and J. Yosinski. Plug & play generative networks: Conditional iterative generation of images in latent space. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3510–3520, 2017. 2, 4

[26] A. Nguyen, A. Dosovitskiy, J. Yosinski, T. Brox, and J. Clune. Synthesizing the preferred inputs for neurons in neural networks via deep generator networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 3387–3395, 2016. 2, 4

[27] A. Odena, C. Olah, and J. Shlens. Conditional image synthesis with auxiliary classifier gans. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, pages 2642–2651, 2017. 2

[28] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. In *International Conference on Learning Representations (ICLR)*, 2016. 2

[29] M. Saito, E. Matsumoto, and S. Saito. Temporal generative adversarial nets with singular value clipping. In *The IEEE International Conference on Computer Vision (ICCV)*, pages 2849–2858, 2017. 1, 2, 6

[30] M. Saito and S. Saito. TGANv2: Efficient training of large models for video generation with multiple subsampling layers. *arXiv preprint arXiv:1811.09245*, 2018. 2, 6

[31] R. Salakhutdinov and H. Larochelle. Efficient learning of deep boltzmann machines. In *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 693–700, 2010. 2

[32] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. Improved techniques for training gans. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2234–2242, 2016. 1, 2, 7

[33] K. Simonyan, A. Vedaldi, and A. Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *International Conference on Learning Representations (ICLR)*, 2014. 2, 4

[34] C. K. Soenderby, J. Caballero, L. Theis, W. Shi, and F. Huszár. Amortised map inference for image super-resolution. In *International Conference on Learning Representations (ICLR)*, 2017. 2

[35] K. Soomro, A. R. Zamir, and M. Shah. Ucf101: A dataset of 101 human actions classes from videos in the wild. Technical report, Center for Research in Computer Vision (CRCV), November 2012. 5

[36] C. Spampinato, S. Palazzo, P. D'Oro, F. Murabito, D. Giordano, and M. Shah. Vos-gan: Adversarial learning of visual-temporal dynamics for unsupervised dense prediction in videos. *arXiv preprint arXiv:1803.09092*, 2018. 2

[37] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research (JMLR)*, 15:1929–1958, 2014. 3

[38] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri. Learning spatiotemporal features with 3d convolutional networks. In *The IEEE International Conference on Computer Vision (ICCV)*, pages 4489–4497, 2015. 5

[39] S. Tulyakov, M.-Y. Liu, X. Yang, and J. Kautz. Mocogan: Decomposing motion and content for video generation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 2, 6

[40] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th International Conference on Machine Learning (ICML)*, pages 1096–1103, 2008. 2

[41] C. Vondrick, H. Pirsiavash, and A. Torralba. Generating videos with scene dynamics. In *Advances in Neural Information Processing Systems (NIPS)*, pages 613–621, 2016. 2

[42] T.-C. Wang, M.-Y. Liu, J.-Y. Zhu, G. Liu, A. Tao, J. Kautz, and B. Catanzaro. Video-to-video synthesis. In *Advances in Neural Information Processing Systems (NIPS)*, 2018. 1, 2

[43] T.-C. Wang, M.-Y. Liu, J.-Y. Zhu, A. Tao, J. Kautz, and B. Catanzaro. High-resolution image synthesis and semantic manipulation with conditional gans. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 2

[44] Z. Wang, E. P. Simoncelli, and A. C. Bovik. Multiscale structural similarity for image quality assessment. In *Asilomar Conference on Signals, Systems & Computers*, volume 2, pages 1398–1402, 2003. 2

[45] J. Yosinski, J. Clune, A. Nguyen, T. Fuchs, and H. Lipson. Understanding neural networks through deep visualization. In *Deep Learning Workshop, International Conference on Machine Learning (ICML)*, 2015. 2

[46] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *European Conference on Computer Vision (ECCV)*, pages 818–833, 2014. 2, 4

24th *Computer Vision Winter Workshop*
Friedrich Fraundorfer, Peter M. Roth, Fabian Schenk (eds.)
Stift Vorau, Austria, February 6–8, 2019

Perspective transformation for accurate detection of 3D bounding boxes of vehicles in traffic surveillance

Viktor Kocur
Faculty of Mathematics, Physics and Informatics, Comenius University in Bratislava
Mlynská Dolina, Bratislava
viktor.kocur@fmph.uniba.sk

Abstract. *Detection and tracking of vehicles captured by traffic surveillance cameras is a key component of intelligent traffic systems. In this paper a novel method of detecting 3D bounding boxes of vehicles is presented. Using the known geometry of the surveilled scene, we propose an algorithm to construct a perspective transformation. The transformation enables us to simplify the problem of detecting 3D bounding boxes to detecting 2D bounding boxes with one additional parameter. We can therefore utilize modified 2D object detectors based on deep convolutional networks to detect 3D bounding boxes of vehicles. Known 3D bounding boxes of vehicles can be utilized to improve results on tasks such as fine-grained vehicle classification or vehicle re-identification. We test the accuracy of our detector by comparing the accuracy of speed measurement on the BrnoCompSpeed dataset with the existing state of the art method. Our method decreases the mean error in speed measurement by 22 % (1.10 km/h to 0.86 km/h) and the median error in speed measurement by 33 % (0.97 km/h to 0.65 km/h mean), while also increasing the recall (83.3 % to 89.3 %).*

1. Introduction

Recent development in commercially available cameras has increased the quality of recorded images and decreased the costs required to install cameras in traffic surveillance scenarios. Automatic traffic surveillance aims to provide information about the surveilled vehicles such as their speed, type and dimensions and as such is an important aspect of intelligent traffic system design.

There are two crucial requirements for accurate monitoring of vehicles. Precise detection of their position and accurate camera calibration. Multiple methods of monocular camera calibration exist. However, they usually require manual measurements of some distances in the road plane. Dubská et al. [8] proposed a fully automatic method of camera calibration based on vanishing point detection. Sochor et al. [25] further improved the method. We employ this method for calibration and focus on the accuracy of vehicle detection.

Object detection is one of the key tasks in computer vision. Current state of the art object detectors rely on convolutional neural network backbones to extract feature maps from images. A structure based on so-called anchorboxes is used to determine the position of bounding boxes of detected objects. There are two types of such object detection networks: one-stage detectors and two-stage detectors. One-stage detectors such as SSD [17] or YOLO [22] have shorter inference times, but suffer from worse accuracy than two-stage detectors such as Faster R-CNN [23] and R-FCN [4]. We employ a recently published one-stage detector RetinaNet [14], which managed to bridge the accuracy gap while keeping the inference times low.

In this paper we propose a perspective image transformation, which utilizes the geometry of a common traffic surveillance scenario to rectify the image. Rectification of the image allows us to reduce the problem of detecting 3D bounding boxes of vehicles to detection of 2D bounding boxes with one additional parameter. Any object detector which utilizes anchorboxes can be easily modified to detect 3D bounding boxes of vehicles in the transformed images. We test our method on the task of speed measurement of vehicles captured by a static monocular traffic camera. Compared to existing approaches, our method achieves lower mean error of measured speeds, while being computationally simpler.

2. Related work

2.1. Camera calibration

Multiple approaches have been proposed to deal with camera calibration in a traffic surveillance scenario. A calibration pattern [7] or manual measurements on the road [19] can be used to determine the camera parameters. Some methods use line segments on the road or traffic signs to find vanishing points [2, 9]. Other methods perform calibration based on vehicle movement [5, 24].

We require the calibration method to be fully automatic and highly accurate. Dubská et al. [8] proposed a fully automatic method of camera calibration by finding three orthogonal vanishing points. The position of the first vanishing point is determined by detecting motion of the surveilled vehicles. Interesting feature points are detected and tracked with a KLT tracker. Hough transform is used to accumulate the found lines of motion in a diamond space based on parallel coordinates. To determine the second vanishing point, the edges of the vehicles which do not correspond to the first vanishing point are again accumulated. The two vanishing points are then determined from the diamond space with the third vanishing point being calculated as orthogonal to the first two.

To detect the scale of the scene, the dimensions of 3D bounding boxes of the passing vehicles are collected. The mean of the collected dimensions is compared against statistical data and thus a scale, which enables measuring distances on the road plane, is determined. This approach has been further improved by Sochor et al. [25] who used edgelets corresponding to relevant car features to detect the second vanishing point more accurately. The detection of the scale was also improved by fitting a 3D model of a common vehicle to the detections of the vehicle in the footage.

2.2. Object detection

Most cameras employed in traffic surveillance are static. Many vehicle detection approaches [3, 19] therefore utilize background substraction methods to detect the vehicles. These methods can fail with quickly changing lighting conditions or small camera movements and may result in single detections containing more than one vehicle due to occlusion. Luvizón et al. use motion detector [1] to detect license plates. Daley et al. [5] find edges in the inter-frame differences [29]. Pham and Lee [28] propose a method based on finding the windshields of vehicles. Zhou et al. [31] propose two deep neural networks to propose and verify the detections.

We opt to base our method on recent object detectors based on deep convolutional neural networks, as these, given proper training data, can handle different light conditions and occlusion. These can be divided in two categories: one-stage and two-stage detectors. Two stage methods (Faster R-CNN [23] and R-FCN [4]) utilize a convolutional neural network to extract features from an image. The first stage comprises of determining which regions, called anchorboxes, in the image could potentially contain foreground objects. In the second stage the features corresponding to the proposed anchorboxes are processed further to determine which object, if any, the candidate anchorbox contains. Along with this classification task, the shape and position of the proposed anchorbox is offset via regression to better fit the object. Finally, non-maximum supression is applied to remove all but one bounding box per detected object. An extension of Faster R-CNN called Mask R-CNN [10] additionally enables pixel-wise segmentation of the detected objects.

One-stage detectors differ from two stage detectors by performing classification and regression on all anchorboxes. This results in disproportionate amount of negative (i.e. background) proposals compared to the amount of positive samples (i.e. objects) and biases the training process. To remedy this, SSD [17] uses hard negative mining and YOLO [22] uses a fixed weight on loss contributions by the background anchorboxes. These approaches are faster than two-stage detectors, but have worse results on standard benchmarks [12].

RetinaNet [14] introduced focal loss to address the issue of negative sample bias and utilized a feature pyramid network to further improve its performance. RetinaNet achieves better results on COCO detection task [15] than Faster R-CNN while keeping the benefits of low inference times of the other one-stage detectors.

We compare our results with a method of object detection proposed in [25] in which a Faster R-CNN object detector is used to find 2D bounding boxes. The bounding boxes are then used to join foreground blobs obtained via background subtraction into masks of vehicles.

2.3. Object tracking

To allow for vehicle counting and speed measurement, the vehicles have to be tracked from frame to frame. Since object detection may sometimes fail a robust tracker is necessary. Kalman filter [13] has been a reliable tool to tackle the task of object tracking. Recent successes with deep neural nets led to development of object tracking methods which utilize convolutional architectures in combinations with recurrent neural networks [20, 16, 21]. For our case we found that a simple object tracker, which compares the positions of bounding boxes in subsequent frames is sufficient.

2.4. Benefits of 3D bounding boxes

The task of detecting 2D bounding boxes of vehicles is in general easier than detecting 3D bounding boxes. However 3D bounding boxes have been shown to be valuable in improving various traffic surveillance algorithms such as fine-grained vehicle classification [27] and vehicle re-identification [30].

2.5. Datasets

Sochor et al. [26] performed a survey of existing traffic surveillance datasets. Most publicly available datasets suffer from too few recorded vehicles and inaccurate measurements of ground truth speeds.

Authors of the survey published their own BrnoCompSpeed dataset. The dataset contains videos from 7 locations. Each location is recorded for approximately one hour from three different viewpoints on an overpass above the recorded roads. The dataset contains ground truth data of over 20 thousand vehicle speeds obtained via LIDAR gate measurements. We train and evaluate our method on this dataset.

After the publication of the survey Luvizón et al. [18] published a dataset comprising of 5 hours of video of smaller sections of roads. Speed measurements of cars using inductive loops and labeled license plate numbers are provided as ground truth.

3. Proposed method

The goal of our method is to detect 3D bounding boxes of cars recorded with a monocular camera installed above the road plane.

3.1. Image transformation

For our method we assume that the camera has been calibrated as per [25]. This calibration method has very few limitations regarding the camera position. The camera has to be positioned above the road plane and the observed road segment has to be straight. The main parameters obtained by the calibration are the positions of the two relevant vanishing points in the image. Assuming that the principal point is in the center of the image, the position of the third vanishing point as well as focal length of the camera can be calculated. This enables us to project any point in the image onto the road plane. To enable measurements of distances on the road plane one additional parameter, denoted as scale, is determined during calibration.

The first detected vanishing point (denoted further as *VP1*) corresponds to the lines on the road plane which are parallel to the direction of the moving vehicles. The second detected vanishing point (*VP2*) corresponds to the lines which lie on the road plane but are perpendicular to the the direction of the moving vehicles. The third vanishing point (*VP3*) corresponds to the lines which are perpendicular to the road plane.

The goal of our transformation is to produce an image in which all lines corresponding to *VP2* will be parallel to the x-axis of the transformed image and the lines corresponding to *VP3* will be parallel to the y axis. Thus in the transformed image plane both *VP2* and *VP3* will be ideal points. Additionally, we require the lines corresponding to *VP3* to be preserved and thus we will use perspective transformation. As a result of such transformation the objects which are aligned with the three vanishing points (most importantly the vehicles on the road) will be rectified in the transformed image.

In most cases this transformation can be performed in a way which preserves the whole captured scene in the resulting image. To find the perspective transformation for such cases we employ an algorithm (see Figure 1 for visual reference):

- In the original image plane, for both *VP2* and *VP3*, construct two lines which originate in the vanishing point and are tangent to the viewing window.
- For both lines originating in *VP2* find the intersections with both of the lines originating in *VP3*. This yields four points.
- Determine which of these four intersection points correspond to which corner points of the viewing window.

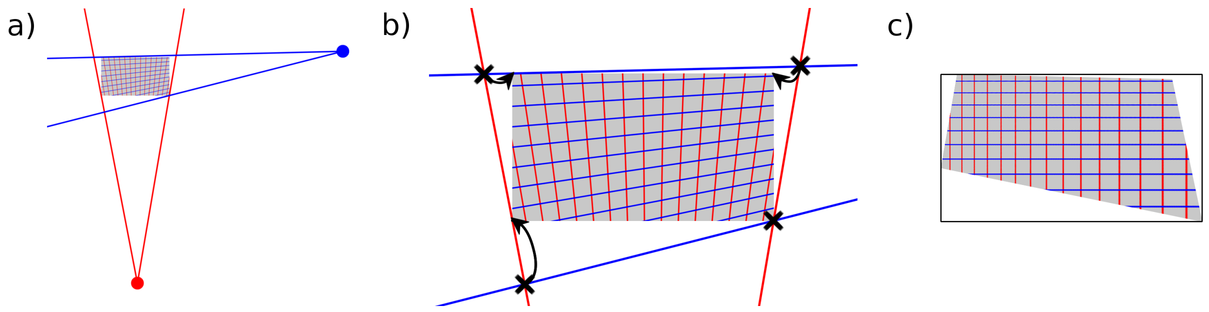


Figure 1. a) Lines starting in *VP2* (blue) and *VP3* (red), which are tangent to the viewing window (gray) are detected. b) Intersections of the lines are found. Points of intersection are paired with the corners of the viewing window. The four pairs are used to find the perspective transform. c) Perspective transform is applied.

- Find a perspective transformation which maps the intersection points to their respective corner points.

This is possible in most cases, however in a case in which the line connecting the two vanishing points intersects the viewing window the algorithm fails. This is expected as in such a case the points on the intersecting line correspond to both vanishing points and thus in the transformed image they would have to form a line which is perpendicular to itself. For this to occur the camera has to be positioned not right above the captured road, but on its side. Usually such situations could be avoided by considering the position in which to install the cameras, for instance in the BrnoCompSpeed dataset there is no such scene where this is the case. Nevertheless it is always possible to crop the viewing window to obtain the transformation.

This strategy can be employed even when the algorithm doesn't fail, but the resulting image is too distorted. In this paper we use this strategy heuristically only for one outlier video and thus we do not specify a rule for its use. Note that the viewing window is reduced only for the calculation of the transformation properties. The pixels which do not fit into the new viewing window can still appear in the transformed image.

3.2. Bounding boxes in the transformed image

The 3D bounding boxes we aim to detect are aligned with the vanishing points as in [25]. In their work Sochor et al. first segment the vehicle and then construct the bounding box around the mask. The construction in this manner is problematic, as the resulting bounding box depends on the order in which the vertices of the bounding box are constructed.

The perspective transformation described above enables us to reduce the problem of finding the 3D bounding box to finding a 2D bounding box with one additional parameter. This is possible, since the transformation rectifies the image in a manner in which all the lines corresponding to *VP2* and *VP3* are parallel to either of the image axes. The remaining parameter denoted as c_c is determined by the relative position of the top frontal edge of the 3D bounding box against the 2D bounding box which encloses the whole 3D bounding box. The construction of the 2D bounding box can be seen in Figure 2.

Reconstructing the 3D bounding box from the 2D version can be achieved by considering the position of *VP1* in the transformed image. A point on the side of the 2D bounding box is determined by the parameter c_c and a relative position of the transformed *VP1*. If the transformed *VP1* is to the left of the bounding box, then the point is on the right side and vice versa. If the transformed *VP1* is directly above the bounding box, then the side can be chosen arbitrarily. With the position of this point known, the 3D bounding box can be constructed in the transformed image. To obtain the 3D bounding box in the original image, the positions of the vertices of the 3D bounding box are transformed to the original image space via inverse perspective transform.

3.3. Bounding box detection

As shown in the previous subsection we only need to detect 2D bounding boxes with the parameter c_c. For this purpose we utilize the RetinaNet object detector [14]. This detector outputs 2D bounding boxes for the detected objects. We modify the method to add c_c to each of the output boxes.

The RetinaNet, as well as other object detecting meta-architectures, use anchorboxes as default positions of bounding boxes to determine where the objects are. The object detection task is separated into

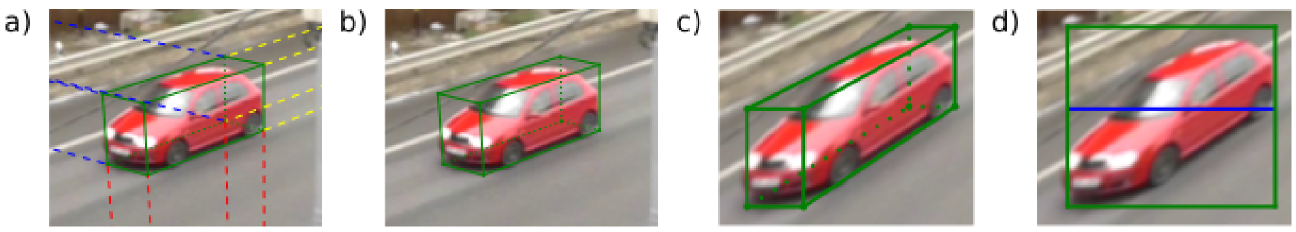


Figure 2. a) 3D bounding box (green) which is aligned with VP1 (yellow), VP2 (blue) and VP3 (red). b) 3D bounding box. c) 3D bouding box after the perspective transform is applied. d) The parametrization of the 3D bouding box as 2D bouding box (green). The parameter c_c is determined as the ratio of the distance from top of the 2D bounding box to the top-front edge of the transformed 3D bounding box (blue) and the height of the 2D bouding box.

three parts: determining which anchorboxes contain which objects, resizing and moving the anchorboxes to better fit the objects and finally performing non-maximum suppression to avoid multiple detections of the same object. To train the network a two-part loss (1) is used.

$$L_{tot} = \frac{1}{N}\left(L_{conf} + L_{loc}\right) \tag{1}$$

The loss is averaged over all N anchorboxes, L_{conf} is the Focal loss used to train a classifier to determine which objects, if any, are in the bounding box. L_{loc} is the regression loss to train the network how to reshape and offset the anchorboxes. To include the parameter c_c we add another regression loss (2).

$$L_c = \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} x_{i,j} s_{L1}\left(c^p_{c,i} - c^g_{c,j}\right) \tag{2}$$

In the loss we sum over all of the N anchorboxes and M ground truth bounding boxes. $x_{i,j}$ determines whether the i-th anchorbox corresponds to the j-th ground truth label [17]. We subtract the ground truth value of c_c denoted as $c^g_{c,j}$ from the predicted value $c^p_{c,i}$ and apply smooth L1 loss.

3.4. Training

To obtain training data we use data from two distinct datasets. The first dataset is BoxCars116k [27], the original purpose of this dataset is fine-grained vehicle classification. The dataset contains over 116 thousand images, each containing one car along with make and model labels, information on positions of vanishing points and the 3D bounding box of the car. We transform these images with the proposed transformation and calculate the 2D bounding boxes and c_c based on the provided 3D bounding boxes. Since each image is only of one car we augment the images by rescaling them and placing them randomly on a black background.

The other used dataset is BrnoCompSpeed [26]. We use the split C of this dataset leaving 9 videos for testing, 9 for training and 3 for validation. The original purpose of this dataset is speed measurement, therefore we test our method by measuring speed on the test set. For training and validation we pick every 25-th frame of the videos. Within the dataset a mask of the region of interest (e.g. the surveilled road) is provided, therefore we keep only the pixels from the region of interest to not confuse the network with cars outside the road, which would get transformed in undesirable ways since they may not be aligned with the vanishing points. We run these frames through Mask-RCNN [10] image segmentation network trained on the COCO dataset [15] to obtain masks of the cars. We transform the masks and the images using our transformation and create the 2D bounding boxes with and without c_c as labels for training.

We train the model on the labeled data in a standard procedure. The validation loss is used to choose the best model for inference. We employ ResNet50 [11] pre-trained on Imagenet [6] as our backbone network. The input of the network is an image with 640 by 360 pixels.

3.5. Speed measurement

We perform speed measurement on the test videos. For each video the network is used to detect the 3D bounding boxes. Each detected 3D bounding box is compared via its encompassing 2D bounding box to the tracks which have been detected in the previous frames.

For each detection the IoU (intersection over union) metric is calculated against the last 2D bounding box of each track. If IoU is higher than 0.1 for at least one track, then the bounding box is added to the track with highest IoU score. If no track has higher IoU against the detection, then a new track is created. If a track hasn't had any bounding boxes added to it in the last 5 frames, then the track is no longer considered active. To detect speed we filter out bounding boxes too close to the edges of the images. We also discard tracks which have less than 5 detected bounding boxes within them.

For the 3D bounding box the speed is determined using a point which is in the middle of the frontal bottom edge of the 3D bounding box. Since this points should under normal circumstances lie on the road plane, we can use the camera calibration to easily determine the distances between various positions within a track. To detect the average speeds of the vehicles we employ the same method as [26] by calculating median speed over the whole track.

4. Results

We compare our results with the results achieved in [25] as these are the best achieved results published in the literature, which we denote as *SochorAuto* for automatic calibration and *SochorManual* for manual calibration. Our method is denoted as *Transform3D*. Examples of the resulting bounding boxes can be seen in Figure 4.

4.1. Ablation experiments

To properly gauge the impact of the image transformation we perform two ablation experiments. We train the standard RetinaNet 2D object detector on the same data as the other models, except that the images are not transformed. We refer to this model as *Orig2D*. We also train the standard RetinaNet 2D object detector on the transformed image. We use the same 2D bounding boxes as in *Transform3D*, but without the parameter c_c. We refer to this method as *Transform2D*. We train these models with the same hyperparameters as our base model.

For our method as well as the ablation experiments we set the confidence necessary to classify a prediction as true to 0.2. This is an unusually low threshold. However in this case it is beneficial as setting the threshold too high may lead to lower recall, while also producing more false positives due to tracks being possibly split into two. Having more detections in the track even if they have lower confidence scores doesn't hurt the overall resulting speed measurement since we determine the speed as the median of interframe speeds.

4.2. Cropping the viewing window

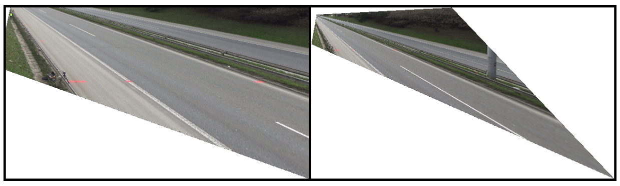


Figure 3. The transformed image when the viewing window is cropped (left) and when the original viewing window is used (right).

Method	Recall	Precision
Cropped	0.8955	0.8360
Uncropped	0.2582	0.4595

Table 1. Precision and recall achieved on *session 5 left* video for the *Transform3D* utilizing cropping of the viewing window (Cropped) and the baseline *Transform3D* (Uncropped).

In our experiments we noticed that for one of the testing videos (*session 5 left*) the recall values were significantly lower than for the rest of the videos. The reason for this was the fact that the second vanishing point position was too close to the center of the image and the resulting transformed image was too distorted for the image detector to work. To remedy this we employed the strategy described in subsection 3.1. We crop the original 1920 by 1080 pixel viewing window from the left by 100 pixels, from top by 200 pixels, and from the right by 480 pixels and use the reduced viewing window to perform the transformation. See Figure 3 for comparison of the transformed images.

In the Table 1 we show the recall and precision values for both the baseline version and the cropped version. Recall is significantly higher for the cropped version. This indicates that camera placement can affect the algorithm significantly. However, even in a case of bad viewing angle a simple manual setting can provide results similar to changing the camera position.

As the abysmal recall would significantly skew the overall recall of the method, we use the cropped version of the *session 5 left* video in the overall results including the ablation experiments. Other videos could also benefit from employment of this strategy, but we opt to not use cropping when not necessary.

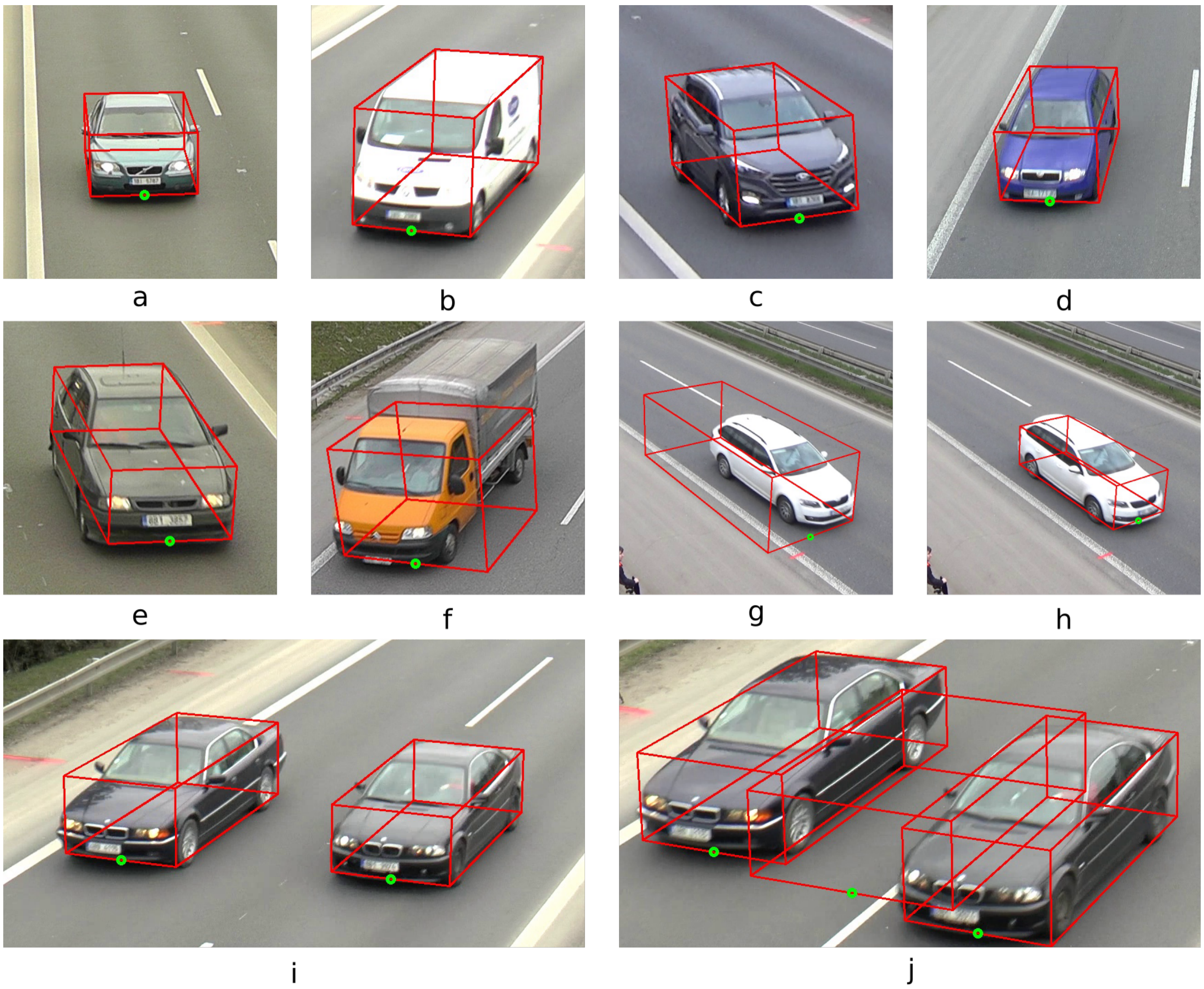


Figure 4. 3D bounding boxes (red) and the reference point for speed measurement (green) detected on the testing data: correct bounding boxes (a-d), bounding boxes with bad dimensions (e,f), the same frame from *session 5 left* without cropping (g) and with cropping (h), a pair of cars detected properly (i), few frames later at the border of the viewing window a false positive appears (j).

Method	Mean error (km/h)	Median error (km/h)	95-th percentile (km/h)	Mean Recall (%)	Mean Precision (%)
Transform3D (ours)	0.86	0.65	2.17	**89.32**	87.67
Transform2D (ours)	**0.83**	**0.60**	**2.04**	82.06	83.53
Orig2D (ours)	0.97	0.79	2.25	85.96	86.44
SochorAuto [25]	1.10	0.97	2.22	83.34	**90.72**
SochorManual [25]	1.32	0.95	3.45	83.34	**90.72**

Table 2. The results of the compared methods. Mean, median and 95-th percentile errors are calculated as means of the corresponding error statistics for each video. Recall and precision are averaged over the videos in the test set.

4.3. Speed measurement accuracy

In the Table 2 the resulting mean absolute error compared to the measured ground truth speed measurement is shown as well as mean recall and precision for the detected tracks. We make our results available online [1]. Our method achieves lower mean error and significantly lower median error than both the fully automatic method *SochorAuto* and a method using manual calibration *SochorManual*, while also

[1] `https://github.com/kocurvik/CVWW2019_results`

significantly increasing recall and decreasing precision. Since we use the same calibration as *SochorAuto* the only difference is in the detection of vehicles. Our method therefore detects the positions of vehicles more accurately.

From the results of the ablation experiments it can be noted that *Transform2D* outperforms *Transform3D* in terms of lower speed measurement error. There is a tradeoff between accuracy and recall. In some cases *Transform2D* produces a bounding box whose parts lie outside of the original image. The point which is used for tracking can therefore be discarded by the evaluation algorithm provided in [26], which may lead to the whole track being discarded. We opted to not modify the evaluation script to keep the results comparable with other research. This effect usually occurs in videos with significant distortion and therefore we expect the omitted cases to be the difficult ones, thus lowering the speed measurement error. However, we cannot conclude that there is any benefit to using 3D bounding boxes over 2D bounding boxes with regards to the speed measurement task.

The ablation method *Orig2D* also outperforms the methods from [25] with respect to errors, but by a significantly lower margin. This indicates that transforming the image is beneficial to speed measurement, but significant improvements can be obtained just by using a better object detector and training data.

4.4. Computational efficiency

Our method runs in real-time (25 FPS) on an Nvidia GTX 970 GPU. We were unable to obtain the FPS of the model we compare against, but we expect it to be lower as it uses the Faster R-CNN object detector, which is in general significantly slower compared to the detector we used [14].

5. Conclusion

We propose a method to detect and track 3D bounding boxes of vehicles in a standard traffic surveillance scenario. Our methods are based on applying a deep convolutional neural network for object detection on an image which has been rectified by a perspective transformation based on known positions of vanishing points. 3D bounding boxes can be detected directly without the need of obtaining the contours of the vehicles.

Our method improved the mean absolute error on a speed measurement task by 22 % (1.10 km/h to 0.86 km/h) and median error by 33 % (0.97 km/h to 0.65 km/h) compared to the existing state-of-the art fully automatic method, while also increasing the recall.

Acknowledgements

I would like to thank my supervisor Milan Ftáčnik for his guidance in this research. I would also like to thank Adam Herout for his valuable comments. I gratefully acknowledge the support of NVIDIA Corporation with the donation of a GPU.

References

[1] A. F. Bobick and J. W. Davis. The recognition of human movement using temporal templates. *IEEE Transactions on pattern analysis and machine intelligence*, 23(3):257–267, 2001. 2

[2] F. Cathey and D. Dailey. A novel technique to dynamically measure vehicle speed using uncalibrated roadway cameras. In *Intelligent Vehicles Symposium, 2005. Proceedings. IEEE*, pages 777–782. IEEE, 2005. 2

[3] E. R. Corral-Soto and J. H. Elder. Slot cars: 3d modelling for improved visual traffic analytics. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2017. 2

[4] J. Dai, Y. Li, K. He, and J. Sun. R-fcn: Object detection via region-based fully convolutional networks. In *Advances in neural information processing systems*, pages 379–387, 2016. 1, 2

[5] D. J. Dailey, F. W. Cathey, and S. Pumrin. An algorithm to estimate mean traffic speed using uncalibrated cameras. *IEEE Transactions on Intelligent Transportation Systems*, 1(2):98–107, 2000. 2

[6] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. Ieee, 2009. 5

[7] V.-H. Do, L.-H. Nghiem, N. P. Thi, and N. P. Ngoc. A simple camera calibration method for vehicle velocity estimation. In *Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON), 2015 12th International Conference on*, pages 1–5. IEEE, 2015. 2

[8] M. Dubská, A. Herout, and J. Sochor. Automatic camera calibration for traffic understanding. In *BMVC*, volume 4, page 8, 2014. 1, 2

[9] L. Grammatikopoulos, G. Karras, and E. Petsa. Automatic estimation of vehicle speed from uncalibrated video sequences. In *Proceedings of International Symposium on Modern Technologies, Educa-*

tion and Professional Practice in Geodesy and Related Fields, pages 332–338, 2005. 2

[10] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask r-cnn. In *Computer Vision (ICCV), 2017 IEEE International Conference on*, pages 2980–2988. IEEE, 2017. 2, 5

[11] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 5

[12] J. Huang, V. Rathod, C. Sun, M. Zhu, A. Korattikara, A. Fathi, I. Fischer, Z. Wojna, Y. Song, S. Guadarrama, et al. Speed/accuracy trade-offs for modern convolutional object detectors. In *IEEE CVPR*, volume 4, 2017. 2

[13] R. E. Kalman. A new approach to linear filtering and prediction problems. *Journal of basic Engineering*, 82(1):35–45, 1960. 3

[14] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár. Focal loss for dense object detection. *IEEE transactions on pattern analysis and machine intelligence*, 2018. 1, 2, 4, 8

[15] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014. 2, 5

[16] M. Liu and M. Zhu. Mobile video object detection with temporally-aware feature maps. *arXiv preprint arXiv:1711.06368*, 2017. 3

[17] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016. 1, 2, 5

[18] D. C. Luvizon, B. T. Nassu, and R. Minetto. A video-based system for vehicle speed measurement in urban roadways. *IEEE Transactions on Intelligent Transportation Systems*, 18(6):1393–1404, 2017. 3

[19] C. Maduro, K. Batista, P. Peixoto, and J. Batista. Estimation of vehicle velocity and traffic intensity using rectified images. In *Image Processing, 2008. ICIP 2008. 15th IEEE International Conference on*, pages 777–780. IEEE, 2008. 2

[20] A. Milan, S. H. Rezatofighi, A. R. Dick, I. D. Reid, and K. Schindler. Online multi-target tracking using recurrent neural networks. In *AAAI*, volume 2, page 4, 2017. 3

[21] G. Ning, Z. Zhang, C. Huang, X. Ren, H. Wang, C. Cai, and Z. He. Spatially supervised recurrent convolutional neural networks for visual object tracking. In *Circuits and Systems (ISCAS), 2017 IEEE International Symposium on*, pages 1–4. IEEE, 2017. 3

[22] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016. 1, 2

[23] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015. 1, 2

[24] T. N. Schoepflin and D. J. Dailey. Dynamic camera calibration of roadside traffic management cameras for vehicle speed estimation. *IEEE Transactions on Intelligent Transportation Systems*, 4(2):90–98, 2003. 2

[25] J. Sochor, R. Juránek, and A. Herout. Traffic surveillance camera calibration by 3d model bounding box alignment for accurate vehicle speed measurement. *Computer Vision and Image Understanding*, 161:87–98, 2017. 1, 2, 3, 4, 6, 7, 8

[26] J. Sochor, R. Juránek, J. Špaňhel, L. Maršík, A. Široký, A. Herout, and P. Zemčík. Brnocompspeed: Review of traffic camera calibration and comprehensive dataset for monocular speed measurement. *arXiv preprint arXiv:1702.06441*, 2017. 3, 5, 6, 8

[27] J. Sochor, J. Špaňhel, and A. Herout. Boxcars: Improving fine-grained recognition of vehicles using 3-d bounding boxes in traffic surveillance. *IEEE Transactions on Intelligent Transportation Systems*, 2018. 3, 5

[28] H. Van Pham and B.-R. Lee. Front-view car detection and counting with occlusion in dense traffic flow. *International Journal of Control, Automation and Systems*, 13(5):1150–1160, 2015. 2

[29] C. Vieren, F. Cabestaing, and J.-G. Postaire. Catching moving objects with snakes for motion tracking. *Pattern recognition letters*, 16(7):679–685, 1995. 2

[30] D. Zapletal and A. Herout. Vehicle re-identification for automatic video traffic surveillance. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 25–31, 2016. 3

[31] Y. Zhou, L. Liu, L. Shao, and M. Mellor. Dave: a unified framework for fast vehicle detection and annotation. In *European Conference on Computer Vision*, pages 278–293. Springer, 2016. 2

24th *Computer Vision Winter Workshop*
Friedrich Fraundorfer, Peter M. Roth, Fabian Schenk (eds.)
Stift Vorau, Austria, February 6–8, 2019

Counting slope regions in the surface graphs

Darshan Batavia, Walter G. Kropatsch
PRIP Group 193/03
TU Wien, Austria
`{darshan,krw}@prip.tuwien.ac.at`

Rocio Gonzalez-Diaz, Rocio M. Casablanca
Applied Maths 1
University of Seville, Spain
`{rogodi,rociomc}@us.es`

Abstract. *The discrete version of a continuous surface sampled at optimum sampling rate can be well expressed in form of a neighborhood graph containing the critical points (maxima, minima, saddles) of the surface. Basic operations on the graph such as edge contraction and removal eliminate non-critical points and collapse plateau regions resulting in the formation of a graph pyramid. If the neighborhood graph is well-composed, faces in the graph pyramid are slope regions. In this paper we focus on the graph on the top of the pyramid which will contain critical points only, self-loops and multiple edges connecting the same vertices. We enumerate the different possible configurations of slope regions, forming a catalogue of different configurations when combining slope regions and studying the number of slope regions on the top.*

1. Introduction

There is a strong correlation between an image and a geographical surface. A digital image can be perceived as a geographical surface where the height of a point in the surface is directly proportional to the pixel intensity at that coordinate. The critical points: maxima, minima and saddle from the image correspond to the hills, dales and ridge in terrain. Configurations of critical points and slope lines of surfaces were discussed by Cayley [3] and Maxwell [17]. Their observations are with respect to the earth's topography but play a significant role in exploiting properties of smooth surfaces. Nackmann Lee [15] represented and studied their possible configurations in form of a graph for a Morse function of two variables (also called 2D Morse function).

Edelsbrunner *et al.* [8, 7] attempt to capture the topological aspect of the images by constructing a hierarchy of increasingly coarse Morse-Smale complexes and decompose a piecewise linear 2D-manifold. Degenerated critical points were excluded in that study. Categorization of critical points and slope points was simplified in [4] with the use of Local Binary Patterns (LBPs).

Cerman *et al.* [5] use LBP to orient the edges of the neighborhood graph of an image. By using contraction and removal operations on the edges of the neighborhood graph, they generate progressively smaller graphs. Stacking such smaller graphs forms a graph pyramid, used for multi-resolution image segmentation. A similar approach can be found in [19] which uses a super pixel hierarchy for the formation of the pyramid used for a similar purpose.

For a 1D Morse function, critical points x are identified if its first derivative is null. For a 2D Morse function, Critical points are determined by vanishing first derivatives. To classify the critical points in 2D we study the determinant of the Hessian matrix H. If it is non-zero, then the point is a non-degenerate critical point. Otherwise, it is a degenerate critical point which is excluded from a Morse function. The signs of the eigenvalues of H are used to classify the point in maximum, minimum or saddle points.

In [12, 11] the authors form a slope complex extending the model developed in [8, 7], to include degenerated critical points. In this paper, we allow the occurrence of degenerated critical points where either the Hessian matrix is nilpotent or semi-definite. If the Hessian matrix is nilpotent, the neighborhood of the point is a plateau region, which is a local surface patch of points with the same height. If the Hessian is a positive semi-definite matrix, then one of it's eigenvalue is zero and the neighborhood of the point contains a level curve inside a 2D surface.

In formation of the graph pyramid, the top level of the pyramid is expected to be preserve only with the critical points. Also there is no unique configuration in which these critical points are connected. Euler's formula [6, theorem 4.2.7] provides the lower bound for the number of faces formed by connecting the vertices, provided the number of edges are known. It further get complicated when we allow the self-loops. In this paper, we study all the different configuration of faces also called as slope regions which are formed by connections of the critical vertices. We also present an approach to count the number of slope regions in a given primal graph which includes self-loops. In this paper we avoid use of derivatives and instead use Local Binary Patterns (LBPs) to determine the category of a point (minimum, maximum, saddle or slope) in discrete domain which is explained in Section 2 along with few other necessary definitions. Section 3 introduces the prototypes of slope regions with minimal number of edges. In Section 4 we summarize the effect of contracting the saddle components. Section 5 is dedicated to the combinations of slope regions which form the basis to represent a surface. We provide a formula to count the number of slope regions of a surface to ski down from its embedded graph in Section 6. We generalize the count made in [11] in the sense that in this paper the primal graph can contain self-loops. Finally, Section 7 is devoted to conclusions and future works. The appendix contains some examples of applying LBP pyramids on the digital images.

2. Basic definitions

A digital image P can be visually perceived as a sampled version of a geographical terrain model which is a continuous surface denoted by S. The sampling frequency to choose the samples should satisfy the Nyquist criterion with respect to the minimum distance between any two critical points. The digital image P can be efficiently represented by a dual pair of plane graphs. The primal graph or neighborhood graph $G = (V, E)$ is formed by vertices $v \in V$ corresponding to pixels $p \in P$ connected to the four adjacent neighbors by edges $e \in E$. The dual of the primal graph is the graph $\overline{G} = (\overline{V}, \overline{E})$ where every dual vertex $\overline{v} \in \overline{V}$ corresponds to a face in the primal graph G and dual edges $\overline{e} \in \overline{E}$ correspond to the border separating the faces in the G [6, Section 4.6]. The gray value (g-value) of the pixel p is visually conceived as the height of the surface and it is denoted by $g(p) = g(v)$ where v is the corresponding vertex of p. There are two operations: contraction and removal defined on the edges of the graph. Contraction of edge [6, Section 1.7] in G will result in merging the two vertices connected by the respective edge. This is equivalent to the removal operation in $\overline{G}$. The removal of an edge $(v, w) \in E$ disconnects the two vertices v and w and merges the two faces which is equivalent to contract $\overline{e} \in \overline{E}$ in $\overline{G}$. There is a one-to-one correspondence between the edges of G and $\overline{G}$. By successively contracting and removing edges, we form a stack of progressively reducing planar graphs $(G_k, \overline{G_k}), k \in \{0, 1, \dots, n\}$ where each graph G_{k+1} is smaller than the graph G_k [10, 1, 9]. The base level of the graph is the primal graph G_0.

Definition 1. *The* ***orientation of an edge*** $(i, j) \in E$ *in the primal graph* $G = (V, E)$ *is directed from vertex* i *to vertex* j *iff* $g(i) > g(j)$.

The edge $e \in E$ connecting two vertices $v, w \in V$ where $g(v) = g(w)$ are non-oriented. Note that we define orientation of edges by considering only the gray values as a feature of an image. The theory stated in this paper remains same for the higher dimensional feature vector if the orientation of edges is well defined for the domain.

The **LBP value** of an edge $e \in E$ connecting two vertices $v, w \in V$ is defined by comparing the g-values of the vertices. The LBP value of e is 1 if $g(v) > g(w)$ and it is 0 if $g(v) < g(w)$. The **LBP code** of vertex $v \in V$ is obtained by concatenating, in clockwise or counterclockwise orientation, the LBP value of edges incident to v (edges connecting vertices $v, w \in V$ such that $g(v) = g(w)$ are not considered when computing the LBP code of vertex v). LBP codes are used for the classification of vertices into maximum, minimum, saddle or slope point.

Definition 2. *A vertex* v *in graph* G_k *is a* ***local maximum*** $\oplus$ *if its LBP code consists of only 1s.*

Definition 3. *A vertex* v *in graph* G_k *is a* ***local minimum*** $\ominus$ *if its LBP code consists of only 0s.*

Definition 4. *A vertex* v *in graph* G_k *is a* ***slope point*** *if the circular permutation*[1] *of it's LBP code has exactly 2 bit switches.*

Definition 5. *A vertex* v *in graph* G_k *is a* ***saddle point*** $\otimes$ *if the circular permutation of it's LBP code has a number of bit switches greater than 2.*

[1]Circular permutation consists of rotating the code clockwise or counter-clockwise by 1 bit.

Fig. 1(a), (b), (c) and (d) are examples of a local maximum, a local minimum, a slope point and a saddle point respectively. The category of the vertex is decided by the orientation of the edges incident on the vertex and the categorization is independent of the number of the incident edges. Thus the theory can be generalized beyond the gray scale digital images, where the vertex may contain a vector of the values (for example: RGB), provided that the metric for the orientation of the edges is well defined.

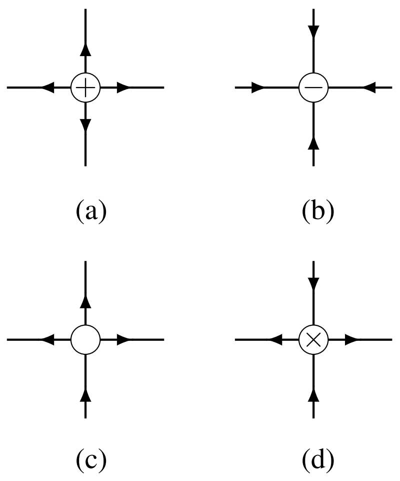


Figure 1. LBP categories and the orientation of the incident edges.

Definition 6. *A* ***path*** *π is a non empty sub-graph of G, consisting of an alternating sequence of vertices and edges $\pi = v_1, e(v_1, v_2), v_2, \ldots, e(v_{r-1}, v_r), v_r$. The LBP code of path π is obtained by consecutively concatenating the LBP value of edge $e(v_{i-1}, v_i)$ for $i \in \{2, 3, \ldots, r\}$. If there are no bit switches in the LBP code of π then π is a* ***monotonic path****. A monotonic path which contains at least one oriented edge is a* ***strictly monotonic path****.*

A **plateau** is a connected, non empty sub-graph $G_P = (V_P, E_P) \subset G = (V, E)$ such that every edge $(v, w) \in E_P$ of the plateau satisfies $g(v) = g(w)$. A **level curve** is a particular case of plateaus: It is a path along which all the vertices have the same g-value. Notice that a self-loop is also a level curve and edges of level curves are not oriented. Observe that a level curve can be a monotonic path but it cannot be a strictly monotonic path since a strictly monotonic path requires requires at least one oriented edge (i.e, an edge with LBP value). After performing the contraction of the edges of a plateau, the sub-graph is reduced to either a single vertex or a set of level curves incident to a common vertex.

A non-well composed configuration is modified to a well-composed configuration [14] by adding a dummy vertex which is a hidden saddle point [5]. Adding a hidden saddle includes addition of four edges incident on the hidden saddle point which ensures that no two local extrema of same category be part of the same face. It decomposes the respective face into four distinct slope regions with degree[2] three each.

The LBP codes are embedded after the contraction of all the edges in the plateau and adding the hidden saddles. The successive operations of contraction on edges may generate self-loops which are included in the model we provide in this document.

Definition 7. *A face in a surface embedded graph G is a* ***slope region*** *$\mathbb{S}$ if all the pairs of points in the surface corresponding to the face can be connected by a continuous monotonic curve inside the face.*

See the example of a slope region bounded by a level curve in Fig. 2d. The slope regions and their different configurations are discussed in [11, 12].

Remark 1. *The boundary $\delta\mathbb{S}$ of the slope region $\mathbb{S}$ can either be decomposed into exactly two monotonic paths or a level curve [11, Lemma 1].*

Remark 2. ***Properties of a slope region:*** *Saddle points can only exist on the boundary $\delta\mathbb{S}$ of the slope region $\mathbb{S}$ with additional edges incident to the saddle point outside the slope region. Saddle points cannot exist in the interior $\mathbb{S} \setminus \delta\mathbb{S}$ of the slope region $\mathbb{S}$ [11, Lemma 2].*

A well-composed sampled surface is a well-composed digital picture [13] which samples a continuous surface. The following property holds.

Lemma 1. *All the faces in the primal graph G_k after contraction of plateau regions in a well-composed sampled surface are slope regions.*

Proof. After collapsing the plateau region to a single vertex, the vertex is encoded by a LBP code which may result in a maximum, minimum, saddle or a slope point. If the plateau collapses into a level curve, a walk on level curves do not require an orientation of its edges. A contraction operation in the interior of graph will reduce the degree of the two face sharing the contracted edge. So after contracting the

[2]The degree of a face in primal graph is the number of edges surrounding the face.

plateau region, the maximum degree of a face in primal graph G_k will be four. After both the above mentioned operations (contraction of plateau and adding hidden saddles), the maximum degree of a face is four with a constraint that no two extrema of same category share the same face. In simple words, no two local maxima and no two local minima are on the same face. Thus we obtain an acyclic configuration of faces of degree 3 or 4 composed of not more than one maximum and one minimum. The remaining vertices are composed of slope points and / or saddles. The border of such face will always be composed of two distinct monotonic paths, i.e. the face is a slope region according to Remark 7. □

Lemma 2. *The minimum possible degree of a face to be a slope region is three.*

Proof. If the degree of a face is reduced to two, it simply means that the two vertices are connected by double edges and one of the edge can be removed to simplify the graph and eliminate redundant information. □

3. Minimal slope regions

In this section we enumerate the different possible configurations of slope regions which can be generated with a minimum number edges and the incident vertices are only critical points, also called *minimal slope regions*.

As mentioned in Lemma 2, the minimum number of oriented edges to form a slope region is three. Fig. 2 shows the possible configurations of minimal slope regions. Fig. 2a is a simple triangle which can be generated by categorizing the vertices A,B and C as maximum $\oplus$, minimum $\ominus$ and a saddle point $\otimes$ in random order. Fig. 2b is a non-simple triangle where vertex A can be a maximum or a minimum and has a self-loop encapsulated by the multiple edges connecting vertex A and B. This self-loop needs a further inside sub-graph S, otherwise it is redundant and removed by simplification. Fig. 2c is the reverse of Fig. 2b, where the face with multiple edges connecting vertices A and B is encapsulated by the self-loop attached to vertex A. Also in this case there must be a further sub-graph D between the double edges to avoid redundancy. Fig. 2d is the simplified version of Fig. 2c where one of the edge connecting vertices A and B is removed.

The motivation behind enumeration of minimal slope regions is to represent a sampled surface with slope regions formed by the critical points only. In this way, we can move ahead to our goal of counting the minimal number of slope regions (faces) in the primal graph G_k which also includes self-loops.

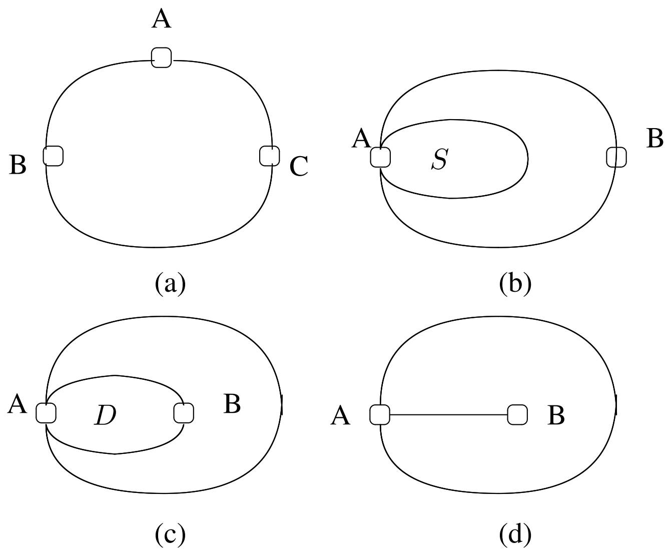


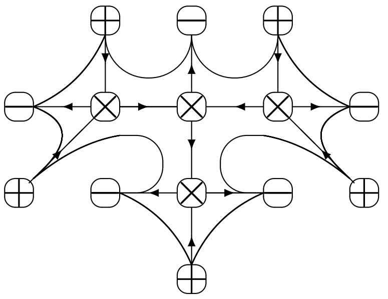
Figure 2. Configuration of slope regions with minimum number of vertices and edges.

4. Contraction of saddle components

Consider a graph G_k with all faces being minimal slope regions. The sub-graph formed by a connected component of saddle points can only have a tree structure [11, Remark 3]. In this section, we summarize the effect of contracting each saddle tree to a single point.

Lemma 3. *The number of faces of graph G_k remains the same after contracting all the connected saddle components of G_k.*

Proof. According to Lemma 1, all the faces in the primal graph G_k of a well-composed sampled surface are slope regions. The proof of Lemma 3 for a sub-graph of G_k composed of adjacent slope regions sharing saddle points can be extended to the whole graph G_k. Consider a sub-graph of G_k formed by a tree of saddle points and their incident edges and neighborhood vertices as shown in Figure 3a. Contraction of an edge in this saddle tree removes exactly one edge and one vertex. Substituting these values in Euler's formula, the number of faces will remain constant. It can be verified in Fig. 3b where the edges of the saddle tree are contracted into a single saddle point. The number of slope regions (faces) in G_k remains the same. □


(a)

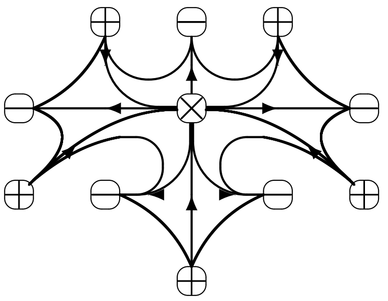

(b)

Figure 3. Sub-graph formed by a saddle tree before (a) and after (b) contraction

5. Combinations of slope regions

In this subsection we construct sub-graphs by combining different configurations of minimal slope regions mentioned in Section 3.

1. We start by generating a sub-graph formed by considering multiple occurrences of configuration Fig. 2a. We get a sub-graph with alternating sequence of maxima and minima surrounding a single saddle point as shown in Fig. 4

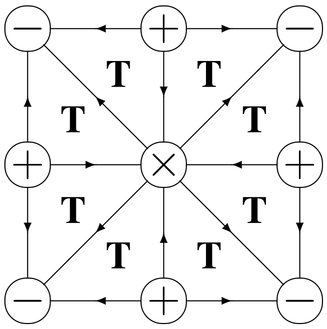
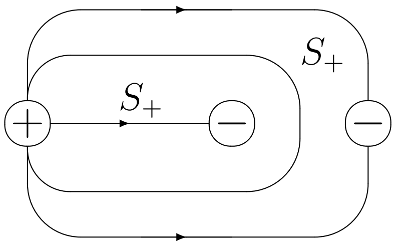


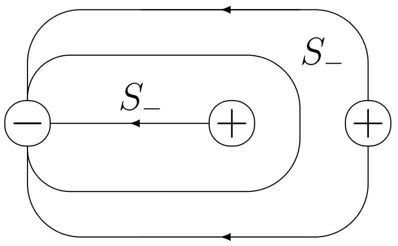
Figure 4. Saddle surrounded by 8 slope regions.

The slope region $\mathbf{T}$ in Fig. 4 is a simple triangle formed by critical points only. The lower limit of the number of slope regions $\mathbf{T}$ that can share the same saddle point is 2, while the upper limit is the total amount of slope regions on the surface.

2. The combination of self-loop and double edge configurations is more tricky. If we encapsulate configuration Fig. 2d inside Fig. 2b, we get a configuration of self-loop S_+ and S_- encapsulated by an alternating sequence of maxima and minima as shown in Fig. 5a, 5b respectively.



(a) Self-loop attached to $\oplus$.



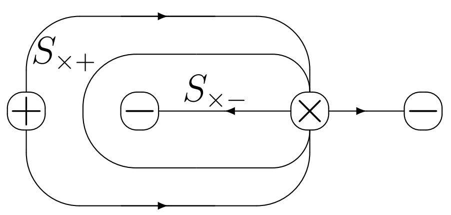
(b) Self-loop attached to $\ominus$.

Figure 5. Combining self-loops and double edges.

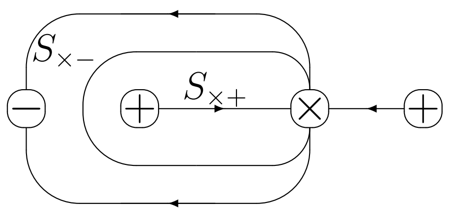
3. The self-loops attached to saddles show a simple $(S_{\times+}, S_{\times-})$ configuration (Fig. 6) and a more complex configuration involving the simple slope region $\mathbf{T}$. The simple configurations can be combined similar to the self-loops at extrema:

   In this case, the orientation of $S_{\times+}$ and $S_{\times-}$ is opposite to each other to yield an outside double edge between the saddle and the extremum. The pending edge connected to the saddle outside the slope region must be complemented by the opposite extremum. It can be completed by a cycle around $\otimes$ similar to the simple triangles $\mathbf{T}$ above.

   The complex configurations $(S_{\times+}, \mathbf{T}, \mathbf{T})$ and $(S_{\times-}, \mathbf{T}, \mathbf{T})$ (Fig. 7) have on the outside a self-loop attached to the saddle. The self-loop of $S_{\times+}$ can be encapsulated into an $S_{\times-}$-configuration and the self-loop $S_{\times-}$ into an $S_{\times+}$- configuration. In both cases the outside is a double edge connecting two different extrema.



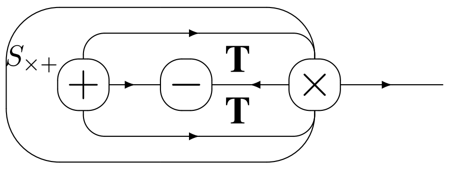
(a) $S_{\times-}$ inside $S_{\times+}$.



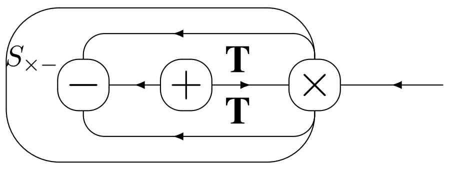
(b) $S_{\times+}$ inside $S_{\times-}$.

Figure 6. Combining self-loop and double edges attached to a saddle $\otimes$.

Hence it can be combined easily with any of the other configurations with alternating extrema on the outside.



(a) $S_{\times+}$ encapsulating two slope regions **T** inside self loop.



(b) $S_{\times-}$ encapsulating two slope regions **T** inside self loop.

Figure 7. Combining simple slope region **T** with self loop and double edges attached to $S_{\times+}$ and $S_{\times-}$

4. Alternatively, any of the configurations bounded by a double edge can be recursively embedded into a self-loop generating one more of the above primitive self-loop configurations. The completion towards the outside is then analogously as in the primitive configurations.

6. Number of slope regions

Let G_k be a graph composed of only critical points and with saddle trees being contracted. Then, the slope regions (faces of the graph) are separated by single edges only. Hence it makes sense to count the number of edges which serve as boundary between the two slope regions instead of counting the slope regions itself.

From now on, a $(\oplus, \ominus)$-edge of G_k is an edge connecting a maximum and a minimum. A $(\oplus, \ominus)$-bridge of G_k is a $(\oplus, \ominus)$-edge satisfying that its removal would disconnect the graph G_k.

Theorem 4. *Given a graph G_k made of critical points only with all the slope regions composed of maximum three edges, we apply following operations on G_k:*

1. *delete all $(\oplus, \ominus)$-edges inside the boundary of graph G_k,*
2. *keep all $(\oplus, \ominus)$-edges on the boundary of graph G_k,*
3. *keep all $(\oplus, \ominus)$-bridges, and*
4. *delete self-loops between $S_{\times+}, S_{\times-}$.*

The resulting graph is plane, all dual faces are slope regions and it cannot be further reduced without destroying the property that all faces are slope regions. The total number of slope regions corresponds to the sum of the following entities:

$$\begin{aligned}&|\{\ \textit{edge}\ (\oplus, \ominus)\ \textit{such that}\ \oplus \in V_\oplus\ \textit{and}\ \ominus \in V_\ominus\}|\\ &+|\{\ \textit{edge}\ (\otimes, \otimes)\ \textit{such that}\ \otimes \in V_\otimes\ \textit{and}\\ &\qquad\qquad\qquad \overline{(\otimes, \otimes)} = (S_{\times+}, S_{\times-})\}|.\end{aligned} \tag{1}$$

Remark that (1) counts for inner $(\oplus, \ominus)$-edges the merged slope regions but only one slope for outer $(\oplus, \ominus)$-edges and $(\oplus, \ominus)$-bridges. Multiple $(\oplus, \ominus)$-edges count only one slope.

Proof. Deletion of an edge does not change the planarity of a graph.

We show that the deletion of a $(\oplus, \ominus)$-edge or of the self-loop merges two slope regions into a new face which is a slope by considering the cases separately. A $(\oplus, \ominus)$-edge of G_k may be a $(\oplus, \ominus)$-bridge or it may be an inner $(\oplus, \ominus)$-edge of G_k. In the first two cases the $(\oplus, \ominus)$-edge bounds a single slope. Notice that a bridge need not be an outer edge and is therefore separately mentioned.

An inner $(\oplus, \ominus)$-edge in a triangular mesh is adjacent to two other triangles each being a slope with the same two local extrema. Hence the quadrilateral formed after the removal of the $(\oplus, \ominus)$-edge is also a slope. The argument remains true after the first $(\oplus, \ominus)$-edge of multiple $(\oplus, \ominus)$-edges is removed. Therefore all multiple $(\oplus, \ominus)$-edges can be removed and the merged slope regions still share the same two extrema in a single slope.

The special configurations involving self-loops enumerated in Section 3 need to be considered with care: Self-loops attached to extrema inherit the property of being extremal from their anchor vertices. They separate inner and outer faces which are both either lower or higher than the self-loop. Every path connecting a point in the inner face with a point in the outer face must cross the self-loop which is extremal and, hence, the path cannot be monotonic.

However, self-loops attached to saddles surround either a higher slope $S_{\times+}$ or a lower slope $S_{\times-}$. $S_{\times+}$ can be embedded only inside $S_{\times-}$ and $S_{\times-}$ only inside $S_{\times+}$. In both cases the removal of the self-loop generates a face which contains one minimum and one maximum and is a valid slope.

Finally we proceed to show the given count of slope regions (1). We have shown in Lemma 3 that all edges attached to saddles do not have any influence on the number of slope regions. All edges between two saddles can be contracted without reducing the number of slope regions and saddles except self-loops cannot form cycles. The first part combines the first three cases while the count of self-loops attached to saddles follows the above arguments.

□

7. Conclusion

In this paper, we have formed a catalogue of different slope configuration which can be formed using critical points. We further enumerated all the possible combinations of slope regions which forms the basis to represent a surface. Then we provide a formula to count the number of slope regions in a graph of a surface. One possible extension of counting the number of slope regions is to serve as an objective quality measure of an algorithm of multi-resolution image segmentation.

References

[1] L. Brun and W. G. Kropatsch. Dual Contraction of Combinatorial Maps. In W. G. Kropatsch and J.-M. Jolion, editors, *2nd IAPR-TC-15 Workshop on Graph-based Representation*, pages 145–154. OCG-Schriftenreihe, Österreichische Computer Gesellschaft, 1999. Band 126. 2, 8

[2] P. J. Burt, T.-H. Hong, and A. Rosenfeld. Segmentation and estimation of image region properties through cooperative hierarchial computation. *IEEE Transactions on Systems, Man, and Cybernetics*, 11(12):802–809, 1981. 8

[3] A. Cayley. Xl. On contour and slope lines. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 18(120):264–268, 1859. 1

[4] M. Cerman, R. Gonzalez-Diaz, and W. G. Kropatsch. LBP and Irregular Graph Pyramids. In N. Petkov and G. Azzopardi, editors, *Proceedings of the CAIP2015*, 2015. 1

[5] M. Cerman, I. Janusch, R. Gonzalez-Diaz, and W. G. Kropatsch. Topology-based image segmentation using LBP pyramids. *Machine Vision and Applications*, 27(8):1161–1174, 2016. 1, 3

[6] R. Diestel. Graph theory. 1997. *Grad. Texts in Math*, 1997. 2

[7] H. Edelsbrunner, J. Harer, V. Natarajan, and V. Pascucci. Morse-Smale complexes for piecewise linear 3-manifolds. In *Proceedings of the nineteenth annual symposium on Computational geometry*, pages 361–370. ACM, 2003. 1

[8] H. Edelsbrunner, J. Harer, and A. Zomorodian. Hierarchical Morse-Smale complexes for piecewise linear 2-manifolds. *Discrete and computational Geometry*, 30(1):87–107, 2003. 1

[9] Y. Haxhimusa and W. G. Kropatsch. Hierarchy of Partitions with Dual Graph Contraction. In E. Michaelis and G. Krell, editors, *DAGM 2003, 25th DAGM Symposium*, volume 2781 of *Lecture Notes in Computer Science*, pages 338–345, Magdeburg, Germany, September 2003. Springer, Berlin Heidelberg. 2

[10] W. G. Kropatsch. Building Irregular Pyramids by Dual Graph Contraction. *IEE-Proc. Vision, Image and Signal Processing*, Vol. 142(No. 6):pp. 366–374, December 1995. 2, 8

[11] W. G. Kropatsch, R. M. Casablanca, D. Batavia, and R. Gonzalez-Diaz. Computing and reducing slope complexes. In *Discrete Geometry for Com-puter Imagery*, volume 11382, pages 12–25. Springer, 2019. 1, 2, 3, 4

[12] W. G. Kropatsch, R. M. Casablanca, D. Batavia, and R. Gonzalez-Diaz. On the space between critical points. In *Computational Topology in Image Context - 7th International Workshop, CTIC 2019, Mlaga, Spain, January 24-25, 2019, Proceedings.* Springer, 2019. 1, 3

[13] L. Latecki, U. Eckhardt, and A. Rosenfeld. Well-composed sets. *Computer Vision and Image Understanding*, 61(1):70 – 83, 1995. 3

[14] L. J. Latecki. 3d well-composed pictures. *CVGIP: Graphical Model and Image Processing*, 59(3):164–172, 1997. 3

[15] R. N. Lee. Two-dimensional critical point configuration graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 4:442–450, 1984. 1

[16] D. Martin, C. Fowlkes, D. Tal, and J. Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, volume 2, pages 416–423. IEEE, 2001. 8

[17] J. C. Maxwell. L. On hills and dales: To the editors of the philosophical magazine and journal. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 40(269):421–427, 1870. 1

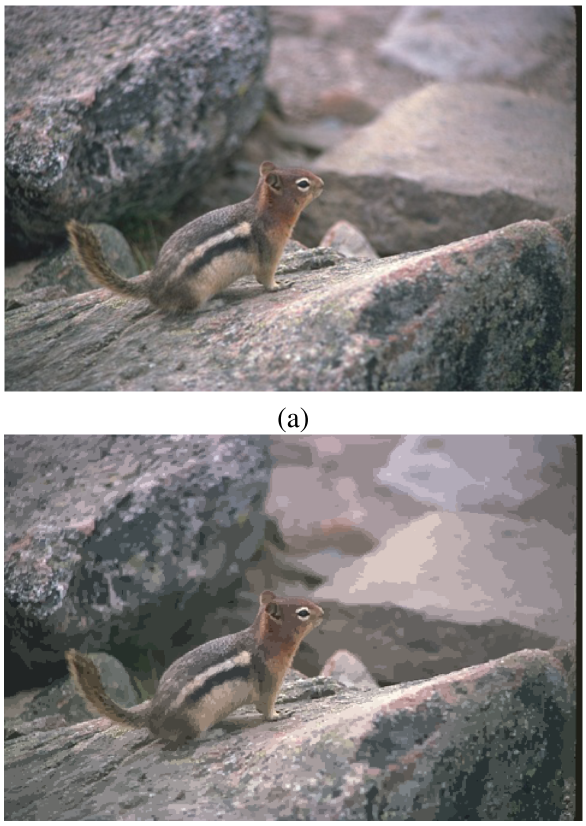
[18] A. Montanvert, P. Meer, and A. Rosenfeld. Hierarchical image analysis using irregular tessellations. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 4:307–316, 1991. 8

[19] X. Wei, Q. Yang, Y. Gong, N. Ahuja, and M.-H. Yang. Superpixel hierarchy. *IEEE Transactions on Image Processing*, 27(10):4838–4849, 2018. 1

8. Appendix

In this section, we show examples of multi-resolution image segmentation using combinatorial graph pyramid. Due to lack of flexibility in the regular pyramid [2], irregular pyramids [18] were introduced. Irregular pyramids can be stored in various data structures like adjacency graphs, dual graphs, combinatorial maps, etc. We use a stack of combinatorial maps to form combinatorial pyramids [10, 1]. Similar to graphs, we define the operation of contraction and removal of edges in the combinatorial pyramid. Edges are removed or contracted in the primal graph as long as the resulting faces continue to be slope regions. Formation of graph pyramid may result in generation of the self-loops and multiple edges between the vertices. Hence it becomes important to count the number of slope regions in primal graph (dual vertices) which essentially are the number of segments at the given level of the graph pyramid, which was the main task of this paper.

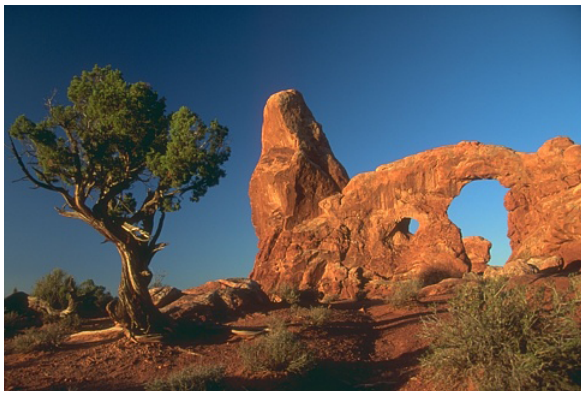
We used the Berkeley Image Segmentation Dataset [16] which consists of images of size $481 \times 321 = 154401$ pixels. That means, at the base level of the graph pyramid, there are 154401 vertices and 153600 faces. The results below show the original image and the processed image with more than 90% reduction of slope regions. It can be clearly observed in Fig. 8 and Fig. 9, that the texture information in the image is preserved and the contour effect can be seen on the region with smooth gradient.



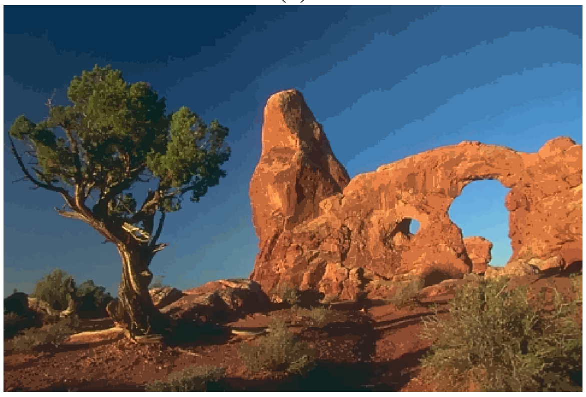
(a)

(b)

Figure 8. (a) Original image of size 481×321 with 154401 regions and (b) has 9264 regions i.e. 94% reduction.



(a)



(b)

Figure 9. (a) Original image of size 481×321 with 154401 regions and (b) has 15440 regions i.e. 90% reduction.

24th *Computer Vision Winter Workshop*
Friedrich Fraundorfer, Peter M. Roth, Fabian Schenk (eds.)
Stift Vorau, Austria, February 6–8, 2019

Leveraging Outdoor Webcams for Local Descriptor Learning

Milan Pultar, Dmytro Mishkin, Jiří Matas
Visual Recognition Group, Dept. of Cybernetics
Faculty of Electrical Engineering, CTU in Prague
milan.pultar@gmail.com, {mishkdmy, matas}@cmp.felk.cvut.cz

Abstract. *We present AMOS Patches, a large set of image cut-outs, intended primarily for the robustification of trainable local feature descriptors to illumination and appearance changes. Images contributing to AMOS Patches originate from the AMOS dataset of recordings from a large set of outdoor webcams.*

The semiautomatic method used to generate AMOS Patches is described. It includes camera selection, viewpoint clustering and patch selection. For training, we provide both the registered full source images as well as the patches.

A new descriptor, trained on the AMOS Patches and 6Brown datasets, is introduced. It achieves state-of-the-art in matching under illumination changes on standard benchmarks.

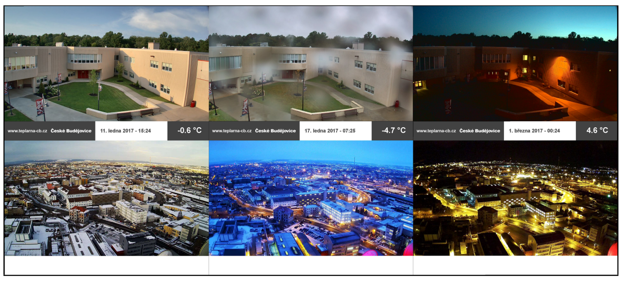
1. Introduction

Local feature descriptors are widely used in tasks such as structure from motion [34, 31], image retrieval [36] and in applications like autonomous driving [9], which benefit from the robustness of the descriptors to acquisition conditions.

Recent years have witnessed a noticeable effort to move from handcrafted descriptors [21] to those obtained by deep learning [26, 37]. Existing work explores possible architectures [5, 37], loss functions [26, 13, 16] and improvements of robustness to viewpoint changes by introducing large scale datasets from 3D reconstruction [29, 22].

Robustness to illumination and appearance changes has received little attention, yet it is a bigger challenge for modern descriptors [28, 4]. We tackle this problem by leveraging information from 24/7 webcams located worldwide [15, 14].

We make the following contributions. First, we present a method for extracting veridical patch correspondences from the "static" cameras. Second, we present the AMOS Patches dataset[1] for training of local feature descriptors with improved robustness to changes in illumination and appearance.


(a)

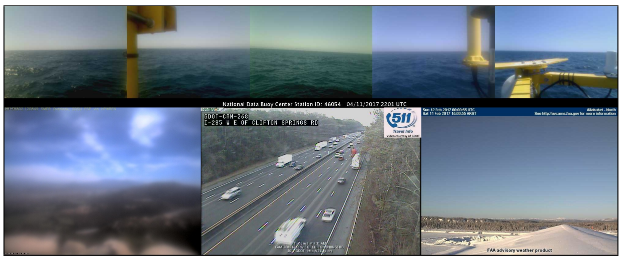

(b)

Figure 1: The AMOS dataset [14, 15] - example images from (a) cameras contributing to the AMOS patches set and (b) cameras unsuitable for descriptor training because of blur, dynamic content or dominant sky.

As a final contribution, HardNet [26] trained with AMOS Patches achieves state-of-the-art results in the commonly used HPatches benchmark [4].

2. Related Work

The literature on local feature descriptors is vast. Here we focus on descriptors which are robust to

[1]The dataset and contributing images are available at https://github.com/pultarmi/AMOS_patches

illumination and appearance changes, refering the reader to Csurka *et al.* [8] for detailed survey on recent advances in local features. There are two main ways towards achieving robustness to illumination change: by descriptor construction and by learning on the appropriate dataset. Normalization of the patch mean and variance is a simple but powerful method, which is implemented in both SIFT [21] and modern learned descriptors [37, 26]. The normalization makes the descriptor invariant to affine changes in pixel intensities in the patch. HalfSIFT [17] treats opposite intensity gradient directions as equal, trading off half of the SIFT dimensionality for being contrast reversal invariant. It works well in medical imaging and infrared-vs-visible matching.

The family of order-based descriptors like LIOP [39] or MROGH [10] operates on the relative order of pixel intensities in the patch instead of on the intensities themselves. Relative order (sorting) is invariant to any monotonically increasing intensity transformation. Descriptors like SymFeat [12], SSIM [35] and learned DASC [18] encode local symmetries and self-similarities. Another possibility is, instead of constructing a descriptor, to apply some transformation to the pixel intensities as done by the learned RGB2NIR [41] or hand-crafted LAT [32], and then use a standard descriptor, e.g. SIFT.

Data-driven approaches mostly include Siamese convolution networks with modality-specific branches, like the Quadruplet Network [3]. The decision which branch to use for a specific patch comes from an external source or a domain classifier. HNet [20] uses an auto-encoder network and style transfer methods like CycleGAN [43] for emulating different modalities.

There is a number of image-level datasets specifically designed for testing illumination-robust recognition: DTU Robot [2], OxfordAffine [25], RobotCar dataset [23], Aachen Day-Night [33], GDB [40], SymBench [12], etc. Despite the importance of the topic, the number of patch-level datasets for illumination-robust descriptors is small, especially those which are suitable for descriptor learning. To our best knowledge, Two Yosemite sequences from the Phototour dataset [6] and the Illumination split of the HPatches dataset [4] are the only ones suitable for descriptor learning and are publicly available.

3. Creating AMOS Patches

AMOS [14, 15] is a continuously growing publicly available dataset collected from outdoor webcams, currently containing over one billion (or 20 TB) images. It is organized into individual camera directories, which are split into folders according to the year and month of the acquisition. The size of the images varies, and so does their quality and the number of images in each camera directory. A typical AMOS camera is static and has approximately 300 times 300 pixel size. Many cameras store images in all seasons and during the whole day.

The advantage of static cameras lies in the fact that they show the same structure under different weather and lighting conditions. Therefore, if observing a static scene, they are highly suitable for training of local feature descriptor robust to illumination and appearance changes.

We learned the hard way that using this type of data is not trivial. Firstly, due to the dataset size, it is not feasible with moderate computing power to load such data into memory. Moreover, preprocessing would take a prohibitive amount of time. Secondly, the training procedure is sensitive to misregistration of the images and the presence and size of moving objects. Many cameras experience technical issues such as: being out of focus, rotating over time, displaying highly dynamic scene (e.g. sky, sea waves), which all significantly hurt the performance of the trained descriptor, as discussed later.

Therefore, we developed a pipeline for the creation of AMOS Patches, shown in Figure 2, which entails several steps to create a clean dataset with veridical patch correspondences. These methods focus on the selection of cameras and images, detection of view switching in a camera and the registration of images. Because of several not easily detectable problems, it was still necessary to perform final manual check of the selected image sets.

3.1. Camera selection

The first step — camera selection — aims at choosing a subset of cameras which are suitable for training, i.e. do not produce very dark images, are sharp and do not display moving objects like cars or boats.

The procedure uses two neural networks, a sky detector [24] and an object detector [1], and computes simple statistics for each of 20 randomly chosen images in each camera directory.

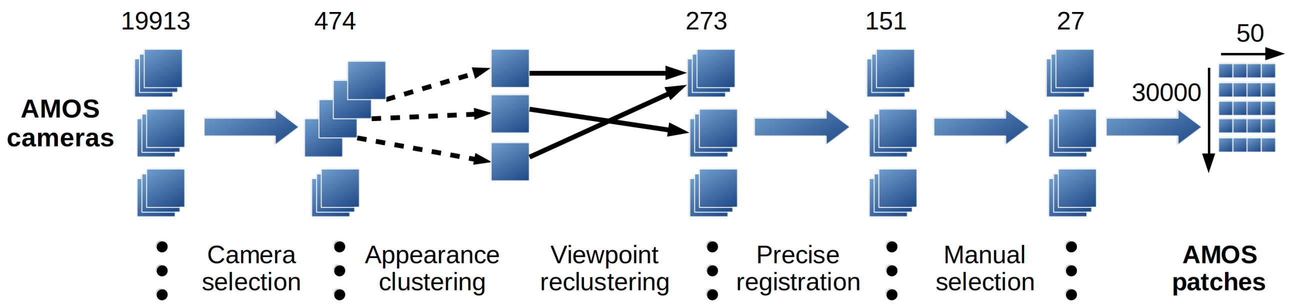


Figure 2: The pipeline of AMOS patches consists of: camera selection to filter out dynamic or empty scenes, appearance clustering to remove redundant images, viewpoint reclustering to tackle switching cameras, precise registration for further filtering, manual pruning for final selection of views and patch sampling.

The camera selection took approximately one week on a single PC (Intel^{X}eon^{C}PU E5-2620) with one GPU GTX Titan X. Processing more images by the neural network detectors increases both the precision of the method and the running time. Our choice is therefore based on the available computation power.

Each image is then checked whether it satisfies the following conditions:

- f_1 : sky area $< 50\%$ *not empty*
- f_2 : no detected cars or boats *not dynamic*
- f_3 : Var(∇^2 image) ≥ 180 *sharp*
- f_4 : mean pixel intensity > 30 *not black*
- f_5 : image size $> (700, 700)$ *large*

A camera is kept if at least 14 out of the 20 images pass the check.

The filter f_5 is the most restrictive, it removes 91% of the cameras – AMOS contains mostly low resolution images. The reasoning behind using f_5 is that images with smaller size often observe a motorway or are blurred. Also, such cameras would not generate many patches. We want to select only a relatively small subset of the cameras with the predefined characteristics and therefore an incorrect removal of a camera is not a problem.

Several cameras were removed because of corrupted image files. The resulting set contains 474 camera folders which were subject to subsequent preprocessing.

3.2. Appearance clustering by K-means

The resulting data is of sufficient quality, but it is highly redundant: images shot in 10 minute intervals are often indistinguishable and very common. To select sufficiently diverse image sets, we run the K-means clustering algorithm with K=120 to keep the most representative images. We use the fc6 layer of the ImageNet-pretrained AlexNet [19] network as the global image descriptor. While not being the state-of-the-art, AlexNet is still the most effective architecture in terms of speed [7], with an acceptable quality.

At this stage of the pipeline, there are K=120 images for each of the C=474 cameras selected, a feasible number for training with the computational resources available.

Feature descriptor training with patches selected from this image set was not successful. We were unable to achieve accuracy higher than 49.1 mean average precision (mAP) in the HPatches matching task; the state-of-the-art is 59.1 mAP – GeoDesc [22].

3.3. Viewpoint clustering with MODS

After examining the data closely, we found that many of the cameras switch between a few views, which breaks our assumption for the generation of ground truth correspondences via identity transformation. In order to filter out the non-matching views, we run MODS [27], a fast method for two-view matching, and split each camera folder into clusters, called views, by applying a threshold on the number of inliers and the difference between the homography matrix and the identity transform.

Let $(x_1, x_2, ..., x_K)$ be a set of images in a camera folder in arbitrary order. MODS matching is first run on pairs $(x_1, x_2), (x_1, x_3), ...(x_1, x_K)$. Image x_1 becomes the reference image in a newly created view, which contains x_i for which the registration yields more than 50 inliers and SAD$(H(x_1, x_i), I_3) < 50$. SAD denotes the sum of absolute differences, H denotes a homography matrix normalized by the element in position $(3, 3)$, I_3 is 3x3 identity matrix. All images in the created view are then removed from the processed image set. The step is repeated until no images remain.

We observed that the number of the resulting views in one camera folder depends on phenomena other than camera movement. For example, in cases where there is a fog or very rainy weather, MODS fails to match most of the image pairs and many of them form a single element cluster, which is excluded from further processing. For each camera, we keep only the view with the largest number of images, if it has more than 50. Each remaining view is reduced to 50 images by random selection.

3.4. Registration with GDB-ICP

While the MODS method is effective in matching and subsequent reclustering of camera sequences, in most of the cases the estimate of the global homography is not suficiently precise. MODS often outputs a homography valid for only small area in the image, see the example shown in Figure 3. Therefore, the views contain also images which are not correctly aligned. To alleviate the problem, we run Generalized Dual Bootstrap-ICP [40] to prune the set of views, keeping those where this second registration is successful.

The registration proceeds as follows. Each view folder contains images $(x_1, x_2, ..., x_{50})$, where image x_1 is the MODS reference. The GDB-ICP registration is run on pairs $(x_1, x_2), (x_1, x_3), ...(x_1, x_{50})$ and warped images $x'_2, x'_3, ..., x'_{50}$ are obtained. If registration fails on any pair, the whole view is removed.

After the precise registration with GDB-ICP, 151 views remained. It is feasible to manually inspect such a set.

3.5. Manual pruning

A few problems remain, see Figure 4, such as dynamic scenes, undetected sky (the sky detector fires mostly on the clear blue sky). As a precaution, we also removed views with very similar content and views from different cameras observing the same place from a different viewpoint. We tried to use the scene segmentation network [42] to detect moving objects, but the result was not satisfactory. The final selection is therefore done by hand, resulting in a set of 27 folders with 50 images each.

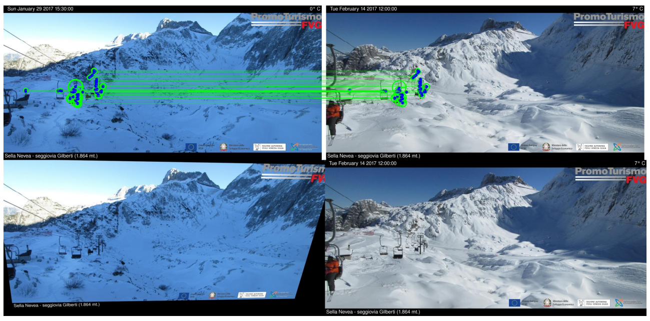


Figure 3: MODS registration failure, most of the correspondences are on moving structures. Top: an image pair with marked inliers. Bottom: wrongly transformed image (left) and the reference.

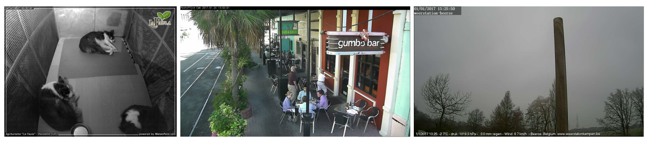


Figure 4: Manually pruned views. Examples of dynamic scenes (left, center) and a cloud-dominated scene not removed during camera selection (right).

3.6. Patch selection, training

The last phase of the AMOS Patches pipeline consists of sampling images to obtain patch centers, scales and angles, and subsequent cropping from source images. We tested two approaches. First, one may average the images in a view and evaluate a response function over the resulting image. Second, one may evaluate the response function over all images in a view and average the outputs. The resulting 2D map is then used as a probability mask for the selection of patch centers. Scales and angles are sampled independently at random from a predefined range.

For training, we use the hard-in-batch triplet margin loss [26]. This structured loss requires corresponding (positive) pairs of patches on input. Therefore, AMOS Patches dataset consists of sets of patches cropped from the same position in each image in a view. The size of each patch set is equal to the number of images in a view directory, which is 50 in our case. Each patch is resampled to 96 times 96 pixels.

During training, we apply random affine transformation and cropping to get patches of smaller size. First, random rotation from range $(-25°, 25°)$, scaling from range $(0.8, 1.4)$ and shear are applied. Second, from a 64 times 64 center of a patch we crop a

32 times 32 region with random scale. These transformed patches are the input for training.

We use the HardNet implementation in Pytorch [30]. For training we use batch size of 1024, 20 epochs, learning rate = 20, SGD optimizer with momentum $= 0.9$.

4. Evaluating influences on precision

We examine the influence of several choices made before and during training. They relate to batch formation, patch selection and the dataset size. Also, we show the importance of registration of images in a view.

Two evaluation tasks are considered. In the matching task, there are two equally sized sets of patches from two different images. The descriptor is used to find a bijection between them. The average precision (AP) over discrete recall levels is evaluated for each such pair of images. Averaging the results over a number of image pairs gives mAP (mean AP). In the verification task there is a set of pairs of patches. The descriptor assigns a score that the two patches in a pair correspond. Precision-recall curve is then plotted based on the sorted (according to the score) list of patch pairs distances.

4.1. Registration

In this experiment we show the importance of the precise alignment of images. We displace each patch by different shifts and observe the influence on the HPatches matching score, see Figure 5. Notice how the performance of the descriptor improves with a small shift, but then quickly deteriorates. We use #source views $= 27$ (all), 30000 patch sets and Hessian weighting without averaging. These parameters are defined below.

4.2. Number of source views

The hard-in-batch triplet margin loss is influenced by the composition of a batch. This experiment shows that lowering the number of views from which we choose patches to form a batch is an effective way to improve training on AMOS Patches, see Figure 6. We interpret this behaviour as follows. Reducing the number of views increases the number of negative patches from the same scene, which are often the most difficult to distinguish.

4.3. AMOS Patches size

Here we examine the influence of the dataset size, i.e. the number of patch sets created from source views, see Figure 7. We use the results from the previous experiment and choose #(source cameras) $= 6$. The graph shows there is a rough increase in HPatches matching score on bigger datasets. Based on the result, we fix the number of patches to be 30 000 to trade off dataset compactness for slightly higher performance.

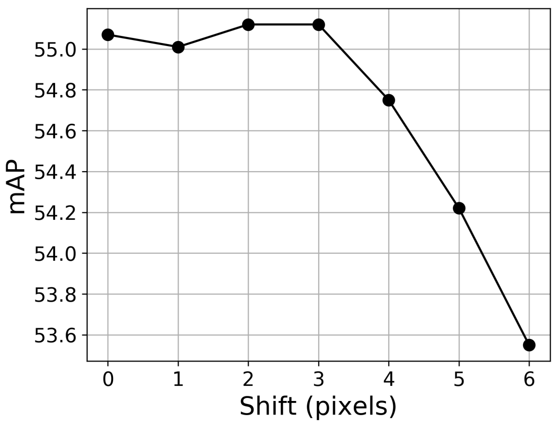


Figure 5: HPatches matching. The mAP score of Hardnet trained on AMOS patches displaced by different shifts.

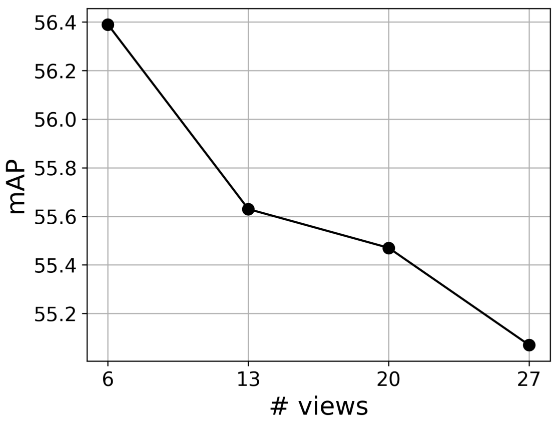


Figure 6: HardNet mAP score in HPatches matching task as a function of the number of source views for a batch. Views are selected randomly in each iteration. Dataset consists of 27 views in total.

4.4. Patch sampling

The patch selection method is partially determined by two independent choices: the response function and the averaging method. First, we find the best response function (Table 1), then we keep it fixed

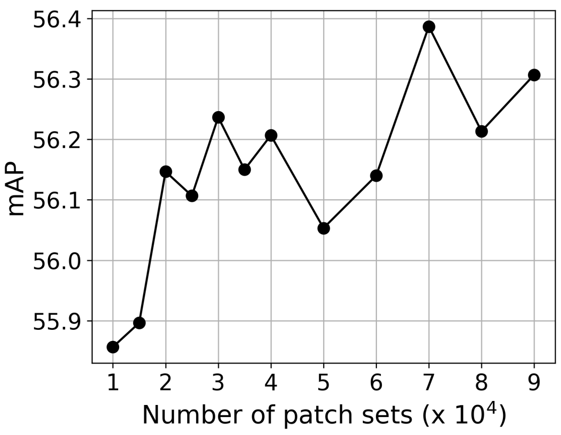


Figure 7: HardNet mAP score in HPatches matching task evaluated for different sizes of AMOS patches training dataset. Each value is an average over 3 different randomly generated datasets of the same size.

and determine the optimal averaging function, which may apply either to outputs from the response function (Table 2) or to images in a view (Table 3).

Table 1: Patch sampling: Influence of the response function on HPatches matching score (mAP).

Weighting	mAP
Uniform	56.20
Hessian	56.39
$\sqrt{\text{Hessian}}$	**56.49**
NMS($\sqrt{\text{Hessian}}$)	56.18

Table 2: Patch sampling: Influence of the response averaging on HPatches matching score (mAP). Weighting function is $\sqrt{\text{Hessian}}$.

Averaging	mAP
none	**56.49**
mean	56.10
median	56.45

5. Evaluation

HPatches and AMOS benchmarks. The evaluation shows that HardNet trained on AMOS Patches and 6Brown dataset outperforms the state-of-the-art descriptors for matching under illumination changes. We also use the new AMOS Patches testing split

Table 3: Patch sampling: Influence of the image averaging on HPatches matching score (mAP). Weighting function is $\sqrt{\text{Hessian}}$.

Image	mAP
random	56.49
median	56.44
mean	**56.58**

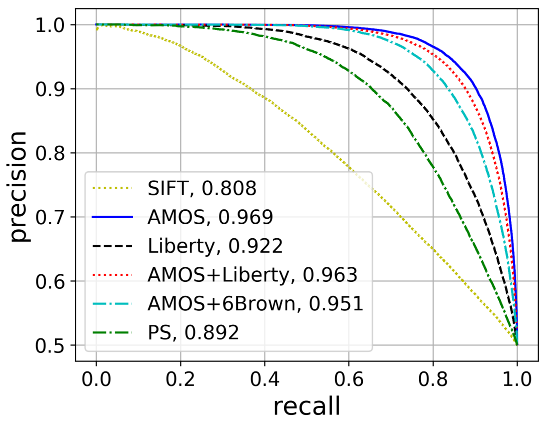


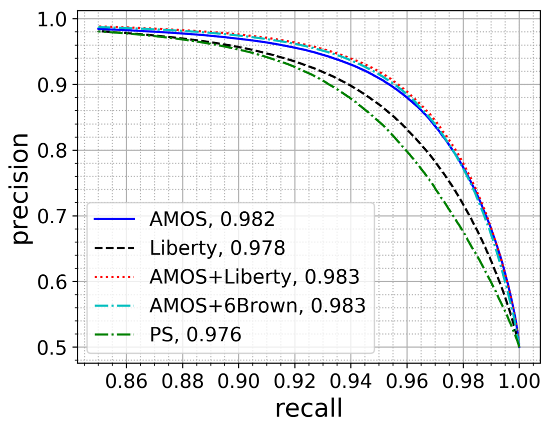
Figure 8: HardNet performance on the AMOS test set, when trained on the AMOS, Liberty, AMOS and Liberty, AMOS and 6Brown and PS [29] datasets. SIFT results are provided as a baseline.

to evaluate robustness to lighting and season-related conditions. See Table 4 for results in the matching task, Figure 9 in the verification task and Figure 8 for comparison on the proposed AMOS Patches test split.

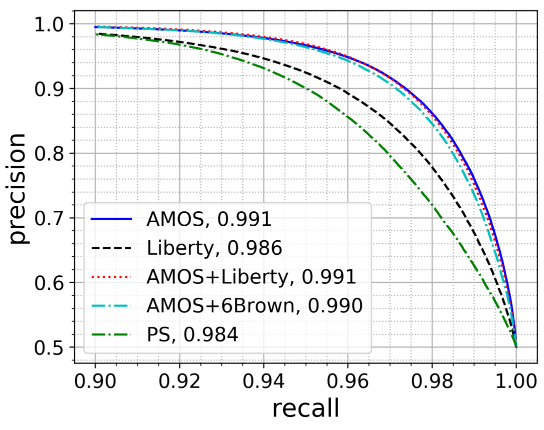
Table 4: HPatches matching scores (mAP).

Training set	HPatches subset		
	illum	view	full
Liberty	49.86	55.62	52.79
6Brown	52.39	59.15	55.83
PS	48.55	**67.43**	58.16
Webcam [38]	51.82	50.77	51.29
AMOS-patches	55.17	57.94	56.58
+Liberty	56.14	60.27	58.24
+6Brown	**56.22**	61.50	**58.91**

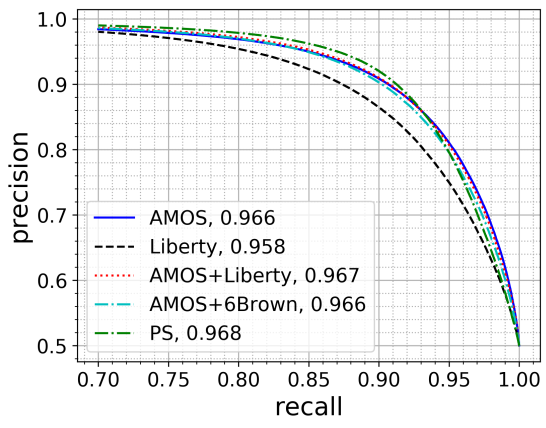
Wide baseline stereo. Finally, we evaluate the descriptors on a real-world task – wide baseline stereo



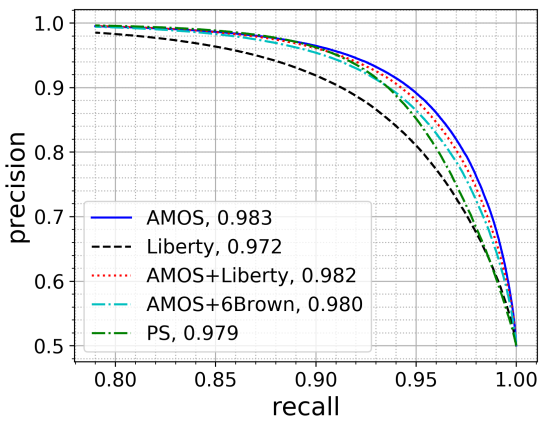
(a) HPatches intra easy



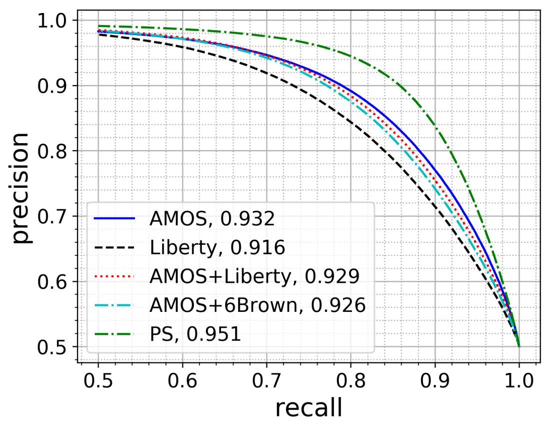
(b) HPatches inter easy



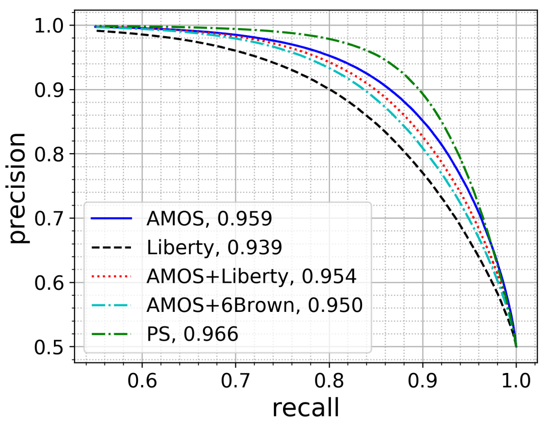
(c) HPatches intra hard



(d) HPatches inter hard



(e) HPatches intra tough



(f) HPatches inter tough

Figure 9: HardNet performance evaluated on the HPatches benchmark. Precision-recall curve is presented based on the output from the verification task. Legend shows the training set name with the corresponding AUC.

Table 5: Comparison of the AMOS+6Br HardNet vs. HardNet++ [26] following the protocol [28]. The number of matched image pairs is shown. The $\boxed{\text{numbers}}$ of image pairs in a dataset are boxed. Best results are in **bold**.

Descriptor	EF [44] $\boxed{33}$	EVD [27] $\boxed{15}$	OxAff [25] $\boxed{40}$	SymB [12] $\boxed{46}$	GDB [40] $\boxed{22}$	map2photo [28] $\boxed{6}$	LTLL [11] $\boxed{172}$
HardNet++ [26]	31	**15**	**40**	40	18	2	**108**
HardNetAMOS+6Br	**33**	**15**	**40**	**45**	**19**	**4**	106

on multiple datasets, following the protocol [28]. Two metrics are reported: the number of successfully matched image pairs and the average number of inliers per matched pair. Results are shown in Table 5. Edge Foci (EF) [44], Extreme view [27] and Oxford Affine [25] benchmarks provide a sanity check — the performance on the benchmark is saturated and they contain (mostly) images taken from a slightly different viewpoint.

SymB [12], GDB [40] and map2photo [28] contain image pairs which are almost perfectly registered, but have severe differences in illumination or modalities, e.g. drawing vs. photo, etc. AMOS+6Br HardNet performs better than baseline HardNet++ on such datasets. The last dataset – LTLL [11] consists of historical photos and old postcards. The landmarks are depicted with significant changes in both viewpoint and illumination. Baseline HardNet++ slightly outperforms our descriptor. Overall, the benchmark confirms that HardNet trained on AMOS Patches is robust to illumination and appearance changes in real-world scenarios.

6. Conclusion

We provide the AMOS Patches dataset for robustification of local feature descriptors to illumination and appearance changes. It is based on registered images from selected cameras from the AMOS dataset. It has both the training and testing split.

We introduce the local feature descriptor trained on AMOS Patches and 6Brown datasets, which achieves the score of 58.91 mAP in HPatches matching task in full split, compared to the current state-of-the-art: 59.1 mAP (GeoDesc). The advantage of the descriptor is the robustness to illumation. It achieves the state-of-the-art score of 56.22 mAP in matching task, illum split, compared to 52.39 mAP of HardNet++.

We conclude with a list of observations and recommendations related to using webcams for descriptor learning:

- Scene parsing methods do not work well in outdoor webcams. The precision of the near state-of-the-art network [42] is not satisfactory.

- For camera selection we recommend to adopt strict "quality" criteria and be prepared to loose many suitable cameras in the process.

- When picking cameras for training manually, a small and diverse subset is better than a bigger one with similar views or imprecise alignment of images.

Acknowledgements

The authors were supported by the Austrian Ministry for Transport, Innovation and Technology, the Federal Ministry of Science, Research and Economy, and the Province of Upper Austria in the frame of the COMET center SCCH, the CTU student grant SGS17/185/OHK3/3T/13, and the OP VVV funded project CZ.02.1.01/0.0/0.0/16_019/0000765 Research Center for Informatics.

References

[1] https://github.com/kuangliu/torchcv. reviewed on December 2018. 2

[2] H. Aanæs, A. Dahl, and K. Steenstrup Pedersen. Interesting interest points. *International Journal of Computer Vision*, 97:18–35, 2012. 2

[3] C. A. Aguilera, A. D. Sappa, C. Aguilera, and R. Toledo. Cross-spectral local descriptors via quadruplet network. *Sensors*, 17(4), 2017. 2

[4] V. Balntas, K. Lenc, A. Vedaldi, and K. Mikolajczyk. HPatches: A benchmark and evaluation of handcrafted and learned local descriptors. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 4, page 6, 2017. 1, 2

[5] V. Balntas, E. Riba, D. Ponsa, and K. Mikolajczyk. Learning local feature descriptors with triplets and shallow convolutional neural networks. In *Proceedings of the British Machine Vision Conference (BMVC)*, 2016. 1

[6] M. Brown and D. G. Lowe. Automatic panoramic image stitching using invariant features. *International Journal on Computer Vision*, 74(1):59–73, 2007. 2

[7] A. Canziani, A. Paszke, and E. Culurciello. An analysis of deep neural network models for practical applications. *arXiv preprint arXiv:1605.07678*, 2016. 3

[8] G. Csurka and M. Humenberger. From handcrafted to deep local invariant features. *arXiv preprint arXiv:1807.10254*, 2018. 2

[9] A. Dewan, T. Caselitz, and W. Burgard. Learning a local feature descriptor for 3d lidar scans. *arXiv preprint arXiv:1809.07494*, 2018. 1

[10] B. Fan, F. Wu, and Z. Hu. Aggregating gradient distributions into intensity orders: A novel local image descriptor. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2011. 2

[11] B. Fernando, T. Tommasi, and T. Tuytelaars. Location recognition over large time lags. *Computer Vision and Image Understanding*, 139:21 – 28, 2015. 8

[12] D. Hauagge and N. Snavely. Image matching using local symmetry features. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 206–213, 2012. 2, 8

[13] K. He, Y. Lu, and S. Sclaroff. Local descriptors optimized for average precision. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 1

[14] N. Jacobs, W. Burgin, N. Fridrich, A. Abrams, K. Miskell, B. H. Braswell, A. D. Richardson, and R. Pless. The global network of outdoor webcams: properties and applications. In *Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 111–120. ACM, 2009. 1, 2

[15] N. Jacobs, N. Roman, and R. Pless. Consistent temporal variations in many outdoor scenes. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–6. IEEE, 2007. 1, 2

[16] M. Keller, Z. Chen, F. Maffra, P. Schmuck, and M. Chli. Learning deep descriptors with scale-aware triplet networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 1

[17] A. Kelman, M. Sofka, and C. V. Stewart. Keypoint descriptors for matching across multiple image modalities and non-linear intensity variations. In *CVPR 2007*, 2007. 2

[18] S. Kim, D. Min, B. Ham, M. N. Do, and K. Sohn. Dasc: Robust dense descriptor for multi-modal and multi-spectral correspondence estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(9), Sept 2017. 2

[19] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012. 3

[20] W. Liu, X. Shen, C. Wang, Z. Zhang, C. Wen, and J. Li. H-net: Neural network for cross-domain image patch matching. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, 7 2018. 2

[21] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004. 1, 2

[22] Z. Luo, T. Shen, L. Zhou, S. Zhu, R. Zhang, Y. Yao, T. Fang, and L. Quan. Geodesc: Learning local descriptors by integrating geometry constraints. In *The European Conference on Computer Vision (ECCV)*, September 2018. 1, 3

[23] W. Maddern, G. Pascoe, C. Linegar, and P. Newman. 1 Year, 1000km: The Oxford RobotCar Dataset. *The International Journal of Robotics Research (IJRR)*, 36(1):3–15, 2017. 2

[24] R. P. Mihail, S. Workman, Z. Bessinger, and N. Jacobs. Sky segmentation in the wild: An empirical study. In *Applications of Computer Vision (WACV), 2016 IEEE Winter Conference on*, pages 1–6. IEEE, 2016. 2

[25] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L. Van Gool. A comparison of affine region detectors. *International Journal of Computer Vision (IJCV)*, 65(1):43–72, 2005. 2, 8

[26] A. Mishchuk, D. Mishkin, F. Radenovic, and J. Matas. Working hard to know your neighbor's margins: Local descriptor learning loss. In *Advances in Neural Information Processing Systems*, pages 4826–4837, 2017. 1, 2, 4, 8

[27] D. Mishkin, J. Matas, and M. Perdoch. Mods: Fast and robust method for two-view matching. *Computer Vision and Image Understanding*, 141:81–93, 2015. 3, 8

[28] D. Mishkin, J. Matas, M. Perdoch, and K. Lenc. Wxbs: Wide baseline stereo generalizations. In *Proceedings of the British Machine Vision Conference (BMVC)*, September 2015. 1, 8

[29] R. Mitra, N. Doiphode, U. Gautam, S. Narayan, S. Ahmed, S. Chandran, and A. Jain. A large dataset for improving patch matching. *arXiv preprint arXiv:1801.01466*, 2018. 1, 6

[30] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in pytorch. In *NIPS-W*, 2017. 5

[31] A. Resindra, A. Torii, and M. Okutomi. Structure from motion using dense cnn features with keypoint relocalization. *IPSJ Transactions on Computer Vision and Applications*, 10, Dec 2018. 1

[32] S. Ryu, S. Kim, and K. Sohn. Lat: Local area transform for cross modal correspondence matching. *Pattern Recognition*, 63, 2017. 2

[33] T. Sattler, W. Maddern, C. Toft, A. Torii, L. Hammarstrand, E. Stenborg, D. Safari, M. Okutomi, M. Pollefeys, J. Sivic, F. Kahl, and T. Pajdla. Benchmarking 6DOF outdoor visual localization in changing conditions. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8601–8610. IEEE Computer Society, 2018. 2

[34] J. L. Schonberger and J.-M. Frahm. Structure-from-motion revisited. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4104–4113, 2016. 1

[35] E. Shechtman and M. Irani. Matching local self-similarities across images and videos. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2007. 2

[36] T. Shen, Z. Luo, L. Zhou, R. Zhang, S. Zhu, T. Fang, and L. Quan. Matchable image retrieval by learning from surface reconstruction. *arXiv preprint arXiv:1811.10343*, 2018. 1

[37] Y. Tian, B. Fan, and F. Wu. L2-net: Deep learning of discriminative patch descriptor in euclidean space. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 1, 2

[38] Y. Verdie, K. M. Yi, P. Fua, and V. Lepetit. TILDE: A temporally invariant learned detector. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. 6

[39] Z. Wang, B. Fan, G. Wang, and F. Wu. Exploring local and overall ordinal information for robust feature description. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(11), Nov 2016. 2

[40] G. Yang, C. V. Stewart, M. Sofka, and C.-L. Tsai. Registration of challenging image pairs: Initialization, estimation, and decision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(11), 2007. 2, 4, 8

[41] T. Zhi, B. R. Pires, M. Hebert, and S. G. Narasimhan. Deep material-aware cross-spectral stereo matching. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 2

[42] B. Zhou, H. Zhao, X. Puig, T. Xiao, S. Fidler, A. Barriuso, and A. Torralba. Semantic understanding of scenes through the ADE20k dataset. *International Journal on Computer Vision*, 2018. 4, 8

[43] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *IEEE International Conference on Computer Vision (ICCV)*, 2017. 2

[44] C. L. Zitnick and K. Ramnath. Edge foci interest points. In *IEEE International Conference on Computer Vision (ICCV)*, pages 359–366, 2011. 8

24th *Computer Vision Winter Workshop*
Friedrich Fraundorfer, Peter M. Roth, Fabian Schenk (eds.)
Stift Vorau, Austria, February 6–8, 2019

Benchmarking Semantic Segmentation Methods for Obstacle Detection on a Marine Environment

Borja Bovcon, Matej Kristan
University of Ljubljana
Faculty of Computer and Information Science
1000 Ljubljana
borja.bovcon@fri.uni-lj.si
matej.kristan@fri.uni-lj.si

Abstract. *Obstacle detection is an important and critical module of autonomous navigation. Majority of modern obstacle detection algorithms are based on semantic segmentation and scene understanding. Most of these methods were developed for autonomous ground vehicles and their performance has not yet been evaluated for autonomous boats. In this paper, we (i) benchmark and analyze the most common segmentation algorithms for autonomous driving on a marine environment, (ii) propose a new, pixel-wise annotated, maritime training set for fine-tuning segmentation methods, (iii) conduct an in-depth study of their performance on Modd2 dataset, pinpoint their drawbacks along with their qualities and (iv) compare the results of classical segmentation metrics against obstacle detection metric in terms of USV safety.*

1. Introduction

Small-sized unmanned surface vehicles (USVs) are an affordable tool for navigating in shallow waters and narrow marinas. They are mainly used for coastal environmental patrol and remote inspection of difficult-to-reach man-made structures. These tasks require a high level of autonomy which primarily depends on timely detection and avoidance of nearby obstacles and floating debris. Lightweight and information-rich sensors, such as cameras, combined with computer vision algorithms are gaining prominence as leading obstacle detection mechanisms.

Obstacles can be detected by various image-processing approaches, for instance background subtraction [33], foreground extraction [12], 3-D reconstruction [34], semantic segmentation [17, 6, 7, 5] etc. Recently, the use of deep learning has contributed significantly to the striking progress in the field of semantic segmentation. The main goal of semantic segmentation methods is to perform a pixel-wise classification of the image, that provides necessary information for scene understanding. Scene understanding is a crucial part for a successful and safe autonomous navigation. Many methods [4, 3, 20, 24, 21], developed for navigation of autonomous ground vehicles, rely on semantic segmentation algorithms to detect obstacles in a scene. Siam *et al.* [32] have done an in-depth comparison of such semantic segmentation methods for autonomous driving and proposed a real-time segmentation benchmarking framework. However, in marine environment different assumptions hold and different segmentation tasks pose a challenge. For instance, the appearance of water varies significantly due to waves and weather conditions. Moreover, submerged and small obstacles might also present a significant threat to the USV.

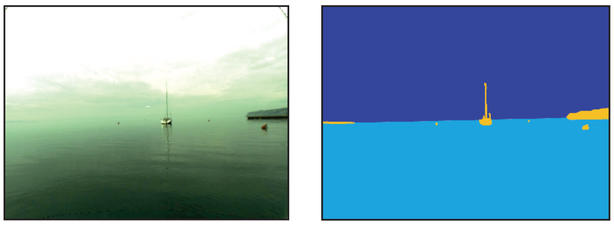


Figure 1 *Sample image and its ground truth segmentation. Sky, obstacles and water are represented with deep blue, yellow and cyan color respectively.*

In this paper we benchmark three commonly used state-of-the-art deep learning semantic segmentation methods (U-Net [31], PSP-Net [37] and DeepLab-v2 [8]) on a marine environment. We evaluate each method based on traditional segmentation metrics and compare the results against metrics used in a marine Multi-modal obstacle detection dataset 2 (Modd2) [7]. The Modd2 is currently one of the largest publicly-available datasets. It consists of a challenging sequences where the sky and the water component are not always distinguishable due to the unfavouring weather conditions. To accurately train selected deep learning segmentation methods, we provide 280 representative images of a marine environment with pixel-wise ground truth annotations.

2. Related work

Obstacle detection for unmanned surface vehicles is still a relatively young research area. A common practice for obstacle detection in marine environment is the use of range sensors - for instance radar [2, 25], sonar [15] etc. Range sensors have difficulties discriminating between water and land in the far field [11] and suffer from scanning rate limitations. Moreover radar also has problems detecting small, non-metallic obstacles. Larson *et al.* [18] presented advances in obstacle avoidance for USVs and pointed out the use of camera as an affordable and information rich alternative. Prasad *et al.* [27] have done an extensive survey of various background subtraction methods and evaluated their performance on Singapore Marine Dataset (SMD) [28] as obstacle detection mechanisms. Analysis shows that spurious dynamics of water and wakes are a leading cause of multiple false detections.

Estimating the water-edge in an image can significantly limit the region of interest (ROI) where obstacles occur. Wang *et al.* [35] combine saliency detection and motion estimation to search for obstacles below the estimated water edge. Their assumption of a sharp boundary between water and sky when estimating the water edge is in practice often violated. In [34], Wang *et al.* introduced the use of a stereo camera system to perform 3-D reconstruction of the scene, which enables them to detect obstacles above the water surface. However, only obstacles that significantly protrude through the water can be detected. Another problem arises in the state of a calm sea, where water lacks a texture, thus leading to a degraded 3-D reconstruction of the scene and consequently inaccurate water surface estimation. Alternatively Kristan *et al.* [17] proposed a graphical model (SSM) for monocular obstacle detection via semantic segmentation. The algorithm generates a water-segmentation mask and treats all blobs inside the water region as obstacles. SSM successfully detects both obstacles protruding through the surface and the floating ones, it does not assume a straight water edge and runs in real-time. Nevertheless, it still fails in the presence of visual ambiguities. For example, when the boat faces open water and the horizon is obscured by haze.

The line separating the water and sky component might not be clear due to the unfavouring weather conditions like haze. Bovcon *et al.* [6] addressed this issue by introducing measurements from the on-board IMU into the segmentation model. The IMU measurements are used to project the horizon into camera view and automatically adjust the priors and hyper-priors of the segmentation model. Their algorithm can correctly estimate the horizon even when obscured. In their recent work [5] the problem of numerous false positive detections has been addressed by a joint stereo image segmentation, where corresponding pixels in the left and right image are assigned to the same semantic region which consequently improves obstacle detection through enforced segmentation consistency. Paccaud *et al.* [26] focus on a lake-deployed USVs, where surrounding land is visible most of the time and the water surface is predominantly calm and without distinct waves. Similar to [6] they use IMU sensor to project the horizon line to the image and define the ROI in which they search for the water edge with RANSAC. On the obtained water component area they use Sobel operator along x- and y-axis in combination with threshold to find blobs representing obstacles. Detected blobs are tracked within consecutive frames to identify false detections caused by glint and reflections. Method assumes that obstacles have sharp edges and is thus unable to detect partially submerged obstacles. Jeong *et al.* [16] use a scene parsing network (PSPNet [37] pre-trained on ADE20k dataset) to perform general segmentation of the image. The horizon approximation is obtained by searching for maximal vertical location corresponding to the sea component in each column of the segmentation mask. Location of the horizon is refined by iteratively applying least-squares regression on its points. The method was evaluated on SMD [28] where it achieved promising results. However, the SMD does not contain images with intense fog, where line between sea and sky is not visible. Lee *et al.* [19] proposed using deep learning network to detect and classify ships. They use a general Faster R-CNN [30], in combination with Bayesian framework to detect ships. Method is able to detect and classify seven different types of ships and cannot be used to distinguish arbitrary obstacles in the water without providing a large amount of additional training data.

3. Semantic Segmentation CNNs

In this section we present three commonly used neural network architectures for semantic segmentation. In Section 3.1 we outline the architecture of the U-Net [31], in Section 3.2 we mark out scene parsing network PSP-Net [37], while in Section 3.3 we describe the model of the DeepLab-v2 [8].

3.1. U-Net [31]

The U-Net, proposed by Ronneberger *et al.* [31], was initially designed for bio-medical image segmentation. Since then, it was used for various segmentation purposes ranging from segmentation of urban planning maps [13] to the road detection through segmentation [22, 36]. Its architecture, shown in Figure 2 top, incorporates an encoder which captures context and a symmetric decoder that provides precise localization. The encoder part consists of a repeated application of convolutions and a max pooling operation which halves the feature map size. After each down-sampling, the number of feature channels is doubled. In contrast, the decoder part of the network is comprised of an up-sampling of the feature map size, followed by a convolution that halves the number of feature channels. A skip connection, in form of a concatenation which combines the information from a corresponding layer in the encoder part, is followed by two convolutions. Each convolution in the network is followed by a rectified linear unit (ReLU). With a proper data augmentation, the network can be trained end-to-end and pixel-to-pixel on a set of very few images and still produce good results [31].

3.2. PSP-Net [37]

Zhao *et al.* [37] designed a state-of-the-art scene parsing network PSP-Net. Its architecture is visualized in Figure 2 middle. They use a pre-trained ResNet-50 [14] backbone with a dilated network strategy to extract features from the input image. The extracted feature map is then fed to pyramid pooling module, where features are fused under four different pyramid scales. After each pyramid level, a convolution is applied to reduce the dimension of context representation and maintain the weight of a global feature. Low dimension feature maps are up-sampled to the size of the original feature map via bi-linear interpolation and concatenated with the initial feature map. Concatenation is sent through a convolution to generate the final prediction map, which is further up-sampled to the original resolution.

3.3. DeepLab-v2 [8]

Chen *et al.* [8] proposed a segmentation model that uses ResNet-101 [14] backbone with atrous convolutions to extract features from the input image. Atrous convolutions enable them to explicitly control the resolution at which feature responses are computed and to enlarge the field-of-view (FOV) of filters. The main benefit of a larger FOV is obtaining a preponderant context without increasing the number of parameters. Passing multiple rescaled versions of the original image to parallel CNN branches allows them to perform a multi-scale semantic segmentation. The responses are combined with a fully connected Conditional Random Field (CRF) which improves the localization of object boundaries. Tuning of the CRF is done separately as a post-processing step. The architecture of DeepLab-v2 is shown in Figure 2 bottom.

4. Experimental setup

The dataset and evaluation protocol are described in Section 4.1 while implementation details of evaluated methods are given in Section 4.2.

4.1. The dataset and evaluation protocol

The performance of segmentation methods was analyzed on Modd2 [7], which consists of 11675 stereo images captured by a small-sized USV in the coastal waters of Marina Koper, Slovenia. The on-board cameras can accurately estimate the depth up to 185 m and their frame-rate is limited to 10 frames per second. Obstacles and water-edge in the dataset were manually annotated with bounding boxes and a polygon respectively. The segmentation CNNs from Section 3 require sufficient training data to produce satisfactory results. We have captured and handpicked 280 images under different weather conditions from Marina Koper using the acquisition system of [7]. These images were pixel-wise annotated

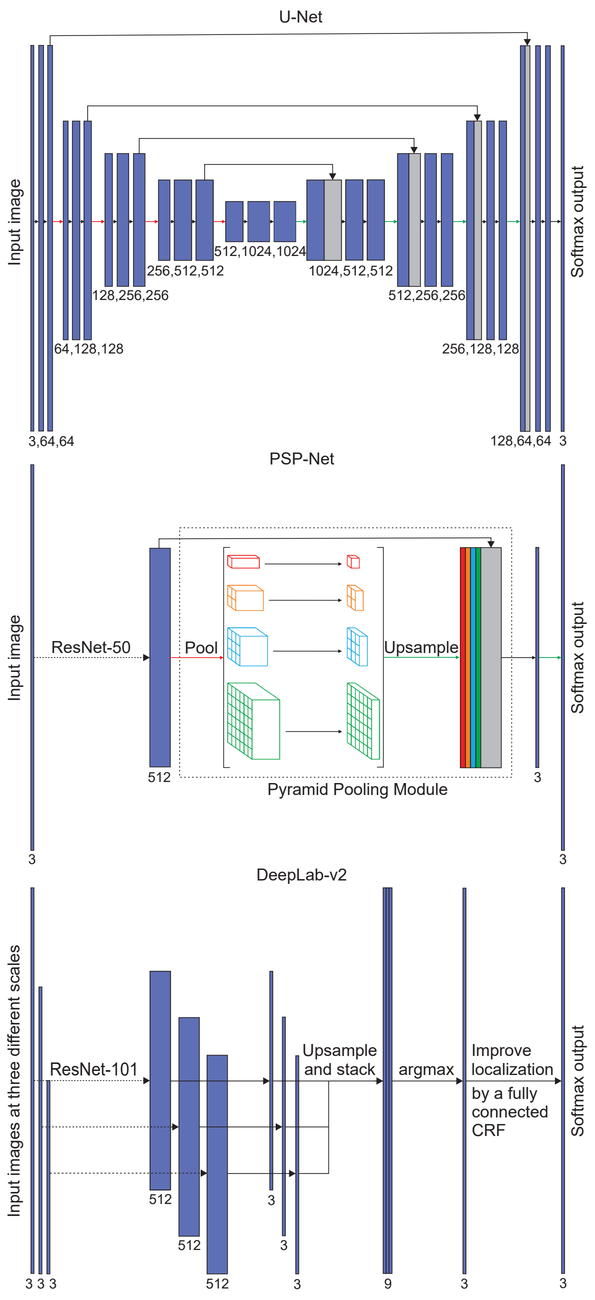


Figure 2 *Architecture illustration of tested CNNs. Black arrows represent 2-D convolutions, red arrows denote max pooling, while unpooling is marked with green arrows. The blue box corresponds to a multi-channel feature map, beneath which a number of feature channels is written. The gray box represents a copied feature map.*

for three classes (water, sky and environment) by human annotators (see Figure 1). The annotated images were further color augmented to increase diversity of the training set and replicate weather conditions from Modd2. For color augmentation we have handpicked seven descriptive images from Modd2

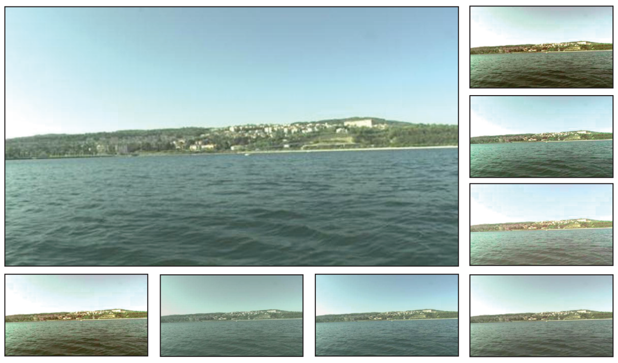


Figure 3 *Sample image from the train set (top left) surrounded by its color augmentation variations.*

which were used as target images in the color transfer method [29] proposed by Reinhard *et al.* With data augmentation we have generated 1960 new training samples with accurate ground truth annotations. Figure 3 shows a sample image from the train set and its color augmentations. Timely and accurate obstacle detection is of central importance for autonomous navigation, so we rescaled images from Modd2 on two different resolutions - low (512×288) and high (896×512) to test the detection accuracy against processing speed.

For image segmentation evaluation purposes we have used metrics inspired by Long *et al.* [23]. These metrics are mean pixel accuracy $\left(\frac{\sum_i n_{ii}}{\sum_i t_i}\right)$, mean IOU $\left(\frac{1}{n_{cl}}\sum_i \frac{n_{ii}}{t_i+\sum_j n_{ji}-n_{ii}}\right)$ and frequency weighted IOU $\left(\left(\sum_k t_k\right)^{-1}\sum_i \frac{t_i n_{ii}}{t_i+\sum_j n_{ji}-n_{ii}}\right)$, where n_{cl} denotes the number of classes in the ground truth, n_{ji} represents the number of pixels of the class j predicted to belong to the class i, while t_i stands for the total number of pixels of class i in the ground truth segmentation. Segmentation metrics do not provide information on how many obstacles were detected, neither how accurately the sea-edge was approximated. For this task, the evaluation protocol of [7] was used. It measures the accuracy of the pixel-wise water-edge estimation by mean-squared error over all sequences, while the accuracy of obstacle detection is measured by the number of true positives (TP), false positives (FP), false negatives (FN) and by the overall F-measure, i.e., a harmonic mean of precision and recall.

4.2. Implementation details

A softmax-cross-entropy loss function and a stochastic gradient descent (SGD) optimization were used to train the segmentation networks. The initial learning rate was set to a low value of 0.009 and a weight decay factor of 0.7 was applied after every 5th epoch.

In our implementation of U-Net (Section 3.1) we have employed batch-normalization after each convolution and before ReLU activation to speed-up the training process. In the PSP-Net (Section 3.2) implementation we have initialized ResNet-50 weights, which were pre-trained on the ADE20k [38, 39] dataset, while in the DeepLab-v2 (Section 3.3) implementation we have restored ResNet-101 weights, which were pre-trained on the ImageNet [10] dataset. Additionally, we have modified the number of output channels in the last layer of both DeepLab-v2 and PSP-Net according to our dataset. The fine-tuning process was carried out for 60 epochs. A single-scale version of DeepLab-v2 (Section 3.3) is denoted as DeepLab-v2S, while its multi-scale counterpart is denoted as DeepLab-v2M.

The semantic segmentation methods were implemented in Python and use Tensorflow [1] back-end. All experiments were run on a desktop computer with Intel Core i7-7700 3.6 GHz CPU and nVidia GTX1080 Ti GPU.

5. Experimental results

We begin our analysis with Section 5.1 where we analyze semantic segmentation results, in Section 5.2 we interpret obstacle detection results, while Section 5.3 serves for qualitative comparison. Results of methods from Section 3 were compared against a baseline method ISSM [7]. The speed of the tested methods is analyzed in Section 5.4.

5.1. Semantic segmentation results

The semantic segmentation results are summarized in Table 1. On low-resolution images DeepLab-v2S achieves the highest accuracy, followed by DeepLab-v2M, PSP-Net and U-Net in the order given. The differences in results between DeepLab-v2S and DeepLab-v2M are 0.11%, 0.14% and 0.10% for the mean pixel accuracy, the mean IOU and the frequency weighted IOU, respectively.

On high-resolution images, DeepLab-v2M achieves the highest accuracy based on the mean pixel accuracy and the frequency weighted IOU, followed by DeepLab-v2S, PSP-Net and U-Net. The differences in results between the top two methods are 0.09% and 0.10% for the mean pixel accuracy

Resolution 512×288

	Mean Pixel Accuracy	Mean IOU	Frequency Weighted IOU
U-Net [31]	93.12	88.82	86.27
PSPNet [37]	96.32	93.33	93.02
DeepLab-v2S [8]	**98.07**	**96.18**	**95.93**
DeepLab-v2M [9]	97.96	96.04	95.83

Resolution 896×512

	Mean Pixel Accuracy	Mean IOU	Frequency Weighted IOU
U-Net [31]	90.91	85.59	82.30
PSPNet [37]	94.69	90.42	89.98
DeepLab-v2S [8]	96.91	**94.26**	93.56
DeepLab-v2M [9]	**97.00**	94.15	**93.66**

Table 1 *Semantic segmentation results with traditional metrics - mean pixel accuracy, mean intersection-over-union and frequency weighted intersection-over-union. All reported results are in percentages.*

and the frequency weighted IOU, respectively. Based on the mean IOU metric, DeepLab-v2S outperforms DeepLab-v2M by 0.11%.

Additional smaller input images of DeepLab-v2M only detriment its performance compared to DeepLab-v2S, because bouys and other tiny obstacles disappear in the process of re-scalling. This is substantiated by a lower number of detections (shown in Table 2).

U-Net is very sensitive to reflections and sun-glitter in water, which causes a lot of false positive detections (Table 2), subsequently leading to a low segmentation accuracy. Based solely on given segmentation metrics and their results we cannot fully determine which method detects more obstacles and how well it approximates navigable surface.

5.2. Obstacle detection results

Table 2 summarizes results based on metrics used in [7]. On low-resolution images DeepLab-v2S approximates the water-edge the most accurately, fallowed by DeepLab-v2M, PSP-Net, ISSM and U-Net. DeepLab-v2S outperforms its multi-scale counterpart DeepLab-v2M by 3.6% on the water-edge estimation task. The highest F-measure score is achieved by PSP-Net, followed by DeepLab-v2S, ISSM, DeepLab-v2M and U-Net. PSP-Net outperforms second-best DeepLab-v2M by 14.1% on the obstacle detection task.

On high-resolution images DeepLab-v2M approximates the water-edge the most accurately, folowed by DeepLab-v2S, ISSM, PSP-Net and U-Net in the order given. DeepLab-v2M outperforms its single-scale counterpart DeepLab-v2S by 2.2% on the water-edge estimation task. It also obtains

Baseline

	μ_{edg}	TP	FP	FN	F-measure
ISSM [5]	0.056 (0.066)	538	1641	144	0.376

Resolution 512×288

	μ_{edg}	TP	FP	FN	F-measure
U-Net [31]	0.098 (0.090)	296	2329	383	0.179
PSP-Net [37]	0.050 (0.063)	**322**	203	**357**	**0.535**
DeepLab-v2S [8]	**0.027** (0.035)	245	121	434	0.469
DeepLab-v2M [8]	0.028 (0.041)	121	**25**	558	0.293

Resolution 896×512

	μ_{edg}	TP	FP	FN	F-measure
U-Net [31]	0.128 (0.115)	153	4686	526	0.055
PSPNet [37]	0.073 (0.101)	318	**94**	361	0.583
DeepLab-v2S [8]	0.045 (0.065)	**388**	447	**291**	0.513
DeepLab-v2M [8]	**0.044** (0.058)	361	117	318	**0.624**

Table 2 *Modd2 [7] reports water-edge estimation error μ_{edg} and its standard deviation, the number of true positive (TP), false positive (FP), false negative (FN) detections and the F-measure.*

the highest F-measure score, followed by PSP-Net, DeepLab-v2S, ISSM and U-Net. DeepLab-v2M outperforms PSP-Net by approximately 7% on the task of obstacle detection.

In general DeepLab-v2 variations approximate the water-edge most accurately. The difference in the number of detections between DeepLab-v2S and DeepLab-v2M is significant, especially on low-resolution images, where multiple re-scalled inputs of DeepLab-v2M suppress small obstacles. This causes a reduction of true positive as well as false positive detections. The difference in the water-edge approximation is less significant, because the water edge does not disappear in the process of re-scaling. PSP-Net is able to detect a lot of true positives, yet it has problems with over- and under-estimating the water edge when overlooking the open sea. Similarly to U-Net, the ISSM method is also sensitive to sun-glitter and reflections, causing a considerable amount of false positive detections and poor water-edge approximation compared to DeepLab-v2, regardless of having an additional IMU sensor. ISSM detects significantly more true positives than any method from Section 3, but its high number of false positive detections deteriorates its overall F-measure score.

5.3. Qualitative comparison

In this section we present a qualitative comparison of methods from Section 3. We limit ourselves to the input resolution of 512×288, where the difference between single-scale and multi-scale version of DeepLab-v2 is most prominent. Figure 4 depicts segmentation performance in various challenging scenarios.

The first row in Figure 4 shows a problem of a small obstacle detection. DeepLab-v2S detects a smaller buoy, while its multi-scale version suppressed the detection. The water-edge is also better estimated in a single-scale version. The water-edge estimation of U-Net is severely over-estimated, however its sensitivity allows it to correctly detect the buoy. PSP-Net is unable to detect obstacles (boat and buoy) in the scene and it drastically under-estimates the water-edge. ISSM correctly detects all obstacles in the scene. These observations are reflected in quantitative results (Table 1,Table 2) as well.

The second row in Figure 4 portraits the difficulty of water segmentation in presence of significant sun-glitter. As stated in Section 5.1, U-Net and ISSM are sensitive to sun-glitter. This causes a lot of false positive detections and poor water-edge estimation. Most of the falsely classified patches are relatively large, which has a negative effect on a segmentation accuracy presented in Table 1. In general, PSP-Net and DeepLab-v2 do not have problems with sun-glitter which is reflected in segmentation (Table 1) and obstacle detection (Table 2) results.

The third row in Figure 4 depicts a challenge of detecting an obstacle (i.e., a green buoy) whose color resembles the surrounding water. U-Net detects only a top part of the obstacle, however the bottom is the more important part for safe navigation. It also drastically over-estimates the water edge, which has a significant negative impact on the segmentation accuracy. The obstacle in a scene is big enough to not get suppressed in a multi-scale version of DeepLab-v2. Moreover, the various scales of DeepLab-v2M allow it to refine the outline of an obstacle more precisely. PSP-Net does not detect obstacle at all and its water-edge approximation is severely over-estimated. ISSM approximates the water-edge the most precisely. It detects obstacle as a whole plus a part of its reflection in the water.

The last row in Figure 4 shows a scene in a harbour with water droplets on a camera lens. The water droplets were correctly ignored by all methods. They have also correctly estimated the water-edge, however none of the CNN methods was able to detect a pole in close proximity, which is a critically dangerous misclassification. On the other hand, ISSM is able to correctly detect the pole, but its water-edge estimation is affected by sun-glitter.

5.4. Speed analysis

The processing speed of methods, described in Section 3, is presented in Table 3. On low-resolution images U-Net is the fastest, followed by DeepLab-v2S, PSP-Net and DeepLab-v2M. Similarly, U-Net is also the fastest on high-resolution images, followed by PSP-Net, DeepLab-v2S and DeepLab-v2M.

U-Net is the fastest method due to its low-complexity architecture and fewer parameters compared to those of PSP-Net and DeepLab-v2. Both PSP-Net and DeepLab-v2 use ResNet backend architecture, however PSP-Net uses ResNet-50 architecture, while DeepLab-v2 uses ResNet-101 architecture. Besides this DeepLab-v2 also has a fully convolutional CRF layer, which explains the slower performance. Despite the segmentation of multi-scale images in DeepLab-v2M is done parallel, we witness a slow-down of approximately 50% compared to DeepLab-v2S. The ISSM method is the fastest, however its performance was measured on images of size 100×100.

The on-board cameras from Modd2 [7] are limited to 10 frames-per-second, meaning that all of the methods from Section 3 would be capable of running in real-time when inputted with low-resolution images. However, only U-Net and PSP-Net would be able to run at real-time when using high-resolution images.

Baseline	t_{seg} [ms]	ω [fps]
ISSM [7]	33.8	29.6
Resolution 512×288	t_{seg} [ms]	ω [fps]
U-Net [31]	**37.6**	**26.6**
PSPNet [37]	57.9	17.3
DeepLab-v2S [8]	48.5	20.6
DeepLab-v2M [8]	98.6	10.1
Resolution 896×512	t_{seg} [ms]	ω [fps]
U-Net [31]	**93.2**	**10.7**
PSPNet [37]	98.1	10.2
DeepLab-v2S [8]	114.8	8.7
DeepLab-v2M [8]	218.7	4.6

Table 3 *Times required for single image segmentation, measured in milliseconds, is denoted with t_{seg}, while the corresponding frame-rate, measured in frames-per-second (fps), is denoted as ω.*

6. Conclusion

In this paper, we benchmarked three popular semantic segmentation methods on a marine environment and prepared an in-depth analysis of their performances. As expected, the results showed that complex networks are able to estimate the water-edge more accurately. DeepLab-v2 produced the most promising results for the task of water-edge estimation as well as for the obstacle detection task. This could be due to deeper backbone model (ResNet-101) compared to PSP-Net (ResNet-50). U-Net performed the worst, which could be a consequence of training it from scratch.

On the task of water-edge approximation CNN methods, described in Section 3, mostly over-estimate the water-edge location. In contrast, non-CNN ISSM does not over-estimate the water-edge location due to embedded IMU sensor, which serves for horizon calculation and segmentation restriction. Nevertheless, due to its sensitivity to sun-glitter, it under-estimates the water-edge location in special cases. This reduces the potential navigable surface, but it does not cause dangerous instances. On the task of obstacle detection, certain obstacles, which visual appearance is similar to water, remain undetected in all compared methods. Detection of buoys far away also proved to be difficult, but such misclassification do not pose an immediate danger to USVs. False positive detections are mainly caused by reflections and prominent sun-glitter.

When processing low-resolution images, all methods are capable of running in real-time. However, low-resolution images also produce low F-measure scores. When processing high-resolution images, presented CNN methods achieve higher F-measure scores due to mostly larger number of true positive detections. DeepLab-v2 cannot run in real-time when processing high-resolution images, while other methods are on the verge of running in real-time.

In our future work, we plan a deeper analysis of tested methods, accompanied by additional state-of-the-art segmentation methods. For a fair comparison we plan on re-train all methods on the same

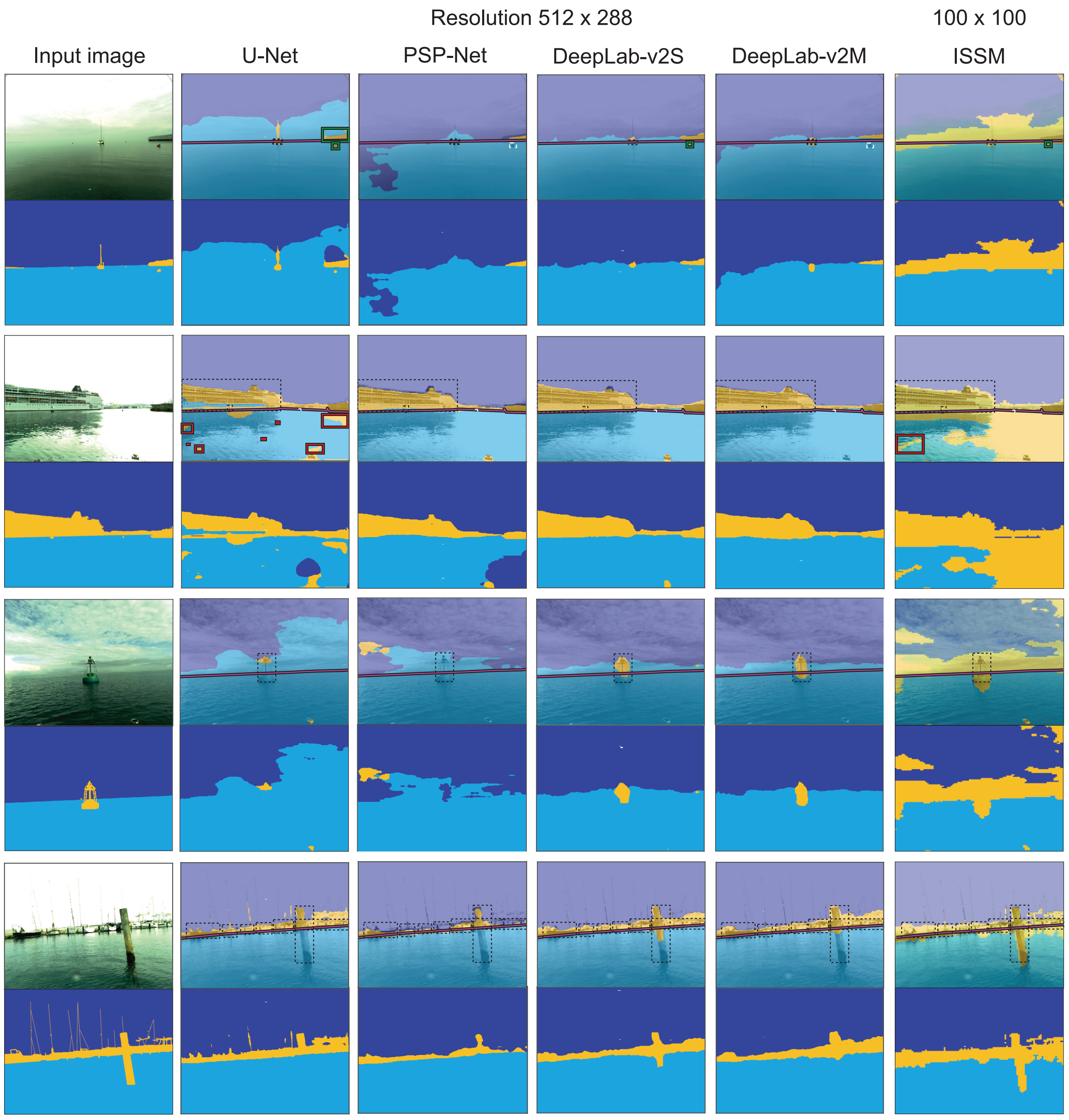


Figure 4 *Qualitative comparison of methods for resolution* 512×288*. The sky, obstacles and water components are denoted with deep-blue, yellow and cyan color, respectively. The ground truth sea edge is annotated with a pink line, while ground truth obstacles are outlined with a dotted bounding box. False positives are marked with a red bounding box, whereas correctly detected obstacles are marked with a green bounding box.*

dataset and use a significantly larger training set for fine-tunning. We will explore a new evaluation metrics, specifically designed for a marine environment, which takes into account the size of obstacles and their distances from the USV. We also plan to experiment with optimization of the segmentation process and embedding different sensor modalities into deep-learning segmentation algorithms.

ACKNOWLEDGMENT

This work was supported in part by the Slovenian research agency (ARRS) programmes P2-0214 and P2-0095, and the Slovenian research agency (ARRS) research project ViAMaRo J2-8175.

References

[1] M. Abadi et al. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[2] C. Almeida, T. Franco, H. Ferreira, A. Martins, R. Santos, J. M. Almeida, J. Carvalho, and E. Silva. Radar based collision detection developments on USV ROAZ II. In *OCEANS - EU*, pages 1–6, May 2009.

[3] J. M. Alvarez, T. Gevers, Y. LeCun, and A. M. Lopez. Road scene segmentation from a single image. In *European Conference on Computer Vision*, pages 376–389. Springer, 2012.

[4] J. M. Alvarez, Y. LeCun, T. Gevers, and A. M. Lopez. Semantic road segmentation via multi-scale ensembles of learned features. In *European Conference on Computer Vision*, pages 586–595. Springer, 2012.

[5] B. Bovcon and M. Kristan. Obstacle detection for usvs by joint stereo-view semantic segmentation. 2018.

[6] B. Bovcon, R. Mandeljc, J. Perš, and M. Kristan. Improving vision-based obstacle detection on USV using inertial sensor. In *ISPA*, pages 1–6, Sept 2017.

[7] B. Bovcon, J. Perš, M. Kristan, et al. Stereo obstacle detection for unmanned surface vehicles by IMU-assisted semantic segmentation. *Robotics and Autonomous Systems*, 104:1–13, 2018.

[8] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE TPAMI*, 40(4):834–848, 2018.

[9] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. *arXiv preprint arXiv:1802.02611*, 2018.

[10] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. Ieee, 2009.

[11] L. Elkins, D. Sellers, and W. R. Monach. The autonomous maritime navigation (AMN) project: Field tests, autonomous and cooperative behaviors, data fusion, sensors, and vehicles. *Journal of Field Robotics*, 27(6):790–818, 2010.

[12] Y. Guo, M. Romero, S. H. Ieng, F. Plumet, R. Benosman, and B. Gas. Reactive path planning for autonomous sailboat using an omni-directional camera for obstacle detection. In *ICM*, pages 445–450, 2011.

[13] Z. Guo, H. Shengoku, G. Wu, Q. Chen, W. Yuan, X. Shi, X. Shao, Y. Xu, and R. Shibasaki. Semantic segmentation for urban planning maps based on U-Net. In *IGARSS 2018-2018 IEEE International Geoscience and Remote Sensing Symposium*, pages 6187–6190. IEEE, 2018.

[14] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[15] H. K. Heidarsson and G. S. Sukhatme. Obstacle detection and avoidance for an autonomous surface vehicle using a profiling sonar. In *ICRA 2011*, pages 731–736, May 2011.

[16] C. Y. Jeong, H. S. Yang, and K. D. Moon. Horizon detection in maritime images using scene parsing network. *Electronics Letters*, 54(12):760–762, 2018.

[17] M. Kristan, V. S. Kenk, S. Kovačič, and J. Perš. Fast image-based obstacle detection from unmanned surface vehicles. *IEEE TCYB*, 46(3):641–654, 2016.

[18] J. Larson, M. Bruch, R. Halterman, J. Rogers, and R. Webster. Advances in autonomous obstacle avoidance for unmanned surface vehicles. Technical report, SPAWAR San Diego, 2007.

[19] S.-J. Lee, M.-I. Roh, H.-W. Lee, J.-S. Ha, I.-G. Woo, et al. Image-based ship detection and classification for unmanned surface vehicle using real-time object detection neural networks. In *The 28th International Ocean and Polar Engineering Conference*. International Society of Offshore and Polar Engineers, 2018.

[20] D. Levi, N. Garnett, E. Fetaya, and I. Herzlyia. Stixelnet: A deep convolutional network for obstacle detection and road segmentation. In *BMVC*, pages 109–1, 2015.

[21] J. Li, X. Liang, S. Shen, T. Xu, J. Feng, and S. Yan. Scale-aware fast R-CNN for pedestrian detection. *IEEE Transactions on Multimedia*, 20(4):985–996, 2018.

[22] L. Liu and Y. Zhou. A closer look at U-Net for road detection. In *ICDIP 2018*, volume 10806, page 108061I. International Society for Optics and Photonics, 2018.

[23] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3431–3440, 2015.

[24] G. L. Oliveira, W. Burgard, and T. Brox. Efficient deep models for monocular road segmentation. In *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*, pages 4885–4891. IEEE, 2016.

[25] C. Onunka and G. Bright. Autonomous marine craft navigation: On the study of radar obstacle detection. In *ICCAR 2010*, pages 567–572, Dec 2010.

[26] P. Paccaud and D. Barry. Obstacle detection for lake-deployed autonomous surface vehicles using RGB imagery. *PloS one*, 13(10):e0205319, 2018.

[27] D. K. Prasad, C. K. Prasath, D. Rajan, L. Rachmawati, E. Rajabally, and C. Quek. Object detection in a maritime environment: Performance evaluation of background subtraction methods. *IEEE Transactions on Intelligent Transportation Systems*, pages 1–16, 2018.

[28] D. K. Prasad, D. Rajan, L. Rachmawati, E. Rajabally, and C. Quek. Video processing from electro-optical sensors for object detection and tracking in a maritime environment: a survey. *IEEE Transactions on Intelligent Transportation Systems*, 18(8):1993–2016, 2017.

[29] E. Reinhard, M. Adhikhmin, B. Gooch, and P. Shirley. Color transfer between images. *IEEE Computer graphics and applications*, 21(5):34–41, 2001.

[30] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[31] O. Ronneberger, P. Fischer, and T. Brox. U-Net: Convolutional networks for biomedical image segmentation. In *MICCAI*, pages 234–241. Springer, 2015.

[32] M. Siam, M. Gamal, M. Abdel-Razek, S. Yogamani, M. Jagersand, H. Zhang, N. Vallurupalli, S. Annamaneni, G. Varma, C. Jawahar, et al. A comparative study of real-time semantic segmentation for autonomous driving. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 587–597, 2018.

[33] D. Socek, D. Culibrk, O. Marques, H. Kalva, and B. Furht. A hybrid color-based foreground object detection method for automated marine surveillance. *LNCS*, 3708:340, 2005.

[34] H. Wang and Z. Wei. Stereovision based obstacle detection system for unmanned surface vehicle. In *ROBIO*, pages 917–921, 2013.

[35] H. Wang, Z. Wei, C. S. Ow, K. T. Ho, B. Feng, and J. Huang. Improvement in real-time obstacle detection system for USV. In *ICARCV*, pages 1317–1322, 2012.

[36] W. Xia, Z. Chen, Y. Zhang, and J. Liu. An approach for road material identification by dual-stage convolutional networks. In *IGARSS 2018 - 2018 IEEE International Geoscience and Remote Sensing Symposium*, pages 7153–7156, July 2018.

[37] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia. Pyramid scene parsing network. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 2881–2890, 2017.

[38] B. Zhou, H. Zhao, X. Puig, S. Fidler, A. Barriuso, and A. Torralba. Semantic understanding of scenes through the ADE20k dataset. *arXiv preprint arXiv:1608.05442*, 2016.

[39] B. Zhou, H. Zhao, X. Puig, S. Fidler, A. Barriuso, and A. Torralba. Scene parsing through ADE20K dataset. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.