SEMANTIC SEGMENTATION AND COMPLETION OF 2D AND 3D SCENES

DISSERTATION

zur Erlangung des Doktorgrades (*Dr. rer. nat.*)

der Mathematisch-Naturwissenschaftlichen Fakultät

der Rheinischen Friedrich–Wilhelms–Universität, Bonn

vorgelegt von

MARTIN GARBADE

aus

Saarburg

Bonn, 2019

Angefertigt mit Genehmigung der Mathematisch-Naturwissenschaftlichen Fakultät der Rheinischen Friedrich–Wilhelms–Universität Bonn

1. Gutachter / 1st Advisor: Prof. Dr. Juergen Gall
2. Gutachter / 2nd Advisor: Prof. Dr. Simone Frintrop
Tag der Promotion / Day of Promotion: 25.11.2019
Erscheinungsjahr / Year of Publication: 2019

Abstract

by Martin Garbade

for the degree of

Doctor Rerum Naturalium

Semantic segmentation is one of the fundamental problems in computer vision. This thesis addresses various tasks, all related to the fine-grained, *i.e.* pixel-wise or voxel-wise, semantic understanding of a scene. In the recent years semantic segmentation by 2D convolutional neural networks has become as much as a default pre-processing step for many other computer vision tasks, since it outputs very rich spatially resolved feature maps and semantic labels that are useful for many higher level recognition tasks. In this thesis, we make several contributions to the field of semantic scene understanding using an image or a depth-measurement, recorded by different types of laser sensors, as input.

Firstly, we propose a new approach to 2D semantic segmentation of images. It consists of an adaptation of an existing approach for real time capability under constrained hardware demands that are required by a real life drone. The approach is based on a highly optimized implementation of random forests combined with a label propagation strategy.

Next, we shift our focus to what we believe is one of the important next forefronts in computer vision: To give machines the ability to anticipate and extrapolate beyond what is captured in a single frame by a camera or depth sensor. This anticipation capability is what allows humans to efficiently interact with their environment. The need for this ability is most prominently displayed in the behaviour of today's autonomous cars. One of their shortcomings is that they only interpret the current sensor state, which prevents them from anticipating events which would require an adaptation of their driving policy. The result is a lot of sudden breaks and non-human-like driving behaviour, which can provoke accidents or negatively impact the traffic flow.

Therefore we first propose a task to spatially anticipate semantic labels outside the field of view of an image. The task is based on the Cityscapes dataset, where each image has been center cropped. The goal is to train an algorithm that predicts the semantic segmentation map in the area outside the cropped input region. Along with the task itself, we propose an efficient iterative approach based on 2D convolutional neural networks by designing a task adapted loss function.

Afterwards, we switch to the 3D domain. In three dimensions the goal shifts from assigning pixel-wise labels towards the reconstruction of the full 3D scene using a grid of labeled voxels. Thereby one has to anticipate the semantics and geometry in the space that is occluded by the objects themselves from the viewpoint of an image or laser sensor. The task is known as 3D semantic scene completion and has recently caught a lot of attention. Here we propose two new approaches that advance the performance of existing 3D semantic scene completion baselines. The first one is a two stream approach where we leverage a multi-modal input consisting of images and Kinect depth measurements in an early fusion scheme. Moreover we propose a more memory efficient input embedding. The second

approach to semantic scene completion leverages the power of the recently introduced generative adversarial networks (GANs). Here we construct a network architecture that follows the GAN principles and uses a discriminator network as an additional regularizer in the 3D-CNN training. With our proposed approaches in semantic scene completion we achieve a new state-of-the-art performance on two benchmark datasets.

Finally we observe that one of the shortcomings in semantic scene completion is the lack of a realistic, large scale dataset. We therefore introduce the first real world dataset for semantic scene completion based on the KITTI odometry benchmark. By semantically annotating alls scans of a 10 Hz Velodyne laser scanner, driving through urban and countryside areas, we obtain data that is valuable for many tasks including semantic scene completion. Along with the data we explore the performance of current semantic scene completion models as well as models for semantic point cloud segmentation and motion segmentation. The results show that there is still a lot of space for improvement for either tasks so our dataset is a valuable contribution for future research into these directions.

Keywords: semantic segmentation, semantic scene completion

Contents

List of Figures

List of Tables

Nomenclature

Abbreviations

An alphabetically sorted list of abbreviations used in the thesis:

AI	Artificial Intelligence
BOW	Bag of words
CAD	Computer-aided design
CRF	Conditional random field
CNN	Convolutional Neural Network
DCNN	Deep Convolutional Neural Network
DNN	Deep Neural Network
EM	Expectation Maximization
GAN	Generative Adversarial Network
HOG	Histogram of Oriented Gradients
IoU	Intersection over Union
m	meters
MAP	Maximum-a-Posteriori
MRF	Markov random field
RGB-D	RGB-Depth
SGD	Stochastic gradient descent
SIFT	Scale invariant feature transform
SLAM	Simultaneous localization and mapping
SVM	Support vector machine

Frequently Used Symbols

$\mathbf{I}$	Image
$\mathcal{G}$	Graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$
$\mathcal{V}$	Vertices of graph $\mathcal{G}$
$\mathcal{E}$	Edges of graph $\mathcal{G}$
E	Energy function
$\mathcal{L}$	Loss function
p	Probability
$\phi(\cdot)$	Unary potential
$\psi(\cdot, \cdot)$	Pairwise potential
θ	Parameters

List of Publications

- **Real-time Semantic Segmentation with Label Propagation**
  Rasha Sheikh, Martin Garbade, and Juergen Gall
  In ECCV Workshop on Computer Vision for Road Scene Understanding and Autonomous Driving (CVRSUAD), 2016
  https://pages.iai.uni-bonn.de/gall_juergen/download/jgall_realtimesegmentation_cvrsuad16.pdf

- **Thinking Outside the Box: Spatial Anticipation of Semantic Categories**
  Martin Garbade, and Juergen Gall
  In British Machine Vision Conference (BMVC), 2017.
  https://pages.iai.uni-bonn.de/gall_juergen/download/jgall_spatialanticipation_bmvc17.pdf

- **Two Stream 3D Semantic Scene Completion**
  Martin Garbade, Yueh-Tung Chen, Johann Sawatzky, and Juergen Gall
  In CVPR Workshop on Multimodal Learning and Applications Workshop (MULA), 2019.
  https://arxiv.org/abs/1804.03550

- **3D Semantic Scene Completion From a Single Depth Image using Adversarial Training**
  Yueh-Tung Chen, Martin Garbade, and Juergen Gall
  In IEEE International Conference on Image Processing (ICIP), 2019.
  https://arxiv.org/abs/1905.06231

- **A Dataset for Semantic Segmentation of Point Cloud Sequences**
  Jens Behley *, Martin Garbade *, Andres Miloto, Jan Quenzel, Sven Behnke, Cyrill Stachniss, and Juergen Gall
  In IEEE International Conference on Computer Vision (ICCV), 2019.
  https://arxiv.org/pdf/1904.01416.pdf

»What is good?« Ye ask.
To teach a machine how to see is good.

NIETZSCHE*, Thus spake Zarathustra. [loosely paraphrased]*

Acknowledgements

Having finished writing my thesis and now sitting in a hotel in Long Beach - Los Angeles - where I just got to attend what was arguably the largest computer vision conference in the history of the field (CVPR'19), it feels like a good moment to look back.

I had a great time during my PhD and the luxury of studying one of the most vibrant and fastest growing research fields of today and thereby being accompanied by some of the most proficient people. I'd like to thank them here!

First of all, I want to thank Prof. Jürgen Gall for accepting me as his PhD student. Having graduated in physics, this was a huge opportunity for me and required a lot of trust from his side. Working with Jürgen has been very efficient and uncomplicated. He was always there for questions and helped with many good ideas. Especially before deadlines, when many papers had to be edited often until late into the night, he helped us getting the job done without batting an eye.

Besides I'd like to thank Prof. Simone Frintrop, Prof. Thomas Schultz and Prof. Stefan Linden for accepting to be part of my PhD evaluation committee.

Next I'd like to thank my working group who made this PhD a memorable time with entertaining conversations and prolific discussions about research. First I'd like to thank Johann Sawatzky with whom I had by far the most frequent and deepest discussions (work- and non-work-related). Another thank goes to my former office mates Umar Iqbal and Mohsen Fayyaz whom I saw more often than anyone in my life at the time, and who always guaranteed for a nice and friendly working environment. A special thanks goes to Alexander Richard, who often gave me good advice and also fueled the efficiency of the entire group by his nice scheduling tool. Next I'd like to thank Hildegard Kühne, Ahsan Iqbal, Sovan Biswas, Andreas Döering, Yasser Souri, Julian Tanke, Rania Briq and Yazan Abu Farha for being such good colleagues that were all very open, friendly and helpful to me. Another big thank goes to Jens Behley, a long-time computer vision companion, whom I had the honor of collaborating with in the final project of my PhD as well as to Rasha Sheikh for smoothly handing over the MoD project to me. Last but not least, I'd like to thank our current and former master students, namely Christian Grund for being a very prolific Hiwi and a crack in Blender visualizations, Yueh-Tung, whom I was happy to supervise during his master thesis and Anna Kukleva, for discussions and being a fun travel mate at the CVPR.

Schließlich möchte ich mich bei meiner Familie und Geschwistern bedanken, die mir bei allem immer zur Seite gestanden haben und die mit mir gemeinsam durch etliche Deadlines gefiebert haben. Darüberhinaus danke ich allen meinen Freunden, die während der gesamten Zeit für die notwendige und erholsame Ablenkung gesorgt haben.

CHAPTER 1

Introduction

Contents

1.1 Motivation

1.1.1 Why Computer Vision?

One of the greatest achievements that humans might solve within our lifetime is construction of intelligent, autonomous robots. The vision already caught Hollywood's attention, although mostly in a negative way. Movies like 'Terminator' or 'The Matrix' describe the future as a war between intelligent machines and humans. We, on the contrary, will focus on the potential for good that lies in this technology.

Along with the discovery of electricity and inventions like the steam engine and computers, the invention of autonomous robots is another milestone in the people's pursuit to replace harsh, dangerous, and costly human labour by machines. It will be interesting to see, if we are the generation to witness armies of robots, harvesting crops, farming land or perhaps even build entire cities at the command of a single person. They could be used to rebuild cities destroyed by war or to restore precious architectural artifacts like Notre-Dame of Paris which recently burned almost to the ground. At least in case of Notre-Dame, 3D reconstruction with the help of computer vision techniques is already likely to play a role in its restoration.

Just as the invention of engines and electricity helped to replace the human muscle by machines, the goal of computer vision and its close relative artificial intelligence are bound to replace the human eyes and brain. At the end of the day, all a man will need is a will and a machine to execute it. In a society equipped with fully autonomous machines, most of the dangerous, laborious and undesired

jobs might be done by machines. Even so much so that mainly creative, social and managing jobs might be left, a vision that scares some people rather than it entertains them. However, history teaches us, that people are very creative when it comes to inventing new jobs, which are likely to be more comfortable than those taken over by machines.

The question is, why aren't we already there? After all, we are able to construct robots in all shapes and forms. The reason is that one final wall on our way there has not been conquered yet: Perception. Though we are able to build robots and manually control them, the goal must be that they interact autonomously with their environment such that all they need is a high level plan and they themselves will figure out how to execute it in detail. This is the motivation for doing computer vision. The goal is to teach machines to see the world, similar to how we humans see it. Thus they can effectively interact with it, navigate through it, make predictions about its state, grasp things and put them somewhere else, mow lawn, harvest crops or drive anyone safely from A to B. This thesis is dedicated to one of the most important technological goals in society today: Bringing perception to machines.

1.1.2 Applications of Computer Vision

Beyond the big dreams, computer vision has been an active field of research for the past 6 decades. It already has reached some impressive goals. We are already surrounded by cleaning robots or cars equipped with computer vision systems to increase the safety.

Since May 2012 Google followed by other companies like Uber and Tesla are already driving with fully autonomous cars through cities of the United States. These cars still have a human pilot that can take over the driving in case the system fails, but the amount of kilometers driven without human interference increases steadily. The chase for the first fully autonomous cars has attracted peoples attention and sparked their ambition in a truly Olympic spirit.

Though many of the first autonomous cars were equipped with a variety of sensors (radars, ultra sound, Velodyne-laser-range scanners), currently most mass produced vehicles with computer vision techniques involve a camera sensor. The advantage over the above mentioned sensors is clear. A camera is extremely cheap ($\sim$1 Eur vs $\sim$60 000 Eur for a Velodyne Laser currently) and has the richest signal of all due to its storage of RGB information and it has by far the best resolution in the far field. On a camera picture one can easily identify objects in a distance of up to 200m whereas a laser range signal becomes rather uninterpretable after 50m distance. On the other hand, RGB images are much harder to interpret and thus machine learning systems become potentially more error prone as the variance in the data increases. In 2017 a Tesla car, equipped with a vision based driving assistant mistook a white truck for the class 'sky' which made the inattentive driver the first victim of a computer vision controlled car[1].

All of the before-mentioned sensors only measure some distance value to an obstacle which is good for collision avoidance, however often crucial parts of the scene, like the difference between sidewalk and road only become obvious in a picture, which has color values. This thesis, therefore, deals with the 'king of the sensors', the RGB camera.

[1] https://www.theverge.com/2016/6/30/12072408/tesla-autopilot-car-crash-death-autonomous-model-s

1.1.3 History of Computer Vision

Computer vision is an extremely fast advancing field and in many ways fields like the Wild West among the sciences. Today state-of-the-art algorithms are usually 'dethroned' after not much more than half a year. The community is open in sharing data and models, so people can immediately start from a state-of-the-art algorithm and improve on it. The usage of common benchmark datasets and evaluation protocols ensures the fair comparison between algorithms. During my PhD a seismic shift shook the entire research community: The successful rise of deep learning techniques above all previous, sophisticated and highly engineered methods. Since then, the chase for better models was partly overshadowed by a chase for more annotated data.

In the early 60s computer vision began as an engineering problem. Datasets were extremely simple and people tried to come up with domain specific expert knowledge in order to recognize objects in images. Apart from these early approaches the field was for the most time of its existence ruled by the following paradigm: Engineer hand-crafted features that capture the important aspects of the object you want to detect, then apply some intermediary processing to either quantize the features or to aggregate them, finally use a classifier to determine which of the predefined classes you have detected.

This paradigm shifted entirely after in 2012 the most successful approach winning the ImageNet image recognition challenge used an entirely different approach: Deep learning[2].

The concepts of deep learning were old. Already in the sixties people tried to emulate the learning behaviour of the human brain by an algorithm[3]. This was the beginning of a new research field called artificial intelligence (AI). Though people believed in the power of AI for some years, the algorithms were lacking far behind other more engineered approaches. This lead to an era called AI winter[4], where the hype around AI burst like a bubble and high ranking conferences refused to accept papers that claimed new advances in the field.

Luckily a small group of people clang on to the idea, such that over 4 decades later, when computers had much more parallel computing power due to GPUs and the amount of annotated data was orders of magnitudes larger, they could outperform all existing approaches with a simple end-to-end system not much unlike one used in the 70s which could only recognize numbers between 0 and 9 on 28×28 pixel images.

1.1.4 Why Semantic Segmentation?

After this short detour into the history of computer vision, we will now explain why we focus on semantic segmentation and its related fields. Compared to one of the oldest computer vision tasks (image classification), semantic segmentation is something like a 'next step'. Since image classification has made tremendous advances, the interest in the field declines. As people see it as 'solved' or rely on non-academic research with their greater access to data and computational resources. Instead people come up with new and more challenging problems. Compared to image classification, semantic segmentation is the next harder step, though being still very basic in its goal: Assign a class label to every pixel in an image. By trying to label every pixel the problem becomes harder, as one has to deal with localization of objects, identifying their shape, boundaries between objects as well as very

[2] https://qz.com/1307091/the-inside-story-of-how-ai-got-good-enough-to-dominate-silicon-valley/

[3] https://towardsdatascience.com/rosenblatts-perceptron-the-very-first-neural-network-37a3ec09038a

[4] https://towardsdatascience.com/history-of-the-first-ai-winter-6f8c2186f80b

small objects that only cover a minuscule part of the image and often need context information in order to be classified.

Again a traditional approach would use a bottom-up graph based segmentation algorithm to compute superpixels, then use an object classifier on these superpixels and occasionally a conditional random field (CRF) in a post-processing step to enforce more label consistence in the local neighbourhood of pixels.

Today, virtually all state-of-the-art approaches rely on fully convolutional neural networks, which are extremely powerful and flexible. There is no need for hand-crafting features anymore and almost any parameter of the model itself is learned during the training of the network so the need for tuning hyperparameters is minimal. Furthermore the same CNN architecture can be used for almost any computer vision related problem with some slight domain specific adaptation. CRFs are still used in some modern approaches to refine the segmentation results.

1.1.5 Open Problems in Semantic Segmentation

Similar to image recognition, semantic segmentation has made impressive advances in a couple of years, so much that even in this field people start moving away in the search for harder problems to be solved. On the other hand, fully convolutional CNNs pre-trained for semantic segmentation have become ubiquitous as feature extractor and starting point for almost any algorithm of related fields (pose estimation, action recognition, tracking, monocular depth estimation).

In this thesis we follow the same pattern. Our first approach which does classical semantic segmentation stems from a time, when the entire field was transitioning towards CNN based methods. It is a classic approach which is highly optimized to achieve real-time capabilities on a GPU-free hardware device (drone). This approach is effective as our experiments show, though it is likely that future drones will carry small GPUs in case the power consumption is not to high as compared to CPUs.

After that we asked ourselves, what could be the next frontier in computer vision / semantic scene understanding. One hint comes from the observation of autonomous cars: Apart from some famous stories that made it to the news of fatal failure of computer vision algorithms: 1) Man crashes his Tesla into a white truck, since the image recognition system mistakes the white truck for "sky" (probably due to greyscale images as used by Mobileye having less color contrast) or 2) Car drives over (seemingly drunken) person that crosses the road, pushing a bike while ignoring the giant Uber SUV approaching him, again with fatal consequences. These accidents hint to the fact that current systems are likely to perform reliably 99.9% of the time, but fail utterly in the occurrence of rare events which are not part of the training data. The above listed examples are shocking as they are easy to manage by a human driver

The latter story could be already related to our problem. The key observation comes from simple observation of the driving performance of current autonomous cars. It looks, like their driving behaviour is very non-human-like, meaning they drive extremely cautiously, stop at every hard interpretable situation and most importantly seem not to understand how the scene surrounding them will evolve in the near future.

This is a new task we address with this thesis: The problem of anticipation. Humans automatically can infer a lot of information in a top down process. They know from experience a lot about the scene, objects in it and their likely behaviour.

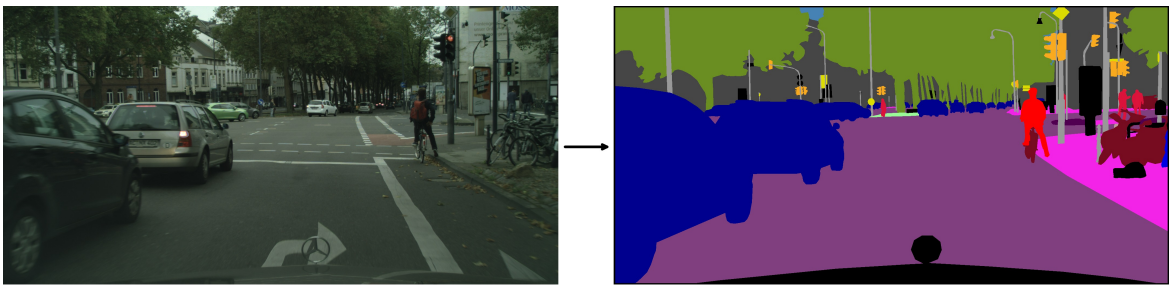

Figure 1.1: Example of an input and output pair for the task of semantic segmentation. Every pixel of the input RGB image has to be assigned to a class label.

There are two types of anticipation that are of interest: Temporal anticipation and spatial anticipation. Temporal anticipation would deal with the dynamic of a given scene in the near future. This is a topic closely related to action recognition which lies outside the focus of this thesis. The other type of anticipation is spatial anticipation. This means given an input image we try to extrapolate the scene to parts that are outside the field of view of a sensor or otherwise occluded. This is the exact part that we focus on in this work. First we start in two dimensions. We ask ourselves: Given a 2D input image of a scene, what is the most likely semantic extrapolation of the scene beyond the image boundary. Thereby we effectively try to anticipate a scene and objects that are not yet visible but likely to occur. This might for example cause an autonomous car to drive safely as it anticipates pedestrians or playing children suddenly emerging from within a row of tightly parked cars. Just as humans learn to drive carefully in such situations as they anticipate the sudden occurrence of a person from experience, likewise we train a system that learns to anticipate similar events from data.

Next we extend the idea to the 3D domain. The reconstruction of a 3D scene from a single input image or depth measurement (*e.g.* by a Kinect or Velodyne laser sensor) has naturally to deal with uncertainties due to sensor occlusions. From every element that composes the 3D scene only the surface that is visible to the sensor is detected. The backside of every element is occluded by the objects themselves. Humans however are able to predict the geometry and semantics of objects even in the occluded space. The field of 3D semantic scene completion tries to emulate this capability. In this thesis we present two new efficient approaches for the task of 3D semantic scene completion as well as a new benchmark suited to boost future research in the field.

1.2 Problem Formulations

We will now clarify for every task, addressed in this thesis, what is the exact input to each algorithm and the respective output.

1.2.1 Semantic Segmentation

Semantic segmentation is the task of assigning a semantic class label to every pixel of a 2D representation of a scene. Thereby elements composing the scene are simultaneously identified, localized and their shape delineated. Therefore the output looks like a map of semantic segments that belong to different class labels. Figure 1.1 shows an example.

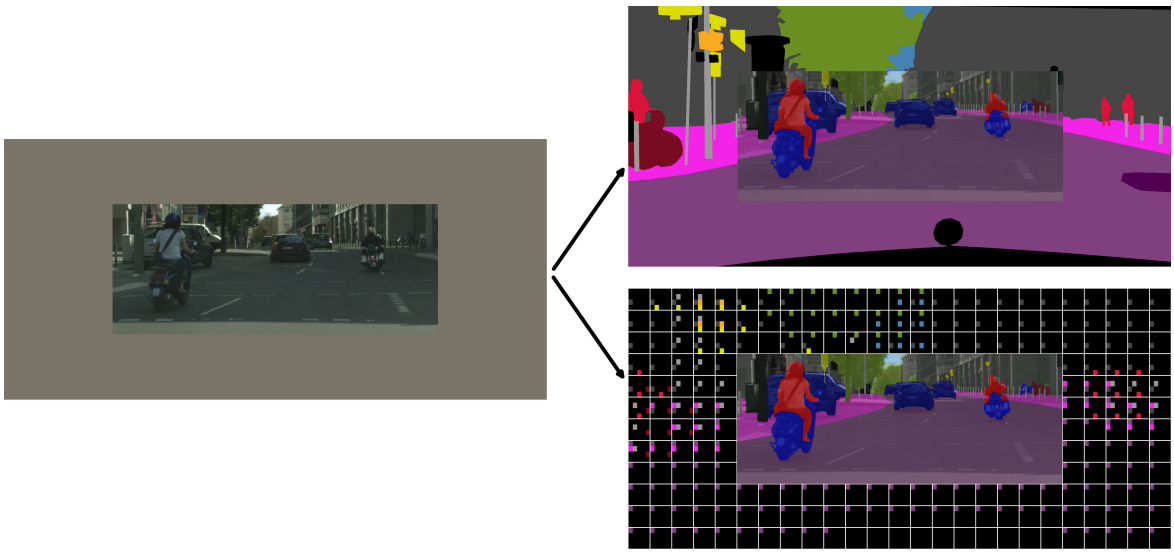

Figure 1.2: Visualization of input and output to the 2D spatial anticipation of semantic categories task. We define two tasks. The first one (top right) requires to predict a pixel-wise semantic segmentation map in the grey area where no sensory input is given. The second task (bottom right) requires a cell wise prediction, where each cell contains a boolean indicator, highlighting which classes are occurring in the respective cell.

Assigning semantic meaning to the world as it is perceived is what a child learns while growing up. Thereby it develops a language which allows it to identify different objects as well as to communicate and consciously reason about objects, matter, etc. and their relation to one-another and to itself. Thereby one has to take into account that words charged with a semantic meaning may stand in intricate relation to one another: Words like object, vegetation, wood, tree for example are in a hierarchical relationship where one word is either an abstraction of the other or includes it as an element of which composes another element. In the definition of today's semantic segmentation benchmarks these hierarchical relations have to be taken into account. Most benchmarks propose a fixed list of predefined semantic class labels that are clearly distinguishable for a human annotator and do not have any semantic overlap with each other. In this thesis, we follow this pattern and only deal with datasets that provide non-hierarchical and unambiguous label categories.

All future tasks that involve interpreting image content enabling an autonomous robot to perceive its environment and to efficiently interact with it will either implicitly or explicitly have to solve the task of semantic segmentation.

1.2.2 Spatial Semantic Anticipation

Another major concept we deal with in this thesis is the task of spatial semantic anticipation. This task is defined as the prediction of semantic categories outside the field of view, i.e. without any sensory input data. In practice we take as input a 2D image of a scene. An example is shown in Figure 1.2. As output we generate a semantic segmentation map for the region surrounding the field of view, captured by the image. The thereby inferred semantic segmentation map is what we call "spatially anticipated". It is an extrapolation of the scene as seen by the camera sensor. The

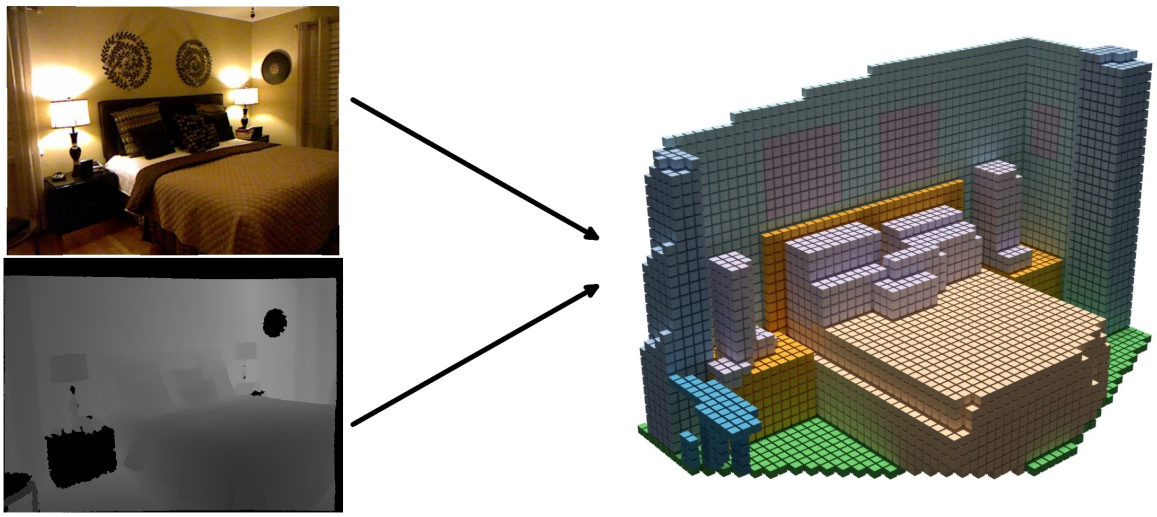

Figure 1.3: Visualization of input and output to the 3D semantic scene completion task. Top left: Input RGB image. Bottom left: Input depth map. Right: Target - a voxelized and semantically labeled representation of the 3D scene.

fundamental idea behind this task is that humans also extrapolate the surroundings of a scene given a single view. Even though the prediction might be ambiguous (since what is a valid extrapolation into the unseen part of a scene is inherently non-deterministic) making plausible predictions of a scene helps to efficiently navigate through it and to come up with an efficient path planning strategy. It also could prevent a robot from the need of exhaustively exploring an entire scene. Moreover, potentially dangerous events can be prevented. Without anticipation capabilities a ride in an autonomous car can be an unintuitive experience. This could lead to unexpected breaks or extreme driving manoeuvres, for example if a small child, initially occluded by cars parking along the edge of the road, suddenly runs onto the road. The probability of such events has to be estimated and the driving policy adapted accordingly.

1.2.3 Semantic Scene Completion

A natural extension of spatial 2D semantic label anticipation into the realm of 3D scene understanding is called 3D scene completion. In this task the input is some kind of sensory input, e.g. from a RGB-D camera or a RGB camera triggered by a Velodyne laser scanner mounted on a car. These sensors only capture appearance and geometric information from surfaces visible to the position of the sensor. As output however we generate a semantically labeled 3D voxel grid including those parts of the scene which are occluded by the visible surfaces. See Figure 1.3 for an example. E.g. if the camera captures the front and top of a bed or table, the output should contain a complete voxelized version of this bed or table. Therefore, like in the 2D case, the algorithm has to anticipate the geometry and semantics of parts of the scene that are beyond what is recorded by the sensors. The motivation for this task is the same as for 2D spatial anticipation of semantic categories. In the 3D case however the geometry and proportionate object part sizes of the scene have to be reproduced as well.

1.3 Contributions

In this work we try to advance the field of semantic scene understanding on several fronts: On the one hand we provide 2 works in the field of 2D semantic image segmentation.

- The first work is a 2D semantic segmentation algorithm with real time performance and competitive accuracy tailored to run on a single threaded device. The idea came from a research project, the goal of which was to equip a research drone (also called "Unmanned Aerial Vehicle" - UAV) with on-device semantic segmentation capability to enable autonomous flight even in case of lacking contact to a ground station server. We propose a new method which is based on texton features and random forests and leverages quad-tree based label propagation as well as spatial Gaussians to achieve a competitive semantic segmentation accuracy while operating in real time ($\sim 30ms$) on a single threaded CPU. We further provide an extensive study on feature and parameter design choices and their impact on computation time and accuracy.
- The second work is a seminal work in 2D spatial semantic label anticipation. To the best of our knowledge we are the first to introduce this kind of task. We introduce two benchmarks based on the Cityscapes dataset. The first one measures how good a proposed extrapolation of a semantic segmentation matches with a pixel-wise segmentation map annotated by humans which we consider as ground truth. The second benchmark relaxes for the need of exact pixel-wise prediction in favour of a cell-wise prediction. In this case the area to predict is cut up into a grid of larger cells. All labels occurring in the pixel-wise annotated ground-truth are counted in a boolean histogram as being present or not present. The goal in this task is to predict the histogram for every cell. In contrast to the pixel-wise accuracy metric, the cell-wise prediction metric puts a less-hard constraint of the exact localization of a pixel which is inherently non-deterministic in the case of scene extrapolation and therefore allows for more practical predictions. Furthermore we propose CNN based models and a new loss function designed to efficiently solve these new tasks.

On the other hand we provide contributions for the task of 3D semantic scene completion

- First we propose a new model in the recently emerged field of semantic scene completion based on fully convolutional 3D-CNNs. We introduce a two stream architecture which fuses data from an RGB and a depth sensor to infer a 3D semantic completion of a scene. Furthermore we show that inference time can be greatly improved while still achieving state-of-the-art performance.
- Secondly, we introduce an approach that examines the potential of generative adversarial networks for the task of 3D semantic scene completion. Our best approach demonstrates that using conditional GANs can greatly improve the performance of 3D CNN architectures modelled for semantic scene completion. Furthermore, we identify that this performance is not robust across all evaluated benchmarks. Given that the concerned dataset is very small and has a high problem of misalignment between real world input data and synthetic annotated labels, we infer that there is a need for a new dataset in semantic scene completion.
- Lastly we introduce a new dataset suitable for 3D scene completion. As hitherto all 3D scene completion works either work on the very small NYUv2 benchmark or on synthetic data taken

from the SUNCG dataset, there is an ardent need for a new large scale dataset in order to effectively train CNN architecture based models on this task. Along with the dataset we provide a quantitative evaluation of common 3D scene completion models on this new dataset. Moreover, since the dataset consists of annotated laser scans from a point cloud dataset, we also explore the performance of current state-of-the-art semantic segmentation algorithms for point clouds as well as motion segmentation.

1.4 Thesis Structure

We begin by revising some theoretical concepts that underlay our models (Chapter 2). Then we give an overview of the related work (Chapter 3). Next, we present a work on 2D semantic segmentation (Chapter 4), followed by a work addressing the anticipation of semantic labels outside the field of view of a 2D image (Chapter 5). Afterwards, we change our focus to 3D where we first propose a new approach to 3D semantic scene completion based on a multi-modal input (Chapter 6) and secondly address the potential of generative adversarial networks in the context of 3D scene completion (Chapter 7). Finally we introduce a new dataset that is the first large scale realistic dataset for 3D semantic scene completion and apart from that is suitable for many other tasks, like semantic segmentation on 3D point clouds (Chapter 8).

CHAPTER 2

Preliminaries

We will now describe the most important models that our works are based on. We begin by discussing random forests, a classifier, which is still of interest today in the context of devices with limited hardware. They allow for quick inference times even without parallel computing hardware like GPUs. Afterwards we focus on the major building block of nearly all computer vision related fields today: Convolutional neural networks (CNNs). Their dominance has only been established within the lifetime of this thesis and is still expanding. Another concept that is based on CNNs are generative adversarial networks (GANs), which also quickly gained popularity from their inception in 2014. One of our works explores the potential of GANs for the task of semantic scene completion. Finally we discuss the conditional random field (CRF), which is a graphical model defined on pixel class probabilities and is frequently used as a post processing step to increase segmentation accuracy throughout this thesis.

Contents

2.1 Random Forest

A random forest is a classifier consisting of an ensemble of binary decision trees. Though each individual tree by itself has a weak classification performance, the combination of several independently trained trees leads to a strong classifier. Random forests leverage the concept of stochastic discrimination to prevent overfitting to a specific training set. We now describe the concept and training procedure of random forests.

Criminisi and Shotton (2013) proposed a random forest for the task of semantic image segmentation. In their model each tree infers for an image pixel $\mathbf{x}$ the class probability $p(c|\mathbf{x};\theta_t)$ where c is a semantic class and θ_t are the parameters of the tree t. The parameters θ_t are learned in a suboptimal

fashion by sampling from both the training data and the parameter space Θ. A robust estimator is then obtained by averaging the predictors

$$p(c|\mathbf{x}) = \frac{1}{T}\sum_t p(c|\mathbf{x};\theta_t), \tag{2.1}$$

where T is the number of trees in the forest. A segmentation of an image can then be obtained by taking the class with highest probability for each pixel.

To learn the parameters θ_t of a tree t, first, pixels from the training data are sampled which provide a set of training pairs $\mathcal{S} = \{(\mathbf{x}, c)\}$. The tree is then constructed recursively, where at each node n a weak classifier is learned by maximizing the information gain

$$\theta_n = \arg\max_{\theta\in\tilde{\Theta}} \left\{ H(\mathcal{S}_n) - \sum_{i\in\{0,1\}} \frac{|\mathcal{S}_{n,i}|}{|\mathcal{S}_n|} H(\mathcal{S}_{n,i}) \right\}. \tag{2.2}$$

While $\mathcal{S}_n$ denotes the training data arriving at the node n, $\tilde{\Theta}$ denotes the set of sampled parameters and

$$H(\mathcal{S}) = -\sum_c p(c;\mathcal{S}) \log p(c;\mathcal{S})$$

where $p(c;\mathcal{S})$ is the empirical class distribution in the set $\mathcal{S}$. Each node becomes a weak classifier $f_\theta(\mathbf{x})$ with parameters θ that splits $\mathcal{S}_n$ into the two sets

$$\mathcal{S}_{n,i} = \{(\mathbf{x}, c) \in \mathcal{S}_n : f_\theta(\mathbf{x}) = i\}$$

with $i \in \{0, 1\}$. After the best weak classifier θ_n is determined, $\mathcal{S}_{n,0}$ and $\mathcal{S}_{n,1}$ is forwarded to the left or right child, respectively. The growing of the tree is terminated when a node becomes pure, meaning it does only contain samples of a single class, or $\mathcal{S}_n$ becomes smaller than some threshold. Finally, the empirical class distribution $p(c;\mathcal{S}_l)$ is stored at each leaf node l. During inference, one can compute $p(c|\mathbf{x};\theta_t)$ for every pixel $\mathbf{x}$ and tree t and infer the final class probability distribution $p(c|\mathbf{x})$ using Equation 2.1.

2.2 Convolutional Neural Network (CNN)

A major building block of nearly all our models are convolutional neural networks (CNNs). We will briefly describe their history and neuroscientific motivation before explaining the architecture and functionality of a typical modern CNN.

2.2.1 History

Frank Rosenblatt, a psychologist and computer scientist, in 1958 published the 'perceptron' model consisting of a single layer of neurons, randomly connected to the input (*Rosenblatt*, 1958). The perceptron was implemented as a hard-wired machine without software using as input a grid of 20×20 photocells which were randomly wired to neurons while using electric motors for updating weights during learning. Nevertheless since the perceptron model consisted of a binary linear classifier it is considered the basis of today's artificial neural networks.

The perceptron then developed into a multilayer perceptron (MLP) consisting of multiple layers of perceptrons and capable of learning non-linear functions. *Rumelhart et al.* (1986) proposed the backpropagation algorithm for training parameters of a multilayer-perceptron. *LeCun et al.* (1989) applied backpropagation to a neural network recognizing digits in 2D images.

LeCun et al. (1998) introduced a convolutional neural network architecture, whose the basic functionality is similar to that of today's architectures: The input was processed by a sequence operations consisting of convolutions, followed by the addition of a bias term, followed by a nonlinear transformation (e.g. hyperbolic tangent or logistic function). The shift from sigmoid activation function to ReLU came due to the observation that sigmoid activations tend to saturate quickly and thus do not contribute to the learning anymore.

Though the field existed for a long time, high expectations and the consequent failure to achieve them, repeatedly led to phases of reduced interest into neural network based methods (phases called 'AI winters'). The interest in the field was completely revived when Krizhevsky et al in 2012 famously won the 2012 ImageNet challenge by a large margin. This success was mainly driven by two effects: The shift towards GPU based parallel processing and the availability of large scale datasets. This can be seen from the fact that the architecture of the Krizhevsky's CNN strongly resembles LeNet-5 by *LeCun et al.* (1998), apart from being significantly deeper.

2.2.2 Neuroscientific Background

The idea for the construction of a neural network was inspired by the study of the functionality of the human brain and perception. It is an attempt to emulate the process of visual perception by an algorithm. We briefly outline the concepts that influenced the conception of CNNs.

The human eye projects a 2D image of the outside world through a lens onto the retina. The retina consists of nerve cells which either register brightness or color. From the retina, the image is transferred via the optic nerve and the lateral geniculate nucleus (LGN) to the primary visual cortex (V1). Researchers have found that neurons in V1 perform operations resembling convolutions. V1 contains 'simple cells' which behave linearly with respect to a local neighborhood of the incoming receptive field and 'complex cells', which are invariant to spatial transformations of the input signal. V1 also shows a 2D spatial structure like the retina, *i.e.* light shined onto different parts of the retina only activate specific regions in V1.

2.2.3 Architectures

CNNs consist of a repetitive pattern of functional blocks. Each functional block processes an input by a sequence of operations to produce an output layer. The output layer is subsequently fed to another functional block. Figure 2.1 gives an overview over some historically influential CNN architectures. The most prominent architectural development, is that CNNs have become deeper over time. While LeNet (1980) only has two convolutional layers, AlexNet (2012) has 5, followed by VGG (2014) with 19 and ResNet (2016) with up to 100. As the networks become deeper, training becomes harder since the problem of 'vanishing gradients' increases.

Some design choices of the presented models are designate as useful and used in most modern architectures. AlexNet relied on Dropout to prevent overfitting, especially in the fully connected layers. VGG, for example, restricted the spatial size of every convolution kernel to 3×3, which became the norm ever since. ResNet on the other hand introduced a new kind of functional block, the

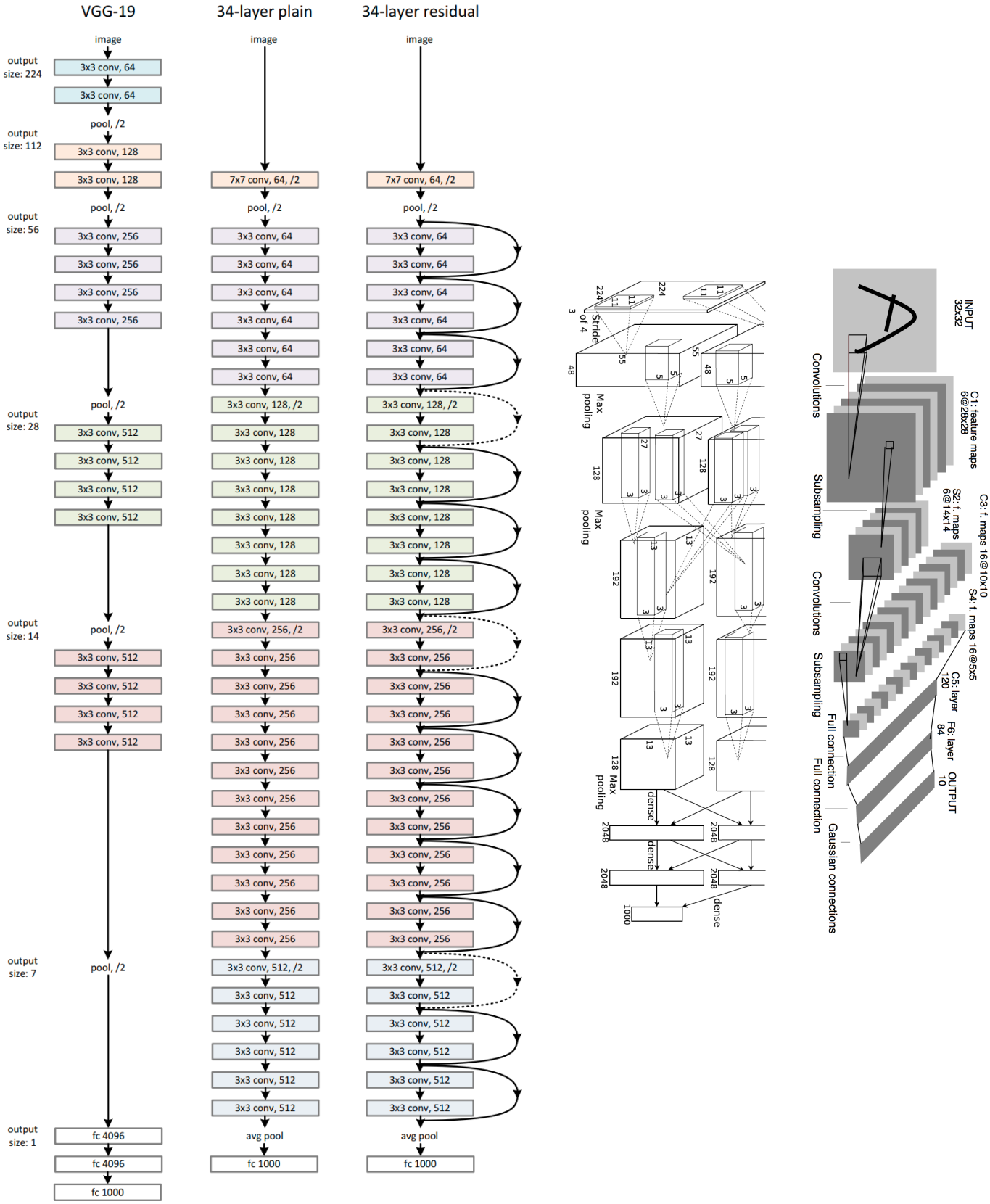

Figure 2.1: Milestones of CNN-architecture development. Top: LeNet-5 (*LeCun et al.*, 1998), Middle: AlexNet (*Krizhevsky et al.*, 2012). Bottom: Comparison of VGG-19 *Simonyan and Zisserman* (2015) and ResNet (*He et al.*, 2016).

'skip-connections'. AlexNet, VGG and ResNet all proved their efficiency by winning the ImageNet challenge (ILSVRC). Afterwards they were quickly adapted and applied in many related fields of computer vision.

For semantic segmentation the most important modification is the replacement of all fully connected layers by a convolutional layer, as first proposed by *Long et al.* (2015). Thereby the last layer becomes a pixel-wise classifier. These fully convolutional neural networks are extremely handy, as they are able to consume inputs and produce outputs of any size. In the following we will explain the behaviour of the individual components of CNNs as well as the training procedure.

2.2.4 Components

Convolutions

Convolutions are at the basis of the success of neural network based learning and image analysis. In contrast to fully connected layers they offer a significant reduction in the number of parameters per layer through weight sharing, *i.e.* that the same kernel is applied to multiple regions of the input feature. The convolution of a two-dimensional image I by a kernel K is defined by

$$S(i,j) = (K * I)(i,j) = \sum_m \sum_n I(i-m, j-n)K(m,n). \tag{2.3}$$

In current CNNs the actual implementation of a convolution slightly deviates from this mathematical definition. First, to account for multiple channels of the input image or feature map, another dimension is added to the kernel. Secondly, in contrast to Equation 2.3, the kernel K is flipped, which technically changes the convolution into a correlation, which is however irrelevant for the functionality of the CNN since the parameters of K are initialized randomly (*Goodfellow et al.*, 2016). More specifically, the computation of an element $Z_{i,j,k}$ of an output tensor $\mathbf{Z}$ from an input tensor $\mathbf{V}$ with entries $V_{i,j,k}$ where the indices i, j, k denote the channel, row and column dimension respectively, is performed as follows: $Z_{i,j,k}$ is obtained by applying a convolution kernel K, then adding a bias b followed by the transformation by some non-linear function σ

$$Z_{i,j,k} = \sigma\Big(\sum_{l,m,n} V_{l,j+m,k+n} K_{i,l,m,n} + b_i\Big). \tag{2.4}$$

where the non-linear function σ is typically the 'rectified linear unit' (ReLU) introduced by *Glorot et al.* (2011)

$$f(x) = \max(0, x). \tag{2.5}$$

Pooling

A pooling layer replaces every neuron of an input feature map by a summary statistic of its local neighbourhood. A typical example is a 2×2 max-pooling operation with stride 2. Thereby only the maximum value of a rectangular 2×2 region is copied to the output feature map, which is downsampled by a factor of 2 with respect to the input feature map.

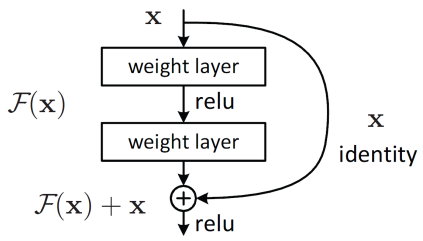

Figure 2.2: Skip-Connection mechanism as proposed by (*He et al.*, 2016).

The pooling operation has several motivations: Apart from reducing the memory consumption of subsequent layers, it also leads to an increase of the receptive field of subsequent convolutions. Moreover by integrating over the local neighbourhood information, the network becomes more robust to small translations of the input features.

Skip-Connection

A skip-connection allows information to flow unmodified from earlier layers to later layers in the network. Figure 2.2 illustrates the mechanism. The input to a convolutional block is forwarded bypassing the block and added to its output layer. *He et al.* (2016) could show significant performance improvements when adding the skip-connections to their CNN architecture. During the forward pass the skip-connections allow deeper layers to use information not only from previous but also from earlier layers. During the backward pass, the skip-connection allows gradients to flow unaltered to early layers, which helps to reduce the problem of vanishing gradients.

Batch-Normalization

A batch-normalization layer transforms every neuron to have approximately 0-mean μ and a variance σ of 1, with respect to the training data. To achieve this a moving average of both values is computed along the batch-dimension during the network training. During inference μ and σ are kept fix.

Batch-normalization has been introduced by *Ioffe and Szegedy* (2015). They showed that batch-normalization greatly speeds up the training time while at the same time making the training more robust towards the choice of hyperparameters like the learning rate or the initial values of the model parameters. However, batch-normalization can only be applied in cases, where the batch-size is larger than 1 and performs poorly for small batch-sizes.

Dilated Convolutions

Dilated convolutions is a concept that is quite frequently used in the domain of semantic segmentation. They are convolutions with an increased kernel size but constant number of parameters. Therefore the learnable parameters are evenly placed on a regular grid and the non-occupied cells are filled up with zeros. These cells remain zero throughout the training. As one can see from Figure 2.3 this has the effect of increasing the receptive field without the need for decreasing the image resolution as in the case of pooling layers.

A strided convolution with stride s, a kernel $\mathbf{K}$ with elements $K_{i,j,k,l}$ on an input $\mathbf{V}$ produces an an element of the output tensor $Z_{i,j,k}$ computed as follows

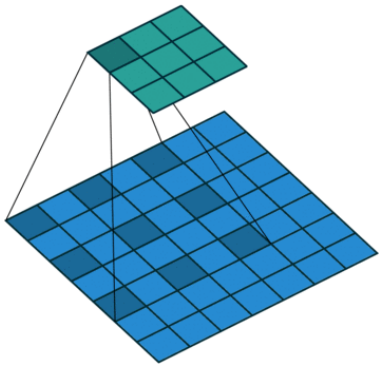

Figure 2.3: Dilated convolution proposed by *Chen et al.* (2015). Input resolution is 7×7 (bottom) and the output resolution 3×3 (top). The output resolution can be increased to the same level as the input by adding a 0 padding to the input layer of half the kernel size. [a]

[a] Source: https://towardsdatascience.com/types-of-convolutions-in-deep-learning-717013397f4d

$$Z_{i,j,k} = \sum_{l,m,n} [V_{l,(j-1)\times s+m,(k-1)\times s+n} K_{i,l,m,n}]. \tag{2.6}$$

Deconvolutions

A frequently used method to increase the resolution of feature maps are deconvolutions. In contrast to convolutions they take as input a single neuron from the previous feature map, scalar multiply it with a convolutional kernel and add the result in the subsequent feature map. The process is repeated for all neurons in the input feature map and the results in the output feature map are aggregated via addition. Figure 2.4 shows an example of the process. There is an equivalent perspective of the same operation, which is what is actually implemented in most libraries. The aggregation of differently weighted kernels in the output layer is equivalent to a strided convolution, where the input feature map is first upsampled by placing the input on a higher resolution grid and setting all of the newly introduced cells to 0 and furthermore adding a 0-padding of half the kernel size at every border. The input is then convolved with the flipped version of the convolution kernel, hence the alternative name 'transpose-convolution'. Transpose convolutions and deconvolutions have different implementations, but yield identical results. Therefore, their names are used exchangeable.

2.2.5 Training

The training of a CNN is modelled as an optimization problem. The goal is to find the parameters θ of a neural network that minimize a cost function $J(\theta)$, which measures the performance of the network on a training set of size N

$$J(\theta) = \sum_{i=1}^{N} \mathcal{L}_i(x, y). \tag{2.7}$$

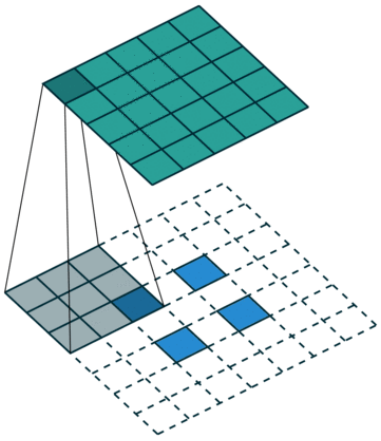

Figure 2.4: Visualization of a 2×2 strided deconvolution with kernel size 3×3, input resolution is 2×2 (bottom), input resolution is 5×5 (top). In order to perform the deconvolution an automatic 0-padding is being added to the input, depending on the kernel size k, the size of the padding is $(k+1)/2$. [a]

[a] Source: https://towardsdatascience.com/types-of-convolutions-in-deep-learning-717013397f4d

where $\mathcal{L}_i(x, y)$ is the loss function evaluated for a training sample i consisting of an input x and a label y. The loss function measures the distance of the output prediction of the neural network to the label in a task specific metric. For classification tasks like semantic segmentation one typically uses a pixel-wise softmax cross-entropy loss

$$\mathcal{L}_i = -\sum_{c \in \mathcal{C}} \hat{y}_{ic} \log \left(\frac{e^{y_{ic}}}{\sum\limits_{c' \in \mathcal{C}} e^{y_{ic'}}} \right) \tag{2.8}$$

where $\hat{y}_{ic}$ is the class probability of the ground truth label of pixel i, which is 1 for the true class and 0 otherwise. y_{ic} denotes the unnormalized predictions of the network for pixel i and class c. In order to tune the initially randomly chosen parameters of the CNN, one applies backpropagation. During backpropagation one computes the gradient of every parameter in the network with respect to the objective function J. Starting from the last layer f_n, one can get the gradient of all earlier layers f_i by successive application of the chain rule, which for simplicity we show here for the one-dimensional case

$$J(x) = f_n \circ \cdots \circ f_1(x) \tag{2.9}$$

$$\frac{\partial J(x)}{\partial f_i} = \frac{\partial f_n}{\partial f_{n-1}} \cdot \frac{\partial f_{n-1}}{\partial f_{n-2}} \cdot \ldots \cdot \frac{\partial f_{i+1}}{\partial f_i}. \tag{2.10}$$

With the help of the gradient with respect to the weights $\nabla_W L$ on can update the trainable weights such that they minimize the cost function J, for example via stochastic gradient descent (SGD):

$$W = W - \lambda \nabla_W L \tag{2.11}$$

Stochastic refers to the fact that L is not computed for the entire training data at once but instead for a small randomly sampled batch of data, before updating the weights. In practice one uses more

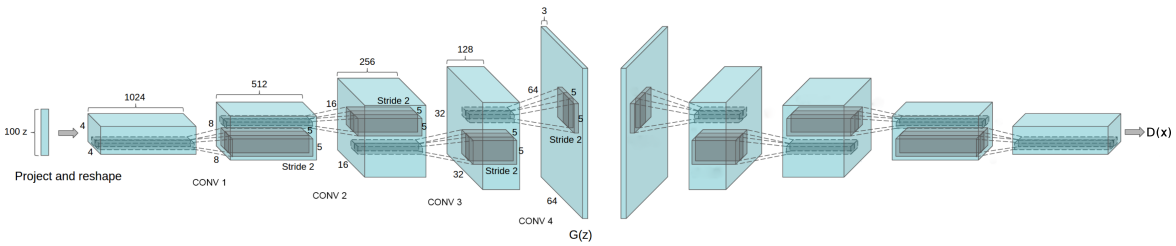

Figure 2.5: Architecture of the DCGAN proposed by (*Radford et al.*, 2015). Both the generator and the discriminator are fully convolutional neural networks. While the generators generates images from a randomly sampled input vector, the discriminator classifies whether the input image he got was real (sampled from the ground truth) or fake (generated by the generator). Image adapted from *Radford et al.* (2015).

refined variants of SGD such as SGD with momentum or ADAM which gather some additional statistics of the gradients in order to accelerate the training. The learning rate λ is one of the most crucial parameters that has to be carefully chosen before every experiment.

2.3 Generative Adversarial Network

A generative adversarial network (GAN) is a generative model that can for example be trained to produce naturally looking images sampled from a random vector. These images resemble but are not identical to those of a training dataset. The training of a GAN has many interesting properties. It learns a latent space, that is a compressed representation of an image. The training is done in an self-supervised way, meaning that no human annotation is needed.

The GAN proposed by *Radford et al.* (2015) consisted of two convolutional neural networks, a generator and a discriminator. The structure is visualized in Figure 2.5. The generator takes as input a random vector z and outputs an image $x = g(z)$. The discriminator gets as input either an image from the training data or from the generator. It then outputs a probability $d(x)$ to assess whether x was 'real', *i.e.* drawn from the training data, or 'fake', *i.e.* it was generated by the generator. Both networks are trained simultaneously thereby competing with each other. The training objective is formulated as a minimax game. While the generator tries to minimize a value function v, it learns to generate images that look similar to those of the training data. On the other hand, the discriminator tries to maximize the value function v by getting better at discriminating between real and fake samples. At convergence both the generator and discriminator will have learned parameters θ to reduce their respective loss, which can be formulated as the following optimization problem

$$\min_{g} \max_{D} v(\theta_g, \theta_d). \tag{2.12}$$

with a value function v defined over the log-likelihood of the discriminator output

$$v(\theta_g, \theta_d) = \mathbb{E}_{\mathbf{x} \sim P_{data}} \log d(\mathbf{x}) + \mathbb{E}_{\mathbf{x} \sim P_{model}} \log(1 - d(\mathbf{x})). \tag{2.13}$$

In practice *Goodfellow et al.* (2014) observed that in the early learning phase, when the generator produces very unrealistic images, the discriminator rejects them with high confidence which means that the term $\log(1 - d(\mathbf{x}))$ saturates. Therefore, they instead propose an objective function which

is modified such that the generator tries to increase the log probability that the discriminator makes a mistake, instead of trying to decrease the log probability that the discriminator makes the correct prediction:

$$v(\theta_g, \theta_d) = \mathbb{E}_{\mathbf{x}\sim P_{data}} \log d(\mathbf{x}) - \mathbb{E}_{\mathbf{x}\sim P_{model}} \log d(\mathbf{x}). \tag{2.14}$$

This trick ensures a stronger gradient signal in the early learning phase and the objective function still results in the same fixed point.

2.4 Conditional Random Field

One problem of current CNN architectures deployed for semantic segmentation is the loss for resolution with respect to the input. This is both due to the use of pooling operations in order to increase the receptive field (which could be avoided using dilated convolutions) and secondly due to the need for a reduction of the memory footprint of deep architectures. The results are often upsampled using bilinear interpolation in order to obtain the desired output resolution. The effect of this is that especially sharp boarders of objects in an image are poorly detected. One way to resolve this issue is a conditional random field (CRF). Thereby the pixels of an image are interpreted as the vertices of a fully connected graph. Using the difference of RGB color values of two pixels and their respective probability of being assigned to the same or different class categories, one can refine semantic segmentation maps to better align with object boundaries in the original resolution of the image.

We will now briefly discuss the fully connected CRF of *Krähenbühl and Koltun* (2011) that is used regularly throughout this thesis in order to refine 2D semantic segmentation results.

A conditional random field is a graphical model characterized by a probability distribution P over a pixel-wise semantic segmentation map $\mathbf{y}$ given an image I. This probability distribution is given as a Gibbs distribution

$$P(\mathbf{y}|I) = \frac{1}{Z(I)} \exp\Big(- \sum_{c\in C_{\mathcal{G}}} \Phi_c(\mathbf{y}|I)\Big) \tag{2.15}$$

where $\mathcal{G}(\mathcal{V}, \mathcal{E})$ is a graph on $\mathbf{y}$, c is a clique in the set of cliques $C_{\mathcal{G}}$ and Φ_c is a potential belonging to that clique. For a labeling $\mathbf{y} \in \mathcal{L}^N$, where $\mathcal{L}^N$ is the space of all possible labelings for an image with N pixels, one can compute a Gibbs energy

$$E(\mathbf{y}|I) = \sum_{c\in C_{\mathcal{G}}} \Phi_c(\mathbf{y}_c|I) \tag{2.16}$$

Given the Gibbs energy one can directly derive the maximum a posteriori label assignment for the conditional random field, which corresponds to most likely semantic segmentation map:

$$\mathbf{y}^* = \underset{\mathbf{y}\in\mathcal{L}^N}{\arg\max}\, P(\mathbf{y}|I) \tag{2.17}$$

In the implementation of *Krähenbühl and Koltun* (2011) the conditional random field is a fully connected pairwise graph over $\mathbf{y}$. It has a Gibbs energy consisting of unary potentials Ψ_u and binary potentials Ψ_p defined at each pixel

$$E(y) = \sum_{i}^{N} \Psi_u(y_i) + \sum_{i<j} \Psi_p(y_i, y_j) \tag{2.18}$$

As unary potential they take the probability assigned to each pixel by a classifier (in our case the softmax-probability assigned by a CNN) as pairwise potential they propose a contrast-sensitive two-kernel potential

$$\Psi_p(y_i, y_j) = \mu(y_i, y_j) \sum_{m=1}^{K} w^m(f_i, f_j) \tag{2.19}$$

with

$$k(f_i, f_j) = w^1 \exp\Big(- \frac{|p_i - p_j|^2}{2\theta_\alpha^2} - \frac{|I_i - I_j|^2}{2\theta_\beta^2} \Big) + w^2 \exp\Big(- \frac{|p_i - p_j|^2}{2\theta_\gamma^2} \Big) \tag{2.20}$$

where I_i, I_j denote pixel intensities, and p_i, p_j are the spatial coordinates of pixel i and j. One can see that the kernel measures the similarity of pixels $|I_i - I_j|$ as well as the spatial distance $|p_i - p_j|$. The parameters θ_α and θ_β control the degree of similarity and distance that is enforced between pixels. They are found through grid search as are the other parameters w^1, w^2 and θ_γ. The factor $\mu(y_i, y_j)$ in Equation 2.19 is the label compatibility function and is in our case given by the Pott's model

$$\mu(y_i, y_j) = \begin{cases} 1 & \text{if } y_i \neq y_j \\ 0 & \text{otherwise} \end{cases} \tag{2.21}$$

The choice of the Pott's model as compatibility function enforces that nearby pixels with the same labeling but different appearance are penalized. The goal is now to find the best label assignment by iteratively minimizing the energy E in Equation 2.18, by updating the potentials at every pixel via message passing.

Therefore the implementation by *Krähenbühl and Koltun* (2011) makes the mean field approximation which stipulates that an individual pixel sees a superposition of all the binary potentials of itself with respect the other pixels. These new potentials are computed for every pixel first, before they are all updated simultaneously.

Furthermore due to the choice of Gaussian edge potentials the message passing step can be implemented very efficiently using convolutions by a truncated Gaussian. To do so they first downsample the feature space using a sparsely stored permuhedral lattice. The Gaussian convolution filter can be linearly separated into one-dimensional filters and be applied along the lattice dimensions of the permuhedral lattice. This leads to inference times of fractions of a second for images with millions of pixels using a single threaded device.

CHAPTER 3

Related Work

We review the body of research that influenced our work. Beginning with a selection of classical works addressing the task of semantic segmentation in images, we continue by exploring what methods deal with the anticipation of semantic categories on the basis of 2D images. After that we move to the 3D domain where we deal with semantic segmentation of point clouds on the one hand as well as semantic scene completion on the other.

Contents

3.1 Semantic Segmentation

The field of semantic segmentation has been addressed in countless works over the past two decades and has been heavily shifted due to the transformation caused by deep learning which also impacted the research trajectory of this thesis. While our first work is a classical approach to semantic segmentation, all the consecutive works use deep learning at the core of every model. We will therefore resume the development of 2D semantic segmentation in two parts. First we give an overview over classical approaches, then we report on recent approaches based on deep learning.

3.1.1 Classical Approaches

A classical approach to semantic segmentation usually consists of a pipeline of several independent components. Each component can be realized in several ways, which leads to a variety of possible combinations between them which is reflected in various works. Each pipeline typically consists of a bottom-up approach to compute coherent image regions or superpixels, hand-crafted feature descriptors to describe individual pixels, a classifier to attribute a certain class label to each feature descriptor and finally a postprocessing step to smoothen noisy semantic segmentation results.

The superpixel computation can be performed by methods based on graph based image segmentation (*Shi and Malik*, 2000; *Felzenszwalb and Huttenlocher*, 2004) or gradient ascent based methods like mean shift (*Comaniciu and Meer*, 2002) or watersheds (*Vincent and Soille*, 1991). Modern methods like SLIC (*Achanta et al.*, 2012), SEEDS (*Van den Bergh et al.*, 2012) rely on methods like k-means or hill-climbing, which can also be combined with heuristics for an adaptive seeding to improve the usually random initialization of the superpixel clustering (*Wilms and Frintrop*, 2017).

As features one can choose between a variety of different hand-engineered descriptors like histograms of oriented gradients (HOG) (*Dalal and Triggs*, 2005), SIFT (*Lowe*, 2004), which are based on orientations found in gradient images or texton (*Leung and Malik*, 2001) which are based on color statistics.

As classifier, one can choose between non-parametric models like k-nearest neighbours or trained classifiers like boosting (*Freund and Schapire*, 1996), a support vector machine (SVM) or random forests (*Breiman*, 2001). Apart from applying the classifier directly to every pixel described by its feature descriptor, one could also aggregate feature vectors over a certain region (*e.g.* a superpixel) and classify the entire region at once. The aggregation can range from a simple max-, sum- or average-pooling to more refined approaches like the bag-of-features (BoF) embedding. Inspired by the bag-of-word (BoW) scheme applied earlier in speech recognition. The concept relies on quantizing features using a codebook. Therefore one first extracts all feature descriptors from the training set, then performs a clustering on the descriptors (*e.g.* k-means clustering) and uses the obtained cluster-centers as codebook to encode an arbitrary number of feature descriptors. Thereby one obtains a normalized frequency histogram, counting how many descriptors belonging to the respective cluster-center can be found in the pooled region. BoF approach is a very effective approach to aggregate feature descriptors and compare regions of varying size and shape.

Finally the resulting segmentation map can be post-processed to refine a noisy classifier result. This can be achieved by enforcing local consistency within a superpixel or across neighbouring superpixels using Markov random fields (MRFs) or conditional random fields (CRFs). This last step is still relevant in today's approaches as deep learning approaches traditionally suffer from low output resolutions and therefore need to be upsampled.

We will now report a selection of classical approaches that were influential to this thesis and the field of semantic segmentation. *Shotton et al.* (2008) and *Brostow et al.* (2008) proposed classical approaches to semantic segmentation using classifiers like random forests or boosting. *Sturgess et al.* (2009) and *Ladickỳ et al.* (2010) deployed CRFs to model the spatial relations of pixels and obtain a smooth segmentation. *Kontschieder et al.* (2011) proposed a structured random forest that predicts not a single label per pixel but the labels of the entire neighborhood. Merging the predicted neighborhoods into a single semantic segmentation of an image, however, is costly. To speed up the segmentation, learning and prediction of random forests, they have been implemented for GPUs (*Schulz et al.*, 2015). Approaches that combine random forests and neural networks have been proposed as well (*Bulo and Kontschieder*, 2014), however, at the cost of increasing the runtime.

3.1.2 Deep Learning Approaches

In 2012 convolutional neural networks have revolutionized the domain of image classification. The network architecture used in 2012, called 'AlexNet' (*Krizhevsky et al.*) was then continuously improved by subsequent models, for example by VGG (*Simonyan and Zisserman*, 2015) and ResNet

(*He et al.*, 2016).

Similarly the field of semantic segmentation transformed towards end-to-end learnable approaches. *Long et al.* (2015) proposed fully convolutional neural network for semantic image segmentation. The architecture of fully convolutional neural networks for image segmentation was adapted in various ways in the following years. Some approaches leverage an encoder-decoder network architecture (*Pohlen et al.*, 2017; *Lin et al.*, 2017). *Ronneberger et al.* (2015), for example, concatenate the output of the encoder layers with the corresponding input feature layers in the decoder while *Badrinarayanan et al.* (2017) use the max-pooling indices stored in the encoder during forward propagation to upsample the feature maps in the decoder. Other approaches focus on context aggregation either through a pyramid pooling strategy (*Zhao et al.*, 2017) or via dilated convolutions (*Chen et al.*, 2015, 2018; *Wang et al.*, 2018a). For real-time applications like autonomous driving, highly efficient approaches have been proposed (*Paszke et al.*, 2016; *Zhao et al.*, 2018a). Newer approaches try to transfer the 'attention mechanism', which became particularly famous in machine translation community (*Vaswani et al.*, 2017)[1], to the task of semantic segmentation (*Zhao et al.*, 2018b; *Yu et al.*, 2018).

In this thesis, we frequently use the approach by *Chen et al.* (2015) as basis for our 2D semantic image segmentation. It is based on the ResNet architecture and uses a couple of adaptations to make it suitable for semantic image segmentation. The main adaptation is the introduction of dilated-convolutions, which are convolutions with increased kernel sizes but with the same amount of parameters. By varying the kernel size of the dilated-convolution one can compute responses from the feature maps at different spatial resolutions which allows to control size of the receptive field. In addition, we usually refine predictions by a conditional random field (*Krähenbühl and Koltun*, 2011).

3.2 Anticipation of Semantic Categories

The task of spatial anticipation of semantic categories, which we address in this thesis, has not been previously studied. Hallucinating semantics has been addressed in very few works. *Liu et al.* (2016) developed an approach to reconstruct 3D scenes by simultaneously predicting depth and semantic labels from incomplete depth data. They propose a two-layer model representing both the visible and the hidden or occluded information. The approach also has some relations to in-painting methods like *Pathak et al.* (2016a), which fill holes inside an image. In-painting methods, however, cannot be applied to anticipate semantics outside an image. For the task of proposal generation for an object detector, *Ristin et al.* (2013) predict from large image patches potential bounding boxes that might contain objects of a relevant object category. While the bounding boxes can be outside the image patch, the approach aims at exploiting the local context within an image to reduce the inference time of an object detector. Moreover our work is connected to those focused on exploiting context for object detection. *Torralba* (2003) made a seminal contribution in this domain. There has been some works that aim for temporal anticipation. *Luc et al.* (2017, 2018), for example, deploy an autoregressive convolutional neural network to iteratively predict semantic segmentations and instances of future frames.

[1] Interestingly their approach does not rely on recurrent neural networks anymore, unlike typical approaches to machine translation. This might indicate a paradigm shift.

3.3 Semantic Scene Completion

Several works addressed the problem of semantic segmentation of RGB-D images (*e.g. Ren et al.*, 2012; *Lai et al.*, 2014; *Gupta et al.*, 2013; *Silberman et al.*, 2012), but they infer semantic labels only for the visible pixels of the image, which means that occluded voxels are not reconstructed.

A possible strategy for semantic scene completion is the generation of 3D object proposals and subsequent 3D shape completion of the respective object. The problem of completing the 3D shape has been addressed in several works (*Rock et al.*, 2015; *Thanh Nguyen et al.*, 2016; *Varley et al.*, 2017; *Wu et al.*, 2015; *Wang et al.*, 2017; *Yang et al.*, 2017; *Han et al.*, 2017; *Dai et al.*, 2017b). In the context of inferring a voxel-wise segmentation, holes between objects have to be filled. As long as these missing parts are small, they can be filled using plane fitting (*Monszpart et al.*, 2015) or object symmetry (*Kim et al.*, 2012; *Mattausch et al.*, 2014). Objects that are not detected, however, heavily disturb the 3D scene completion. Completing the scene geometry without predicting the semantics has been addressed by *Firman et al.* (2016). Their model assumes that objects of semantically dissimilar classes can still be represented by similar 3D shapes, *i.e.* it is possible to predict the unobserved voxels from the frontal geometry. However, this approach fails for complex scenes where these geometric constraints are violated.

An alternative is to use instances of 3D mesh models and fit them to the scene geometry (*Geiger and Wang*, 2015; *Gupta et al.*, 2015; *Kim et al.*, 2012; *Lai and Fox*, 2010; *Mattausch et al.*, 2014; *Nan et al.*, 2012; *Song and Xiao*, 2016; *Shao et al.*, 2012; *Li et al.*, 2015; *Shi et al.*, 2016). The mesh models, however, do not model shape variations of objects of the same category and increasing the number of mesh models per category is not practical since the number of available CAD models is limited and the shape retrieval becomes more expensive. The approaches by *Jiang and Xiao* (2013) and *Lin et al.* (2013) even neglect fine-grained details and simply fit 3D primitives to the scene.

Various contextual cues proved to be helpful for semantic scene completion. While physical reasoning is employed in *Zheng et al.* (2016), *Kim et al.* (2013) predict voxel labels with a CRF whose unary potentials are determined by floor plans. The CRF, however, only models contextual information within a short distance. In *Blaha et al.* (2016) and *Häne et al.* (2013), semantic scene completion and multi-view reconstruction are jointly performed. The approaches do not rely on end-to-end learning approaches, but they use predefined features and heuristics to integrate context information.

The seminal approach in 3D semantic scene completion using an end-to-end approach, which motivated our own research, was proposed by *Song et al.* (2017). They also proposed the SUNCG dataset, a large scale synthetic dataset for semantic scene completion based on rendered depth images. Moreover they generate another training set for the NYUv2, which consists of realistic indoor scenes, using CAD models provided by *Rock et al.* (2015). However the realistic NYUv2 datasets only consists of 795 training samples which are badly aligned with the cluttered RGB-D images of NYUv2. This is a strong limitation for end-to-end learning approaches.

In the following, many different works have addressed the problem of semantic scene completion using CNN based approaches. *Zhang et al.* (2018) proposed a new network architecture which leverages sparse feature map encodings and allows for much deeper network architectures and *Dai et al.* (2018) propose a three-stage multi-resolution, autoregressive model. Parallel to our work, approaches that follow related ideas have been released. *Liu et al.* (2018) also try to leverage a multimodal input consisting of RGB and depth images for semantic scene completion, *Wang et al.* (2018c) propose a

model based on an adversarial training scheme.

Although *Dai et al.* (2017a) and *Chang et al.* (2017) collected large scale, semantically annotated datasets with real world data, they are not suitable for training semantic scene completion models. Instead, *Chang et al.* focus on surface normal estimation while *Dai et al.* focus on tasks like 3D object classification, voxel labeling and CAD model retrieval. Therefore there is still a need for large, realistic datasets for semantic scene completion, which we address in this thesis.

3.4 Generative Adversarial Network

In 2014 *Goodfellow et al.* introduced the concept of generative adversarial networks (GANs). They proposed an architecture consisting of 2 convolutional neural networks, a generator and a discriminator. Their model was able to generate images that looked distinct yet still resembling those of a training set. The idea quickly took off, not least since Yan LeCun, in an interview, called it 'the most interesting idea in the last 10 years'[2] - thereby carelessly trashing the successes of convolutional neural networks in image classification two years earlier. The result was a plethora of works that tried to extend the idea to almost all fields in computer vision and machine learning.

One year after the GAN concept was proposed, *Radford et al.* (2015) introduced a deep convolutional generative adversarial architecture (DCGAN) which became the standard architecture for many of the subsequent works on GANs. The DCGAN architecture strongly resembles that of a autoencoder, where the generator has the form a decoder and the discriminator that of an encoder.

In the following years, researchers were trying to either improve the notoriously difficult training, increase the output resolution of the generator (*Karras et al.*, 2017; *Shrivastava et al.*, 2017), extend the model architecture or adapt it to a plethora of new tasks. Some influential conceptual changes have been proposed by *Dosovitskiy et al.* (2015), who extended the GAN approach by using conditioning variables and thereby making the image generation process more controllable.

Other works focussing on controlling the output of the generator involved conditioning on discrete labels (*Gauthier*, 2014; *Denton et al.*, 2015; *Mirza and Osindero*, 2014), conditioning on images for tasks such as image generation from a normal map (*Wang and Gupta*, 2016) or image manipulation guided by user constraints (*Zhu et al.*, 2016).

Another new concept was the introduction of a new loss function based on the Wasserstein metric by *Arjovsky et al.* (2017), which makes the training more robust towards the 'mode collapse' problem as well as to the choice of the model parameters. However training GANs is still notoriously unstable, which is why many people stick to architectures resembling that of *Radford et al.* (2015), while following a collection of shared training heuristics (*Karras et al.*, 2017; *Salimans et al.*, 2016; *Roth et al.*, 2017).

Apart from generating images, GANs were applied in many different computer vision domains such as style transfer (*Li and Wand*, 2016), superresolution (*Ledig et al.*, 2017), inpainting (*Pathak et al.*, 2016b), and future state prediction (*Zhou and Berg*, 2016) to name a few. *Luc et al.* (2016) were the first to apply GANs to 2D semantic segmentation by adding an adversarial network after a segmentation network to discriminate segmentation maps coming from either the ground truth or the segmentation network. Some other works applied GANs to the 3D space, focussing on either single object reconstruction (*Wu et al.*, 2016; *Yang et al.*, 2018b) or dealing with a scene as a composition

[2] https://www.quora.com/What-are-some-recent-and-potentially-upcoming-breakthroughs-in-deep-learning

of objects (*Yang et al.*, 2018a).

3.5 Semantic Point Cloud Segmentation

Semantic segmentation or point-wise classification of point clouds is a long-standing topic (*Anguelov et al.*, 2005), which was traditionally solved using a feature extractor, such as Spin Images (*Johnson and Hebert*, 1999), in combination with a traditional classifier, like support vector machines (*Agrawal et al.*, 2009) or even semantic hashing (*Behley et al.*, 2010). Many approaches used Conditional Random Fields (CRF) to enforce label consistency of neighboring points (*Triebel et al.*, 2006; *Munoz et al.*, 2008, 2009a,b; *Xiong et al.*, 2011).

With the advent of deep learning approaches in image-based classification, the whole pipeline of feature extraction and classification has been replaced by end-to-end deep neural networks. Voxel-based methods transforming the point cloud into a voxel-grid and then applying convolutional neural networks (CNN) with 3D convolutions for object classification (*Maturana and Scherer*, 2015) and semantic segmentation (*Huang and You*, 2016) were among the first investigated models, since they allowed to exploit architectures and insights known for images.

To overcome the limitations of the voxel-based representation, such as the exploding memory consumption when the resolution of the voxel grid increases, more recent approaches either upsample voxel-predictions (*Tchapmi et al.*, 2017) using a CRF or use different representations, like more efficient spatial subdivisions (*Klukov and Lempitsky*, 2017; *Riegler et al.*, 2017; *Zeng and Gevers*, 2017; *Wang and Lu*, 2018; *Graham et al.*, 2018), rendered 2D image views (*Boulch et al.*, 2017), graphs (*Landrieu and Simonovsky*, 2018; *Te et al.*, 2018), splats (*Su et al.*, 2018), or even directly the points (*Qi et al.*, 2017b,a; *Hua et al.*, 2018; *Groh et al.*, 2018; *Rethage et al.*, 2018; *Jiang et al.*, 2018; *Engelmann et al.*, 2018).

3.6 Benchmarks

The progress of computer vision has always been driven by benchmarks and datasets (*Torralba and Efros*, 2011), but the availability of especially large-scale datasets, such as *ImageNet* (*Deng et al.*, 2009), was even a crucial prerequisite for the rise of deep learning. As there are countless datasets, related to the field of semantic scene understanding, we will now list a subjective subset of datasets that we were most influential to our work.

3.6.1 2D Semantic Segmentation

For a long time, 2D semantic segmentation was driven by benchmarks, focussing on iconic images of arbitrary objects, animals or humans. Over the years ever larger and more complex datasets were published. Prominent examples are the PASCAL VOC 2012 challenge (*Everingham et al.*, 2015) comprising of $\sim$10,000 annotated images for 21 classes and Microsoft COCO (*Lin et al.*, 2014) providing $\sim$200,000 annotated images for 80 classes.

Task-specific datasets geared towards self-driving cars were also proposed. *Brostow et al.* (2008) introduced the CamVid dataset, which comprises of a few hundred densely annotated training images of street scenes. In comparison to CamVid, the *Cityscapes* dataset (*Cordts et al.*, 2016) provides an order of magnitude more of pixel-wise labeled images which makes it suitable for deep learning.

The *Mapillary Vistas* dataset (*Neuhold et al.*, 2017) surpasses the amount and diversity of labeled data compared to *Cityscapes*.

3.6.2 3D Semantic Segmentation

Also in point cloud-based interpretation, *e.g.*, semantic segmentation, RGB-D based datasets enabled tremendous progress. *ShapeNet* (*Chang et al.*, 2015) is especially noteworthy for point clouds showing a single object, but such data is not directly transferable to other domains. Specifically, LiDAR sensors usually do not cover objects as densely as an RGB-D sensor due to their lower angular resolution, in particular in vertical direction.

For indoor environments, there are several datasets (*Silberman et al.*, 2012; *Ros et al.*, 2016; *Hua et al.*, 2016; *Armeni et al.*, 2017; *Dai et al.*, 2017a; *McCormac et al.*, 2017; *Li et al.*, 2018; *Dai et al.*, 2018) available, which are mainly recorded using RGB-D cameras or synthetically generated. However, such data shows very different characteristics compared to outdoor environments, which is also caused by the size of the environment, *i.e.* indoors, point clouds tend to be much denser due to the range at which objects are scanned. Furthermore, the sensors have different properties regarding sparsity and accuracy. While laser sensors are more precise than RGB-D sensors, they usually only capture a sparse point cloud compared to the latter.

For outdoor environments, datasets were recently proposed that are recorded with a terrestrial laser scanner (TLS), like the *Semantic3d* dataset (*Hackel et al.*, 2017), or using automotive LiDARs, like the *Paris-Lille-3D* dataset (*Roynard et al.*, 2018). However, the *Paris-Lille-3D* provides only the aggregated scans with point-wise annotations for 50 classes from which 9 are selected for evaluation. A large dataset for autonomous driving (*Wang et al.*, 2018b), but with fewer classes, is not publicly available.

The *Virtual KITTI* dataset (*Gaidon et al.*, 2016) provides synthetically generated sequential images with depth information and dense pixel-wise annotation. The depth information can also be used to generate point clouds. However, these point clouds do not show the same characteristics as a real rotating LiDAR, including defects like reflections and outliers.

In contrast to these datasets, our dataset combines a large amount of labeled points, a large variety of classes, and sequential scans generated by a commonly employed sensor used in autonomous driving, which is distinct from all publicly available datasets, also shown in Table 8.1.

CHAPTER 4

Real-Time Semantic Segmentation with Label Propagation

Our first work addresses the classic task of semantic image segmentation, *i.e.* given an image, we infer a class label for every pixel in the image. Despite of the success of convolutional neural networks for semantic image segmentation, CNNs cannot be used for some applications due to limited computational resources. Efficient approaches based on random forests are not efficient enough for real-time performance in some cases. In this Chapter, we propose an approach based on superpixels and label propagation that reduces the runtime of a random forest by factor 192 compared to the baseline while increasing the segmentation accuracy.

Contents

4.1 Introduction

Although convolutional neural networks have shown a great success for semantic image segmentation in the last years, fast inference can only be achieved by massive parallelism as offered by modern GPUs. For many applications like mobile platforms or unmanned aerial vehicles, however, the power consumption matters and GPUs are often not suitable. A server-client solution is not always an option due to latency and limited bandwidth. Therefore there is a need for very efficient approaches that segment images in real-time on single-threaded architectures.

In this Chapter, we analyze in-depth how design choices affect the accuracy and runtime of random forests and propose an efficient superpixel-based approach with label propagation for videos. As illustrated in Figure 4.1, we use a very efficient quadtree representation for superpixels. The superpixels are then classified by random forests. For classification, we investigate two methods. For the first method, we use the empirical class distribution and for the second method we model the spatial distributions of class labels by Gaussians. For video data, we propose label propagation to reduce the runtime without substantially decreasing the segmentation accuracy. An additional spatial smoothing even improves the accuracy.

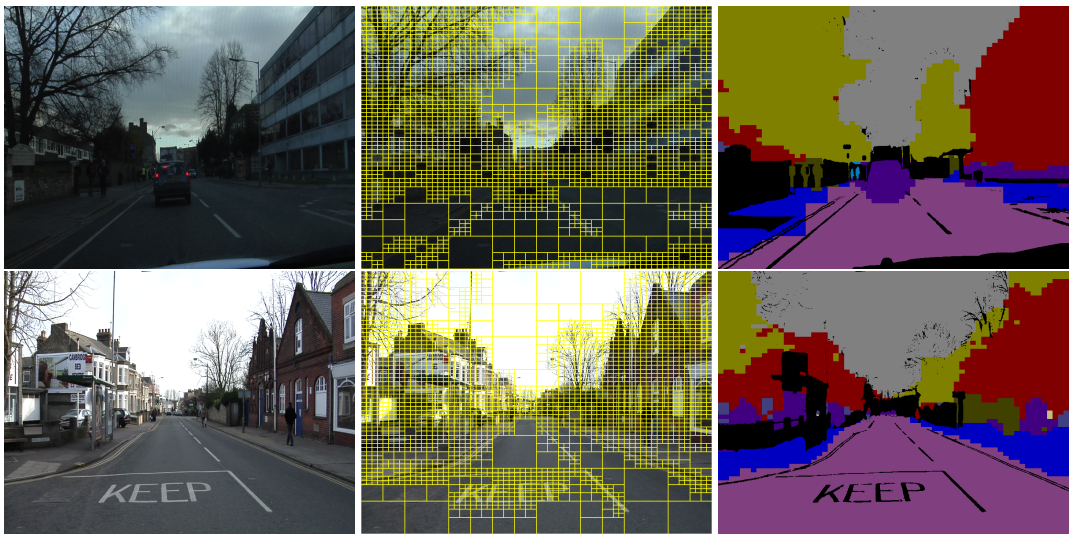

Figure 4.1: For efficient segmentation, we use a quadtree to create superpixels and classify the superpixels by a random forests.

We evaluate our approach on the CamVid dataset (*Brostow et al.*, 2008). Compared to a standard random forest, we reduce the runtime by factor 192 while increasing the global pixel accuracy by 4 percentage points. A comparison with state-of-the-art approaches in terms of accuracy shows that the accuracy of our approach is competitive while achieving real-time performance on a single-threaded architecture.

4.2 Semantic Segmentation

We already described a standard random forests for semantic image segmentation and the training procedure in Section 2.1. In Section 4.2.1 we will specify the parameters and features used in our implementation. In Section 4.2.2, we propose a superpixel approach that can be combined with label propagation in the context of videos.

4.2.1 Random Forests for Semantic Segmentation

For 2D semantic segmentation we can use a forest as described in Chapter 2.1 in combination with the following four types of weak classifiers $f_\theta(\mathbf{x})$, that were proposed by *Shotton et al.* (2008):

$$R(\mathbf{x}+\mathbf{x}_1, w_1, h_1, k) - R(\mathbf{x}+\mathbf{x}_2, w_2, h_2, k) \leqslant \tau \tag{4.1}$$

$$R(\mathbf{x}+\mathbf{x}_1, w_1, h_1, k) + R(\mathbf{x}+\mathbf{x}_2, w_2, h_2, k) \leqslant \tau \tag{4.2}$$

$$|R(\mathbf{x}+\mathbf{x}_1, w_1, h_1, k) - R(\mathbf{x}+\mathbf{x}_2, w_2, h_2, k)| \leqslant \tau \tag{4.3}$$

$$R(\mathbf{x}+\mathbf{x}_1, w_1, h_1, k) \leqslant \tau. \tag{4.4}$$

The term $R(\mathbf{x}+\mathbf{x}_1, w_1, h_1, k)$ denotes the average value of feature channel k in the rectangle region centered at $\mathbf{x}+\mathbf{x}_1$ with $\mathbf{x}_1 \in [-100, \ldots, 100]$, width $w_1 \in [1, \ldots, 24]$, and height $h_1 \in [1, \ldots, 24]$. As feature channels, we use the CIELab color space and the x- and y-gradients extracted by a Sobel

filter. To generate the set of parameters $\tilde{\Theta}$, we randomly sample 500 weak classifiers without threshold value τ and for each sampled weak classifier we sample τ 20 times, i.e., $\tilde{\Theta}$ consists of 10,000 randomly sampled weak classifiers.

During training, we terminate the growing of a tree either when a node becomes pure, or the number of training samples $\mathcal{S}_n$ found in it becomes smaller than some 100 (found using cross-validation).

4.2.2 Superpixels with Label Propagation

A single tree as described in Section 4.2.1 requires on a modern single-threaded architecture 1,500 ms for segmenting an image with a resolution of $960\text{x}720$. This is insufficient for real-time applications and we therefore propose to classify superpixels. In order to keep the overhead by computing superpixels as small as possible, we use an efficient quadtree structure. As shown in Figure 4.1, the regions are not quadratic but have the same aspect ratio as the original image. Up to depth 3, we divide all cells. For deeper quadtrees, we divide a cell into four cells if the variance of the intensity, which is in the range of 0 and 255, within a cell is larger than 49. Instead of classifying each pixel in the image, we classify the center of each superpixel and assign the predicted class to all pixels in the superpixel. For training, we sample 1000 superpixels per training image and assign the class label that occurs most frequently in the superpixel.

While Equation (2.1) uses the empirical class distribution $p(c; \mathcal{S}_l)$ stored in the leaves for classification, it discards the spatial distribution of the class labels within and between the superpixels ending in a single leaf. Instead of reducing the pixel-wise labels of the training data to a single label per superpixel, we model the spatial distribution by a Gaussian per class. To this end, we use the pixel-wise annotations of the superpixels ending in a leaf denoted by $\mathcal{S}_l = \{(\mathbf{x}_l, c_l)\}$. From all pixels $\mathbf{x}_l$ with class label $c_l = c$, we estimate a spatial Gaussian distribution $\mathcal{N}(\mathbf{y}; \mu_{c,l}, \Sigma_{c,l})$ where $\mathbf{y}$ is a vector pointing to a location in the image and $\mu_{c,l}, \Sigma_{c,l}$ are the mean and the covariance of the class specific Gaussian. In our implementation, $\Sigma_{c,l}$ is simplified to a diagonal matrix.

For inference, we convert a superpixel with width w, height h, and centered at $\mathbf{x}$ also into a Gaussian distribution $\mathcal{N}(\mathbf{y}; \mu_\mathbf{x}, \Sigma_\mathbf{x})$ where $\mu_\mathbf{x} = \mathbf{x}$ and $\Sigma_\mathbf{x}$ is a diagonal matrix with diagonal $((\frac{w}{2})^2, (\frac{h}{2})^2)$. The class probability for a single tree and a superpixel ending in leaf l is then given by the integral

$$p(c|\mathbf{x}; \theta_t) = \int \mathcal{N}(\mathbf{y}; \mu_{c,l}, \Sigma_{c,l}) \mathcal{N}(\mathbf{y}; \mu_\mathbf{x}, \Sigma_\mathbf{x}) d\mathbf{y} = \mathcal{N}(\mu_\mathbf{x}; \mu_{c,l}, \Sigma_{c,l} + \Sigma_\mathbf{x}) \tag{4.5}$$

$$\propto \exp\left(-\frac{1}{2}\left(\mu_\mathbf{x} - \mu_{c,l}\right)^T \left(\Sigma_{c,l} + \Sigma_\mathbf{x}\right)^{-1} \left(\mu_\mathbf{x} - \mu_{c,l}\right)\right). \tag{4.6}$$

In our implementation, we omit the normalization constant and use Equation (4.6). Several trees are combined as in Equation (2.1). Instead of using only one Gaussian per class, a mixture of Gaussians can be used as well.

The accuracy can be further improved by smoothing. Let $N_\mathbf{x}$ be the neighboring superpixels of $\mathbf{x}$ including $\mathbf{x}$ itself. The class probability for the superpixel $\mathbf{x}$ is then estimated by

$$p(c|\mathbf{x}) = \frac{1}{|N_\mathbf{x}|} \sum_{\mathbf{y} \in N_\mathbf{x}} p(c|\mathbf{y}). \tag{4.7}$$

To reduce the runtime for videos, the inferred class for a superpixel can be propagated to the next frame. We propagate the label of a cell in the quadtree to the next frame, if the location and size

	Global Pixel Accuracy	Average Class Accuracy	Average time (ms)
pixel stride 1	63.54	40.61	1549
pixel stride 15	64.92	41.21	277
superpixel (sp)	65.11	41.13	27.50
sp - RGB	62.25	36.42	26.47
sp - fix region (sp-fr)	65.29	42.48	28.11
sp - 1 Gaussian	65.16	41.36	29.64
sp - 2 Gaussians	66.41	42.29	26.58
sp-fr - 2 Gaussians (sp-fr-Gauss2)	67.13	43.19	26.25
sp-fr-Gauss2 + smoothing	75.76	47.93	45.24
sp-fr-Gauss2 + propagate	66.49	42.88	18.24
sp-fr-Gauss2 + sm. + prop.	75.03	47.45	37.10
sp-fr-Gauss2 (367 images)	67.06	43.47	23.26
sp-fr-Gauss2 + smoothing (367 images)	74.80	48.00	41.71
sp-fr-Gauss2 + propagate (367 images)	67.00	43.21	15.89
sp-fr-Gauss2 + sm. + prop. (367 images)	74.67	47.19	34.50

Table 4.1: Results for one tree trained on all 468 training images. The last 4 rows report the results when only the sequences recorded with 30 Hz are used for training (367).

does not change and if the mean intensity of the pixels in the cell does not change by more than 5. Otherwise, we classify the cell by the random forest.

4.3 Experiments

For the experimental evaluation, we use the CamVid dataset (*Brostow et al.*, 2008). The images in this dataset have a resolution of 960x720 pixels. The CamVid dataset consists of 468 training images and 233 test images taken from video sequences. There is one sequence where frames are extracted at 15Hz and 30Hz and both are included in the training set. Most approaches discard the frames that were extracted at 15Hz resulting in 367 training images. We report results for both settings. The dataset is annotated by 32 semantic classes, but most works use only 11 classes for evaluation, namely *road, building, sky, tree, sidewalk, car, column pole, fence, pedestrian, bicyclist, sign symbol*. We stick to the 11 class protocol and report the global pixel accuracy and the average class accuracy (*Badrinarayanan et al.*, 2017). The runtime is measured on a CPU with 3.3GHz single-threaded.

Our implementation is based on the publicly available CURFIL library (*Schulz et al.*, 2015), which provides a GPU and CPU version for random forests. As baseline, we use a random forest as described in Section 4.2.1. In Table 4.1, we report the accuracy and runtime for a single tree. The baseline denoted by *pixel stride 1* requires around 1500 ms for an image, which is insufficient for real-time applications. The runtime can be reduced by downsampling the image or classifying only a subset of pixels and interpolation. We achieved the best trade-off between accuracy and runtime for

a stride of 15 pixels in x and y-direction. The final segmentation is then obtained by nearest-neighbor interpolation. Larger strides decreased the accuracy substantially. While this reduces the runtime by factor 5.6 without reducing the accuracy, the approach requires still 280 ms.

We now evaluate the superpixel based approach proposed in Section 4.2.2. We first evaluate superpixel classification based on the empirical class distribution $p(c; \mathcal{S}_l)$, which is denoted by *sp*. Compared to the baseline the runtime is reduced by factor 56 and compared to interpolation by factor 10 without reducing the accuracy. The proposed approach achieves real-time performance with a runtime of only 27.5 ms. Due to the efficient quadtree structure the computational overhead of computing the superpixels is only 2 ms.

In the following, we evaluate a few design choices. Converting an RGB image into the CIELab color space takes 1 ms. The comparison of *sp* (CIELab) with *sp - RGB* (RGB) in Table 4.1, however, reveals that the RGB color space degrades the accuracy substantially. We also investigated what happens if the number of parameters of the weak classifiers $f_\theta(\mathbf{x})$ Equation (4.1)-(4.4) are reduced by setting $\mathbf{x}_1 = 0$, which is denoted by *sp-fr*. It slightly increases the average class accuracy compared to *sp* since one region R is fixed to the pixel location which improves the accuracy for small semantic regions. Small regions, however, have a low impact on the global pixel accuracy. If we use Equation (4.6) instead of the empirical class distribution to classify a superpixel, denoted by *sp - 1 Gaussian*, the accuracy does not improve but the runtime increases by 2 ms. If we use two Gaussians per class, one for the left side of the image and one for the right side, the accuracy increases slightly. Note that the runtime even decreases since Equation (4.6) becomes more often zero for *2 Gaussians* than for *1 Gaussian*.

For the further experiments, we use the superpixel classification with fixed region and two Gaussians, denoted by *sp-fr-Gauss2*. As mentioned in Section 4.2.2 the superpixel classification can be improved by spatial smoothing, which is denoted by *smoothing*. This increases the accuracy substantially but also the runtime to 45 ms. The label propagation on the contrary reduces the runtime to 18 ms without a substantial decrease in accuracy. The smoothing can also be combined with label propagation. This gives nearly the same accuracy as *sp-fr-Gauss2 + smoothing*, but the runtime is with 37 ms lower.

If we use only 367 images instead of 468 images for training, the accuracy is the same but the runtime is reduced by around 3 ms. Since the larger set is based on sampling one sequence twice at 15 Hz and 30 Hz, the larger set does not contain additional information and the accuracy therefore remains the same. The additional training data, however, increases the depth of the trees and thus the runtime. The classification without feature computation takes around 4 ms for a tree of depth 20 and 8-10 ms for a tree of depth 100. For 1000 superpixels sampled from each of the 468 training images, the trees can reach a depth of 100.

In Table 4.2, we report the accuracy and runtime for 10 trees. Increasing the number of trees from one to ten increases the global pixel accuracy of the baseline by 11 percentage points and the average class accuracy by 8 percentage points. We also evaluated the use of convolutional channel features (CCF) (*Yang et al.*, 2015) which are obtained by the VGG-16 network (*Simonyan and Zisserman*, 2015) trained on the ImageNet (ILSVRC-2012) dataset. As in the work of *Iqbal et al.* (2017), the features are combined with axis-aligned split functions to build weak classifiers. Without finetuning the features do not perform better on this dataset. The extraction of CCF features is furthermore very expensive without a GPU. Similar to the baseline, the global pixel accuracy and average class accuracy is also increased for *sp-fr-Gauss2* by 10 and 8 percentage points, respectively.

	Global Pixel Accuracy	Average Class Accuracy	Average time (ms)
pixel stride 1	74.60	48.56	11288
pixel stride 15	74.58	48.69	301.7
CCF features	71.68	51.19	28476
sp-fr-Gauss2	77.49	51.29	105.3
sp-fr-Gauss2 + smoothing	79.62	51.77	131.5
sp-fr-Gauss2 + propagate	76.62	49.99	40.19
sp-fr-Gauss2 + sm. + prop.	78.56	50.79	58.75
sp-fr-Gauss2 (367 images)	77.43	51.22	102.5
sp-fr-Gauss2 + smoothing (367 images)	79.99	52.20	111.5
sp-fr-Gauss2 + propagate (367 images)	76.82	50.48	36.48
sp-fr-Gauss2 + sm. + prop. (367 images)	79.30	51.68	55.47

Table 4.2: Results for 10 trees trained on all 468 training images. The last 4 rows report the results when only the sequences recorded with 30 Hz are used for training (367).

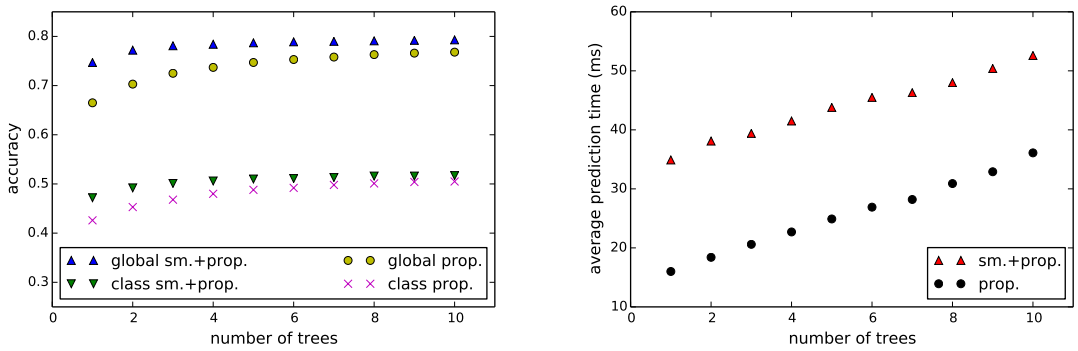

Figure 4.2: Accuracy and average prediction time with respect to the number of trees.

Only if spatial smoothing is added the increase is only 4 percentage points, but it still improves the accuracy. The runtime increases by factor 4, 2.9, 2.2, 1.6 for *sp-fr-Gauss2*, *sp-fr-Gauss2 + smoothing*, *sp-fr-Gauss2 + propagate*, *sp-fr-Gauss2 + sm. + prop.*, respectively. Compared to the baseline *pixel stride 1*, the runtime is reduced by factor 192 while increasing the accuracy if label propagation and smoothing are used. Figure 4.2 plots the accuracy and runtime of *sp-fr-Gauss2 + propagate* and *sp-fr-Gauss2 + sm. + prop.* while varying the number of trees.

The impact of the depth of the quadtree is shown in Figure 4.3. The accuracy but also the runtime increases with the depth of the quadtree since the cells get smaller the deeper the trees are. Limiting the depth of the quadtrees to seven gives a good trade-off between accuracy and runtime. This setting is also used in our experiments.

We finally compare our approach with the state-of-the-art in terms of accuracy in Table 4.3. The first part of the table uses all training images for training. Our approach outperforms CURFIL (*Schulz et al.*, 2015) in terms of accuracy and runtime on a single-threaded CPU. Although the approach by

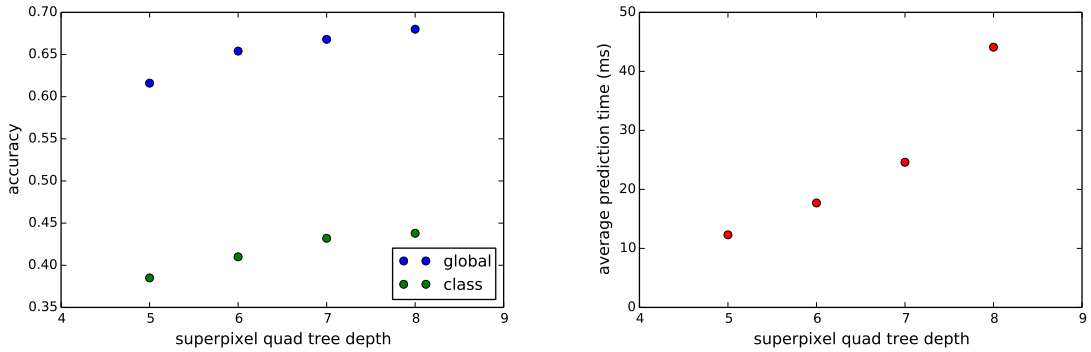

Figure 4.3: Accuracy and average prediction time for one tree using different quadtree depths when creating superpixels. The results are reported for *sp-fr-Gauss2*.

Tighe and Lazebnik (2013) achieves a higher global pixel accuracy, it is very expensive and requires 16.6 seconds for an image with resolution of 800x600 pixels. Our fastest setting requires only 40 milliseconds.

The second part of the table uses the evaluation protocol with 367 images. The numbers are taken from *Badrinarayanan et al.* (2017). The convolutional neural network proposed in *Badrinarayanan et al.* (2017) achieves the best accuracy and requires around 2 seconds per image on a GPU. The methods based on CRFs (*Sturgess et al.*, 2009) require 30 to 40 seconds for an image. The method by *Brostow et al.* (2008) is based on random forests and structure-from-motion. It requires one second per image if the point cloud is already computed by structure-from-motion. The methods by *Bulo and Kontschieder* (2014) and *Kontschieder et al.* (2011) are also too slow for real-time applications. In contrast, our approach segments an image not in the order of seconds but milliseconds while still achieving competitive accuracy. A few qualitative results are shown in Figure 4.4.

4.4 Summary

In this Chapter, we proposed a real-time approach for semantic segmentation on a single-threaded architecture. Compared to the baseline we reduced the runtime by factor 192 while increasing the accuracy. This has been achieved by combining an efficient superpixel representation based on quadtrees with random forests and combining label propagation with spatial smoothing. Compared to the state-of-the-art in terms of accuracy, our approach achieves useful results and runs in real-time without the need of a GPU. This makes the approach ideal for applications with limited computational resources.

In the course of this thesis, large scale datasets like Cityscapes (*Cordts et al.*, 2016) emerged and helped CNN based approaches to increase the accuracy of semantic segmentation algorithms far beyond what is possible with random forest based approaches. At the same time more power saving GPUs for mobile devices have been released, which are capable to perform semantic segmentation using CNNs in real-time. Our work is a contribution to a very specific problem. In fact the work was part of a project to bring real-time semantic segmentation capability to an existing drone, which simply did not feature a GPU. From now on, all our proposed models will be based on CNNs and powered by GPUs.

	Global Pixel Accuracy	Average Class Accuracy	Average time (ms)
Super Parsing (*Tighe and Lazebnik*, 2013)	83.3	51.2	
CURFIL (*Schulz et al.*, 2015)	65.9	49.8	34163
sp-fr-Gauss2	77.5	51.3	105.3
sp-fr-Gauss2 + smoothing	79.6	51.8	131.5
sp-fr-Gauss2 + propagate	76.6	50.0	40.2
sp-fr-Gauss2 + sm. + prop.	78.6	50.8	58.8
Appearance (*Brostow et al.*, 2008)	66.5	52.3	
SfM + Appearance (*Brostow et al.*, 2008)	69.1	53.0	
Boosting (*Sturgess et al.*, 2009)	76.4	59.8	
Dense Depth Maps (*Zhang et al.*, 2010)	82.1	55.4	
Structured Random Forests (*Kontschieder et al.*, 2011)	72.5	51.4	
Neural Decision Forests (*Bulo and Kontschieder*, 2014)	82.1	56.1	
Local Label Descriptors (*Yang et al.*, 2012)	73.6	36.3	
SegNet - 4 layer (*Badrinarayanan et al.*, 2017)	84.3	62.9	2000
Boosting + pairwise CRF (*Sturgess et al.*, 2009)	79.8	59.9	
Boosting + Higher order (*Sturgess et al.*, 2009)	83.8	59.2	
Boosting + Detectors + CRF (*Ladickỳ et al.*, 2010)	83.8	62.5	
sp-fr-Gauss2 (367 images)	77.4	51.2	102.5
sp-fr-Gauss2 + smoothing (367 images)	80.0	52.2	111.5
sp-fr-Gauss2 + propagate (367 images)	76.8	50.5	36.5
sp-fr-Gauss2 + sm. + prop. (367 images)	79.3	51.7	55.5

Table 4.3: Comparison with state-of-the-art approaches. The first six rows shows results for all 468 training images. The lower part report the results when only the sequences recorded with 30 Hz are used for training (367).

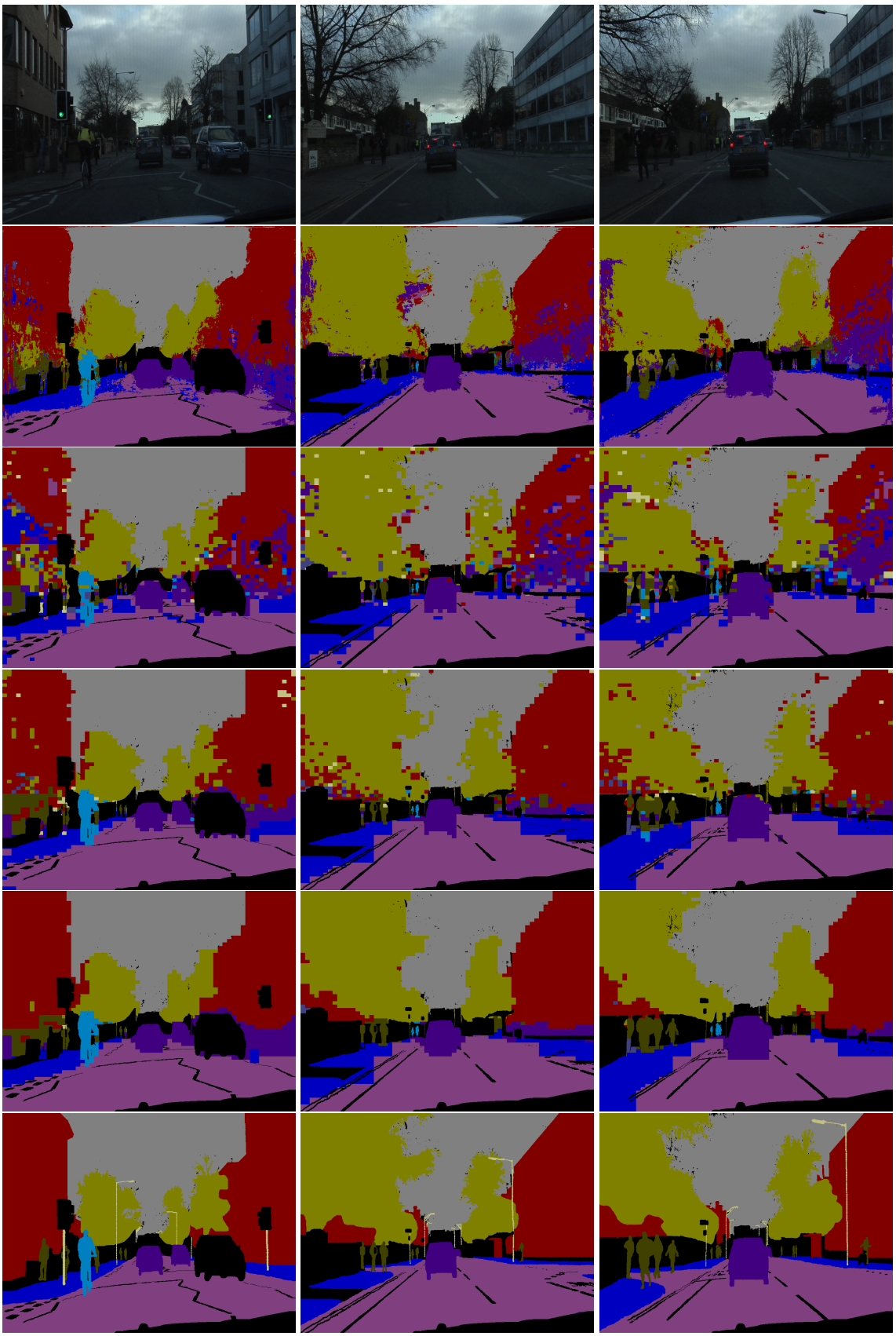

Figure 4.4: Examples of segmentation results. First row: original image. Second row: *pixel stride 1*. Third row: *sp-fr*. Fourth row: *sp-fr-Gauss2 + propagate*. Fifth row: *sp-fr-Gauss2 + sm. + prop*. Sixth row: ground truth.

CHAPTER 5

Spatial Anticipation of Semantic Categories

After providing a real-time approach to semantic segmentation tailored to GPU-free, low energy devices, we now switch our focus to a new, but highly related field of research: The spatial anticipation of semantic categories outside the field of view of the camera sensor, *i.e.*, given an image we'd like to predict the semantics that are likely to occur outside of it. For certain applications like autonomous systems it is insufficient to interpret only the observed data. Instead, objects or other semantic categories, which are close but outside the field of view, need to be anticipated as well. In this Chapter, we propose an approach for anticipating the semantic categories that surround the scene captured by a camera sensor. This task goes beyond current semantic labeling tasks since it requires to extrapolate a given semantic segmentation. Using the challenging Cityscapes dataset, we demonstrate how current deep learning architectures are able to learn this extrapolation from data. Moreover, we introduce a new loss function that prioritizes on predicting multiple labels that are likely to occur in the near surrounding of an image.

Contents

5.1 Introduction

One of the core capabilities of humans intelligence is to make predictions about the environment. Humans are able to predict how the world around them will evolve in the near future and how their actions will affect it. Even without observing an entire scene, they can anticipate objects or surfaces that are close. This ability allows them to plan ahead and to efficiently interact with the world. Similar anticipation capabilities are also required for autonomous systems. For instance, the presence of semantic categories like pedestrians, bicyclists, cars, roads or sidewalks in the near surrounding of an autonomous vehicle has implications for the driving policy and safety measurements. These object

categories, however, are not always within the field of view of the sensors attached to the vehicle and therefore need to be anticipated.

To the best of our knowledge, we are the first to propose an approach that anticipates semantic categories outside the field of view of a camera. In order to evaluate this task, we propose a novel protocol for the large-scale Cityscapes dataset (*Cordts et al.*, 2016), which is the state-of-the-art benchmark for semantic urban scene understanding. In contrast to semantic image segmentation, which requires to infer the labels for each observed pixel, anticipation of semantic categories outside the field of view requires to infer the semantic labels in regions that are not observed. The anticipation task is not only more difficult due to missing data, it is also inherently non-deterministic since many solutions could be plausible. Since the true distribution of all plausible solutions for a single image is unknown, we propose an evaluation metric that does not require a pixel-wise prediction but measures if the occurrence of a semantic class within a predefined region outside the image is correctly predicted.

Since the proposed task has not been addressed before, we introduce a baseline that infers the pixel-wise semantic labels in the observed region and the unobserved region outside the image. The baseline builds on a state-of-the-art convolutional neural network for image segmentation (*Chen et al.*, 2015). In addition, we propose a novel approach that consists of two networks. While the first network infers semantic labels for each observed pixel, the second network gradually anticipates the semantic categories outside the field of view from the previous output. For the second network, two different loss functions are investigated. We evaluate the proposed approach on the Cityscapes dataset using the new protocol for spatial anticipation of semantic categories. The experimental evaluation shows that the proposed approach improves the baseline by a large margin.

5.2 Dataset for Anticipation of Semantic Categories

We propose the new task of spatial anticipation of semantic categories outside the field of view. The task requires to predict for a given image the categories that are most likely to occur outside of it as illustrated in Figure 5.3. For evaluation, we introduce a new protocol for the Cityscapes dataset. The Cityscapes dataset is recorded by a RGB camera mounted at the front of a car driving through urban scenes. We only use the images provided with fine-grained annotation. There are 2,975 images in the training set, 500 in the validation set and 1,525 images in the test set. For evaluation, we take the images from the validation set since the ground truth annotations for the test set are not publicly available. Following *Cordts et al.* (2016), we evaluate the performance on 19 classes (road, sidewalk, building, wall, fence, pole, traffic-light, traffic-sign, vegetation, terrain, sky, person, rider, car, truck, bus, train, motorcycle and bicycle) ignoring the background class. The original images have the size of 1024×2048 pixels. For our task, we crop the validation images such that only the center region of 642×1282 pixels remains. The invisible region outside the cropped area is used to evaluate the anticipation performance.

For the evaluation, we report the accuracy for two evaluation criteria. The first evaluation criterion is a standard semantic image segmentation metric and compares the ground-truth segmentation map for the invisible region with the inferred pixel-wise semantic segmentation. It assumes that exactly one label is predicted for each pixel outside the cropping area of the original images. As for standard semantic image segmentation, we use the Jaccard index, also referred to as intersection over union (IoU), to measure the quality of the prediction. This evaluation approach has the weakness

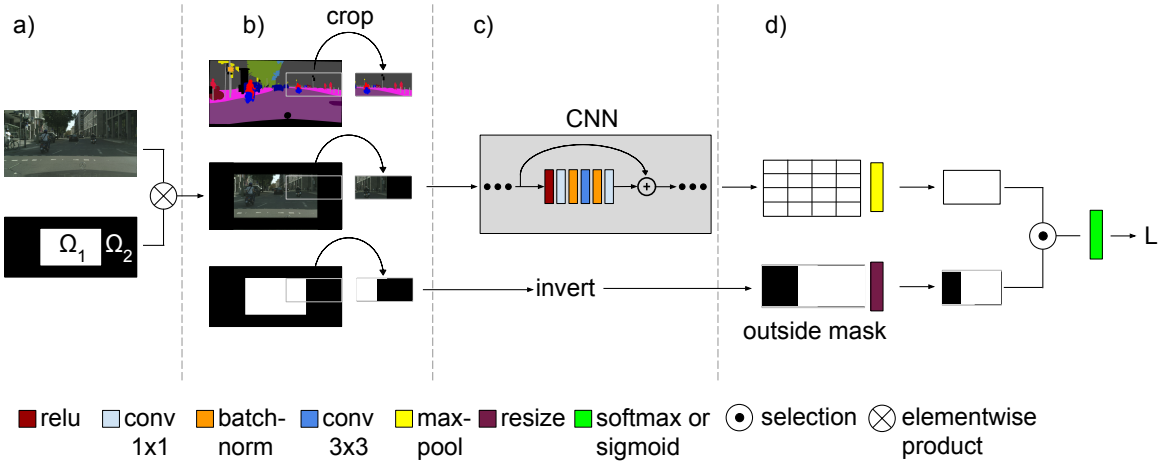

Figure 5.1: Training procedure for SASNet. The SASNet is trained on crops of masked images. The mask marks the visible (Ω_1) and the invisible regions (Ω_2). A detailed description is given in Section 5.3.

that it assumes that the ground-truth is deterministic and can be predicted at a pixel level. However, not even humans will be able anticipate semantic classes with such an accuracy. Moreover, an exact localization of the anticipated labels is unnecessary in a practical context. To account for this fact, we introduce an alternative evaluation metric. The unobserved area is subdivided into a grid of cells as shown in Figure 5.2. All labels that occur in the same cell are collected. If a label occurs in the same cell in both the ground-truth and the prediction map, it is counted as a true positive. Labels only occurring in the prediction map are false positives and labels exclusively occurring in the ground-truth map are considered as false negatives. We sum the true positives, false positives, true negatives, and false negatives over all cells and images for each class and compute the F_1 score, which is defined as the harmonic mean of precision and recall:

$$F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}. \tag{5.1}$$

The scores are then averaged over all classes.

5.3 SASNet: Network for Spatial Anticipation of Semantic Categories

To explore the task of spatial anticipation of semantic categories outside the field of view, we propose an approach that is based on a convolutional neural network architecture that is common for state-of-the-art approaches for semantic image segmentation. However, we modify the last layers and loss function due to the new task at hand. Moreover, we train the model in a different way.

Figure 5.1 gives an overview of the proposed SASNet and the training procedure. The network is trained by providing masks for the visible and invisible regions for each training image. The mask divides the original image, which has a resolution of 1024×2048 pixels in our dataset, into an inner region Ω_1 of 642×1282 pixels and an outer region Ω_2 that is set to zero as shown in a). Then we sample random crops from the images as shown in b). The crops of size 321×321 pixels are taken

from the image, its ground-truth segmentation mask, and the visibility mask. Note that the ratio of the visible and invisible area varies among the crops. The random crops are our training set $\mathcal{T}$.

Figure 5.1 c) illustrates the first part of the network. As base architecture for the convolutional neural network, we choose the DeepLab model (*Chen et al.*, 2015) based on the ResNet 101 structure (*He et al.*, 2016). We omit the conditional random field as well as the loss layer and instead process the unnormalized network output y to compute the loss for the unobserved region Ω_2. We investigate two different loss functions $\mathcal{L}_1$ and $\mathcal{L}_2$. The first loss $\mathcal{L}_1$ is given by the softmax cross entropy:

$$\mathcal{L}_1 = -\sum_{t\in\mathcal{T}}\sum_{i\in\Omega_2^t}\sum_{c\in\mathcal{C}} \hat{y}_{ic}\log\left(\frac{e^{y_{ic}}}{\sum\limits_{c'\in\mathcal{C}} e^{y_{ic'}}}\right) \tag{5.2}$$

where $\hat{y}_{ic}$ is the class probability of the ground truth label of pixel i, which is one for the true class and zero otherwise. y_{ic} denotes the unnormalized predictions of the network for pixel i and class c.

The second loss $\mathcal{L}_2$ measures the anticipation error in accordance with the proposed second evaluation criterion described in Section 5.2, *i.e.*only the classes occurring in each cell in the region Ω_2 should be predicted. This can be efficiently realized as illustrated in Figure 5.1 d) by adding a max pooling layer with kernel size and stride k:

$$\tilde{y}_{i_k c} = \max_{\Delta i\in\mathcal{N}_{i_k}} \{y_{i_k+\Delta i,c}\} \tag{5.3}$$

where $\mathcal{N}_{i_k}$ is the $k \times k$ neighborhood of pixel i_k, *i.e.*$\tilde{y}_{i_k c}$ is the maximum value for each class c in each cell i_k. It is important to note that the kernel size does not need to be equal to the cell size used for evaluation as we will show in Section 5.4.1. Due to the max pooling, Ω_2 has been reduced to the number of cells $\Omega_{k,2}$. We therefore also resize the mask to $\Omega_{k,2}$. For the cells of the invisible region, we compute the second loss $\mathcal{L}_2$ using the sigmoid cross entropy:

$$\mathcal{L}_2 = -\sum_{t\in\mathcal{T}}\sum_{i_k\in\Omega_{k,2}^t}\sum_{c\in\mathcal{C}} \hat{y}_{i_k c}\log\left(\frac{1}{1+e^{-\tilde{y}_{i_k c}}}\right) \tag{5.4}$$

where $\hat{y}_{i_k c}$ is one if the cell i_k in the invisible region Ω_2^t of random crop $t \in \mathcal{T}$ contains the label of class $c \in \mathcal{C}$ and it is zero otherwise.

For inference, the network processes an image with binary mask, which is one for the image pixels (Ω_1) and zero for the regions where the semantic categories should be anticipated (Ω_2). For the first loss function $\mathcal{L}_1$, the network predicts for each pixel i the semantic label given by $\arg\max_c \frac{e^{y_{ic}}}{\sum_{c'} e^{y_{ic'}}}$. For the second loss function $\mathcal{L}_2$, the network predicts for each cell i_k the set of labels

$$\left\{c \in \mathcal{C} : \frac{1}{1+e^{-\tilde{y}_{i_k c}}} \geqslant 0.5\right\}. \tag{5.5}$$

Figure 5.2 shows an example of such a prediction.

In Section 5.4.1, we show that SASNet performs better when we first perform standard semantic image segmentation on the visible region Ω_1 and then use the inferred labels as input for SASNet instead of the RGB values of the image. The accuracy can be further improved by performing the anticipation in successive steps where the region Ω_2 outside the image is gradually increased and the intermediate results are used as input for the next step as shown in the last row of Figure 5.3.

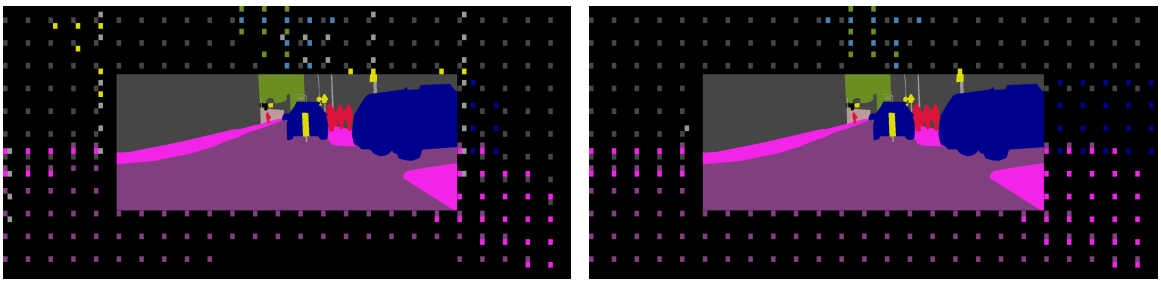

Figure 5.2: The F_1 score is computed for cells outside the visible region and measures if for each cell the same labels are predicted (right) as they occur in the ground-truth segmentation map (left).

5.4 Experimental Evaluation

5.4.1 Implementation and Evaluation Details

We augment the training data by random scaling between 0.5 and 1.5, as well as random mirroring and random cropping. The batch size is set to 10 and the learning rate is set to $2.5 \cdot 10^{-4}$. The learning rate of the batch normalization layer parameters are set to 0. This has shown to stabilize the training process (*Chen et al.*, 2015). The number of training iterations is 20,000. One training takes about 15 hours.

As described in Section 5.2, we report intersection over union (IoU) and the F_1 score computed for four different cell sizes. As cell size c, we choose 16×16, 24×24, 40×40 and 80×80 pixels with respect to the original resolution of the Cityscapes images. Both measures are only evaluated on the unobserved region Ω_2.

We evaluate the two loss functions $\mathcal{L}_1$ and $\mathcal{L}_2$ discussed in Section 5.3. For $\mathcal{L}_2$, we have to define the kernel size k. In our experiments, we evaluate $\mathcal{L}_2$ with the four different kernel sizes 2×2, 3×3, 5×5 and 10×10. Since the previous layers of the network reduce the size of the input image by factor 8, this corresponds to the cell sizes 16×16, 24×24, 40×40 and 80×80 pixels with respect to the input resolution.

As mentioned in Section 5.3, SASNet can be used to anticipate semantic categories outside the field of view from the raw RGB image data or from a pre-segmentation of the visible region Ω_1. We evaluate both cases and use the approach of *Chen et al.* (2015) for image segmentation in the latter case. We denote the first version by 'color-SASNet' and the second version by 'label-SASNet'. In addition, the anticipation can be performed gradually. Depending on the number of steps, we subdivide Ω_2 into either 2, 3 or 4 enclosing regions as can be seen in Figure 5.3. For each step, we use the prediction of the previous step as input and anticipate the semantic categories for the next enclosing region until Ω_2 is fully covered. For initialization, we use the inferred semantic segmentation of the visible region Ω_1.

The scripts, source code and models used for evaluation are publicly available.

5.4.2 Results

The quantitative results for the dataset described in Section 5.2 are summarized in Table 5.1. The first six rows compare the two loss functions $\mathcal{L}_1$ and $\mathcal{L}_2$ if SASNet anticipates the semantic categories from the RGB image (color-SASNet). For both, the intersection over union (IoU) and the F_1 accuracy

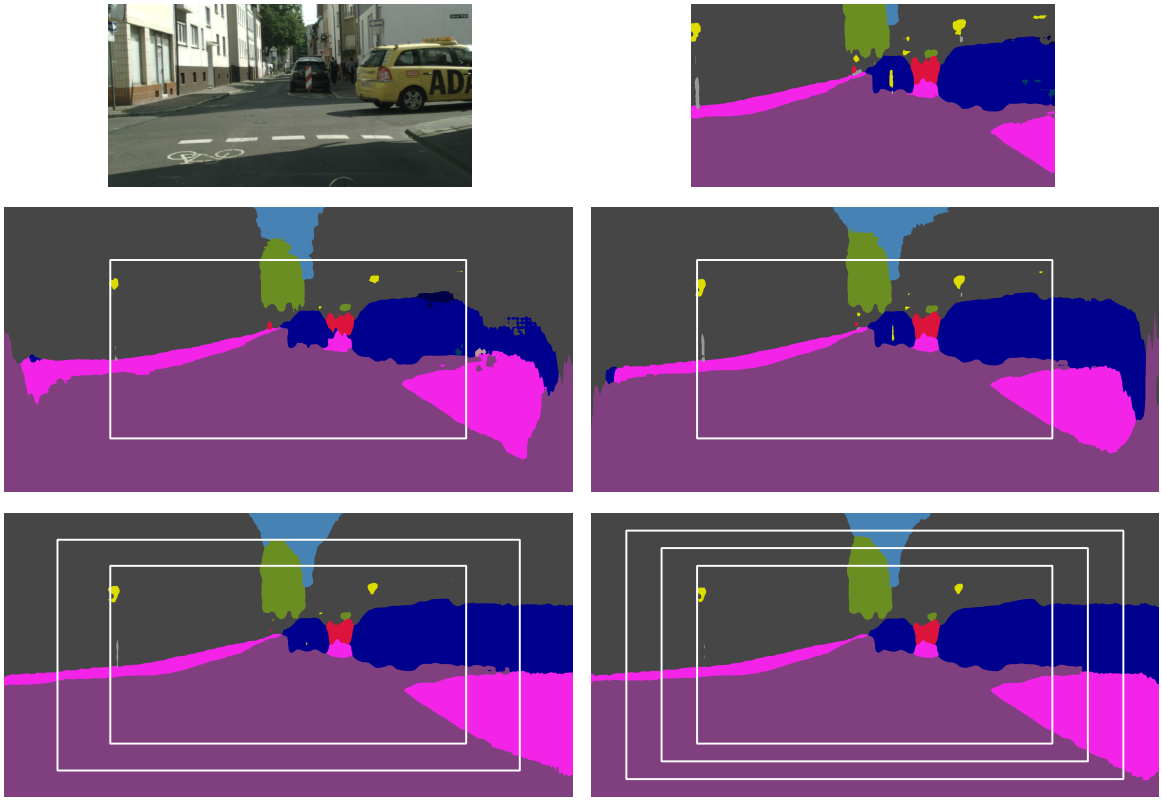

Figure 5.3: Qualitative results for the pixel-wise label prediction using $\mathcal{L}_1$ loss. The first row shows an RGB image and the inferred semantic segmentation using the approach of *Chen et al.* (2015). The second row shows the result for color-SASNet (left), which uses the RGB image of the first row as input, and for label-SASNet (right), which uses the inferred labels as input. The inner white rectangle marks the boundary between observed and unobserved regions Ω_1 and Ω_2. The label-SASNet anticipates the semantic labels in Ω_2 better than color-SASNet. The last row shows the result of label-SASNet if the prediction is performed in two (left) or three steps (right). The additional white rectangles mark the growing regions that are predicted in each step. Compared to the second row, the labels are better anticipated at the border.

$\mathcal{L}_1$ performs better than $\mathcal{L}_2$ in the case of prediction in a single step. For the prediction in two consecutive steps $\mathcal{L}_2$ loss outperforms $\mathcal{L}_1$ in both metrics.

The F_1 score increases for larger c values since this increases the cell size, which requires a lower localization accuracy. If we compare different values of k for $\mathcal{L}_2$, we observe that IoU is slightly higher for $k = 1 \times 1$ since a smaller k enforces the network to learn a better localization of the categories. The setting with $k = 1 \times 1$ is also the best for all c values of the F_1 score.

We now compare the difference of having one network (color-SASNet) or two networks (label-SASNet), one for semantic image segmentation and one for spatial anticipation. A qualitative comparison is also shown in Figure 5.3. If we compare the $\mathcal{L}_1$ loss, IoU increases from 26.1 to 30.7. For the $\mathcal{L}_2$ loss with $k = 1$, IoU increases from 22.0 to 26.6. The F_1 scores also increase for both $\mathcal{L}_1$ and $\mathcal{L}_2$ by about 4 to 5% for all c values, except for $\mathcal{L}_1$ in the case $c = 80$. We can conclude that label-SASNet outperforms color-SASNet.

As illustrated in Figure 5.3, the anticipation accuracy decreases if the distance to the visible image border becomes large. The anticipation can therefore performed gradually where the region Ω_2 grows in each step as described in Section 5.3. The qualitative results using 2, 3 or 4 steps are reported in Table 5.1. We first compare the impact of the number of steps for label-SASNet with $\mathcal{L}_1$ loss. The IoU increases from 30.7 to 33.5 if anticipation is performed in two steps. Further steps increase the accuracy only slightly. The F_1 scores are also slightly improved by estimating the semantic categories outside the image region gradually. If the $\mathcal{L}_2$ loss is used, we observe a large improvement for all values of k. The best results are achieved with two steps. For $k = 5 \times 5$, the F_1 scores increase from 31.4, 31.8, 34.6, 36.3 to 42.7, 43.5, 44.9, 45.3, for $c = 16$, 24, 40, 80 respectively. The IoU also increases from 26.6 to 35.0 for $k = 1 \times 1$. It actually even achieves a higher IoU than the best setting with $\mathcal{L}_1$ loss (33.9). We can conclude that anticipating semantic categories with two steps improves the accuracy by a large margin. The proposed $\mathcal{L}_2$ loss performs better than the $\mathcal{L}_1$ loss with respect to F_1 score as well as IoU. Although the impact of k is very low, $k = 1 \times 1$ is best if the accuracy is measured by IoU and $k = 5 \times 5$ works very well for any c value of the F_1 score. For the case of prediction in a single step $\mathcal{L}_1$ performs better with respect to IoU and F1 score than $\mathcal{L}_2$. But as the number of iterations for the prediction and the size of the evaluation cell is increasing $\mathcal{L}_2$ again outperforms $\mathcal{L}_1$. Finally we can compare our system to a simple border replication of labels at the periphery of the cropped image. This results in 34.7 IoU and 16.8 F1-score at $c = 80$. We can see that a simple border replication baseline performs quite well on the task of pixel-wise prediction (our best system yields 35.0 IoU), but loses strongly in performance in the task of cell-wise prediction (our best system performed at 45.4 F1-score for c = 80).

5.5 Summary

We introduced a new task of anticipating semantic label information outside of an image. Therefore, we investigated two evaluation metrics to asses the quality of the prediction. While the first metric measures pixel-wise accuracy, the second metric relaxes the required localization accuracy and requires only the prediction of categories occurring in cells. In addition, we have proposed a neural network for spatial anticipation and investigated two different loss functions. From our experimental evaluation, we conclude that the most effective configuration uses two networks. The first one infers a pixel-wise segmentation within the visible area and the second one anticipates categories outside of the image from the segmented image. If the second network is applied gradually, the anticipation

Input	$\mathcal{L}$	steps	k	% IoU	c = 16	c = 24	c = 40	c = 80
RGB	$\mathcal{L}_1$			26.1	34.7	35.5	37.5	38.9
RGB	$\mathcal{L}_2$		1x1	22.0	26.6	27.2	29.6	32.7
RGB	$\mathcal{L}_2$		2x2	21.2	25.6	26.2	28.4	31.8
RGB	$\mathcal{L}_2$		3x3	21.2	25.8	26.4	28.5	32.0
RGB	$\mathcal{L}_2$		5x5	21.4	26.4	26.9	29.0	32.4
RGB	$\mathcal{L}_2$		10x10	20.3	25.2	25.6	27.5	30.7
Label	$\mathcal{L}_1$			30.7	39.7	40.2	41.7	38.0
Label	$\mathcal{L}_1$	2		33.5	41.3	42.3	43.2	44.0
Label	$\mathcal{L}_1$	3		33.9	42.0	42.7	43.3	43.4
Label	$\mathcal{L}_1$	4		33.9	42.3	43.0	43.6	43.9
Label	$\mathcal{L}_2$		1x1	26.6	30.8	31.4	34.0	35.3
Label	$\mathcal{L}_2$		2x2	26.3	30.3	30.9	33.4	35.3
Label	$\mathcal{L}_2$		3x3	26.2	31.0	31.5	33.9	35.8
Label	$\mathcal{L}_2$		5x5	26.7	31.4	31.8	34.6	36.3
Label	$\mathcal{L}_2$		10x10	26.6	30.5	31.1	34.1	36.3
Label	$\mathcal{L}_2$	2	1x1	**35.0**	42.3	43.0	44.1	43.7
Label	$\mathcal{L}_2$	2	2x2	34.8	42.6	**43.5**	44.7	45.1
Label	$\mathcal{L}_2$	2	3x3	34.8	42.5	43.4	44.6	44.9
Label	$\mathcal{L}_2$	2	5x5	34.6	**42.7**	**43.5**	**44.9**	45.3
Label	$\mathcal{L}_2$	2	10x10	34.6	42.3	43.3	44.6	**45.4**
Label	$\mathcal{L}_2$	3	1x1	33.9	41.2	41.9	43.1	43.1
Label	$\mathcal{L}_2$	3	2x2	33.7	41.4	42.1	43.6	44.1
Label	$\mathcal{L}_2$	3	3x3	33.7	41.6	42.1	43.5	44.0
Label	$\mathcal{L}_2$	3	5x5	33.7	41.7	42.3	43.8	44.3
Label	$\mathcal{L}_2$	3	10x10	33.7	41.2	41.8	43.3	44.1
Label	$\mathcal{L}_2$	4	1x1	32.8	40.1	40.6	42.0	42.5
Label	$\mathcal{L}_2$	4	2x2	32.8	40.1	40.7	42.4	42.8
Label	$\mathcal{L}_2$	4	3x3	32.7	40.4	40.9	42.5	43.0
Label	$\mathcal{L}_2$	4	5x5	32.7	40.4	40.9	42.7	43.0
Label	$\mathcal{L}_2$	4	10x10	32.8	39.8	40.3	42.3	43.4
Border Rep.				34.7	11.6	12.8	14.0	16.8

Table 5.1: Quantitative results for spatial anticipation on the Cityscapes dataset (*Cordts et al.*, 2016). RGB or Label denote if color-SASNet or label-SASNet are used. $\mathcal{L}_1$ stands for pixel-wise loss and $\mathcal{L}_2$ for the cell-wise loss. k is the kernel size and stride used to compute the $\mathcal{L}_2$ loss during training. The third column indicates if the SASNet was applied gradually using 2, 3 or 4 steps. The fifth column is the pixel-wise evaluation using % IoU. The other columns are F1 scores expressed in % computed for the cell-wise evaluation. The size of the cells is specified as c. The last line shows the result for a simple border replication.

accuracy increases by a large margin. In this configuration, training the second network using the cell-wise loss performs for both evaluation metrics better than a pixel-wise loss. For the pixel-wise metric, it is most effective to choose the smallest possible kernel size for the loss function. For the cell-wise metric, the kernel size $k = 5 \times 5$ has shown to perform very well for any cell size used for evaluation. The proposed protocol and evaluation measure allows to study the problem of anticipating semantic categories outside the field of view. We have demonstrated the anticipation capabilities of the proposed approach. It will be a first step to address this problem in the future.

CHAPTER 6

Two Stream 3D Semantic Scene Completion

After dealing with the semantic extrapolation of scenes in two dimensions, we now shift our focus to the 3D domain. In 3D, the semantic and geometric extrapolation of a scene beyond what is captured by sensor is addressed in an established field called 3D semantic scene completion. Inferring the 3D geometry and the semantic meaning of surfaces, which are occluded, is a very challenging task. Recently, a first end-to-end learning approach has been proposed that completes a scene from a single depth image. The approach voxelizes the scene and predicts for each voxel if it is occupied and, if it is occupied, the semantic class label. In this Chapter, we propose a two stream approach that leverages depth information and semantic information, which is inferred from the RGB image, for this task. The approach constructs an incomplete 3D semantic tensor, which uses a compact three-channel encoding for the inferred semantic information, and uses a 3D CNN to infer the complete 3D semantic tensor. In our experimental evaluation, we show that the proposed two stream approach substantially outperforms the state-of-the-art for semantic scene completion.

Contents

6.1 Introduction

Humans quickly infer the 3D semantics of a scene, i.e., an estimate of the 3D geometry and the semantic meaning of the surfaces. While RGB-D sensors in combination with CNNs provide geometry and semantic information, the resulting representation is very sparse since large parts of the 3D scene

are occluded and not visible. The perception, however, is not limited to the visible part of the scene. When looking at a mug on a table, a human can estimate the full geometry of both objects including parts which are invisible since they are occluded by the objects themselves. This information is obtained from semantic understanding of the scene which allows to estimate the spatial extent of the objects from experience. Such an ability is highly desirable for autonomous agents, e.g., to navigate or interact with objects. A robot that has an intuition about the geometry behind the surface it sees, for example, could plan ahead given a single view instead of exhaustively explore the occluded parts of a scene first.

In this work, we aim to estimate the semantics not only of the visible part, but of the entire scene including the occluded space. To this end, we build on the work of *Song et al.* (2017). They show that semantic scene understanding and 3D scene completion benefit from each other. On one hand, recognizing a part of the object helps to estimate its location in the 3D space and the voxels it occupies. On the other hand, knowing the occupancy in the 3D space gives information on form and size of the object and thus facilitates semantic recognition. For estimating for each voxel in the scene the occupancy and semantic label, they proposed an end-to-end trainable 3D convolutional neural network (3D CNN) which incorporates context from a large field of view via dilated convolutions. The approach, however, only uses depth as input and neglects the RGB image. This means that the semantic label has to be inferred from the geometry alone and properties such as color, texture, or reflectance are not taken into account.

We therefore extend the approach by keeping its beneficial context incorporation and end-to-end trainability while modifying it to leverage semantic information inferred from the RGB image at the input stage as well as at the loss. Given a single RGB-D image, we first use a 2D CNN to infer the semantic labels from the RGB data and construct an incomplete 3D semantic tensor. To this end, we map the inferred semantic labels to the 3D space and label each visible surface voxel by the inferred class label. The 3D semantic tensor is incomplete since it only contains the labels of the visible voxels but not of the occluded voxels. The 3D projection is performed using the depth image. The tensor is then used as input for a 3D CNN that infers a complete 3D semantic tensor, which includes the occupancy and semantic labels for all voxels.

Using the RGB images as input leads to a significant performance gain in scene completion and semantic scene completion as our experiments show. We outperform *Song et al.* (2017) by a substantial margin of up to 9.4 $\%$ on NYU. This implies that RGB images provide a rich discriminative signal.

6.2 Two Stream Semantic Scene Completion

6.2.1 Semantic Scene Completion

The goal of 3D semantic scene completion is to classify every voxel in the view frustum into one of $K + 1$ labels $c = c_0, ..., c_K$ where c_0 represents an empty voxel and $c_1, ..., c_K$ represents one of $K = 11$ class labels like ceiling, floor, wall, window, chair, bed, sofa, table, TV, furniture and object. As illustrated in Figure 6.1, the camera observes only a part of the scene while other voxels are occluded. The occluded voxels can either be empty (c_0) or belong to one of the K classes.

To address the task of 3D semantic scene completion, we propose an approach that leverages two input streams, namely RGB and depth. An overview of the approach is given in Figure 6.2 a).

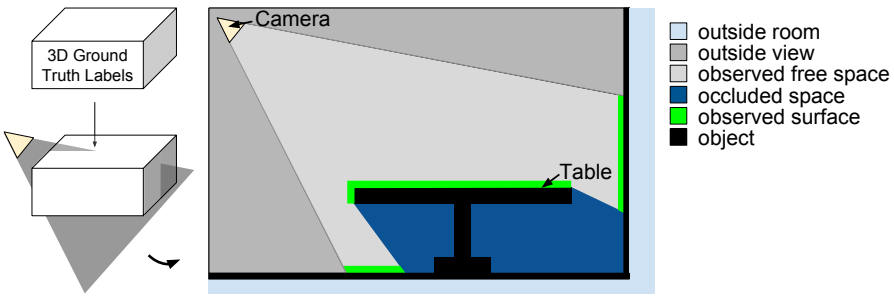

Figure 6.1: Using the protocol of *Song et al.* (2017), ground truth labels are provided for all voxels of a 3D volume. Voxels that are outside the intersection of the camera frustum and ground truth volume are outside the room or outside the view and not taken into account. Within the intersection, there are observed surface voxels (green) and observed non-occupied voxels (light gray), but other voxels are not observed by the camera. These voxels are either non-occupied (blue) or belong to an object (black).

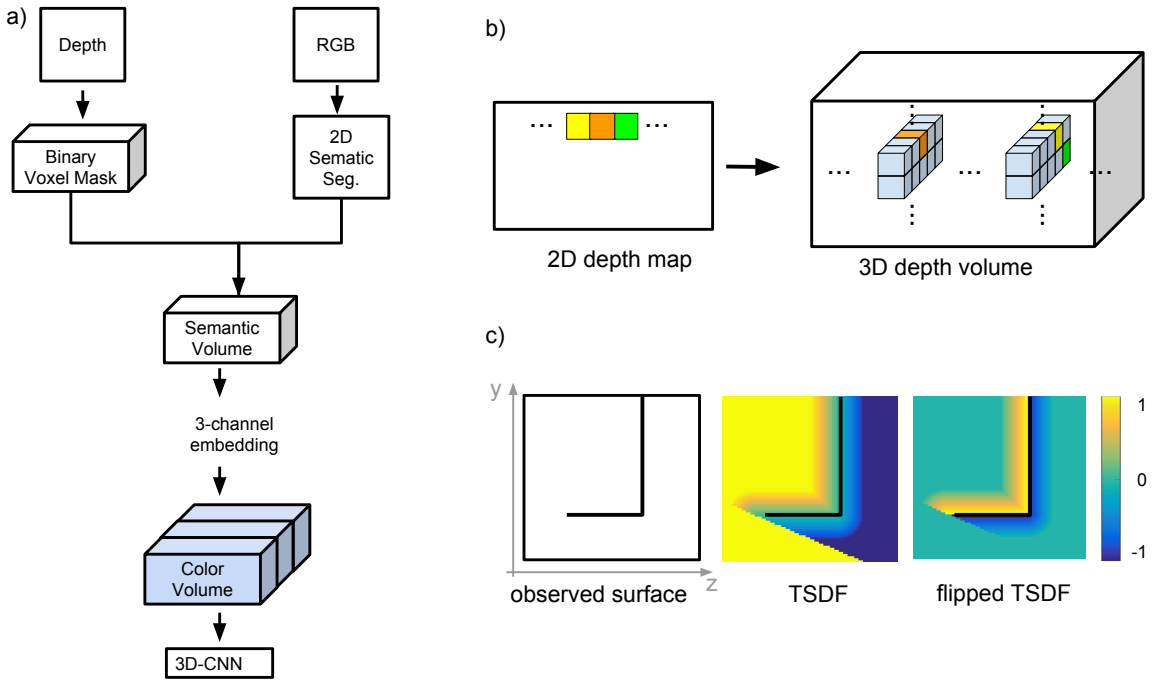

Figure 6.2: a) The proposed two stream approach for semantic scene completion transforms first the depth data and RGB image into a volumetric representation, which represents the geometry and semantic of the visible scene and then uses a 3D-CNN to infer a 3D semantic tensor for the entire scene. b) Given 2D depth map and camera pose, a binary voxel mask is created by setting each voxel that belongs to a depth pixel to one and all other voxels to zero (blue). c) Visualization of TSDF vs. flipped TSDF. One can see the long 'shadow' caused by the observed surface which produces high gradients at the occlusion boundary (between -1 and 1). In the flipped TSDF, this effect is suppressed. The gradient is highest at the surface.

While the depth data is converted into a volumetric representation, the RGB image is first processed in a separate branch to infer 2D semantic segmentation maps and then transformed into a volumetric representation referred to as color-volume (Section 6.2.3). The volumetric representation is then fed to a 3D convolutional neural network (3D-CNN). The 3D-CNN infers a 3D semantic tensor where every voxel is classified as either being empty or belonging to one of the K semantic classes. In the following, each step will be discussed in detail.

6.2.2 Depth Input Stream

To obtain the volumetric input encoding, the depth map is projected into a regular voxel grid using the camera pose, which is provided along with each image. The voxel grid is of size 240 x 144 x 240 voxels and encodes a scene of 4.80m horizontally, 2.88m vertically, and 4.80m in depth with a resolution of 0.02m. For every pixel in the depth map, its corresponding voxel in the 3D input volume is computed using the camera pose. The obtained binary voxel mask encodes the location of surface points that are visible to the camera, see Figure 6.2 b).

As pre-processing, all 3D scenes are rotated such that the room orientations are aligned. For indoor room scenes, one can assume that most of the observed surface normals are oriented either like the normals of the walls, floor or ceiling, which are usually planar. Therefore a principal component analysis of the surface normals is used to infer the room orientation, which is used to align the scene.

6.2.3 Color Input Stream

The input RGB image is first processed by a 2D-CNN (*Chen et al.*, 2015), which is an adaptation of the Resnet101 architecture (*He et al.*, 2016) for semantic segmentation. While all but one pooling layer are omitted, dilated convolutions are used to keep the output resolution high while simultaneously increasing the receptive field. The 2D-CNN predicts the softmax probabilities for every class and pixel at a resolution which is four times smaller than the input image. The output is then upsampled to the original resolution of the image using bilinear interpolation. A densely connected CRF (*Krähenbühl and Koltun*, 2011) is then used in combination with the inferred class probabilities and the RGB image to refine the semantic segmentation map. For training, we use the same setting as in *Chen et al.* (2015). As initialization, we use a model that is pre-trained on MSCOCO (*Lin et al.*, 2014) and fine-tune it on the dataset for 3D semantic scene completion. Furthermore, we present results using the more recent model Deeplab v3+(*Chen et al.*, 2018) which is pretrained on ADE-20k and finetune it on NYUv2 using an initial learning rate of 0.001. We also apply a CRF (*Krähenbühl and Koltun*, 2011) on the resulting outputs.

As in Section 6.2.2, we convert the 2D segmentation map into a volumetric representation. Since each pixel in the depth map corresponds to a pixel in the 2D semantic segmentation map, every class pixel can be projected into the 3D volume at the location of its corresponding depth value. This yields an incomplete 3D semantic tensor that assigns to every surface voxel its corresponding class label. The class labels can be encoded by one-hot encoding, i.e., a channel for each class, or by a single channel for the class label. In our experiments, however, we show that none of them is optimal. Encoding semantic classes with only one channel implies a semantic proximity of classes by the numerical proximity of their class values, which introduces undesirable artifacts based on the class values. The one-hot encoding has the disadvantage that it is insufficient in terms of memory consumption since it requires to store a K dimensional vector per voxel. We therefore represent the

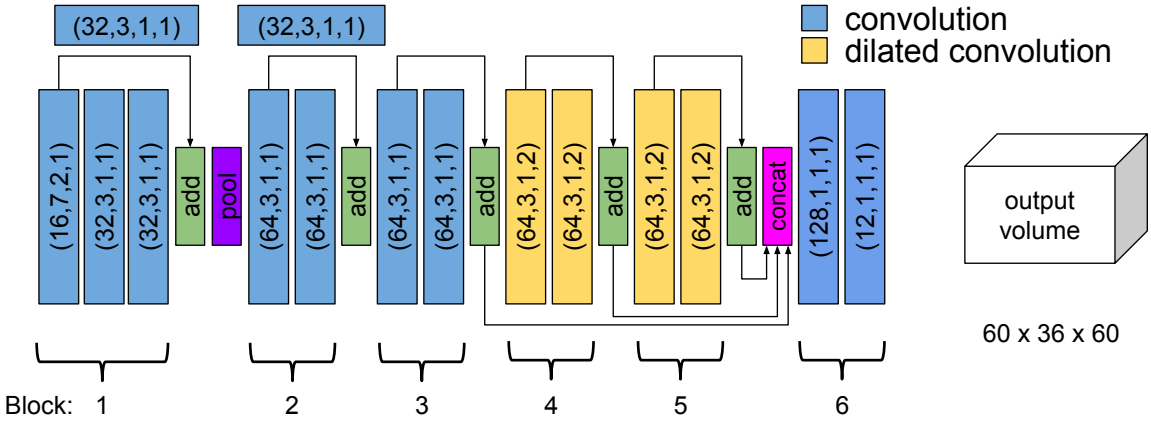

Figure 6.3: Architecture of the 3D-CNN. The parameters of the convolution kernels are denoted as (number of filters, kernel size, stride, dilation). All but the last convolution layer have a ReLU activation function assigned to it. Arrows indicate skip connections *He et al.* (2016) where the output of one convolution layer is added to another output at a later stage. Pool denotes max pooling. The output is a volume that is 4 fold downsampled with respect to the input of the 3D CNN and encodes for every voxel the probability of it being empty (label 0) or to belong to one of K semantic classes.

semantic information by a lower dimensional vector. We use a three-dimensional vector and encode the classes linearly from $(0, 0, 1)$ over $(0, 1, 1)$, $(0, 1, 0)$, $(1, 1, 0)$ to $(1, 0, 0)$.

6.2.4 3D-CNN

For the 3D-CNN, we adapt the architecture of *Song et al.* (2017) by increasing the number of input channels of the first convolutional layer such that it fits to our input. The architecture is illustrated in Figure 6.3. It is inspired by the 2D-CNN for semantic segmentation. The major difference apart from using 3D instead of 2D convolutions is that the network only has a depth of 14 convolutional layers. The network has therefore significantly less parameters than its two dimensional counter part. Moreover, batch-normalization layers are omitted due to the small size of the batches.

We adapt the training protocol of *Song et al.* (2017) as follows. We train for 150,000 steps with a learning rate of 0.01 that is reduced by a factor of 0.1 after 100,000 iterations. As optimizer, stochastic gradient (SGD) with momentum is applied. As initialization, we chose a random initialization with a Gaussian distribution with mean $\mu = 0$ and a standard deviation of $\sigma = 0.01$.

The output of the 3D-CNN is a semantic tensor of size 60 x 36 x 60 x $(K + 1)$, where K is the number of object classes and an additional class is added for empty voxels. We compute a softmax cross entropy loss on the unnormalized network outputs y:

$$\mathcal{L} = -\sum_{i,c} w_{ic}\hat{y}_{ic} \log\left(\frac{e^{y_{ic}}}{\sum_{c' \in \mathcal{C}} e^{y_{ic'}}}\right) \tag{6.1}$$

where $\hat{y}_{ic}$ are the binary ground truth vectors, i.e., $\hat{y}_{ic} = 1$ if voxel i is labeled by class c, and w_{ic} are the loss weights. Since the ratio of empty vs. occupied voxels is 9:1, the empty space is randomly subsampled. Therefore w_{ic} is chosen as binary mask such that only 2N empty voxels are selected for loss calculation where N is the number of occupied voxels in the scene.

6.3 Experimental Evaluation

6.3.1 Evaluation Metric

For evaluation, we follow the evaluation protocol of *Song et al.* (2017), which evaluates the accuracy on a subset of voxels. The evaluation considers only voxels that are part of the occluded space and within both the room and the field-of-view as shown in Figure 6.1. While generating the 3D semantic labels from the annotated CAD models, every voxel in the input volume is marked as being on surface, free space, occluded space, outside field of view, outside room or outside ceiling. For semantic scene completion, a binary evaluation mask is computed such that the evaluation metric is only computed for voxels which are either occluded, on surface or close to the surface (within the range of the TSDF function defined by *Song et al.* (2017)). For scene completion another mask is computed which comprises all voxels in the occluded space. To assess the quality of 3D scene completion, several metrics are computed. First we compute the Jaccard index, which measures the intersection over union (IoU) between ground truth and predicted voxel for every object category $c_1, ..., c_K$. As an overall segmentation performance, we compute the average across all classes. For scene completion all voxels are considered to belong to one of the two classes empty vs. non-empty. All object categories $c_1, ..., c_K$ are counted as 'non-empty'. For completion, IoU as well as precision and recall are computed.

Note that computing the average Jaccard index of all semantic classes is analogous to the evaluation of the 2D semantic segmentation and anticipation approaches (Chapter 4 and 5). The cell-wise prediction metric proposed in Chapter 5 is not used in the case of 3D semantic scene completion, as one of its goals is to accurately reconstruct the 3D scene whereas in the 2D case an exact localization of semantic labels was not necessary. Also the voxelized 3D scene is sparse especially requires the prediction of empty voxels in the occluded space, a problem which does not exist in the 2D case and would not be accounted for by the cell-wise metric.

6.3.2 Datasets

We evaluate our method on the NYUv2 dataset, which is in the following denoted as NYU. NYU consists of indoor scenes that are captured via a Kinect sensor. For 3D semantic scene completion labels, we use the annotated 3D labels provided by *Rock et al.* (2015). They provide 1449 scenes, annotated with 11 classes, 795 of which are used for training and 654 for testing. These annotations consist of CAD models that are fitted into the scene. Since the CAD models do not exactly fit the shape of the annotated objects and neglect small objects such as clutter, there is a significant mismatch between the Kinect input data and the output labels. To address this problem, depth maps generated from the projections of the 3D annotations as in *Rock et al.* (2015) are used for training. For evaluation, we consider two test sets. The first test set, which is denoted by NYU Kinect, consists of the depth maps from the Kinect sensor and the second test set, which is denoted as NYU CAD, uses the depth maps generated by projection.

6.3.3 Ablation Study

We conduct an ablation study on Kinect to analyze the design choices of our model.

	Scene Completion	Semantic Scene Completion										
semantic	IoU	ceil.	floor	wall	win.	chair	bed	sofa	table	TV	furn.	objs	avg
DLv2	60.1	7.0	93.1	25.9	16.8	14.7	53.3	46.0	16.8	22.7	34.1	13.8	31.3
DLv3+	60.0	6.7	93.2	26.2	20.6	17.8	56.9	49.2	16.5	29.4	37.6	18.3	33.8
GT	61.0	6.6	93.4	27.5	24.6	23.0	61.6	57.9	24.3	33.5	46.4	24.5	38.5
RGB image	58.2	4.2	93.4	19.3	4.4	10.8	34.9	20.2	11.8	4.9	17.2	10.3	21.0

Table 6.1: Impact of the quality of the semantic input. For the version 'RGB image', the 2D-CNN is omitted and the color values of the pixels instead of the semantic information is stored in the semantic volume.

	ceil.	floor	wall	win.	chair	bed	sofa	table	TV	furn.	objs.	avg.
Deeplab v2	58.1	85.7	76.6	62.9	58.5	65.8	62.8	37.9	56.8	56.5	54.7	61.5
Deeplab v3+	71.1	89.8	82.8	72.8	65.8	72.4	66.1	50.7	63.0	64.7	62.9	69.3

Table 6.2: 2D semantic segmentation accuracies on the NYUv2 dataset ($\%IoU$). In both cases, a CRF is used.

6.3.3.1 Effect of Semantic Input

As mentioned in Section 6.2.3, we compare two network architectures for the 2D-CNN, namely Deeplab v2 (DLv2) by *Chen et al.* (2015) and Deeplab v3+ (DLv3+) by *Chen et al.* (2018). Table 6.1 shows that DLv3+ increases the accuracy for semantic scene completion from 31.3 % to 33.8 %. This is expected since DLv3+ provides a better 2D segmentation accuracy compared to DLv2 as shown in Table 6.2. For scene completion, IoU is slightly higher for DLv2 than for DLv3+.

We compare the results to a setting when we use ground truth semantic segmentation masks for the RGB images as input to the 3D-CNN, which is denoted by GT in Table 6.1. This also serves as an upper bound for our method when the used 2D-CNN provides perfect segmentation masks. As expected, using ground truth segmentation masks improves the semantic scene completion compared to DLv3+ by +4.7 %. For scene completion, the improvements compared to DLv3+ are +1.0 %. This shows that the quality of the 2D-CNN has a strong impact on the accuracy of semantic scene completion but only a minor impact on scene completion, which is also not the focus of this work.

As it is illustrated in Figure 6.2 a), the RGB images are processed by a 2D-CNN and the inferred pixel-wise labels are used to construct the semantic volume. We also evaluate in Table 6.1 what happens if the three-channel encoding is not based on the semantic labels but if the RGB values of the pixels are directly used for the encoding, i.e., without inferring semantic information from the visible part of the scene. This setting is denoted by 'RGB image' and performs as expected poorly. Besides of the class 'floor', all categories are poorly estimated.

6.3.3.2 Input Encoding

As we discussed in Section 6.2.3, the encoding of the semantic information in the semantic volume should provide a numerical equidistance between classes, which can be achieved by using one-hot encoding. However, this approach has a high memory footprint. As an alternative, we evaluate a one-channel and a three-channel input encoding. In the one-channel setup, the numeric class values

	Scene Completion	Semantic Scene Completion										
channels	IoU	ceil.	floor	wall	win.	chair	bed	sofa	table	TV	furn.	objs	avg
1	59.3	8.3	93.3	25.0	13.4	11.5	43.0	31.8	11.2	2.4	26.5	16.8	25.8
3	60.0	6.7	93.2	26.2	20.6	17.8	56.9	49.2	16.5	29.4	37.6	18.3	33.8
12 (one-hot)	60.0	9.7	93.4	25.5	21.0	17.4	55.9	49.2	17.0	27.5	39.4	19.3	34.1

Table 6.3: Impact of the number of channels for the semantic volume. 12 channels refers to one-hot encoding.

	Scene Completion	Semantic Scene Completion										
input	IoU	ceil.	floor	wall	win.	chair	bed	sofa	table	TV	furn.	objs	avg
proposed, no fTSDF	60.0	6.7	93.2	26.2	20.6	17.8	56.9	49.2	16.5	29.4	37.6	18.3	33.8
with fTSDF, early fusion	60.2	8.1	94.4	25.6	17.1	17.8	53.6	48.0	17.0	28.0	36.0	18.4	33.1
with fTSDF, fusion 1	60.0	4.8	94.1	25.5	21.5	16.6	56.9	47.2	16.7	27.5	37.3	18.1	33.3
with fTSDF, fusion 2	54.4	5.9	93.6	22.0	11.0	16.5	50.4	41.0	12.4	23.1	31.9	12.4	29.1
with fTSDF, fusion 5	59.1	5.1	92.9	23.0	19.4	15.1	53.9	46.7	16.3	28.2	34.6	15.0	31.8
with fTSDF, late fusion	60.4	5.7	93.9	25.7	20.3	15.9	55.7	44.8	17.0	28.1	34.9	16.0	32.5
RGB image	58.2	4.2	93.4	19.3	4.4	10.8	34.9	20.2	11.8	4.9	17.2	10.3	21.0

Table 6.4: Impact of the input for the 3D-CNN. The proposed architecture is shown in Figure 6.2 a). The versions 'with fTSDF' refers to a version where not only the semantic volume but also the flipped TSDF volume *Song et al.* (2017) are used.

are normalized to the range from 0 to 1. For the proposed 3-channel input encoding, every label is mapped to a 3 dimensional vector as described in Section 6.2.3. Table 6.3 shows that using only one channel performs poorly since it introduces undesirable artifacts based on the class values. While some classes like 'floor' and 'bed' are well recognized, the accuracy for 'window' and 'TV' is very low. Using one-hot encoding (12 channels) performs much better than 1 channel but it is expensive in terms of memory consumption. The proposed three-channel encoding requires less memory while it only slightly decreases the accuracy. Also the training time of the 3 channel setup is by a factor of 1.7 faster which reduces the training time from 4 to 2.5 days. Therefore we adopt the 3 channel setup as it provides an efficient alternative to the one-hot encoding.

6.3.3.3 Fusion with flipped TSDF

Furthermore, we have conducted an experiment where we combine our input with the flipped truncated signed distance function (fTSDF) proposed by *Song et al.* (2017) and evaluate different fusing schemes.

The fTSDF is computed as follows: The previously computed binary voxel mask (Figure 6.2 b) is used to first compute a truncated signed distance function (TSDF) encoding as illustrated in Figure 6.2 c). In the TSDF, every voxel contains as value the distance d to the next surface point. The sign of the distance value indicates whether a voxel lies in the empty (1) or occluded space (-1). The TSDF has the disadvantage of having high gradients at the occlusion boundary, i.e., the boundary between observed and unobserved space behind a surface. Therefore in the TSDF encoding every surface yields a shadow into the unobserved space as shown in Figure 6.2 c).

To provide a more meaningful input signal, the signed distance function is transformed into a

NYU CAD		Scene Completion	Semantic Scene Completion										
method	trained on	IoU	ceil.	floor	wall	win.	chair	bed	sofa	table	TV	furn.	objs	avg
Zheng et al. (2016)	NYU	34.6												
Firman et al. (2016)	NYU	50.8												
SSCNet (*Song et al.*, 2017)	NYU	70.3												
SSCNet (*Song et al.*, 2017)	SUNCG+NYU	73.2	32.5	92.6	49.2	8.9	33.9	57.0	59.5	28.3	8.1	44.8	25.1	40.0
Two-Stream (ours), 3ch	NYU	76.1	28.3	94.0	48.6	33.0	33.4	67.9	54.7	31.1	33.8	50.8	30.6	46.0
Two-Stream (ours), one-hot	NYU	76.1	25.9	93.8	48.9	33.4	31.2	66.1	56.4	31.6	38.5	51.4	30.8	46.2

NYU Kinect		Scene Completion	Semantic Scene Completion										
Lin et al. (2013)	NYU	36.4	0.0	11.7	13.3	14.1	9.4	29.0	24.0	6.0	7.0	16.2	1.1	12.0
Geiger and Wang (2015)	NYU	44.4	10.2	62.5	19.1	5.8	8.5	40.6	27.7	7.0	6.0	22.6	5.9	19.6
SSCNet (*Song et al.*, 2017)	NYU	55.1	15.1	94.7	24.4	0.0	12.6	32.1	35.0	13.0	7.8	27.1	10.1	24.7
SSCNet (*Song et al.*, 2017)	SUNCG+NYU	56.6	15.1	94.6	24.7	10.8	17.3	53.2	45.9	15.9	13.9	31.1	12.6	30.5
ESSCN (*Zhang et al.*, 2018)	NYU	56.2	17.5	75.4	25.8	6.7	15.3	53.8	42.4	11.2	0.0	33.4	11.8	26.7
Two-Stream (ours), 3ch	NYU	60.0	6.7	93.2	26.2	20.6	17.8	56.9	49.2	16.5	29.4	37.6	18.3	33.8
Two-Stream (ours), one-hot	NYU	60.0	9.7	93.4	25.5	21.0	17.4	55.9	49.2	17.0	27.5	39.4	19.3	34.1

Table 6.5: Comparison to the state-of-the-art.

flipped TSDF (*Song et al.*, 2017), where every signed distance value d is converted into a distance d_f which is 1 or -1 at a surface and linearly falls to 0 at a distance d_{max} from the surface:

$$d_f = sign(d)\mathcal{H}(d_{max} - |d|)\frac{d_{max} - |d|}{d_{max}} \tag{6.2}$$

where d_{max} is the maximum distance of 24 cm and $\mathcal{H}$ is the Heaviside function:

$$\mathcal{H}(x) = \begin{cases} 1 & \text{if } x \geqslant 0 \\ 0 & \text{if } x < 0. \end{cases} \tag{6.3}$$

We perform different fusion experiments to evaluate whether the proposed fTSDF encoding can give us a meaningful signal for semantic scene completion. As one can see from Figure 6.3 the 3D-CNN consists of several blocks. We concatenate the fTSDF before block 1 (early fusion) and also after block 1, 2 and 5 (fusion 1, 2 and 5). Before concatenation, both the color and the fTSDF input stream are processed separately. In the case of "late fusion" we take the maximum of the softmax probabilities of both streams.

As can be seen from Table 6.4, all fusion schemes perform slightly worse than our approach. This indicates that fTSDF provides a superfluous signal for our approach. This is interesting since computing the flipped TSDF volume is the most time-consuming part for inference and our approach provides a substantial faster alternative while also increasing the accuracy.

6.3.4 Comparison to the State-of-the-Art

We evaluate our approach, in the following denoted as 'Two-Stream', on the two test sets NYU CAD and NYU Kinect, which are in the following denoted as CAD and Kinect, and we compare our approach to the state-of-the-art.The results for scene completion and semantic scene completion are

reported in Table 6.5. Both of our approaches with 3-channel input encoding and one-hot encoding perform comparably. Since one-hot encoding yields a slightly higher accuracy, we only discuss the difference between the latter and other approaches from the literature.

Our approach sets the new state-of-the-art for semantic scene completion. We achieve 46.2 % on CAD and 34.1 % on Kinect and outperform the baseline approach by *Song et al.* (2017) by +6.2 % on CAD and +3.6 % on Kinect, although they use SUNCG as additional training data. If the same training data, i.e. only NYU, is used, our approach outperforms *Song et al.* (2017) by +9.4 % on Kinect. For scene completion, we outperform *Song et al.* (2017) by +5.8 % on CAD and +4.9 % on Kinect if NYU is used as training data. However, even if *Song et al.* (2017) uses additional training data from SUNCG, our approach still outperforms it by +2.9 % on CAD and +3.4 % on Kinect.

Compared to the recent ESSCN approach (*Zhang et al.*, 2018), we perform better in both scene completion (+3.8 %) and semantic scene completion (+7.4 %). Note also that pretraining on SUNCG (*Song et al.*, 2017) and using a stronger 3D-CNN architecture (*Zhang et al.*, 2018) are orthogonal to our proposed method. One can assume that our performance would further increase by incorporating both ideas.

Table 6.5 also includes the results of other approaches that do not rely on end-to-end learning (*Zheng et al.*, 2016; *Firman et al.*, 2016; *Lin et al.*, 2013; *Geiger and Wang*, 2015). Furthermore, the approaches of *Zheng et al.* (2016) and *Firman et al.* (2016) only address scene completion but not semantic scene completion. These methods perform substantially worse than the end-to-end learning approaches.

6.4 Summary

In this work, we have proposed a two stream approach for 3D semantic scene completion. In contrast to previous works, the proposed approach leverages depth and semantic information of the visible part of the scene for this task. In our experiments, we have shown that the proposed three-channel encoding for the semantic volume is not only memory efficient but it also results in higher accuracies compared to a single-channel encoding and is competitive to a memory expensive one-hot encoding. The proposed approach achieves state-of-the-art results for semantic scene completion on the NYUv2 dataset while also providing much faster inference times than approaches based of TSDF input features.

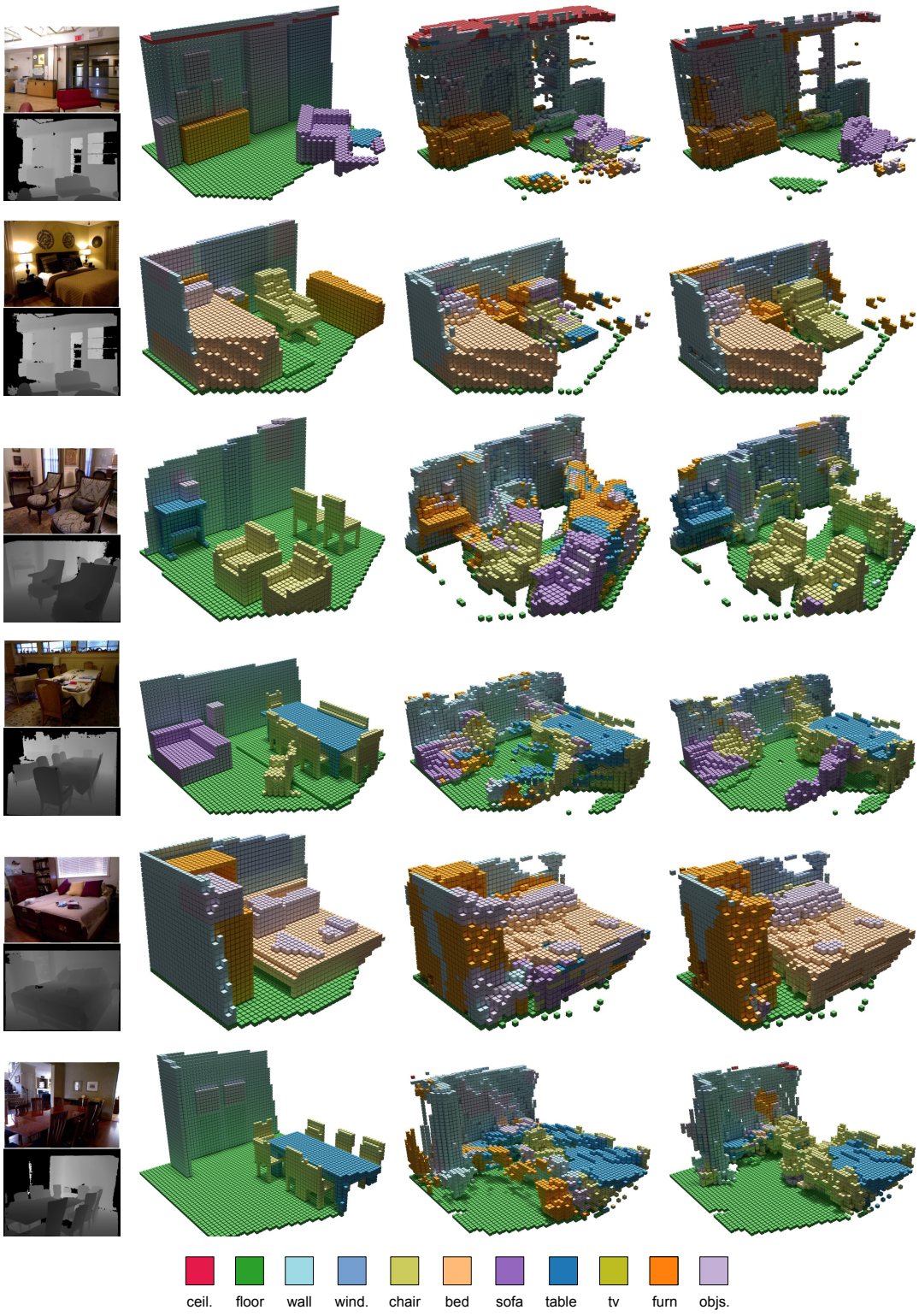

Figure 6.4: Qualitative results on NYUv2 Kinect. From left to right: Input RGB-D image, ground truth, result obtained by *Song et al.* (2017), and result obtained by our approach. Overall, our completed semantic 3D scenes are less cluttered and show a higher voxel class accuracy compared to *Song et al.* (2017).

CHAPTER 7

3D Semantic Scene Completion using Adversarial Training

We continue with the exploration of the field of 3D semantic scene completion. In this Chapter we explore the potential of the recently emerged generative adversarial networks for the task of semantic scene completion. In contrast to our "Two Stream" approach (last Chapter) the GAN approach will only use depth information as input, since preliminary experiments were showing that combining both on the NYU dataset slightly hurts the performance as compared to the vanilla two stream setup. We attribute this to the fact that NYU with its low number of training samples is not suitable to explore the combination of both methods. As our experiments will show, the full potential of adversarial training can only be seen on a the large scale, synthetic SUNCG dataset (*Song et al.*, 2017), which in turn is not suited for our Two Stream approach for its lack of RGB information. In light of the recently introduced generative adversarial networks (GAN), our goal is to explore the potential of this model and the efficiency of various important design choices. Our results show that using conditional GANs outperforms the vanilla GAN setup. We evaluate these architecture designs on several datasets. Based on our experiments, we demonstrate that GANs are able to outperform a baseline 3D CNN in case of clean annotations as provided in NYU CAD, but they suffer from poorly aligned annotations as in NYU Kinect. We propose a conditional generative adversarial network (GAN) to perform 3D semantic scene completion. We thoroughly evaluate the proposed approach and compare it with a standard generative adversarial network as well as in combination with a local adversarial loss. We observe that the conditional generative adversarial network performs best, but also that generative adversarial networks struggle if the ground truth is not well aligned with the depth data as in NYU Kinect.

Contents

7.1 Semantic Scene Completion with GANs

Inspired by the successful application of GANs in other domains, we introduce a novel model to perform semantic scene completion using GANs.

7.1.1 Network Architecture

Our model takes an encoded Truncated Signed Distance Function (TSDF) 3D scene from a depth image as input and predicts the fully voxelized 3D scene by a generator network. In this work we use the SSCNet (*Song et al.*, 2017) network architecture as our generator network. The discriminator network takes two kind of inputs, one is the generated 3D volume from the generator which should be classified as 'fake'. The generator network contains a softmax layer in the final layer. Therefore, it creates a probability map over C classes of size $H \times W \times D$, where H, W and D are the height, width and depth of the 3D volume. The input volume for the discriminator is then $C \times H \times W \times D$. We transform the ground truth samples to have the same size using a one-hot encoding. Although the discriminator network might easily distinguish the ground truth and the generated label maps by detecting whether the map consists of zeros and ones (one-hot encoding) or values between zero and one (output of semantic segmentation network), *Luc et al.* (2016) have shown that this encoding mechanism does not strongly affect the performance of the discriminator network.

For the architecture of the discriminator network we follow the design of *Wu et al.* (2016) and use several convolutional blocks which consist of a convolution layer with kernel size $4 \times 4 \times 4$ and stride $2 \times 2 \times 2$, a normalization layer and a Leaky ReLU activation layer. Due to the different size of the input dimension, we modified the kernel size accordingly. The output of the last convolutional layer with the size of $5 \times 3 \times 5 \times 16$ is reshaped to a vector of 1200 dimensions. After that, it is processed by three fully-connected layers with output sizes of 256, 128 and 1 respectively. Hence, the final output is a binary indicator to determine whether the predicted volumetric data is from the ground truth samples or not. An overview of our proposed architecture is shown in Figure 7.1.

7.1.2 Loss Function

We propose to use a hybrid loss function that is a weighted sum of two terms. The first term is a multi-class cross-entropy loss $\mathcal{L}_{mce}$ that is used for the generator to predict the right class label at each voxel location independently. We use $g(x)$ to denote the class probability map over C classes of size $C \times H \times W \times D$ that is produced by the generator network.

The second loss term is based on the output of the discriminator network. This loss term is large if the discriminator can differentiate between the output of the predicted data and ground truth label maps. We use $d(x, \hat{y}) \in [0, 1]$ to represent the scalar probability with which the discriminator network predicts that $\hat{y}$ is the ground truth label map of x, as opposed to being a label map produced by the generator model $g(\cdot)$. Given a dataset of N training images x_n and a corresponding 3D ground truth volume $\hat{y}_n$, we define the loss as:

$$\begin{aligned} \mathcal{L}_{GAN}(\theta_g, \theta_d) = &\sum_{n=1}^{N} \mathcal{L}_{mce}(g(x_n), \hat{y}_n) \\ &- \lambda[\mathcal{L}_{bce}(d(x_n, y_n), 1) + \mathcal{L}_{bce}(d(x_n, g(x_n)), 0)] \end{aligned} \tag{7.1}$$

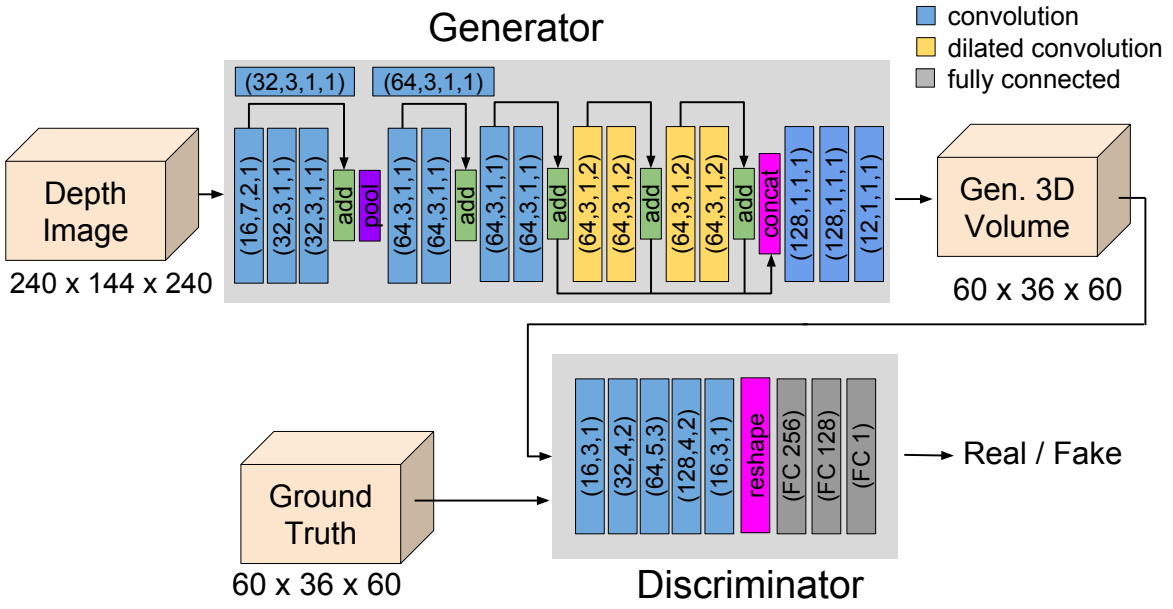

Figure 7.1: **Proposed network architecture.** The generator network takes a depth image as input and predicts a 3D volume. The discriminator network takes either the generated 3D volume or the ground truth volume as input, then classifies them as real or fake. The parameters of each layer are shown as (number of filters, kernel size, stride) in the case of convolutions and as (number of output channels) in the case of fully connected layers.

where θ_g and θ_d denote the parameters of the generator and discriminator network respectively. The multi-class cross-entropy loss for prediction y is given by:

$$\mathcal{L}_{mce}(y, \hat{y}) = -\sum_{i=1}^{H \times W \times D} \sum_{c=1}^{C} \hat{y}_{ic} \log y_{ic} \tag{7.2}$$

which equals the negative log-likelihood of the target ground truth volume $\hat{y}$ in a one-hot encoding representation. Similarly, the binary cross-entropy loss is denoted as:

$$\mathcal{L}_{bce}(z, \hat{z}) = -[\hat{z} \log(z) + (1 - \hat{z}) \log(1 - z)] \tag{7.3}$$

We then minimize the loss according to the parameters θ_g of the generator network, while maximizing it with respect to the parameters θ_d of the discriminator network as explained in Chapter 2.3.

7.1.3 Conditional GANs

Conditional GANs have been recently proposed in the literature. Since these works deal with different tasks ($e.g.$ 2D image generation), it is helpful to examine their potential for 3D semantic scene completion.

Using a conditional GAN, the output of the discriminator $d(x, \hat{y})$ is conditioned on the input x

which we denote as $d(x, \hat{y}|x)$. This leads to the following new objective function:

$$\begin{aligned}\mathcal{L}_{cGAN}(\theta_g, \theta_d) = &\sum_{n=1}^{N} \mathcal{L}_{mce}(g(x_n), \hat{y}_n) \\ &- \lambda[\mathcal{L}_{bce}(d(x_n, \hat{y}_n|x_n), 1) + \mathcal{L}_{bce}(d(x_n, g(x_n)|x_n), 0)]\end{aligned} \tag{7.4}$$

In practice, we achieve the conditioning by concatenating the input depth image x_n with the two kinds of inputs which are fed into the discriminator network respectively.

7.1.4 Local Adversarial Loss

A key observation for the discriminator network is that it should learn to model the input sample features equally within the whole input space. When we train a single strong discriminator network, the generator network tends to over-emphasize certain part of features to fool the current discriminator network. In other words, any local patch sampled from the input samples should have similar statistics to a real ground truth patch. Therefore, the idea of a local adversarial loss was proposed to overcome this problem in 2D images (*Shrivastava et al.*, 2017). Here, however, we extend this trick to the 3D domain. Rather than defining a global discriminator network, we can define a discriminator network that classifies each voxel separately. This division strategy can not only enhance the capacity of the discriminator network, but also provides more samples per input volume for learning. The generator network can also be improved by having multiple values per sample which gives more feedback for the generator network to learn.

In practice, we design the discriminator network to be a fully convolutional network that outputs the same dimension $C \times H \times W \times D$ as the input of discriminator. Instead of using fully connected layers to reduce the output into a single binary indicator, we upsample the output to match the ground truth dimensions. Since the discriminator has shrunk the input dimension within the middle three layers by the factor of 12, we again upsample the dimension by the factor of 12 using trilinear upsampling. We then calculate the loss term with binary cross-entropy.

7.2 Experiments

We implement our network architecture in PyTorch (*Paszke et al.*, 2017) and use batch size of 4. For our generator network, we use a SGD optimizer with weight decay of 0.0005 and learning rate of 0.01. For the discriminator network, we use an Adam optimizer with a learning rate of 0.0001. Besides, label smoothing is applied for improving the training process in all the experiments (*Salimans et al.*, 2016). We perform some experiments to determine the optimal value for the loss weight parameter λ in Equation (7.1) and (7.4). It turns out that $\lambda = 1$ performs the best.

We separate our evaluation results mainly in two parts: Semantic scene completion (SSC) and scene completion (SC). While scene completion only considers whether a voxel is occupied or empty, semantic scene completion also evaluates whether an occupied voxel is given the correct semantic label. As in *Song et al.* (2017) we measure the precision, recall and Jaccard index (IoU) for scene completion and the average (avg.) of the IoU across all categories for semantic scene completion.

7.2.1 Evaluation on NYU Depth v2

We first evaluate our models on the NYU Depth v2 dataset (*Silberman et al.*, 2012) and as in the previous Chapter we differentiate between NYU Kinect consisting of depth images captured by the Kinect sensor and NYU CAD consisting of rendered depth images generated by projection from the annotated CAD models. Table 7.1 and 7.2 show the results for 4 different design choices. Firstly, we examine using the standard GANs vs. using conditional GANs which we denote as 'GAN' and 'cGAN'. Secondly, the usage of global adversarial loss vs. local adversarial loss is examined, denoted as 'GL' and 'LL' respectively.

Table 7.1 shows the results on the NYU Kinect test set. On the one hand, applying global adversarial loss increases the accuracy from 20.6% to 22.7% when switching from GAN to conditional GAN on semantic scene completion. On the other hand, applying local adversarial loss decreases the accuracy by 1.2% for cGAN and by 0.7% for GAN. For scene completion, the IoU decreases by around 1% by applying conditional GAN. Applying local adversarial loss decreases the performance by less than 0.7%. Overall our model performs worse than the baseline (*Song et al.*, 2017), which achieves 24.7% on NYU Kinect, whereas our best model only achieves 22.7%. This suggests that our model is less robust to noise in the test data of NYU Kinect than that of *Song et al.* (2017). Due to the noisiness of the data in NYU Kinect in combination with the small amount of available training data, we conclude that NYU Kinect is not suited to examine the potential of adversarial learning. We therefore focus on the results on the NYU CAD test set.

Table 7.2 shows the results on the NYU CAD test set. Here the accuracy improves by around +2% (from 39.8% to 42.0%) by the usage of conditional GANs and further increases slightly to 42.3% by applying the local adversarial loss. For scene completion the performance also increases over the baseline of 70.3% when using adversarial loss and achieves maximum performance in the SSC-cGAN-LL model with 74.9%. Overall, the conditional GAN outperforms the standard GAN setup across both NYU test settings. Local and global adversarial loss, however perform differently on different test settings. The usage of the local adversarial loss tends to lay more emphasis on the finer local detail of each voxel. Since all networks are only trained on NYU CAD and the NYU Kinect test data contains noisy data points, the use of a local loss further decreases the performance on this test set. For scene completion, all of our network models provide higher precision with lower recall. We observe that *Song et al.* (2017) achieve a very high recall but a low precision for scene completion. In other words, this method tends to predict more occupied voxels than our approaches which results in cluttered scene completions. Nevertheless, on NYU CAD, our models outperform the baseline (*Song et al.*, 2017) by a large margin for both scene completion and semantic scene completion. As shown in Table 7.2, our model SSC-cGAN-LL achieves 42.3% accuracy and outperforms the approach by *Song et al.* (2017) by +4.7% for semantic scene completion. For scene completion, our model outperforms *Song et al.* (2017) with +4.6% IoU.

Apart from the quantitative results, we provide some qualitative results visualizing the effect of applying different GAN models in Figure 7.2. We can observe that the model using the global GAN loss (SSC-cGAN-GL) suffers from partial mode collapse, meaning it constantly generates fixed voxels in parts of the scene ($e.g$ ceilings, walls). Using a conditional GAN and the local adversarial loss simultaneously seems to significantly reduce this problem. On the other hand, the predicted scenes from SSC-cGAN-GL are visually more plausible since they display more fine structure of the objects, tend to contain more empty voxels and therefore look less cluttered. Thus, we propose

NYU Kinect	SC			SSC
method	prec.	recall	IoU	avg.
SSCNet (*Song et al.*, 2017)	57.0	94.5	55.1	24.7
Zhang et al. (2018)	71.9	71.9	56.2	26.7
Two-Stream (ours)	65.6	87.2	60.0	34.1
Liu et al. (2018) *	67.3	85.8	**60.6**	**34.4**
SSCNet*, *Song et al.* (2017)	59.3	92.9	56.6	30.5
SSC-GAN-GL	65.3	84.8	**58.2**	20.6
SSC-GAN-LL	64.5	85.9	58.1	19.9
SSC-cGAN-GL	63.1	87.8	57.8	**22.7**
SSC-cGAN-LL	64.0	84.8	57.1	21.5

Table 7.1: Comparison of models for semantic scene completion results on the NYU Kinect. * denotes that the network is trained on SUNCG and fine-tuned on NYU.

NYU CAD	SC			SSC
method	prec.	recall	IoU	avg.
SSCNet (*Song et al.*, 2017)	75.0	92.3	70.3	37.6
SSCNet* (*Song et al.*, 2017)	75.4	96.3	73.2	40.0
Two-Stream (ours)	81.6	92.4	**76.1**	**46.2**
SSC-GAN-GL	81.1	90.6	74.8	39.8
SSC-GAN-LL	80.6	91.3	73.9	40.6
SSC-cGAN-GL	80.7	91.1	74.8	42.0
SSC-cGAN-LL	81.0	91.0	74.9	42.3

Table 7.2: Comparison of models for semantic scene completion on the NYU CAD (*Firman et al.*, 2016). * denotes that the network is trained on SUNCG and fine-tuned on NYU.

SSC-cGAN-GL as the most effective model, due to the fact that it performs comparably to the model with the highest accuracy (SSC-cGAN-LL), while being able to generate more realistic results.

Finally, we compare our approach to the state-of-the-art. On NYU Kinect we perform worse than the baseline by *Song et al.* (2017), but the only approach that fairly outperforms the baseline is that of *Zhang et al.* (2018). All the other approaches either use additional input information inferred from RGB images ('Two-Stream'), pretrain on SUNCG (*Song et al.*, 2017) or both (*Liu et al.*, 2018). On NYU CAD, we outperform *Song et al.* (2017) even if they pretrain on SUNCG and perform competitively with the Two-Stream approach. It is worth noting that the approach 'Two-Stream' is the our approach which has been described in the previous Chapter.

7.2.2 Evaluation on SUNCG

The SUNCG dataset *Song et al.* (2017) is a synthetic dataset, which provides a large amount of training data with rendered depth images and volumetric ground truth. It contains 45,622 different scenes with realistic room and furniture layouts. However, due to the high computational cost (around 15 days for 10 epochs) of the training procedure, we only run our SSC-cGAN-GL model on SUNCG.

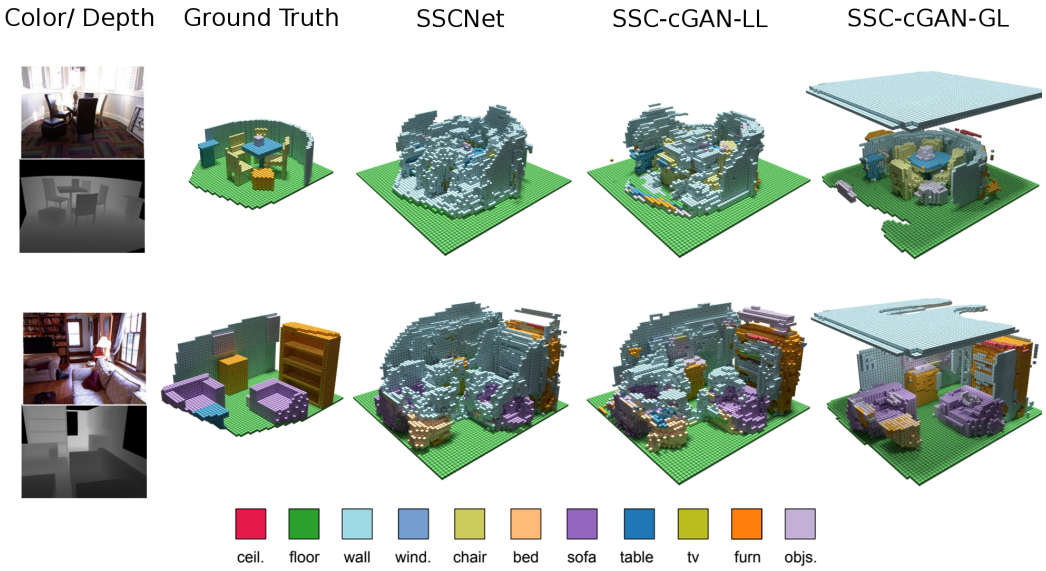

Figure 7.2: **Qualitative results on NYU CAD.** The first three columns show the input depth image with its corresponding color image, ground truth volume, and the results obtained by *Song et al.* (2017). The fourth and fifth columns show the results obtained by our approaches.

The result is shown in Table 7.3. Our model SSC-cGAN-GL outperforms the baseline *Song et al.* (2017) by a large margin of +10% on semantic scene completion. For scene completion the IoU increases from 72.9% to 78.1% with both a higher precision and recall. As a result, we infer that the GAN structure benefits from the large amount of training samples provided by SUNCG.

Compared to other recent approaches, the effect of using adversarial learning is smaller. However, since our approach is orthogonal to the approaches of *Zhang et al.* (2018) and *Liu et al.* (2018), we expect that they could be combined and potentially benefit each other. *Wang et al.* (2018c) also used an adversarial learning approach in combination with a encoder-decoder network. However they do not follow the original evaluation protocol. Instead of reporting numbers on the SUNCG test set, they perform a 10-fold cross validation using random splits. For comparison to the state-of-the-art, we follow the standard evaluation protocol *Song et al.* (2017).

7.2.3 Loss Behaviour of the Discriminator

We design an experiment that allows us to assess whether the discriminators show the expected behaviour of producing high losses for unrealistic scene inputs. Therefore, we gradually add noise to the ground truth samples of the NYU CAD test data and feed it as input to the trained discriminator networks. We simulate the noise by randomly changing voxel labels in the occluded space. While increasing the percentage of noise, we calculate the binary cross-entropy loss value using Equation (7.3). As one can see from Figure 7.3, the loss curve for SSC-GAN remains stable whereas for SSC-cGAN it increases. This suggests that the conditional GAN model behaves in the expected way while the standard GAN is not sensitive to the noise.

SUNCG	SC			SSC
method	prec.	recall	IoU	avg.
SSCNet *Song et al.* (2017)	79.8	89.5	72.9	45.0
Wang et al. (2018c) *	-	-	-	51.4
Zhang et al. (2018)	92.6	90.4	**84.5**	**70.5**
Liu et al. (2018)	80.7	96.5	78.5	64.3
SSC-cGAN-GL	83.4	92.4	78.1	55.6

Table 7.3: Comparison with the state-of-the-art networks on the SUNCG dataset. * denotes that the model uses different training / testing splits and performs a 10-fold cross-validation (*Wang et al.*, 2018c)

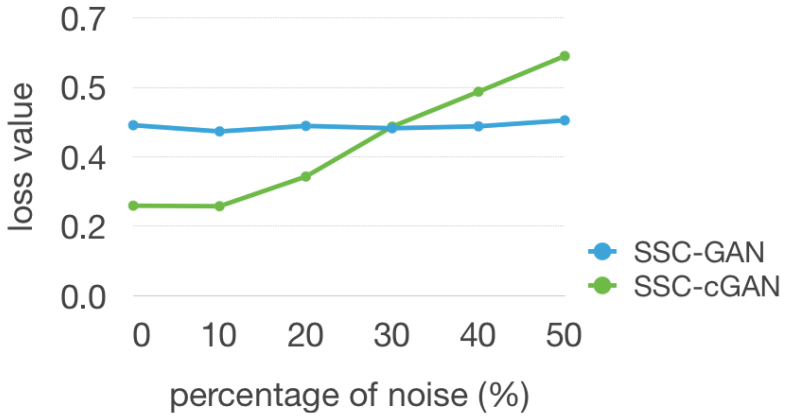

Figure 7.3: Comparison of the loss behaviour of the discriminator networks.

7.3 Summary

We presented a novel GAN architecture to perform 3D semantic scene completion. Also, we evaluated two variations of the architecture design: conditional GANs and local adversarial loss. The results show that the conditional GAN improves the network performance on both test sets, while the local adversarial loss only improves the performance on NYU CAD but not on NYU Kinect. In comparison to the baseline *Song et al.* (2017), our models yield a significant improvement on NYU CAD. On SUNCG our models outperform the baseline by a large margin. Qualitatively, our proposed model SSC-cGAN-GL produces significantly more realistic appearing results. For the effective combination of our 'Two-Stream' approach explained in the previous Chapter and the GAN setup introduced in this Chapter, we suggest that a much larger dataset than NYU is needed.

CHAPTER 8

A Dataset for Semantic Segmentation of Point Cloud Sequences

We have seen in the previous two Chapters that deep learning approaches to 3D semantic scene completion suffer from the lack of data provided in the NYU dataset. The SUNCG dataset is purely synthetic and does not provide RGB images along with the depth measurements. Therefore there is a need for a realistic, large scale dataset to perform 3D semantic scene completion, which we address in this Chapter. We introduce a large dataset, called '*SemanticKITTI*', to propel research on semantic scene completion as well as laser-based semantic segmentation. Therefore, we annotated all sequences of the KITTI Vision Odometry Benchmark and provide dense point-wise annotations for the complete 360^o field-of-view of the employed automotive LiDAR. Based on this new dataset, we propose three benchmark tasks : (i) semantic segmentation of point clouds using a single scan, (ii) semantic segmentation using sequences comprising of multiple past scans, and (iii) semantic scene completion, which, in our case, requires to anticipate the semantic scene in the future. We provide baseline experiments and show that there is a need for more sophisticated models to efficiently tackle all of these tasks. Our dataset opens the door for the development of more advanced methods, but also provides plentiful data to investigate new research directions.

Contents

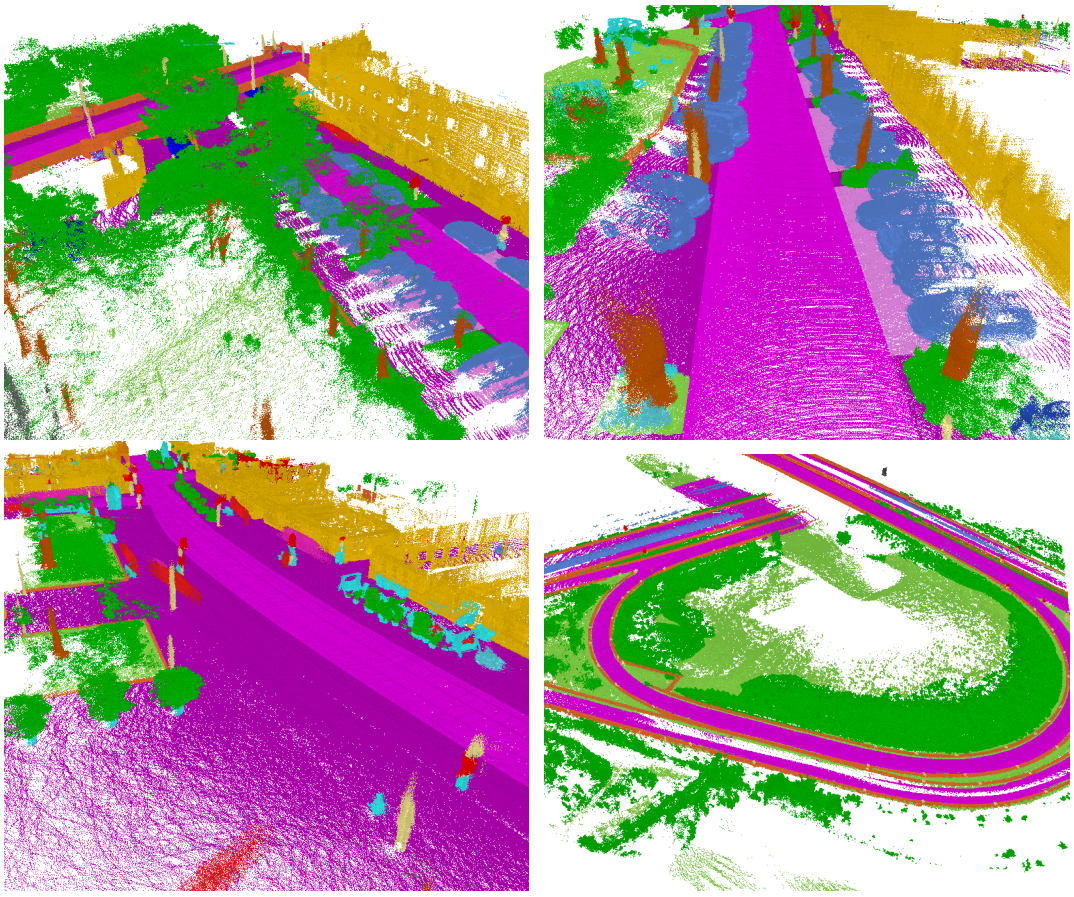

Figure 8.1: Our dataset provides dense annotations for each scan of all sequences from the KITTI Odometry Benchmark (*Geiger et al.*, 2012). Here, we show multiple scans aggregated using pose information estimated by a SLAM approach. See also the video[a] to get a better impression of the consistency and quality of the provided annotation.

[a] `http://www.ipb.uni-bonn.de/html/projects/semantic_kitti/videos/teaser.mp4`

	#scans[1]	#points[2]	#classes[3]	sensor	annotation	sequential
SemanticKITTI (Ours)	**23201/20351**	**4549**	**25** (28)	Velodyne HDL-64E	point-wise	✓
Oakland3d	17	1.6	5 (44)	SICK LMS	point-wise	✗
Freiburg	77	1.1	4 (11)	SICK LMS	point-wise	✗
Wachtberg	5	0.4	5 (5)	Velodyne HDL-64E	point-wise	✗
Semantic3d	15/15	4009	8 (8)	Terrestrial Laser Scanner	point-wise	✗
Paris-Lille-3D	3	143	9 (50)	Velodyne HDL-32E	point-wise	✗
Zhang et al.	140/112	32	10 (10)	Velodyne HDL-64E	point-wise	✗
KITTI	7481/7518	1799	3	Velodyne HDL-64E	bounding box	✗

Table 8.1: Overview of other point cloud datasets with semantic annotations. Ours is by far the largest dataset with sequential information. [1]Number of scans for train and test set, [2]Number of points is given in millions, [3]Number of classes used for evaluation and number of classes annotated in brackets.

8.1 Introduction

Semantic scene understanding is essential for many applications and an integral part of self-driving cars. Particularly, fine-grained understanding provided by semantic segmentation is necessary to distinguish drivable and non-drivable surfaces and to reason about functional properties, like parking areas and sidewalks. Currently, such understanding, represented in so-called high definition maps, is mainly generated in advance using surveying vehicles. However, self-driving cars should also be able to drive in unmapped areas and adapt their behaviour if there are changes in the environment.

Most self-driving cars currently use multiple different sensors to perceive the environment. Complementary sensor modalities enable to cope with deficits or failures of particular sensors. Besides cameras, light detection and ranging (LiDAR) sensors are often used as they provide precise distance measurements that are not affected by lighting.

Publicly available datasets and benchmarks are crucial for empirical evaluation of research. They mainly fulfill three purposes: (i) they provide a basis to measure progress, since they allow to provide results that are reproducible and comparable, (ii) they uncover shortcomings of the current state-of-the-art and therefore pave the way for novel approaches and research directions, and (iii) they make it possible to develop approaches without the need to first painstakingly collect and label data. While multiple large datasets for *image-based* semantic segmentation exist (*Cordts et al.*, 2016; *Neuhold et al.*, 2017), publicly available datasets with point-wise annotation of three-dimensional point clouds are still comparably small, as shown in Table 8.1.

To close this gap we propose *SemanticKITTI*, a large dataset showing unprecedented detail in point-wise annotation with 28 classes, which is suited for various tasks. In this paper, we mainly focus on *laser-based* semantic segmentation, but also semantic scene completion. The dataset is distinct from other laser datasets as we provide accurate scan-wise annotations of sequences. Overall, we annotated all 22 sequences of the odometry benchmark of the *KITTI Vision Benchmark* (*Geiger et al.*, 2012) comprised of over 43 000 scans. Moreover, we labeled the complete horizontal 360° field-of-view of the rotating laser sensor. Figure 8.1 shows example scenes from the provided dataset. In summary, our main contributions are:

- We present a point-wise annotated dataset of point cloud sequences with an unprecedented

number of classes and unseen level-of-detail for each scan.

- We furthermore provide an evaluation of state-of-the-art methods for semantic segmentation of point clouds.
- We investigate the usage of sequence information for semantic segmentation using multiple scans.
- Based on the annotation of sequences of a moving car, we furthermore introduce a real-world dataset for semantic scene completion and provide baseline results.
- Together with a benchmark website, we will also publish our point cloud labeling tool enabling other researchers to generate other labeled datasets in future.

We believe that providing this large dataset to the research community will stimulate the development of novel algorithms, make it possible to investigate new research directions, and puts evaluation and comparison of these novel algorithms on a more solid ground.

8.2 The SemanticKITTI Dataset

Our dataset is based on the odometry dataset of the KITTI Vision Benchmark (*Geiger et al.*, 2012) showing inner city traffic, residential areas, but also highway scenes and countryside roads around Karlsruhe, Germany. The original odometry dataset consists of 22 sequences, splitting sequences 00 to 10 as training set, and 11 to 21 as test set. For consistency with the original benchmark, we adopt the same division for our training and test set. Moreover, we do not interfere with the original odometry benchmark by providing labels only for the training data. Overall, we provide 23 201 full 3D scans for training and 20 351 for testing, which makes it, to the best of our knowledge, the largest dataset publicly available by a wide margin.

We decided to use the KITTI dataset as a basis for our labeling effort, since it allowed us to exploit one of the largest available collections of raw point cloud data captured with a car. We furthermore expect that there are also potential synergies between our annotations and the existing benchmarks and this will enable the investigation and evaluation of additional research directions, such as the usage of semantics for laser-based odometry estimation.

Compared to other datasets (cf. Table 8.1), we provide labels for sequential point clouds generated with a commonly used laser range sensor in autonomous driving applications, *i.e.* the Velodyne HDL-64E. Other publicly available datasets, like *Paris-Lille-3D* (*Roynard et al.*, 2018) or *Wachtberg* (*Behley et al.*, 2012), also use such sensors, but only provide the aggregated point cloud of the whole acquired sequence or some individual scans of the whole sequence, respectively. Since we provide the individual scans of the whole sequence, one can also investigate how aggregating multiple consecutive scans influences the performance of the semantic segmentation and use the information to recognize moving objects.

We annotated 28 classes, where we ensured a large overlap of classes with the *Mapillary Vistas* dataset (*Neuhold et al.*, 2017) and *Cityscapes* dataset (*Cordts et al.*, 2016) and made modifications where necessary to account for the sparsity and vertical field-of-view. More specifically, we do not distinguish between persons riding a vehicle and the vehicle, but label the vehicle and the person as either *bicyclist* or *motorcyclist*.

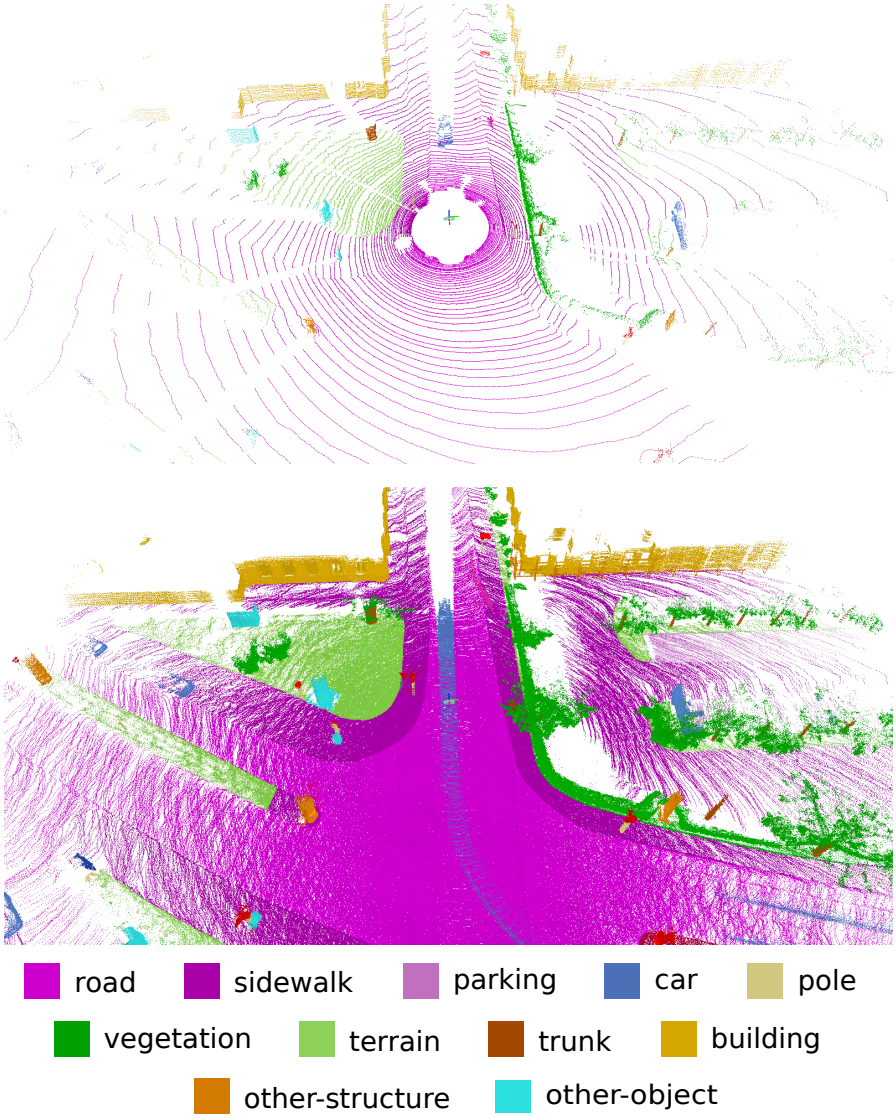

Figure 8.2: (Top) Single scan and (Bottom) multiple superimposed scans with labels. Also shown is a moving car in the center of the image resulting in a trace of points.

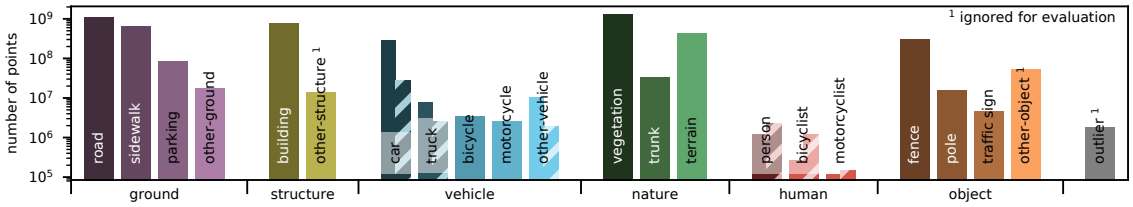

Figure 8.3: Label distribution. Number of labeled points per class. Also shown are the root categories for the classes. For movable classes, we also show the number of points on non-moving (solid bars) and moving objects (hatched bars).

We furthermore distinguished between moving and non-moving vehicles and humans, *i.e.* a vehicle or humans gets the corresponding moving class if they moved in some scan while observing them, as shown in the lower part of Figure 8.2. All annotated classes are listed in Figure 8.3 and a more detailed discussion and definition of the different classes can be found in the Section 8.6. In summary, we have 28 classes, where 6 classes are assigned the attribute moving or non-moving, and one *outlier* class is included for erroneous laser measurements caused by reflections or other effects.

We will make the dataset publicly available through a benchmark website, following common best practices to provide only the training set with ground truth labels and perform the test set evaluation online. We furthermore will also limit the number of possible test set evaluations to prevent overfitting to the test set (*Torralba and Efros*, 2011).

8.2.1 Labeling Process

To make the labeling of point cloud sequences practical, we superimpose multiple scans above each other, which conversely allows us to label multiple scans consistently. To this end, we first register and loop close the sequences using an off-the-shelf laser-based SLAM system (*Behley and Stachniss*, 2018). This step is needed as the provided information of the inertial navigation system (INS) often results in map inconsistencies, *i.e.* streets that are revisited after some time have different height. For three sequences, we had to manually add loop closure constraints to get correctly loop closed trajectories, since this is essential to get consistent point clouds for annotation. The loop closed poses allow us to load all overlapping point clouds for specific locations and visualize them together, as depicted in Figure 8.2.

We subdivide the sequence of point clouds into tiles of 100 m by 100 m. For each tile, we only load scans overlapping with the tile. This enables us to label all scans consistently even when we encounter temporally distant loop closures. To ensure consistency for scans overlapping with more than one tile, we show all points inside each tile and a small boundary overlapping with neighboring tiles. Thus, it is possible to continue labels from a neighboring tile.

Following best practices, we compiled a labeling instruction and provided instructional videos on how to label certain objects, such as cars and bicycles standing near a wall. Compared to image-based annotation, the annotation process with point clouds is more complex, since the annotator often needs to change the viewpoint. An annotator needs on average 4.5 hours per tile, when labeling residential areas corresponding to the most complex encountered scenery, and needs on average 1.5 hours for labeling a highway tile.

We explicitly did not use bounding boxes or other available annotations for the KITTI dataset,

since we want to ensure that the labeling is consistent and the point-wise labels should only contain the object itself.

We provided regular feedback to the annotators to improve the quality and accuracy of labels. Nevertheless, a single annotator also verified the labels in a second pass, *i.e.* corrected inconsistencies and added missing labels. In summary, the whole dataset comprises 518 tiles and over $1\,400$ hours of labeling effort have been invested with additional $10 - 60$ minutes verification and correction per tile, resulting in a total of over $1\,700$ hours.

8.2.2 Dataset Statistics

Figure 8.3 shows the distribution of the different classes, where we also included the root categories as labels on the x-axis. The ground classes, *road*, *sidewalk*, *building*, *vegetation*, and *terrain* are the most frequent classes. The class *motorcyclist* only occurs rarely, but still more than $100\,000$ points are annotated.

The unbalanced count of classes is common for datasets captured in natural environments and some classes will be always under-represented, since they do not occur that often. Thus, an unbalanced class distribution is part of the problem that an approach has to master. Overall, the distribution and relative differences between the classes is quite similar in other datasets, *e.g. Cityscapes* (*Cordts et al.*, 2016).

8.3 Semantic Point Cloud Segmentation

In this section, we provide the evaluation of several state-of-the-art methods for semantic segmentation of a single scan. We also provide experiments exploiting information provided by sequences of multiple scans.

8.3.1 Single-Scan Input

Task and Metrics

In semantic segmentation of point clouds, we want to infer the label of each three-dimensional point. Therefore, the input to all evaluated methods is a list of coordinates of the three-dimensional points along with their remission, *i.e.* the strength of the reflected laser beam which depends on the properties of the surface that was hit and its distance. Each method should then output a label for each point of a scan, *i.e.* one full turn of the rotating LiDAR sensor.

To assess the labeling performance, we rely on the commonly applied mean Jaccard Index or mean intersection-over-union (mIoU) metric (*Everingham et al.*, 2015) over all classes, given by

$$\frac{1}{C} \sum_{c=1}^{C} \frac{\mathrm{TP}_c}{\mathrm{TP}_c + \mathrm{FP}_c + \mathrm{FN}_c}, \tag{8.1}$$

where TP_c, FP_c, and FN_c correspond to the number of true positive, false positive, and false negative predictions for class c, and C is the number of classes.

As the classes *other-structure* and *other-object* have either only a few points and are otherwise too diverse with a high intra-class variation, we decided to not include these classes in the evaluation.

Thus, we use 25 instead of 28 classes, ignoring *outlier*, *other-structure*, and *other-object* during training and inference.

Furthermore, we cannot expect to distinguish moving from non-moving objects with a single scan, since this Velodyne LiDAR cannot measure velocities like radars exploiting the Doppler effect. We therefore combine the moving classes with the corresponding non-moving class resulting in a total number of 19 classes for training and evaluation.

Baselines

We provide the results of six state-of-the-art architectures for the semantic segmentation of point clouds in our dataset: PointNet (*Qi et al.*, 2017a), PointNet++ (*Qi et al.*, 2017b), Tangent Convolutions (*Tatarchenko et al.*, 2018), SPLATNet (*Su et al.*, 2018), Superpoint Graph (*Landrieu and Simonovsky*, 2018), and SqueezeSeg (*Wu et al.*, 2018). Furthermore, we investigate two extensions of SqueezeSeg: Darknet21Seg and Darknet53Seg.

PointNet and PointNet++ use the raw un-ordered point cloud data as input. Core of these approaches is max pooling to get an order-invariant operator that works surprisingly well for semantic segmentation of shapes and several other benchmarks. Due to this nature, however, PointNet fails to capture the spatial relationships between the features. To alleviate this, PointNet++ applies individual PointNets to local neighborhoods and uses a hierarchical approach to combine their outputs. This enables it to build complex hierarchical features that capture both local fine-grained and global contextual information.

Tangent Convolutions also handles unstructured point clouds by applying convolutional neural networks directly on surfaces. This is achieved by assuming that the data is sampled from smooth surfaces and defining a tangent convolution as a convolution applied to the projection of the local surface at each point into the tangent plane.

SPLATNet takes an approach that is similar to the aforementioned voxelization methods and represents the point clouds in a high-dimensional sparse lattice. As with voxel-based methods, this scales poorly both in computation and in memory cost and therefore they exploit the sparsity of this representation by using bilateral convolutions, which only operates on occupied lattice parts.

Similarly to PointNets, Superpoint Graph, captures the local relationships by summarizing geometrically homogeneous groups of points into superpoints, which are later embedded by local PointNets. The result is a superpoint graph representation that is more compact and rich than the original point cloud exploiting contextual relationships between the superpoints.

SqueezeSeg also discretizes the point cloud in a way that makes it possible to apply 2D convolutions to the point cloud data exploiting the sensor geometry of a rotating LiDAR. In the case of a rotating LiDAR, all points of a single turn can be projected to an image by using a spherical projection. A fully convolutional neural network is applied and then finally filtered with a CRF to smooth the results. Due to the promising results of SqueezeSeg and its fast training, we investigate how the labeling performance is affected by the number of model parameters. To this end, we use a different 2D CNN backbone based on the Darknet architecture (*Redmon and Farhadi*, 2018) with 21 and 53 layers, and 25 and 50 million parameters respectively. We furthermore eliminate the vertical downsampling used in the architecture.

We modify the available implementations such that the methods can be trained and evaluated on our large-scale dataset with very sparse point clouds due to the LIDAR sensor. Note that most of

Approach	road	sidewalk	parking	other-ground	building	car	truck	bicycle	motorcycle	other-vehicle	vegetation	trunk	terrain	person	bicyclist	motorcyclist	fence	pole	traffic sign	mIoU
PointNet	61.6	35.7	15.8	1.4	41.4	46.3	0.1	1.3	0.3	0.8	31.0	4.6	17.6	0.2	0.2	0.0	12.9	2.4	3.7	14.6
SPGraph	45.0	28.5	0.6	0.6	64.3	49.3	0.1	0.2	0.2	0.8	48.9	27.2	24.6	0.3	2.7	0.1	20.8	15.9	0.8	17.4
SPLATNet	64.6	39.1	0.4	0.0	58.3	58.2	0.0	0.0	0.0	0.0	71.1	9.9	19.3	0.0	0.0	0.0	23.1	5.6	0.0	18.4
PointNet++	72.0	41.8	18.7	5.6	62.3	53.7	0.9	1.9	0.2	0.2	46.5	13.8	30.0	0.9	1.0	0.0	16.9	6.0	8.9	20.1
SqueezeSeg	85.4	54.3	26.9	4.5	57.4	68.8	3.3	16.0	4.1	3.6	60.0	24.3	53.7	12.9	13.1	0.9	29.0	17.5	24.5	29.5
TangentConv	83.9	63.9	33.4	15.4	83.4	90.8	15.2	2.7	16.5	12.1	79.5	49.3	58.1	23.0	28.4	8.1	49.0	35.8	28.5	40.9
DarkNet21Seg	91.4	74.0	57.0	26.4	81.9	85.4	18.6	26.2	26.5	15.6	77.6	48.4	63.6	31.8	33.6	4.0	52.3	36.0	50.0	47.4
DarkNet53Seg	91.8	74.6	64.8	27.9	84.1	86.4	25.5	24.5	32.7	22.6	78.3	50.1	64.0	36.2	33.6	4.7	55.0	38.9	52.2	49.9

Table 8.2: Single scan results (19 classes) for all baselines on sequences 11 to 21 (test set). All methods were trained on sequences 0 to 10, except for 8 which is left as validation.

these approaches have so far only been evaluated on small RGB-D indoor datasets.

We restrict the number of points within a single scan due to memory limitations on some approaches (PointNet, PointNet++) to 50,000 via random sampling.

For SPLATNet, we use the SPLATNet3D[1] architecture. The input consists of the 3D position of each point as well as its normal. The normals have been pre-estimated given 30 closest neighbors.

For TangentConv[2] we use the existing configuration for Semantic3D. We speed up the training and validation procedures by precomputing scan batches and add an asynchronous data loading. Complete single scans are provided during training. In the multi scan experiment we fix the number of points per batch to 500,000 due to memory constraints and start training from the single scan weights.

For SqueezeSeg and its Darknet backbone equivalents, we use a spherical projection of the scans in the same way as the original SqueezeSeg approach. The projection contains 64 lines in height corresponding to the separate beams of the sensor, and extrapolating the configuration of SqueezeSeg which only uses the front 90° and a horizontal resolution of 512, we use 2,048 for the entire scan. Because some points are duplicated in this sampling process, we always keep the closest range value, and during inference of each scan we iterate over the entire point list and check it's semantic value in the output grid.

An overview of the used parameters is given in Table 8.8. We furthermore provide the number of trained epochs and if we could get a results which seems to be converged in the given amount of time.

Results and Discussion

Table 8.2 shows the results of our baseline experiments for various approaches using either directly the point cloud information (PointNet, PointNet++, SPGraph, TangentConv, SPLATNet) or a projection of the point cloud (SqueezeSeg, DarkNet21Seg, DarkNet53Seg). The results show that the current state-of-the-art for point cloud semantic segmentation falls short for the size and complexity of our dataset.

We believe that this is mainly caused by the limited complexity of the used architectures (see

[1] `https://github.com/NVlabs/splatnet`

[2] `https://github.com/tatarchm/tangent_conv`

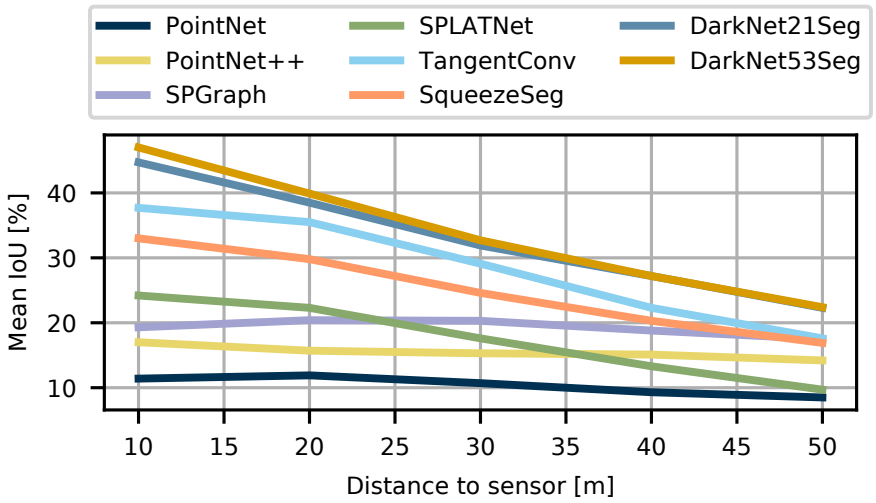

Figure 8.4: IoU vs. distance to the sensor.

Approach	num. parameters (million)	train time $\left(\frac{\text{GPU hours}}{\text{epoch}}\right)$	inference time $\left(\frac{\text{seconds}}{\text{point cloud}}\right)$
PointNet	3	4	0.5
PointNet++	6	16	5.9
SPGraph	0.25	6	5.2
TangentConv	0.4	6	3.0
SPLATNet	0.8	8	1.0
SqueezeSeg	1	0.5	0.015
DarkNet21Seg	25	2	0.055
DarkNet53Seg	50	3	0.1

Table 8.3: Approach statistics.

Table 8.3), because the number of parameters of these approaches is much lower than the number of parameters used in leading image-based semantic segmentation networks. As mentioned above, we add Darknet21Seg and DarkNet53Seg to test this hypothesis and the results show that this simple modification improves the accuracy from $29.5\,\%$ for SqueezeSeg to $47.4\,\%$ for DarkNet21Seg to $49.9\,\%$ for Darknet53Seg.

Another reason is that the point clouds generated by LiDAR are relatively sparse, especially as the distance to the sensor increases. This is partially solved in SqueezeSeg, which exploits the way the rotating scanner captures the data to generate a dense range image, where each pixel corresponds roughly to a point in the scan. Finally, occlusions may pose a problem for approaches that rely on raw point clouds due to more discontinuities in the surface of occluded objects.

These effects are further analyzed in Figure 8.4, where the mIoU is plotted w.r.t the distance to the sensor. It is clearly visible that all approaches show a correlation of distance and classification performance, *i.e.* results get worse with increasing distance. This further confirms our hypothesis that the sparsity is the main reason for worse results at large distances. However, the results also show that some methods, like SPGraph, are less affected by the distance-dependent sparsity and this might be a promising direction for future research to combine the strength of both paradigms.

Approach		Scans	mIoU	mAcc
TangentConv		1	35.9	83.3
		5	**39.8**	**84.9**
Darknet53Seg	64×2048	1	45.1	85.8
		5	46.3	86.7
	96×2048	5	**46.5**	**87.1**

Table 8.4: Single scan vs. multiple scan results.

Finally, the best performing approach (Darknet53Seg) with 49.9% mIoU is still far from achieving results that are on par with image-based approaches, *e.g.* 80% on the *Cityscapes* benchmark (*Cordts et al.*, 2016). Figure 8.5 shows qualitative results from every used approach.

8.3.2 Effect of Sequential Information

Since we provide sequences of laser scans with locally consistent pose information, we can also use an aggregated scan to increase the density of the cloud. With this aggregated information, we run additional experiments with approaches capable of exploiting this information, *i.e.* approaches that are not limited by memory constraints and can be easily adapted to larger point clouds. To this end, we retrained Tangent Convolutions and our ablation of SqueezeSeg using the Darknet53 backbone, both of which performed best in the single scan task and scale well to the number of points in the scans.

In the interest of showing how more information can help the projective methods, Figure 8.6 shows the effect of increasing the number of scans used from 1 to 5. The figure illustrates the effect of aggregating the point clouds to make a richer range image, which allows us to increase the resolution of the input. Due to the nature of this projection, however, not all points of the aggregated scan can be inferred, but only the ones that are visible by the sensor in the last scan. In order to achieve this, we render the range images using Z-buffering and we only evaluate how the multi-scan inference can help the current scan, and not the entire history. Another problem with this naive aggregation of scans is that it does not deal properly with dynamic objects, and therefore they will generate blurred clouds that may affect the inference quality.

Even though the approach taken for the aggregation of the scans was very simple, and didn't deal with the dynamic objects explicitly, Table 8.4 shows promising results when using multiple scans vs. single scan inference. For the projective method, as Figure 8.6 suggests, the aggregation of the scans allows us to generate richer and higher resolution range images, filling in the holes that hurt the performance. In case of Tangent Convolutions, which also turns the point cloud into a representation where planar convolutions can be locally applied, the multiple scans makes the data respect the assumption of locally Euclidean surfaces more closely, increasing the performance. We expect that our dataset will spawn a new category of methods that deals with this type of sequential information to increase the performance of the semantic segmentation task.

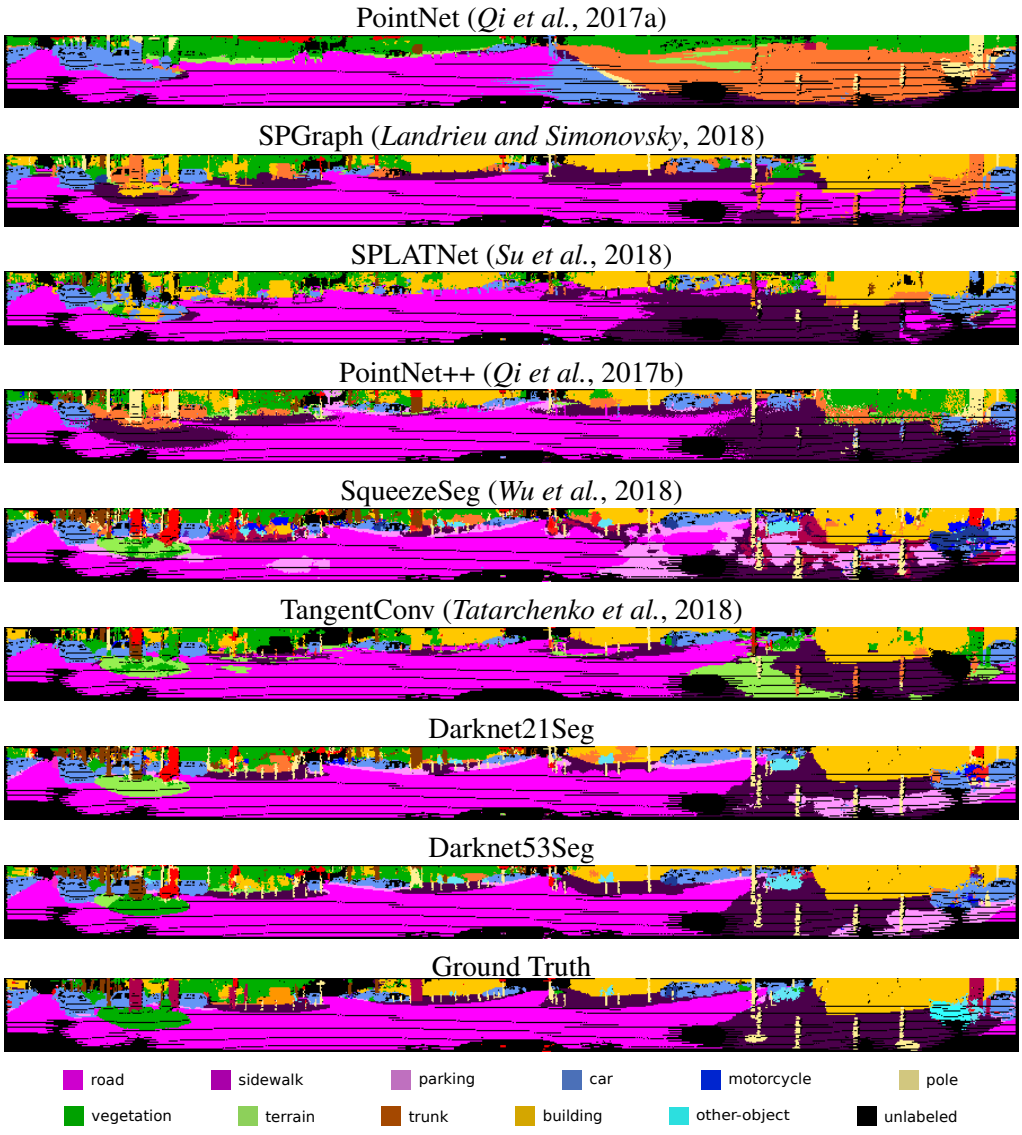

Figure 8.5: Examples of inference for all methods. The point clouds were projected to 2D using a spherical projection to make the comparison easier.

Approach	car	truck	other-vehicle	person	bicyclist	motorcyclist	mIoU
TangentConv	84.9	21.1	18.5	1.6	0.0	0.0	34.1
	40.3	42.2	30.1	6.4	1.1	1.9	
DarkNet53Seg	84.1	20.0	20.7	7.5	0.0	0.0	41.6
	61.5	37.8	28.9	15.2	14.1	0.2	

Table 8.5: IoU results using a sequence of multiple past scans (in %). Shaded cells correspond to the IoU of the moving classes, while unshaded entries are the non-moving classes.

8.3.3 Multi-Scan Input for Motion Segmentation

Task and Metrics

In this task, we allow methods to exploit information from a sequence of multiple past scans to improve the segmentation of the current scan. We furthermore want the methods to distinguish moving and non-moving classes, *i.e.* all 25 classes must be predicted, since this information should be visible in the temporal information of multiple past scans. The evaluation metric for this task is still the same as in the single scan case, *i.e.* we evaluate the mean IoU of the current scan no matter how many past scans are used to compute the results.

Baselines

We exploit the sequential information by combining 5 scans into a single, large point cloud, *i.e.* the current scan at timestamp t and the 4 scan before at timestamps $t-1, \ldots, t-4$. We evaluate DarkNet53Seg and TangentConv, since these approaches can deal with a larger number of points without downsampling of the point clouds and could still be trained in a reasonable amount of time.

However, we expect that new approaches could explicitly exploit the sequential information by using multiple input streams to the architecture or even recurrent neural networks to account for the temporal information, which again might open a new line of research.

Results and Discussion

Table 8.5 shows the per-class results for the movable classes and the mean IoU (mIoU) over all classes. For each method, we show in the upper part of the row the IoU for non-moving (unshaded) and in the lower part of the row the IoU for moving objects (shaded). The performance of the remaining static classes is similar to the single scan results. The full per class IoU results for the multiple scans experiment are listed in Table 8.6.

The general trend that the projective methods perform better then the point-based methods is still apparent, which can be also attributed to the larger amount of parameters as in the single scan case. Both approaches show difficulties in separating moving and non-moving objects, which might be caused by our design decision to aggregate multiple scans into a single large point cloud. The results show that especially bicyclist and motorcyclist never get correctly assigned the non-moving class, which is most likely a consequence from the generally sparser object point clouds.

Approach	road	sidewalk	parking	other-ground	building	car	car (moving)	truck	truck (moving)	bicycle	motorcycle	other-vehicle	other-vehicle (moving)	vegetation	trunk	terrain	person	person (moving)	bicyclist	bicyclist (moving)	motorcyclist	motorcyclist (moving)	fence	pole	traffic-sign	mIoU
TangentConv	83.9	64.0	38.3	15.3	85.8	84.9	40.3	21.1	42.2	2.0	18.2	18.5	30.1	79.5	43.2	56.7	1.6	6.4	0.0	1.1	0.0	1.9	49.1	36.4	31.2	34.1
DarkNet53Seg	91.6	75.3	64.9	27.5	85.2	84.1	61.5	20.0	37.8	30.4	32.9	20.7	28.9	78.4	50.7	64.8	7.5	15.2	0.0	14.1	0.0	0.2	56.5	38.1	53.3	41.6

Table 8.6: IoU results using a sequence of multiple past scans (in %).

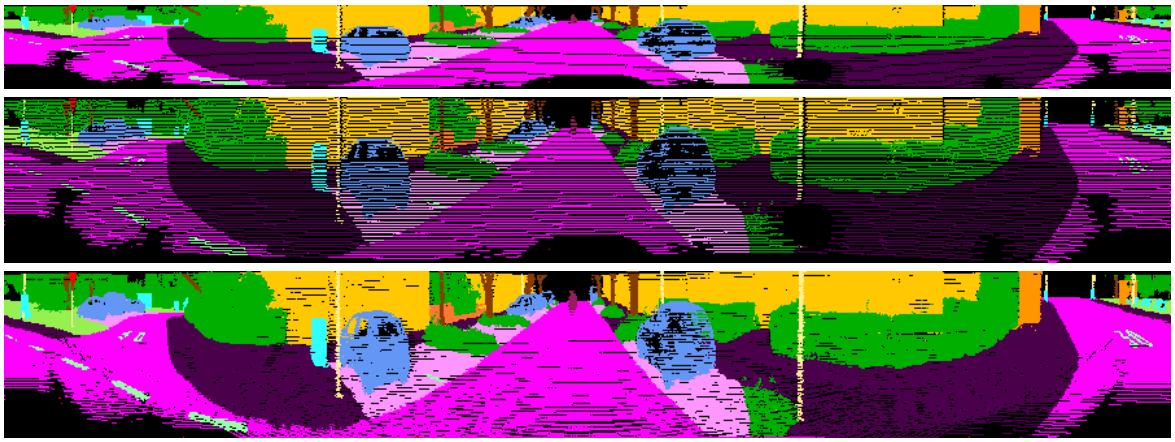

Figure 8.6: Example of using sequence information from SLAM poses to aggregate history. Top: Original scan projected to a 64×900 image. Middle: Same scan projected to a 128×900 image. The image becomes sparse because the laser scanner only has 64 beams. Bottom: Result of aggregating the last 5 scans using the SLAM poses and projecting into to 128×900 resolution. Projection becomes densely populated again.

Figure 8.7: Left: Visualization of the incomplete input for the semantic scene completion benchmark. Note that here we also show the labels for better visualization. However, the real input is a single raw laser scan without any labels. Right: Corresponding target output representing the completed and fully labeled 3D scene.

8.4 Semantic Scene Completion

After leveraging a sequence of past scans for semantic point cloud segmentation, we now show a scenario that makes use of future scans. Due to its sequential nature, our dataset provides the unique opportunity to be extended for the task of 3D semantic scene completion. Note that this is the first real world outdoor benchmark for this task. Existing point cloud datasets cannot be used to address this task, as they do not allow for aggregating labeled point clouds that are sufficiently dense in both space and time.

In semantic scene completion, one fundamental problem is to obtain ground truth labels for real world datasets. In case of NYUv2 (*Silberman et al.*, 2012) CAD models were fit into the scene (*Rock et al.*, 2015) using an RGB-D image captured by a Kinect sensor. New approaches often resort to prove their effectiveness on the larger, but synthetic SUNCG dataset (*Song et al.*, 2017). However, a dataset combining the scale of a synthetic dataset and usage of real-world data is still missing.

In the case of our proposed dataset, the car carrying the LiDAR moves past 3D objects in the scene and thereby records their backsides, which are hidden in the initial scan due to self-occlusion. This is exactly the information needed for semantic scene completion as it contains the full 3D geometry of all objects while their semantics are provided by our dense annotations.

Dataset Generation

By superimposing an exhaustive number of future laser scans in a predefined region in front of the car, we can generate pairs of inputs and targets that correspond to the task of semantic scene completion. As proposed by *Song et al.* (2017), our dataset for the scene completion task is a voxelized representation of the 3D scene.

We select a volume of $51.2\,\text{m}$ ahead of the car, $25.6\,\text{m}$ to every side and $6.4\,\text{m}$ in height with a voxel resolution of $0.2\,\text{m}$, which results in a volume of $256 \times 256 \times 32$ voxels to predict. We assign a single label to every voxel based on the majority vote over all labeled points inside a voxel. Voxels that do not contain any points are labeled as *empty*.

To compute which voxels belong to the occluded space, we check for every pose of the car which voxels are visible to the sensor by tracing a ray. Some of the voxels, *e.g.* those inside objects or behind walls are never visible, so we ignore them during training and evaluation.

Overall, we extracted $19\,130$ pairs of input and target voxel grid for training, 815 for validation and $3\,992$ for testing. To speed up the training process, we subsampled training data by taking every

5th scan. For the test set, we only provide the unlabeled input and withhold the target voxel grids. Figure 8.7 shows an example of a input and target pair.

Task and Metrics

In semantic scene completion, we are interested in predicting the complete scene inside a certain volume from a single initial scan. More specifically, we use as input a voxel grid, where each voxel is marked as empty or occupied, depending on whether or not it contains a laser measurement. For semantic scene completion, one needs to predict whether a voxel is occupied and its semantic label in the completed scene.

For evaluation, we follow the evaluation protocol as in Chapter 6 and 7 using the same 19 classes that were used for the single scan semantic segmentation task (see Section 8.3).

Baselines

We report the results of three semantic scene completion approaches. First, we apply SSCNet (*Song et al.*, 2017) without the flipped TSDF as input feature. This has minimal impact on the performance, but significantly speeds up the training time due to faster pre-processing as shown in Chapter 6. Second, we use the Two-Stream approach, likewise described in Chapter 6, which makes use of the additional information from the RGB image corresponding to the input laser scan. Therefore the RGB image is first processed by a 2D semantic segmentation network, using the approach DeepLab v2 based on ResNet-101 (*Chen et al.*, 2015) and trained on Cityscapes to generate a semantic segmentation. The depth information from the single laser scan and the labels inferred from the RGB image are combined in an early fusion. Finally, we modify the Two-Stream approach by directly using labels from the best LiDAR-based semantic segmentation approach (DarkNet53Seg).

Results and Discussion

Table 8.7 shows the class-wise results for semantic scene completion as well as precision and recall for scene completion.

	Scene Completion			Semantic Scene Completion																			
Approach	precision	recall	IoU	road	sidewalk	parking	other-ground	building	car	truck	bicycle	motorcycle	other-vehicle	vegetation	trunk	terrain	person	bicyclist	motorcyclist	fence	pole	traffic sign	mIoU
SSCNet	31.71	83.40	29.83	27.55	16.99	15.60	6.04	20.88	10.35	1.79	0.0	0.0	0.11	25.77	11.88	18.16	0.0	0.0	0.0	14.40	7.90	3.67	9.53
Two-Stream	31.58	84.18	29.81	28.00	16.98	15.65	4.86	23.19	10.72	2.39	0.0	0.0	0.19	24.73	12.46	18.32	0.03	0.05	0.0	13.23	6.98	3.52	9.54
Two-Stream + DarkNet53Seg	25.85	88.25	24.99	27.53	18.51	18.89	6.58	22.05	8.04	2.19	0.08	0.02	3.96	19.48	12.85	20.22	2.33	0.61	0.01	15.79	7.57	6.99	10.19

Table 8.7: Results for scene completion and class-wise results for semantic scene completion (in %).

The Two-Stream network incorporating 2D semantic segmentation of the RGB image outperforms SSCNet which only uses depth information. However, the usage of the best semantic segmentation directly working on the point cloud (Two-Stream + DarkNet53Seg) performs best in this scenario. Nevertheless, the scene completion performance is still low, suggesting the need for further research into more sophisticated models to address this task.

One possible reason for the poor performance is the problem of dealing with the output resolution. SSCNet performs a 4 fold downsampling in a forward pass, followed by a trilinear upsampling to match the target resolution, which renders it incapable of dealing with the details of the scene. Approaches that produce higher output resolutions currently fail on our benchmark due to memory limitations. These approaches need some significant adaptation to cope with the desired output resolution and are not ready for off-the-shelf usage. Another challenge for current models is the sparsity of the laser input signal in the far field as can be seen from Figure 8.7. To obtain a higher resolution input signal in the far field, approaches would have to exploit more efficiently information from high resolution RGB images provided along with each laser scan.

8.5 Consistent Labels for LiDAR Sequences

In this section, we explain more detailed the implementation of our point cloud labeling tool and the rationale behind our decision to subdivide the sequences spatially, but not temporally for getting consistently labeled point cloud sequences. The labeling tool itself was critical to the successful completion of the endeavor to provide such amounts of data with such fine-grained labels.

Overall, we developed an OpenGL-based labeling tool, which exploits parallelization on the GPU to determine which must be labeled. Figure 8.8 shows our point cloud annotation program visualizing an aggregated point cloud of over 20 million points. We provide interactive tools like a brush, a polygon tool, and different filtering methods to hide particular parts of the scene. Even with that many points, we are still able to maintain interactive labeling capabilities. Changes to the label of the points inside the aggregated point cloud are reflected in the individual scans.

Since we are labeling each point, we are able to annotate objects, even with complex occlusions, more precisely than using bounding volumes (*Xie et al.*, 2016). For instance, we ensured that ground points below a car are labeled accordingly, which was enabled by our filtering capabilities of the visualizer.

To accelerate the search for points that must be labeled, we used a projective approach to assign labels. To this end, we determine for each point the two-dimensional projection on the screen and then determine for the projection if the point is near to the clicked position (in case of the brush) or inside the polygon. Therefore, annotators had to ensure that they did not choose a view that essentially destroyed previously assigned points.

Usually, an annotator performed the following cycle to annotate points: (1) mark points with a specific label and (2) filter points with the label. Due to the filtering of already labeled points, one can resolve occlusions and furthermore ensure that the aforementioned projective labeling does not destroy already labeled points.

8.5.0.1 Tile-based labeling

An important detail is the aforementioned spatial subdivision of the complete point cloud into tiles (also shown on the left upper part of Figure 8.8). Initially, we simply rendered all scans in a range of timestamps, say $100 - 150$, and then moved on the next part, say $150 - 200$. However, this leads quickly to inconsistencies in the labels, since scans from such parts still overlap and therefore must be relabeled to match labels from before. Since we, furthermore, encounter loop closures with a

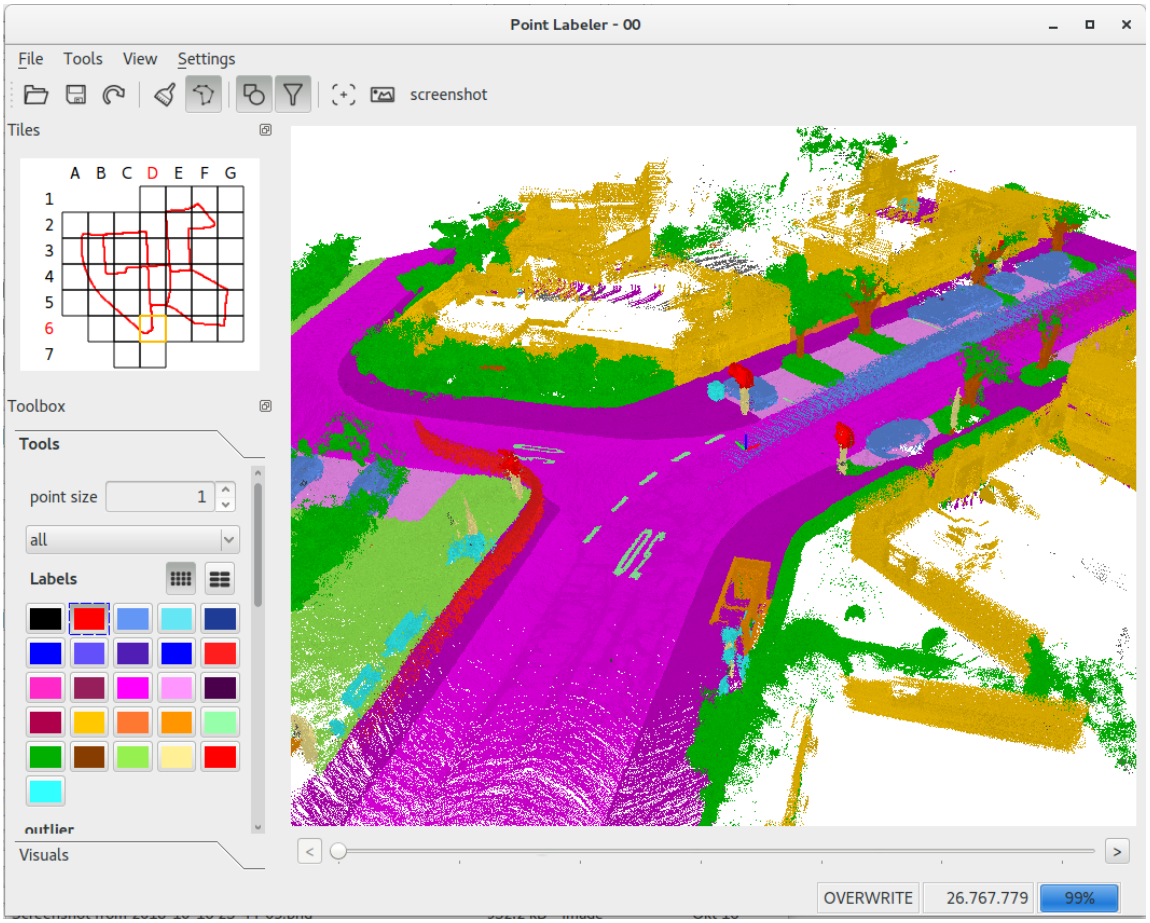

Figure 8.8: Point cloud labeling tool. In the upper left corner the user sees the tile and the sensor's path indicated by the red trajectory.

considerable temporal distance, this overlap can even happen between parts of the sequences that are not temporally close, which even more complicated the task.

It quickly was apparent that such an additional effort to ensure consistent labels would quickly lead to unreasonable complicated and insufficient results. Therefore, we decided to subdivide the sequence spatially into tiles, where each tile contains all points from scans overlapping with this tile. Consistency at the boundaries between tiles was achieved by having a small overlap between the tiles, which enabled to consistently continue the labels from one tile into another neighboring tile.

8.5.0.2 Moving objects

We annotated all moving objects, i.e., car, truck, person, bicyclist, and motorcyclist, and each moving object is represented by a different class to distinguish it from its non-moving corresponding class. In our case, we assigned an object the corresponding moving class when it moved at some point in time while observing it with the sensor.

Since moving objects will appear at different places when aggregating scans captured from different sensor locations, we had to take special care to annotate moving objects. We annotated moving

objects either by filtering ground points or by labeling each scan individually, which was often necessary to label points of tires of a car and bicycles or the feet of persons. It often was the first step when annotating a tile, since this allowed us to filter all moving points and then concentrate on the static parts of the environment.

8.6 Class Definition

In the process of labeling such large amounts of data, we had to decide which classes we want to be annotated at some point in time. In general, we followed the class definitions and selection of the *Mapillary Vistas* dataset (*Neuhold et al.*, 2017) and *Cityscapes* (*Cordts et al.*, 2016) dataset, but did some simplifications and adjustments for the data source used.

First, we do not explicitly consider a *rider* class for persons riding a motorcycle or a bicycle, since the available point clouds do not provide the density for a single scan to distinguish the person riding a vehicle. Furthermore, we get for such classes only moving examples and therefore cannot easily aggregate the point clouds to increase the fidelity of the point cloud and make it easier to distinguish the rider of a vehicle and the vehicle.

We furthermore had to keep the class selection fixed, even though we encountered while labeling situation where we thought that adding a class would have been beneficial. In particular, the classes *other-structure*, *other-vehicle*, and *other-object* are fallback classes of their respective root category in unclear cases or missing classes, since this simplified the labeling process and might be used to distinguish these categories further in future.

Annotators often annotated some object or part of the scene and then hide the labeled points to avoid overwriting or removing the labels (the labeling tool is described more in the next paragraph.) Thus, assigning the fallback class in ambiguous cases or cases where a specific class was missing made it possible to simply hide that class to avoid overwriting it. If we had instructed the annotators to label such parts as unlabeled, it would have caused problems to consistently label the point clouds.

For the vehicle category (except bicycle and motorcycle, since we assume that these cannot move without a human) and human category, we additionally distinguished between non-moving and moving, *i.e.* it moved while observing the object. We furthermore distinguished between moving and non-moving vehicles and humans, *i.e.* a vehicle gets the moving vehicle class if it moved in some scan while observing the vehicle.

Overall, we annotated 28 classes and all annotated classes with their respective definitions are listed in Table 8.9.

	class	definition
Ground-related	road	Drivable areas where cars are allowed to drive on including service lanes, bike lanes, crossed areas on the street. Only the road surface is labeled excluding the curb.
	sidewalk	Areas used mainly by pedestrians, bicycles, but not meant for driving with a car. This includes curbs and spaces where you are not allowed to drive faster than $5\,\mathrm{km\,h^{-1}}$. Private driveways are also labeled as sidewalk. Here cars should also not drive with regular speeds (such as 30 or $5.0 \times 10^1\,\mathrm{km\,h^{-1}}$).

	parking	Areas meant explicitly for parking and that are clearly separated from sidewalk and road by means of a small curb. If unclear then *other-ground* or *sidewalk* can be selected. Garages are labeled as *building* and not as parking.
	other-ground	This label is chosen whenever a distinction between sidewalk and terrain is unclear. It includes (paved/plastered) traffic islands which are not meant for walking. Also the paved parts of a gas station are not meant for parking.
structure	building	The whole building including building walls, doors, windows, stairs, etc. Garages count as building.
	other-structure	This includes other vertical structures, like tunnel walls, bridge posts, scaffolding on a building from a construction site or bus stops with a roof.
vehicle	car	Cars, jeeps, SUVs, vans with a continuous body shape (i.e. the driver cabin and cargo compartment are one) are included.
	truck	Trucks, vans with a body that is separate from the driver cabin, pickup trucks, as well as their attached trailers.
	bicycle	Bicycles without the cyclist or possibly other passengers. If the bicycle is driven by a person or a person stands nearby the vehicle, we label it as bicyclist.
	motorcycle	Motorcycles, mopeds without the driver or other passengers. Includes also motorcycles covered by a cover. If the motorcycle is driven by a person or a person stands nearby the vehicle, we label it as motorcyclist.
	other-vehicle	Caravans, Trailers and fallback category for vehicles not explicitly defined otherwise in the meta category *vehicle*. Included are buses intended for 9+ persons for public or long-distance transport. This further includes all vehicles moving on rails, e.g., trams, trains.
nature	vegetation	Vegetation are all bushes, shrubs, foliage, and other clearly identifiable vegetation.
	trunk	The tree trunk is labeled as *trunk* separately from the treetop which gets the label *vegetation*.
	terrain	Grass and all other types of horizontal spreading vegetation, including soil.
human	person	Humans moving by their own legs, sitting, or any unusual pose, but not meant to drive a vehicle.
	bicyclist	Humans driving a bicycle or standing in close range to a bicycle (within *arm reach*). We do not distinguish between riders and bicyclist.
	motorcyclist	Humans driving a motorcycle or standing in close range to a motorcycle (within *arm reach*).
object	fence	Separators, like fences, small walls and crash barriers.
	pole	Lamp posts and the poles of traffic signs.
	traffic sign	Traffic sign excluding its mounting. Spurious points in a layer in front and behind the traffic sign are also labeled as traffic sign and not as outlier.
	other-object	Fallback category that includes advertising columns.
outlier	outlier	Outlier are caused by reflections or inaccuracies in the deskewing of scans, where it is unclear where the points came from.

Table 8.9: Class definitions.

8.7 Dataset and Baseline Access API

Along with the annotations and the labeling tool, we also provide a public API implemented in Python.

In contrast with our labeling tool which is intended at allowing users to easily extend this dataset, and generate others for other purposes, this API is intended to be used to easily access the data, calculate statistics, evaluate metrics, and access several implementations of different state-of-the-art semantic segmentation approaches. We hope that this API will serve as a baseline to implement new point cloud semantic segmentation approaches, and will provide a common framework to evaluate them, and compare them more transparently with other methods. The choice of Python as the underlying language for the API is that it is the current language of choice for the front end for deep learning framework developers, and therefore, for deep learning practitioners.

8.8 Overview and Example scenes

To show the quality and the variety of Figure 8.9 gives an overview of the labeled sequences showing the estimated trajectories and the aggregated point cloud over the whole sequence.

8.9 Summary and Outlook

In this Chapter, we have presented a large-scale dataset showing unprecedented scale in point-wise annotation of point cloud sequences. We provide a range of different baseline experiments for three tasks: (i) semantic segmentation using a single scan, (ii) semantic segmentation using multiple scans, and (iii) semantic scene completion. These experimental results revealed limitations of the baselines, which need to be addressed in the future.

The dataset will be provided along with instance-level annotations for vehicles and humans over the whole sequence, which will enable future tasks that involve distinguishing distinct objects and identifying the same object over time. Another potential future task enabled by our dataset, could be the evaluation of semantic SLAM algorithms.

With respect to semantic scene completion, an obvious thing to try, would be to combine the approaches proposed in Chapter 6 and 7 which unfortunately could not be explored within the time frame of this thesis. Another interesting future task would be to try to get rid of the laser-scan as an input to the model and only use the RGB image instead. In this scenario the laser sensor would only provide the supervision signal during training during inference only a RGB camera is needed. A similar problem is already addressed in a research field called ‘monocular depth estimation’. Apart from the expected loss of localization accuracy, the big advantage of such an approach would be that it becomes very cheap as mentioned in Chapter 1.1.2.

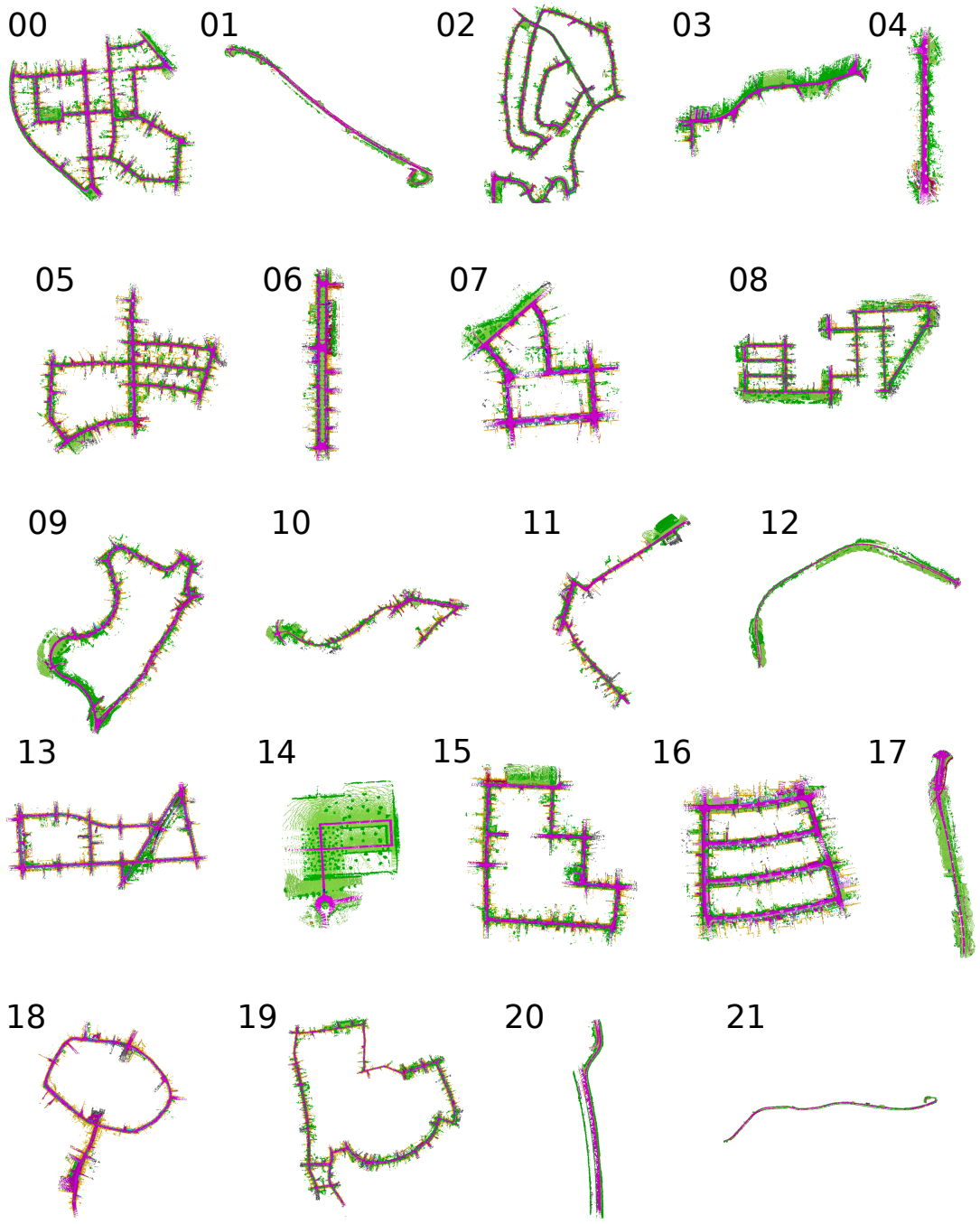

Figure 8.9: Qualitative overview of labeled sequences and trajectories.

Approach	scan size	projected	learning rate	epochs trained	converged
PointNet	50 000	-	$3e^{-4} \times 0.9^{\text{epoch}}$	33	✓
PointNet++	45 000	-	$3e^{-3} \times 0.9^{\text{epoch}}$	25	✓
SPGraph	120 000	-	$1e^{-2}$	40	✓
TangentConv	120 000	-	$1e^{-4}$	10	✓
SPLATNet	50 000	-	$1e^{-3}$	20	-
SqueezeSeg	$64 \times 2\,048$	✓	$1e^{-2} \times 0.99^{\text{epoch}}$	200	✓
DarkNet21Seg	$64 \times 2\,048$	✓	$1e^{-3} \times 0.99^{\text{epoch}}$	40	✓
DarkNet53Seg	$64 \times 2\,048$	✓	$1e^{-3} \times 0.99^{\text{epoch}}$	120	✓
TangentConv	500 000	-		5*	✓
DarkNet53Seg64	$64 \times 2\,048$	✓	$1e^{-3} \times 0.99^{\text{epoch}}$	40*	✓
DarkNet53Seg96	$96 \times 2\,048$	✓	$5e^{-4} \times 0.99^{\text{epoch}}$	40*	✓

Table 8.8: Approach statistics. $*$ in number of epochs means that it was started from the pretrained weights of the single scan version.

CHAPTER 9

Conclusion

In this thesis we provided several contributions to the field of semantic segmentation as well as to its new offsprings: The anticipation of semantic categories in two and three dimensions. We will now quickly summarize the individual contributions of all our works in those fields.

Contents

9.1 Summary

In our first work, we presented a real-time adaptation of a 2D semantic segmentation algorithm based on texton features and random forests. The combination with a superpixel computation using quadtrees as well as a label propagation strategy leads to reduction of the runtime by a factor of 192 over the baseline while increasing the accuracy. The approach is capable of running in real-time on a single threaded CPU. This makes it suitable for devices with limited hardware resources.

Next we introduced the new task of spatially anticipating 2D semantic labels outside an image. We proposed two evaluation metrics. The first one focus on a pixel-wise segmentation map. The second one relaxes the need for an exact localization of labels and instead requires the prediction of labels within a coarse grid of cells. We proposed a CNN for spatial anticipation and evaluated two different loss functions. We found through experiments that the most effective approach to solve the tasks is a combination of 2 CNNs: The first one converts the visible pixels into a semantic segmentation map, whereas the second one is used to iteratively extrapolate this segmentation map into the unobserved region outside the image.

Afterwards we focused on the 3D domain by proposing two new efficient approaches to 3D semantic scene completion. Our first approach is a two stream approach for 3D semantic scene completion which leverages a depth and a image input in an early fusion scheme. Moreover we propose an effective 3-channel linear input embedding that outperforms the 1-channel embedding and performs competitively to a memory expensive one-hot encoding. The approach achieves state-of-the-art results on the NYUv2 dataset while also featuring a much faster inference time than approaches using a TSDF based input encoding. Our second approach uses the recently introduced concept of generative adversarial networks to improve the performance of 3D semantic scene completion. We evaluate

several modifications of the classical GAN setup like using local adversarial loss and conditional GANs. Our experiments show that conditional GANs give a robust performance improvement while local adversarial loss only improves quantitative results on NYU CAD but not on NYU Kinect. On SUNCG our model is able to outperform the baseline by a large margin and sets a new state-of-the-art. Qualitatively the model using global adversarial loss and conditional GANs is giving significantly more realistically looking results.

Our final contribution is SemanticKITTI, a new, large scale dataset with point-wise annotated laser scans of the KITTI odometry benchmark. It has been evaluated in several settings using existing approaches: (i) for semantic segmentation of single point clouds (ii) for semantic segmentation on several aggregated scans and (iii) for 3D semantic scene completion. The experimental evaluation shows that there is still a lot of room for improvement for either task, which needs to be addressed by the research community.

9.2 Outlook

9.2.1 Semantic Segmentation and Completion

As the field of semantic segmentation progresses fast, it is hard to tell for how long the research community will keep up its interest in the field, rather than moving towards new more challenging tasks - which was also the trajectory of this thesis. For many problems including semantic segmentation, anticipation of semantic labels and 3D semantic scene completion, it seems that increasing the amount, quality and granularity of humanly annotated data will overcome some of the current obstacles in training more robust models. However there seems to be a limit on how much current CNN architectures can scale, when confronted with a vastly increased variance of training data as well as a heavily increased number of labels to distinguish.

Apart from that, a typical problem for CNN based semantic segmentation algorithms is the loss of resolution during forward propagation through the network. Although there exist many approaches how to achieve higher output resolutions, like using dilated convolutions, decoder network architectures with skip connections from earlier higher-resolution layers or post-processing of the segmentation map using a CRF, there is still a need for improvement.

As for 3D semantic scene completion an additional problem of current models is the high memory demand and long training times of the involved 3D-CNN architectures. The memory demand of a regular voxel grid grows cubically with its resolution. As the resolution grows the percentage of occupied voxels decreases in the case the scene contains thin, hollow objects or surfaces. This problem is termed ‘curse of dimensionality’. To save computational resources, one can use sparse implementation of 3D convolutions (*Engelcke et al.*, 2017). However, for a traditional convolution the sparsity of subsequent feature maps decreases and therefore the memory demand increases. This issue has been recently addressed by *Zhang et al.* (2018) using sparsely-implemented ‘submanifold convolutions’. However, the submanifold convolutions suffer from a reduced receptive field and reduce interactions between spatially proximate neurons. *Riegler et al.* (2017) also proposed a sparse feature map encoding based on stacked, shallow octrees, but the growing of octrees to determine the neighbourhood structure of the voxel grid, gives an additional computational overhead. Therefore the need for memory has just been replaced by a need for more computation.

Another interesting direction for semantic scene completion would be to train models using

purely image data as input and get rid of the need for a depth sensor. Such an approach could still use the depth information as a supervision signal during training, *e.g.* by either implicitly or explicitly performing monocular depth estimation. The general goal would be to extract as much information from an RGB image as humans can, using different sensor modalities to provide a supervision signal. There might even be a synergy for a neural network to perform depth estimation and semantic scene completion simultaneously. In any case however, the resulting hardware setup would be intriguingly cheap and simple.

9.2.2 How Robust are CNNs?

We know that, given a large dataset, CNNs are pretty effective in learning tasks like semantic segmentation on them. When confronted with a new dataset, people usually take the pre-trained network, add a new, randomly initialized classification layer and finetune the network for the new task (potentially with a new number of classes). However, in doing so, the network becomes incapable of solving the earlier task. A solution would be to only finetune the last layers and keep the bulk of the earlier layers fixed. Though the method is working, it is seldom applied, since it results in a poorer performance than finetuning the entire network. The effectiveness of the finetuning depends on how different the new task or dataset is compared to the earlier one. The phenomenon is also discussed under the term 'catastrophic forgetting', where unlike the human brain which is able to learn new information without forgetting the older one, artificial neural networks still miss this capability. *Goodfellow et al.* (2013) suggest that training while heavily relying on dropout alleviates the problem. However, since they only perform experiments on MNIST, there is still progress to be made until CNNs reach a desired level of robustness and flexibility towards incremental learning on current semantic segmentation benchmarks.

Another problem of CNNs is their notorious sensitivity to 'adversarial examples'. These are images that are transformed by some small amount of pixel noise, in fact so small that a human cannot see a difference to the original input image. However this invisible noise is enough to fool the network recognition pipeline entirely. This problem is inherent to CNNs. As CNNs process RGB images raw without much pre-processing or feature normalization, the small amount of noise is amplified during forward propagation, leading to misclassifications. Since the problem has already been identified, researchers proposed several approaches to attenuate the problem. Concepts like 'adversarial training' or 'defensive distillation' are able to reduce the problem. However, we are likely confronted with a new quality of vulnerability due to the usage of end-to-end trained systems in contrast to hand-crafted features, which were less expressive but also more robust to this kind of noise.

9.2.3 One CNN architecture for Different Problems?

So far it seems that CNN architectures found to perform strongly on one task (*e.g.* image classification) are also suitable for all other vision related tasks. However different tasks often require different adaptations of existing architectures. For semantic segmentation, for example, the output needs a high resolution, so the usage of pooling-layers, which decrease the feature map resolution, is undesirable. Until now, it seems, that small adaptations are useful, when it comes to performing different tasks. However, the example of the human brain, suggests that there should be one structure, able to perform all of the tasks. It will be interesting to see, whether computer vision and artificial

intelligence evolve into the same directions or whether the models and architectures stay different for every individual field.

Another interesting observation comes from machine translation, where the same CNN architecture can be trained to take different languages as input and to generate sentences in different languages as output. The result is a model that can translate from any source into any target language although the model has never been trained with a single sentence pair in those two languages. This hints into another interesting direction to unify CNN models while simultaneously addressing the problem of data shortage.

Bibliography

Achanta, Radhakrishna; Shaji, Appu; Smith, Kevin; Lucchi, Aurelien; Fua, Pascal, and Süsstrunk, Sabine. Slic superpixels compared to state-of-the-art superpixel methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 34(11):2274–2282, 2012. (Not cited.)

Agrawal, Anuraag; Nakazawa, Atsushi, and Takemura, Haruo. MMM-classification of 3D Range Data. In *IEEE Int. Conference on Robotics and Automation (ICRA)*, 2009. (Not cited.)

Anguelov, Dragomir; Taskar, Ben; Chatalbashev, Vassil; Koller, Daphne; Gupta, Dinkar; Heitz, Geremy, and Ng, Andrew. Discriminative Learning of Markov Random Fields for Segmentation of 3D Scan Data. In *IEEE Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 169–176, 2005. (Not cited.)

Arjovsky, Martin; Chintala, Soumith, and Bottou, Léon. Wasserstein gan. *arXiv preprint: 1701.07875*, 2017. (Not cited.)

Armeni, Iro; Sax, Sasha; Zamir, Amir R., and Savarese, Silvio. Joint 2D-3D-Semantic Data for Indoor Scene Understanding. *arXiv preprint: 1702.01105*, 2017. (Not cited.)

Badrinarayanan, Vijay; Kendall, Alex, and Cipolla, Roberto. SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation. In *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 2017. (Not cited.)

Behley, Jens and Stachniss, Cyrill. Efficient Surfel-Based SLAM using 3D Laser Range Data in Urban Environments. In *Proc. of Robotics: Science and Systems (RSS)*, 2018. (Not cited.)

Behley, Jens; Kersting, Kristian; Schulz, Dirk; Steinhage, Volker, and Cremers, Armin B. Learning to Hash Logistic Regression for Fast 3D Scan Point Classification. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 5960–5965, 2010. (Not cited.)

Behley, Jens; Steinhage, Volker, and Cremers, Armin B. Performance of Histogram Descriptors for the Classification of 3D Laser Range Data in Urban Environments. In *IEEE Int. Conference on Robotics and Automation (ICRA)*, 2012. (Not cited.)

Blaha, M; Vogel, C; Richard, A; Wegner, D; Pock, T, and Schindler, K. Large-scale semantic 3d reconstruction: An adaptive multi-resolution model for multi-class volumetric labeling. In *IEEE Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2016. (Not cited.)

Boulch, Alexandre; Guerry, Joris; Le Saux, Bertrand, and Audebert, Nicolas. SnapNet: 3D point cloud semantic labeling with 2D deep segmentation networks. *Computers & Graphics*, 2017. (Not cited.)

Breiman, Leo. Random forests. *Machine Learning*, 45(1):5–32, 2001. (Not cited.)

Brostow, Gabriel J; Shotton, Jamie; Fauqueur, Julien, and Cipolla, Roberto. Segmentation and recognition using structure from motion point clouds. In *Computer Vision–ECCV 2008*, pages 44–57. 2008. (Not cited.)

Bulo, Samuel and Kontschieder, Peter. Neural decision forests for semantic image labelling. In *IEEE Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 81–88, 2014. (Not cited.)

Chang, Angel; Dai, Angela; Funkhouser, Thomas; Halber, Maciej; Niessner, Matthias; Savva, Manolis; Song, Shuran; Zeng, Andy, and Zhang, Yinda. Matterport3D: Learning from RGB-D Data in Indoor Environments. In *International Conference on 3D Vision*, 2017. (Not cited.)

Chang, Angel X.; Funkhouser, Thomas A.; Guibas, Leonidas J.; Hanrahan, Pat; Huang, Qi-Xing; Li, Zimo; Savarese, Silvio; Savva, Manolis; Song, Shuran; Su, Hao; Xiao, Jianxiong; Yi, Li, and Yu, Fisher. ShapeNet: An Information-Rich 3D Model Repository. In *arXiv preprint: 1512.03012*, 2015. (Not cited.)

Chen, Liang-Chieh; Papandreou, George; Kokkinos, Iasonas; Murphy, Kevin, and Yuille, Alan L. Semantic Image Segmentation with Deep Convolutional Nets and Fully Connected CRFs. In *Int. Conf. on Learning Representations (ICLR)*, 2015. (Not cited.)

Chen, Liang-Chieh; Zhu, Yukun; Papandreou, George; Schroff, Florian, and Adam, Hartwig. Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation. In *arXiv preprint: 1802.02611*, 2018. (Not cited.)

Comaniciu, Dorin and Meer, Peter. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, (5):603–619, 2002. (Not cited.)

Cordts, Marius; Omran, Mohamed; Ramos, Sebastian; Rehfeld, Timo; Enzweiler, Markus; Benenson, Rodrigo; Franke, Uwe; Roth, Stefan, and Schiele, Bernt. The Cityscapes Dataset for Semantic Urban Scene Understanding. In *IEEE Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2016. (Not cited.)

Criminisi, Antonio and Shotton, Jamie. *Decision Forests for Computer Vision and Medical Image Analysis*. Springer Science & Business Media, 2013. (Not cited.)

Dai, A; Rithie, D; Bokeloh, M; Reed, S; Sturm, J, and Nießner, M. ScanComplete: Large-Scale Scene Completion and Semantic Segmentation for 3D Scans. In *IEEE Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018. (Not cited.)

Dai, Angela; Chang, Angel X.; Savva, Manolis; Halber, Maciej; Funkhouser, Thomas A., and Nießner, Matthias. ScanNet: Richly-annotated 3D Reconstructions of Indoor Scenes. In *IEEE Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017a. (Not cited.)

Dai, Angela; Qi, Charles Ruizhongtai, and Nießner, Matthias. Shape Completion using 3D-Encoder-Predictor CNNs and Shape Synthesis. In *IEEE Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017b. (Not cited.)

Dalal, Navneet and Triggs, Bill. Histograms of oriented gradients for human detection. In *IEEE Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 886–893. IEEE Computer Society, 2005. (Not cited.)

Deng, Jia; Dong, Wei; Socher, Richard; Li, Li-Jia; Li, Kai, and Fei-Fei, Li. ImageNet: A Large-Scale Hierarchical Image Database. In *IEEE Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2009. (Not cited.)

Denton, Emily L; Chintala, Soumith; Fergus, Rob, and others, . Deep generative image models using aï¿¼ laplacian pyramid of adversarial networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1486–1494, 2015. (Not cited.)

Dosovitskiy, Alexey; Tobias Springenberg, Jost, and Brox, Thomas. Learning to Generate Chairs with Convolutional Neural Networks. In *IEEE Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 1538–1546, 2015. (Not cited.)

Engelcke, Martin; Rao, Dushyant; Wang, Dominic Zeng; Tong, Chi Hay, and Posner, Ingmar. Vote3deep: Fast object detection in 3d point clouds using efficient convolutional neural networks. In *IEEE Int. Conference on Robotics and Automation (ICRA)*, pages 1355–1361. IEEE, 2017. (Not cited.)

Engelmann, Francis; Kontogianni, Theodora; Schult, Jonas, and Leibe, Bastian. Know What Your Neighbors Do: 3D Semantic Segmentation of Point Clouds. *arXiv preprint: 1810.01151*, 2018. (Not cited.)

Everingham, Mark; Eslami, Ali; van Gool, Luc; Williams, Chris; Winn, John, and Zisserman, Andrew. The Pascal Visual Object Classes Challenge - a Retrospective. In *Int. Journal on Computer Vision (IJCV)*, volume 111, pages 98–136, 2015. (Not cited.)

Felzenszwalb, Pedro F and Huttenlocher, Daniel P. Efficient graph-based image segmentation. *Int. Journal on Computer Vision (IJCV)*, 59(2):167–181, 2004. (Not cited.)

Firman, Michael; Mac Aodha, Oisin; Julier, Simon, and Brostow, Gabriel J. Structured Prediction of Unobserved Voxels From a Single Depth Image. In *IEEE Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2016. (Not cited.)

Freund, Yoav and Schapire, Robert E. Experiments with a new boosting algorithm. In *Int. Conf. on Machine Learning (ICML)*, pages 148–156, 1996. (Not cited.)

Gaidon, Adrien; Wang, Qiao; Cabon, Yohann, and Vig, Eleonora. Virtual Worlds as Proxy for Multi-Object Tracking Analysis. In *IEEE Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2016. (Not cited.)

Gauthier, Jon. Conditional generative adversarial nets for convolutional face generation. *Class Project for Stanford CS231N: Convolutional Neural Networks for Visual Recognition, Winter semester*, 2014(5):2, 2014. (Not cited.)

Geiger, Andreas and Wang, Chaohui. Joint 3D Object and Layout Inference from a single RGB-D Image. In *German Conference on Artificial Intelligence (GCPR)*, volume 9358, pages 183–195, 2015. (Not cited.)

Geiger, Andreas; Lenz, Philipp, and Urtasun, Raquel. Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite. In *IEEE Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 3354–3361, 2012. (Not cited.)

Glorot, Xavier; Bordes, Antoine, and Bengio, Yoshua. Deep Sparse Rectifier Neural Networks. In *Aistats*, volume 15, page 275, 2011. (Not cited.)

Goodfellow, Ian; Pouget-Abadie, Jean; Mirza, Mehdi; Xu, Bing; Warde-Farley, David; Ozair, Sherjil; Courville, Aaron, and Bengio, Yoshua. Generative Adversarial Nets. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2672–2680, 2014. (Not cited.)

Goodfellow, Ian; Bengio, Yoshua, and Courville, Aaron. *Deep learning*. MIT press, 2016. (Not cited.)

Goodfellow, Ian J; Mirza, Mehdi; Xiao, Da; Courville, Aaron, and Bengio, Yoshua. An empirical investigation of catastrophic forgetting in gradient-based neural networks. *arXiv preprint: 1312.6211*, 2013. (Not cited.)

Graham, Benjamin; Engelcke, Martin, and van der Maaten, Laurens. 3D Semantic Segmentation with Submanifold Sparse Convolutional Networks. In *IEEE Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018. (Not cited.)

Groh, Fabian; Wieschollek, Patrick, and Lensch, Hendrik. Flex-Convolution (Million-Scale Point-cloud Learning Beyond Grid-Worlds). In *Asian Conference on Computer Vision (ACCV)*, Dezember 2018. (Not cited.)

Gupta, Saurabh; Arbelaez, Pablo, and Malik, Jitendra. Perceptual organization and recognition of indoor scenes from RGB-D images. In *IEEE Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 564–571, 2013. (Not cited.)

Gupta, Saurabh; Arbeláez, Pablo Andrés; Girshick, Ross B., and Malik, Jitendra. Aligning 3D models to RGB-D images of cluttered scenes. In *IEEE Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 4731–4740, 2015. (Not cited.)

Hackel, Timo; Savinov, Nikolay; Ladicky, Lubor; Wegner, Jan D.; Schindler, Konrad, and Pollefeys, Marc. SEMANTIC3D.NET: A new Large-Scale Point Cloud Classification Benchmark. In *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, volume IV-1-W1, pages 91–98, 2017. (Not cited.)

Han, Xiaoguang; Li, Zhen; Huang, Haibin; Kalogerakis, Evangelos, and Yu, Yizhou. High-Resolution Shape Completion Using Deep Neural Networks for Global Structure and Local Geometry Inference. In *Int. Conf. on Computer Vision (ICCV)*, 2017. (Not cited.)

Häne, Christian; Zach, Christopher; Cohen, Andrea; Angst, Roland, and Pollefeys, Marc. Joint 3D Scene Reconstruction and Class Segmentation. In *IEEE Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 97–104, 2013. (Not cited.)

He, Kaiming; Zhang, Xiangyu; Ren, Shaoqing, and Sun, Jian. Deep Residual Learning for Image Recognition. In *IEEE Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2016. (Not cited.)

Hua, Binh-Son; Pham, Quang-Hieu; Nguyen, Duc Thanh; Tran, Minh-Khoi; Yu, Lap-Fai, and Yeung, Sai-Kit. SceneNN: A Scene Meshes Dataset with aNNotations. In *Proc. of the International Conference on 3D Vision (3DV)*, 2016. (Not cited.)

Hua, Binh-Son; Tran, Minh-Khoi, and Yeung, Sai-Kit. Pointwise Convolutional Neural Networks. In *IEEE Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018. (Not cited.)

Huang, Jing and You, Suya. Point Cloud Labeling using 3D Convolutional Neural Network. In *Proc. of the Internation Conference on Pattern Recognition (ICPR)*, 2016. (Not cited.)

Ioffe, Sergey and Szegedy, Christian. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Int. Conf. on Machine Learning (ICML)*, 2015. (Not cited.)

Iqbal, Umar; Garbade, Martin, and Gall, Juergen. Pose for Action - Action for Pose. In *IEEE Conference on Automatic Face and Gesture Recognition*, 2017. (Not cited.)

Jiang, H and Xiao, J. A linear approach to matchig cuboids in RGBD images. In *IEEE Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2013. (Not cited.)

Jiang, Mingyang; Wu, Yiran, and Lu, Cewu. PointSIFT: A SIFT-like Network Module for 3D Point Cloud Semantic Segmentation. *arXiv preprint: 1807.00652*, 2018. (Not cited.)

Johnson, Andrew and Hebert, Martial. Using spin images for effcient object recognition in cluttered 3D scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 21(5):433–449, 1999. (Not cited.)

Karras, Tero; Aila, Timo; Laine, Samuli, and Lehtinen, Jaakko. Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint: 1710.10196*, 2017. (Not cited.)

Kim, Byung-soo; Kohli, Pushmeet, and Savarese, Silvio. 3D scene understanding by Voxel-CRF. In *IEEE Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2013. (Not cited.)

Kim, Young Min; Mitra, Niloy J.; Yan, Dong-Ming, and Guibas, Leonidas. Acquiring 3D Indoor Environments with Variability and Repetition. In *ACM Transactions on Graphics*, volume 31, pages 138:1–138:11, 2012. (Not cited.)

Klukov, Roman and Lempitsky, Victor. Escape from Cells: Deep Kd-Networks for the Recognition of 3D Point Cloud Models. In *Int. Conf. on Computer Vision (ICCV)*, 2017. (Not cited.)

Kontschieder, Peter; Rota Bulò, S; Bischof, Horst, and Pelillo, Marcello. Structured class-labels in random forests for semantic image labelling. In *Int. Conf. on Computer Vision (ICCV)*, pages 2190–2197, 2011. (Not cited.)

Krähenbühl, Philipp and Koltun, Vladlen. Efficient Inference in Fully Connected CRFs with Gaussian Edge Potentials. In *Advances in Neural Information Processing Systems (NIPS)*, 2011. (Not cited.)

Krizhevsky, Alex; Sutskever, Ilya, and Hinton, Geoffrey E. Imagenet classification with deep convolutional neural networks. In *Neural Information Processing Systems*, pages 1097–1105, 2012. (Not cited.)

Ladickỳ, L'ubor; Sturgess, Paul; Alahari, Karteek; Russell, Chris, and Torr, Philip HS. What, where and how many? Combining object detectors and crfs. In *European Conf. on Computer Vision (ECCV)*, pages 424–437, 2010. (Not cited.)

Lai, Kevin and Fox, Dieter. Object Recognition in 3D Point Clouds Using Web Data and Domain Adaptation. In *International Journal of Robotics Research*, volume 29, pages 1019–1037, 2010. (Not cited.)

Lai, Kevin; Bo, Liefeng, and Fox, Dieter. Unsupervised feature learning for 3d scene labeling. In *IEEE Int. Conference on Robotics and Automation (ICRA)*, pages 3050–3057, 2014. (Not cited.)

Landrieu, Loic and Simonovsky, Martin. Large-scale Point Cloud Semantic Segmentation with Superpoint Graphs. In *IEEE Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018. (Not cited.)

LeCun, Yann; Boser, Bernhard; Denker, John S; Henderson, Donnie; Howard, Richard E; Hubbard, Wayne, and Jackel, Lawrence D. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989. (Not cited.)

LeCun, Yann; Bottou, Léon; Bengio, Yoshua, and Haffner, Patrick. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. (Not cited.)

Ledig, Christian; Theis, Lucas; Huszár, Ferenc; Caballero, Jose; Cunningham, Andrew; Acosta, Alejandro; Aitken, Andrew; Tejani, Alykhan; Totz, Johannes; Wang, Zehan, and others, . Photo-realistic single image super-resolution using a generative adversarial network. In *IEEE Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 4681–4690, 2017. (Not cited.)

Leung, Thomas K. and Malik, Jitendra. Representing and recognizing the visual appearance of materials using three-dimensional textons. *Int. Journal on Computer Vision (IJCV)*, 43(1):29–44, 2001. (Not cited.)

Li, Chuan and Wand, Michael. Precomputed real-time texture synthesis with markovian generative adversarial networks. In *European Conf. on Computer Vision (ECCV)*, pages 702–716. Springer, 2016. (Not cited.)

Li, Wenbin; Saeedi, Sajad; McCormac, John; Clark, Ronald; Tzoumanikas, Dimos; Ye, Qing; Huang, Yuzhong; Tang, Rui, and Leutenegger, Stefan. InteriorNet: Mega-scale Multi-sensor Photo-realistic Indoor Scenes Dataset. In *British Machine Vision Conference (BMVC)*, 2018. (Not cited.)

Li, Yangyan; Dai, Angela; Guibas, Leonidas, and Niessner, Matthias. Database-Assisted Object Retrieval for Real-Time 3D Reconstruction. In *Computer Graphics Forum*, volume 34, pages 435–446, 2015. (Not cited.)

Lin, D; Fidler, S, and Urtasun, R. Holistic scene understanding for 3D object detection with RGBD cameras. In *Int. Conf. on Computer Vision (ICCV)*, 2013. (Not cited.)

Lin, Guosheng; Milan, Anton; Shen, Chunhua, and Reid, Ian. Refinenet: Multi-path refinement networks for high-resolution semantic segmentation. In *IEEE Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, July 2017. (Not cited.)

Lin, Tsung-Yi; Maire, Michael; Belongie, Serge; Hays, James; Perona, Pietro; Ramanan, Deva; Dollár, Piotr, and Zitnick, C Lawrence. Microsoft COCO: Common objects in context. In *European Conf. on Computer Vision (ECCV)*, pages 740–755. Springer, 2014. (Not cited.)

Liu, Miaomiao; He, Xuming, and Salzmann, Mathieu. Building Scene Models by Completing and Hallucinating Depth and Semantics. In *European Conf. on Computer Vision (ECCV)*, pages 258–274, 2016. (Not cited.)

Liu, Shice; HU, YU; Zeng, Yiming; Tang, Qiankun; Jin, Beibei; Han, Yainhe, and Li, Xiaowei. See and Think: Disentangling Semantic Scene Completion. In *Advances in Neural Information Processing Systems (NIPS)*, pages 261–272, 2018. (Not cited.)

Long, Jonathan; Shelhamer, Evan, and Darrell, Trevor. Fully convolutional networks for semantic segmentation. In *IEEE Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 3431–3440, 2015. (Not cited.)

Lowe, David G. Distinctive image features from scale-invariant keypoints. *Int. Journal on Computer Vision (IJCV)*, 60:91–110, 2004. (Not cited.)

Luc, Pauline; Couprie, Camille; Chintala, Soumith, and Verbeek, Jakob. Semantic Segmentation using Adversarial Networks. In *NIPS Workshops*, 2016. (Not cited.)

Luc, Pauline; Neverova, Natalia; Couprie, Camille; Verbeek, Jakob, and LeCun, Yann. Predicting deeper into the future of semantic segmentation. In *IEEE Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 648–657, 2017. (Not cited.)

Luc, Pauline; Couprie, Camille; Lecun, Yann, and Verbeek, Jakob. Predicting future instance segmentation by forecasting convolutional features. In *European Conf. on Computer Vision (ECCV)*, pages 584–599, 2018. (Not cited.)

Mattausch, Oliver; Panozzo, Daniele; Mura, Claudio; Sorkine-Hornung, Olga, and Pajarola, Renato. Object detection and classification from large-scale cluttered indoor scans. In *Computer Graphics Forum*, volume 33, pages 11–21, 2014. (Not cited.)

Maturana, Daniel and Scherer, Sebastian. VoxNet: A 3D Convolutional Neural Network for Real-Time Object Recognition. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2015. (Not cited.)

McCormac, John; Handa, Ankur; Leutenegger, Stefan, and J.Davison, Andrew. SceneNet RGB-D: Can 5M Synthetic Images Beat Generic ImageNet Pre-training on Indoor Segmentation? In *Int. Conf. on Computer Vision (ICCV)*, 2017. (Not cited.)

Mirza, Mehdi and Osindero, Simon. Conditional generative adversarial nets. *arXiv preprint: 1411.1784*, 2014. (Not cited.)

Monszpart, Aron; Mellado, Nicolas; Brostow, Gabriel J., and Mitra, Niloy J. RAPter: Rebuilding Man-made Scenes with Regular Arrangements of Planes. In *ACM Transactions on Graphics*, volume 34, pages 103:1–103:12, 2015. (Not cited.)

Munoz, Daniel; Vandapel, Nicholas, and Hebert, Marial. Directional Associative Markov Network for 3-D Point Cloud Classification. In *Proc. of the International Symposium on 3D Data Processing, Visualization and Transmission*, pages 63–70, 2008. (Not cited.)

Munoz, Daniel; Bagnell, J Andrew; Vandapel, Nicolas, and Hebert, Martial. Contextual Classification with Functional Max-Margin Markov Networks. In *IEEE Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2009a. (Not cited.)

Munoz, Daniel; Vandapel, Nicholas, and Hebert, Martial. Onboard Contextual Classification of 3-D Point Clouds with Learned High-order Markov Random Fields. In *IEEE Int. Conference on Robotics and Automation (ICRA)*, 2009b. (Not cited.)

Nan, Liangliang; Xie, Ke, and Sharf, Andrei. A Search-classify Approach for Cluttered Indoor Scene Understanding. In *ACM Transactions on Graphics*, volume 31, pages 137:1–137:10, New York, NY, USA, 2012. (Not cited.)

Neuhold, Gerhard; Ollmann, Tobias; Rota Bulo, Samuel, and Kontschieder, Peter. The Mapillary Vistas Dataset for Semantic Understanding of Street Scenes. In *Int. Conf. on Computer Vision (ICCV)*, 2017. (Not cited.)

Paszke, Adam; Chaurasia, Abhishek; Kim, Sangpil, and Culurciello, Eugenio. Enet: A deep neural network architecture for real-time semantic segmentation. *arXiv preprint: 1606.02147*, 2016. (Not cited.)

Paszke, Adam; Gross, Sam; Chintala, Soumith; Chanan, Gregory; Yang, Edward; DeVito, Zachary; Lin, Zeming; Desmaison, Alban; Antiga, Luca, and Lerer, Adam. Automatic differentiation in PyTorch. In *Advances in Neural Information Processing Systems - Workshops*, 2017. (Not cited.)

Pathak, Deepak; Krähenbühl, Philipp; Donahue, Jeff; Darrell, Trevor, and Efros, Alexei. Context Encoders: Feature Learning by Inpainting. In *IEEE Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2016a. (Not cited.)

Pathak, Deepak; Krahenbuhl, Philipp; Donahue, Jeff; Darrell, Trevor, and Efros, Alexei A. Context encoders: Feature learning by inpainting. In *IEEE Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 2536–2544, 2016b. (Not cited.)

Pohlen, Tobias; Hermans, Alexander; Mathias, Markus, and Leibe, Bastian. Full-resolution residual networks for semantic segmentation in street scenes. In *IEEE Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 4151–4160, 2017. (Not cited.)

Qi, Charles R; Su, Hao; Mo, Kaichun, and Guibas, Leonidas J. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. In *IEEE Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017a. (Not cited.)

Qi, Charles Ruizhongtai; Yi, Li; Su, Hao, and Guibas, Leonidas J. PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space. In *Advances in Neural Information Processing Systems (NIPS)*, 2017b. (Not cited.)

Radford, Alec; Metz, Luke, and Chintala, Soumith. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint: 1511.06434*, 2015. (Not cited.)

Redmon, Joseph and Farhadi, Ali. YOLOv3: An Incremental Improvement. *arXiv preprint: 1804.02767v1*, 2018. (Not cited.)

Ren, X; Bo, L, and Fox, Dieter. RGB-(D) scene labeling: Features and Algorithms. In *IEEE Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2012. (Not cited.)

Rethage, Dario; Wald, Johanna; Sturm, Jürgen; Navab, Nassir, and Tombari, Federico. Fully-Convolutional Point Networks for Large-Scale Point Clouds. *European Conf. on Computer Vision (ECCV)*, 2018. (Not cited.)

Riegler, Gernot; Ulusoy, Ali Osman, and Geiger, Andreas. OctNet: Learning Deep 3D Representations at High Resolutions. In *IEEE Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017. (Not cited.)

Ristin, Marko; Gall, Juergen, and Van Gool, Luc. Local Context Priors for Object Proposal Generation. In *Asian Conference on Computer Vision (ACCV)*, pages 57–70, 2013. (Not cited.)

Rock, Jason; Gupta, Tanmay; Thorsen, Justin; Gwak, JunYoung; Shin, Daeyun, and Hoiem, Derek. Completing 3D object shape from one depth image. In *IEEE Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2015. (Not cited.)

Ronneberger, Olaf; Fischer, Philipp, and Brox, Thomas. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015. (Not cited.)

Ros, German; Sellart, Laura; Materzynska, Joanna; Vazquez, David, and Lopez, Antonio. The SYNTHIA Dataset: A Large Collection of Synthetic Images for Semantic Segmentation of Urban Scenes. In *IEEE Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, June 2016. (Not cited.)

Rosenblatt, Frank. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958. (Not cited.)

Roth, Kevin; Lucchi, Aurelien; Nowozin, Sebastian, and Hofmann, Thomas. Stabilizing training of generative adversarial networks through regularization. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2018–2028, 2017. (Not cited.)

Roynard, Xavier; Deschaud, Jean-Emmanuel, and Goulette, François. Paris-Lille-3D: A large and high-quality ground-truth urban point cloud dataset for automatic segmentation and classification. *The International Journal of Robotics Research*, 37(6):545–557, 2018. (Not cited.)

Rumelhart, David; Hinton, Geoffrey, and Williams, Ronald. Learning representations by back-propagating errors. *Cognitive modeling*, 323(6088):533–536, 1986. (Not cited.)

Salimans, Tim; Goodfellow, Ian; Zaremba, Wojciech; Cheung, Vicki; Radford, Alec, and Chen, Xi. Improved techniques for training gans. In *Advances in Neural Information Processing Systems (NIPS)*, 2016. (Not cited.)

Schulz, Hannes; Waldvogel, Benedikt; Sheikh, Rasha, and Behnke, Sven. CURFIL: Random Forests for Image Labeling on GPU. In *International Conference on Computer Vision Theory and Applications*, 2015. (Not cited.)

Shao, Tianjia; Xu, Weiwei; Zhou, Kun; Wang, Jingdong; Li, Dongping, and Guo, Baining. An Interactive Approach to Semantic Modeling of Indoor Scenes with an RGBD Camera. In *ACM Transactions on Graphics (TOG)*, 2012. (Not cited.)

Shi, Jianbo and Malik, Jitendra. Normalized cuts and image segmentation. *Departmental Papers (CIS)*, page 107, 2000. (Not cited.)

Shi, Yifei; Long, Pinxin; Xu, Kai; Huang, Hui, and Xiong, Yueshan. Data-driven Contextual Modeling for 3D Scene Understanding. In *Computers & Graphics*, volume 55, pages 55–67, 2016. (Not cited.)

Shotton, Jamie; Johnson, Matthew, and Cipolla, Roberto. Semantic texton forests for image categorization and segmentation. In *IEEE Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2008. (Not cited.)

Shrivastava, Ashish; Pfister, Tomas; Tuzel, Oncel; Susskind, Joshua; Wang, Wenda, and Webb, Russell. Learning from Simulated and Unsupervised Images through Adversarial Training. In *IEEE Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017. (Not cited.)

Silberman, Nathan; Hoiem, Derek; Kohli, Pushmeet, and Fergus, Rob. Indoor Segmentation and Support Inference from RGBD Images. In *European Conf. on Computer Vision (ECCV)*, 2012. (Not cited.)

Simonyan, Karen and Zisserman, Andrew. Very Deep Convolutional Networks for Large-Scale Image Recognition. In *Int. Conf. on Learning Representations (ICLR)*, 2015. (Not cited.)

Song, Shuran and Xiao, Jianxiong. Deep Sliding Shapes for Amodal 3D Object Detection in RGB-D Images. In *IEEE Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2016. (Not cited.)

Song, Shuran; Yu, Fisher; Zeng, Andy; Chang, Angel X; Savva, Manolis, and Funkhouser, Thomas. Semantic Scene Completion from a Single Depth Image. In *IEEE Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017. (Not cited.)

Sturgess, Paul; Alahari, Karteek; Ladicky, Lubor, and Torr, Philip HS. Combining appearance and structure from motion features for road scene understanding. In *British Machine Vision Conference (BMVC)*, 2009. (Not cited.)

Su, Hang; Jampani, Varun; Sun, Deqing; Maji, Subhransu; Kalogerakis, Evangelos; Yang, Ming-Hsuan, and Kautz, Jan. SPLATNet: Sparse Lattice Networks for Point Cloud Processing. In *IEEE Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018. (Not cited.)

Tatarchenko, Maxim; Park, Jaesik; Koltun, Vladlen, and Zhou, Qian-Yi. Tangent Convolutions for Dense Prediction in 3D. In *IEEE Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018. (Not cited.)

Tchapmi, Lyne; Choy, Christopher; Armeni, Iro; Gwak, JunYoung, and Savarese, Silvio. SEGCloud: Semantic Segmentation of 3D Point Clouds. In *Proc. of the International Conference on 3D Vision (3DV)*, 2017. (Not cited.)

Te, Gusi; Hu, Wei; Guo, Zongming, and Zheng, Amin. RGCNN: Regularized Graph CNN for Point Cloud Segmentation. *arXiv preprint: 1806.02952*, 2018. (Not cited.)

Thanh Nguyen, Duc; Hua, Binh-Son; Tran, Khoi; Pham, Quang-Hieu, and Yeung, Sai-Kit. A field model for repairing 3D shapes. In *IEEE Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2016. (Not cited.)

Tighe, Joseph and Lazebnik, Svetlana. Superparsing. *Int. Journal on Computer Vision (IJCV)*, 101 (2):329–349, 2013. (Not cited.)

Torralba, A. and Efros, A. Unbiased Look at Dataset Bias. In *IEEE Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2011. (Not cited.)

Torralba, Antonio. Contextual priming for object detection. *Int. Journal on Computer Vision (IJCV)*, 53(2):169–191, 2003. (Not cited.)

Triebel, Rudolph; Kersting, Krisitian, and Burgard, Wolfram. Robust 3D Scan Point Classification using Associative Markov Networks. In *IEEE Int. Conference on Robotics and Automation (ICRA)*, pages 2603–2608, 2006. (Not cited.)

Van den Bergh, Michael; Boix, Xavier; Roig, Gemma; de Capitani, Benjamin, and Van Gool, Luc. Seeds: Superpixels extracted via energy-driven sampling. In *European Conf. on Computer Vision (ECCV)*, pages 13–26. Springer, 2012. (Not cited.)

Varley, J; DeChant, C; Richardson, A; Nair, A andRuales J, and Allen, P. Shape completion enabled robotic grasping. In *IEEE Conference on Intelligent Robots and Systems*, 2017. (Not cited.)

Vaswani, Ashish; Shazeer, Noam; Parmar, Niki; Uszkoreit, Jakob; Jones, Llion; Gomez, Aidan N; Kaiser, Łukasz, and Polosukhin, Illia. Attention is all you need. In *Advances in Neural Information Processing Systems (NIPS)*, pages 5998–6008, 2017. (Not cited.)

Vincent, Luc and Soille, Pierre. Watersheds in digital spaces: an efficient algorithm based on immersion simulations. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, (6): 583–598, 1991. (Not cited.)

Wang, Panqu; Chen, Pengfei; Yuan, Ye; Liu, Ding; Huang, Zehua; Hou, Xiaodi, and Cottrell, Garrison. Understanding convolution for semantic segmentation. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1451–1460. IEEE, 2018a. (Not cited.)

Wang, Shenlong; Suo, Simon; Ma, Wei-Chiu; Pokrovsky, Andrei, and Urtasun, Raquel. Deep Parametric Continuous Convolutional Neural Networks. In *IEEE Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018b. (Not cited.)

Wang, Weiyue; Huang, Qiangui; You, Suya; Yang, Chao, and Neumann, Ulrich. Shape Inpainting using 3D Generative Adversarial Network and Recurrent Convolutional Networks. In *arXiv preprint: 1711.06375*, 2017. (Not cited.)

Wang, Xiaolong and Gupta, Abhinav. Generative image modeling using style and structure adversarial networks. In *European Conf. on Computer Vision (ECCV)*, pages 318–335. Springer, 2016. (Not cited.)

Wang, Yida; Tan, David Joseph; Navab, Nassir, and Tombari, Federico. Adversarial Semantic Scene Completion from a Single Depth Image. In *Proc. of the International Conference on 3D Vision (3DV)*, pages 426–434, 2018c. (Not cited.)

Wang, Zongji and Lu, Feng. VoxSegNet: Volumetric CNNs for Semantic Part Segmentation of 3D Shapes. *arXiv preprint: 1809.00226*, 2018. (Not cited.)

Wilms, Christian and Frintrop, Simone. Edge adaptive seeding for superpixel segmentation. In *German Conference on Artificial Intelligence (GCPR)*, pages 333–344. Springer, 2017. (Not cited.)

Wu, Bichen; Wan, Alvin; Yue, Xiangyu, and Keutzer, Kurt. SqueezeSeg: Convolutional Neural Nets with Recurrent CRF for Real-Time Road-Object Segmentation from 3D LiDAR Point Cloud. In *IEEE Int. Conference on Robotics and Automation (ICRA)*, 2018. (Not cited.)

Wu, Jiajun; Zhang, Chengkai; Xue, Tianfan; Freeman, Bill, and Tenenbaum, Josh. Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling. In *Advances in Neural Information Processing Systems (NIPS)*, pages 82–90, 2016. (Not cited.)

Wu, Zhirong; Song, Shuran; Khosla, Aditya; Yu, Fisher; Zhang, Linguang; Tang, Xiaoou, and Xiao, Jianxiong. 3D ShapeNets: A deep representation for volumetric shapes. In *IEEE Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2015. (Not cited.)

Xie, Jun; Kiefel, Martin; Sun, Ming-Ting, and Geiger, Andreas. Semantic Instance Annotation of Street Scenes by 3D to 2D Label Transfer. In *IEEE Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2016. (Not cited.)

Xiong, Xuehan; Munoz, Daniel; Bagnell, J. Andrew, and Hebert, Martial. 3-D Scene Analysis via Sequenced Predictions over Points and Regions. In *IEEE Int. Conference on Robotics and Automation (ICRA)*, pages 2609–2616, 2011. (Not cited.)

Yang, Bin; Yan, Junjie; Lei, Zhen, and Li, Stan Z. Convolutional Channel Features. In *Int. Conf. on Computer Vision (ICCV)*, pages 82–90, 2015. (Not cited.)

Yang, Bo; Wen, Hongkai; Wang, Sen; Clark, Ronald; Markham, Andrew, and Trigoni, Niki. 3D Object Reconstruction from a Single Depth View with Adversarial Learning. In *arXiv preprint: 1708.07969*, 2017. (Not cited.)

Yang, Bo; Lai, Zihang; Lu, Xiaoxuan; Lin, Shuyu; Wen, Hongkai; Markham, Andrew, and Trigoni, Niki. Learning 3D Scene Semantics and Structure from a Single Depth Image. In *IEEE Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 309–312, 2018a. (Not cited.)

Yang, Bo; Rosa, Stefano; Markham, Andrew; Trigoni, Niki, and Wen, Hongkai. Dense 3D Object Reconstruction from a Single Depth View. In *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 2018b. (Not cited.)

Yang, Yiqing; Li, Zhouyuan; Zhang, Li; Murphy, Christopher; Ver Hoeve, Jim, and Jiang, Hongrui. Local label descriptor for example based semantic image labeling. In *European Conf. on Computer Vision (ECCV)*, pages 361–375. 2012. (Not cited.)

Yu, Changqian; Wang, Jingbo; Peng, Chao; Gao, Changxin; Yu, Gang, and Sang, Nong. Learning a discriminative feature network for semantic segmentation. In *IEEE Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 1857–1866, 2018. (Not cited.)

Zeng, Wei and Gevers, Theo. 3DContextNet: K-d Tree Guided Hierarchical Learning of Point Clouds Using Local and Global Contextual Cues. *arXiv preprint: 1711.11379*, 2017. (Not cited.)

Zhang, Chenxi; Wang, Liang, and Yang, Ruigang. Semantic segmentation of urban scenes using dense depth maps. In *European Conf. on Computer Vision (ECCV)*, pages 708–721. 2010. (Not cited.)

Zhang, Jiahui; Zhao, Hao; Yao, Anbang; Chen, Yurong; Zhang, Li, and Liao, Hongen. Efficient Semantic Scene Completion Network with Spatial Group Convolution. In *European Conf. on Computer Vision (ECCV)*, pages 733–749, 2018. (Not cited.)

Zhao, Hengshuang; Shi, Jianping; Qi, Xiaojuan; Wang, Xiaogang, and Jia, Jiaya. Pyramid scene parsing network. In *IEEE Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 2881–2890, 2017. (Not cited.)

Zhao, Hengshuang; Qi, Xiaojuan; Shen, Xiaoyong; Shi, Jianping, and Jia, Jiaya. Icnet for real-time semantic segmentation on high-resolution images. In *European Conf. on Computer Vision (ECCV)*, pages 405–420, 2018a. (Not cited.)

Zhao, Hengshuang; Zhang, Yi; Liu, Shu; Shi, Jianping; Change Loy, Chen; Lin, Dahua, and Jia, Jiaya. Psanet: Point-wise spatial attention network for scene parsing. In *European Conf. on Computer Vision (ECCV)*, pages 267–283, 2018b. (Not cited.)

Zheng, B; Zhao, Y; Yu, J C; Ikeuchi, K, and Sx-Cx, Zhu. Beyond point clouds: Scene understanding by reasoning geometry and physics. In *IEEE Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2016. (Not cited.)

Zhou, Yipin and Berg, Tamara L. Learning temporal transformations from time-lapse videos. In *European Conf. on Computer Vision (ECCV)*, pages 262–277. Springer, 2016. (Not cited.)

Zhu, Jun-Yan; Krähenbühl, Philipp; Shechtman, Eli, and Efros, Alexei A. Generative visual manipulation on the natural image manifold. In *European Conf. on Computer Vision (ECCV)*, pages 597–613. Springer, 2016. (Not cited.)