Utrecht University

Neural multi-view segmentation-aggregation for joint Lidar and image object detection

MASTER THESIS
ICA-3769240

Author:
Berry van Someren

Supervisor:
Dr. Ir. Ronald W. Poppe

External Supervisors:
Dr. Ir. Bas Boom
Arjen Swart

Second Examiner:
Prof. Remco C. Veltkamp

August 6, 2017

Abstract

Combining different types of data from multiple views makes it easier to perform object detection. Our novel method enables multi-view deep convolutional neural networks to combine color information from panoramic images and depth information derived from Lidar point clouds for improved street furniture detection. Our focus is on the prediction of world positions of light poles specifically. In contrast to related methods, our method operates on data from real world environments consisting of many complex objects and supports the combination of information from recording locations that do not have fixed relative positions.

We make five contributions in this work. First, we propose a single-view pipeline which produces segmentations that are reconstructed as labeled point clouds, after which a three-dimensional clustering method extracts the world positions for the segmented objects. Then, a scalable method to generate ground truth to train the segmentation-network is described. Different depth-derived features are explored to find an optimal data representation to transfer information to the neural segmentation-network. A novel method is then introduced that refocuses and reprojects images, based on the depth information of Lidar point clouds, so that correlation between images from different camera poses is directly defined and no longer has to be inferred through complex matching algorithms. Finally, a multi-view pipeline is introduced that leverages refocusing and reprojection to combine data from multiple recording locations. This combination of geometric and deep learning methods has never been performed before.

Results show that a single-view pipeline using solely depth-information outperforms a pipeline with only color information. A deep convolutional neural network trained on the combined representation creates higher quality segmentations than networks trained on color or depth information alone. It is also shown that better segmentations generally result in better clustering and world position estimates. The multi-view pipeline operating on just RGB-images correctly predicts 66% of all light poles.

The method is easily extensible to other types of street furniture objects and currently utilizes neural network architectures with relatively small computation and memory requirements. However, the pipeline does not dictate any specific neural network architecture and can in the future easily be updated with more modern networks for improved performance. The method will be productized by Cyclomedia to obtain data for remote inspection and inventory purposes, and will be extended with support for road markings.

Contents

Chapter 1

Introduction

Object detection performance can be improved by combining different types of data from multiple sources. We discuss how information from street-level panoramic images and Lidar point clouds can be combined in the detection of street furniture objects such as light poles and traffic signs. Detection of street furniture objects such as light poles and traffic signs can be automated using machine learning techniques. Deep learning is currently one of the most powerful machine learning tools for data processing and deep neural networks can be trained to capture abstract features that would otherwise be too complex to model by hand. The main challenge in this work is to effectively combine the information from multiple images and point clouds into a single representation.

1.1 Context

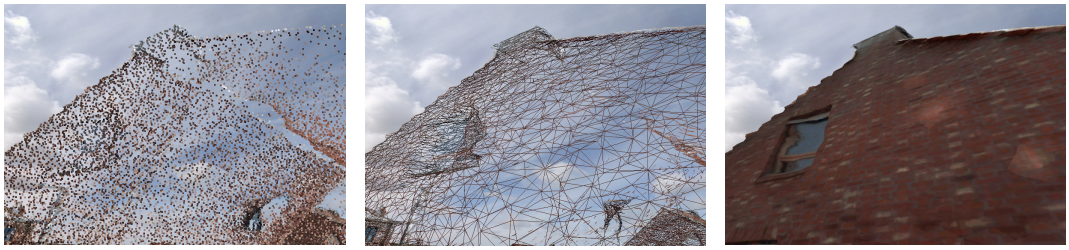
CycloMedia is a company that records the public space with 100 megapixel, 360 degree cylindrical panoramic images (cycloramas) and Lidar point clouds. Cyclomedia's data allows users to virtually perform tasks such as tax assessment, inspection, inventory and planning for infrastructure, public safety and transportation sectors, without the need to visit each location physically. The precision of the coordinates is 2 centimeters on average, which allows users to perform highly accurate positioning and measurements within the virtual data. By combining information from point clouds and images a user can measure for example the area and angle of a roof, just by clicking on the corners of the roof within an image.

The data is obtained from the Digital Cyclorama Recorder (DCR) system, which is mounted on top of driving cars that record the entirety of the Netherlands and parts of France, Germany, Scandinavia and the USA. The panoramic images are recorded with a camera system that uses smart triggering to combine images from multiple cameras into a single panoramic image. The panoramic images are recorded at an interval of 5 meters. The point clouds are sets of three-dimensional data points that represents the surrounding scene. The point clouds a measured with a Lidar laser sensor, which continuously fires lasers in different directions to to scan the surroundings. The DCR system is illustrated in Figure 1.1.

(A) (B) (C)

FIGURE 1.1: A) CycloMedia's DCR10 system. The system contains five cameras whose focal points lie on a line in order to create parallax-free cycloramas. B) A Velodyne HDL-32E Lidar sensor. C) A Cyclomedia Car; a Ford Fiesta with a DCR10L system, including Lidar scanner, mounted on top.

However, point clouds offer only a sparse sampling of the scene. While information of the measured points is available, the areas and relations between points are still unknown, and the point clouds cannot directly be used to extract depth values for each pixel. Therefore, surface reconstruction algorithms are applied to reconstruct the point cloud as a **mesh**. Meshes are the conventional representation of 3D models in many applications that explicitly store which points are connected to each other, approximating the original surface. The surface can be colored with images in a process called texturing. Figure 1.2 illustrates a point cloud reconstructed as a textured mesh.



(A) (B) (C)

FIGURE 1.2: a) A point cloud of the facade of a house. b) The edges of the reconstructed mesh. c) The textured reconstructed mesh.

The mesh can then be used as an approximation to the actual scene surfaces. For every recorded panorama, the corresponding camera location, orientation, focal length and other lens parameters are known. The image recorded from a camera can be considered a virtual plane that is placed in front of the camera at a distance determined by the lens. Each pixel can therefore be associated with a direction vector, which is the vector from the camera's lens, through that pixel in the image plane, into the world. By calculating where this vector intersects with the mesh, a depth value can be computed for each pixel. A simplified camera model illustrating how each pixel can be related to a direction is illustrated in Figure 1.3. This combination of the original cyclorama and depth values is called a **depth cyclorama**. Figure 1.4 illustrates a cyclorama, its mesh reconstruction, and a depth image. It should be noted, that as point clouds and reconstructed meshes can contain inaccuracies, the depth values do not always intuitively correspond to the depicted scene. Moving objects have been filtered from the point clouds and sometimes the point clouds and images have been measured at slightly different timestamps, further increasing this effect.

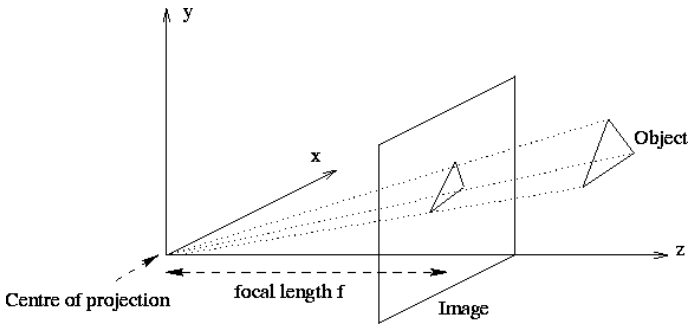
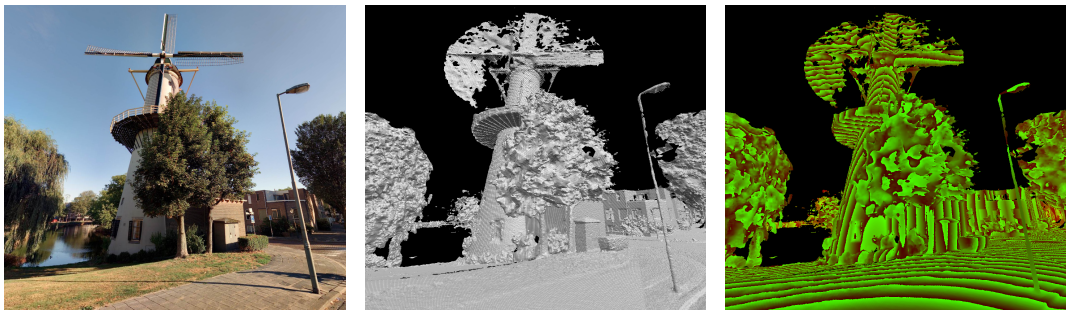


FIGURE 1.3: This image illustrates the geometric relationship between a three-dimensional point and its corresponding two-dimensional projection onto the image plane of a camera. The virtual image plane is a plane located at a focal length distance from the center of projection. The focal length is determined by the camera lens. For a pair of pixel coordinates we can then compute the vector from the center of projection through that pixel. By computing where this vector intersects the geometry of the scene, a depth value is computed. In reality the model is a bit more complex since rays from the center of projection are refracted through a lens, and the center of projection is not an infinitely small point, but an area defined by the camera aperture.



(A) (B) (C)

FIGURE 1.4: A) A region of a cycloramic image. B) The corresponding mesh as reconstructed from the point cloud. Note that the sails of the windmill were moving during recording, resulting in superfluous mesh parts. Additionally, some parts of the trees and the light pole are missing. The water is not represented in the point cloud altogether since the lasers from the Lidar sensor were not properly reflected back. C) The depth image as created from the mesh. The depth is encoded in the red and green image channels for higher precision depth storage and improved local depth contrast.

One of CycloMedia's goals is to automate the detection of street furniture such as light poles and traffic signs in the data, for inventory purposes. The detection task consist of finding the objects that belong to a certain class within an image. CycloMedia already performs object detection in regular images without depth information. However, classification can be difficult when no geometrical data is available. For example, when objects with similar colors appear next to each other within an image, it is difficult to estimate the boundary between the objects. Another limitation of regular images, is that a location within an image, cannot easily be translated to a position in world coordinates. One solution to this problem is to detect the same object in multiple images, obtained from nearby recording locations. The vector from the camera towards the object can then be computed in each view, and the intersection of these vectors can be calculated as the position of the object. However, this method involves many inaccuracies as the directional vectors might diverge form each other, or might have intersections at locations where no object is present. It is expected that the additional depth information would improve accuracy for both the detection within an image and the world positioning task. Object detection within the image could be improved since the additional depth information allows for more descriptive and discriminative features. Even if object with the same colors appear next to each other, they could be distinguished by their depth, or three-dimensional shape. The step to map the detection within the image into a world position also becomes less complex as depth information is readily available and does not have to be inferred through multiple images. It is therefore interesting to further explore how the color information from images and depth information from Lidar point clouds could be combined.

The traditional process of detecting objects automatically, involves the design of descriptive and discriminative features. It is inefficient to design such features manually, since they are complex, and new features have to be designed for every class. Machine learning models can be trained to define such features automatically as long as a representative data set is available. The benefit of this is that the architecture does not need to be changed when a new type of object has to be recognized. Since Cyclomedia deals with large volumes of data, and the set of objects that need to be detected keeps expanding, machine learning techniques offer a flexible and scalable solution. Artificial Neural Networks are currently one of the most powerful machine learning techniques. However, finding efficient data representations, creating large annotated datasets and designing a network that can capture the required features efficiently still poses a challenge.

To train a neural network, a large annotated dataset is required that defines a golden standard, also called the ground truth, consisting of examples of what the neural network should learn. Such a dataset might not be available for the chosen data representation. Creating such a dataset in a fashion that is similarly flexible and scalable as the neural network is an additional challenge.

Since Cyclomedia deals with large quantities of data, it is desired that the neural network is fast and has low memory requirements so that it can be invoked many times in parallel. While this is not a problem that we focus on in this work, it is something we bear in mind. Another desired property is that the pipeline should be general enough to deal with a range of street furniture objects such as garbage bins, road markings and traffic signs. However, to limit the scope of this work, the primary class of interest in this work consists of light poles.

1.2 Research questions

The goal is to introduce a neural network pipeline which computes the world coordinates of street furniture objects, using information from both images and point clouds. It is expected that the additional information from the point clouds improves performance over regular images. The main research question of this work is as follows.

- *Research Question*
  How could cycloramas and Lidar point clouds be combined for improved street furniture detection?

Although the additional depth information is expected to improve detection performance, the data representation significantly affects its usability. It might therefore be beneficial to design a neural network that directly operates on point clouds, directly operates on depth cycloramas, or operates on any representation that is a combined derivative of both. This problem introduces a new sub-question.

- *Sub-question 1*
  In what representation could the panoramic images and Lidar point clouds from a single recording location best be combined for optimal neural network processing?

Since Cyclomedia obtains data from many recording locations, this information can be leveraged to overcome some of the limitations of considering recording locations individually. If an object is missing or incomplete within the data from a single recording locations, this data could be supplemented with data from other recording locations. How the information from multiple recording locations could be combined poses another challenge where the representation is crucial.

- *Sub-question 2*
  How could data from multiple recording locations be combined, so that inaccuracies within individual recording locations can be overcome with supplementary information from others?

1.3 Contributions

Five main contributions are made in this work.

- **A single-view pipeline is introduced which estimates light pole positions from images with depth information.** First a deep convolutional neural network is used for image segmentation. Then, using the available depth information, the segmentation is reconstructed as a labeled point cloud. A three-dimensional clustering algorithm then extracts the positions of the segmented objects from the segmentation-derived point cloud. Only few methods have used depth information in segmentation-networks, and the combination of segmentation with point cloud clustering is unique. The extraction of depth information from Lidar point clouds for segmentation purposes specifically, has not been illustrated before.
- **A method to generate ground truth segmentation images using the depth information from Lidar point clouds is described.** The data that is available to us consists of object positions, panoramic images and Lidar point clouds. However, annotated images that can directly be used for training our neural segmentation-network are not available. Our method produces high quality ground truth segmentations fully automatically.

- **A comparison is made between neural networks using color information from panoramic images, depth information form Lidar point clouds, and derived combinations of both.** Several representations that combine color and depth-derived features are explored. Furthermore, in the literature depth images are mostly explored independently from color, applied on virtual scenes with few objects, for solving classification tasks. Our problem is more difficult as real world scenes are more complex and consist of many objects. Additionally our method combines both color and depth information and does not only involve classification but also the extraction of real world positions.
- **A novel method is introduced that supports the combination of information from recording locations that do not have fixed relative positions.** In related work dealing with virtual environments, recording locations with fixed relative positions are chosen. This enables a neural network to easily correlate the data. However, in the real world, recording locations can not be chosen arbitrarily. Because our recording cars are bound to the roads and have to make turns, the trajectories at which scenes are approached vary wildly. In our novel method, images are refocused and reprojected to a target pose using the depth information from Lidar point clouds so that they directly overlap and correlation no longer has to be inferred.
- **A novel type of multi-view pipeline is illustrated which leverages the power of image refocusing and reprojections to combine information from multiple recording locations that do not have fixed relative positions.** In related work, information from multiple recordings locations is often combined through pooling operations followed by a shallow convolutional neural network. However, pooling operations quickly reduce data dimensionality, without extracting features first, and therefore valuable information is lost. We investigate the application of a deep convolutional neural network that jointly performs pooling and feature estimation for aggregation purposes. This specific combination of geometric and deep learning methods has not been illustrated before.

1.4 Outline

A literature study providing an overview of neural networks related to our problem is provided in Chapter 2. Chapter 3 discusses our pipeline in which information from both images and Lidar point clouds is combined for segmentation, and segmentation-derived point clouds are clustered to extract world positions for the objects. In addition to using depth information directly, several other depth-derived features are explored for improved performance. A novel method is then proposed to refocus and reproject images to a target camera, so that information from multiple recording locations can be directly correlated, without the need for complex matching methods. This method is then incorporated in our pipeline enabling for improved performance. In Chapter 4 the implementation of the method, the dataset, ground truth creation method and training methodology and evaluation metrics are discussed. In Chapter 5 a quantitative analysis illustrates the performance of the proposed method. Both the quality of the segmentations and the quality of the extracted object positions are evaluated. Finally, Chapter 6 concludes this work with a summary and a discussion of future work.

Chapter 2

Literature Study

Deep learning is one of the most powerful tools in machine learning. We therefore primarily focus on deep neural networks throughout this work. In this Chapter an overview is provided of the neural network developments that are most relevant to our problem. Section 2.1 provides an introduction to artificial networks. In the following sections we focus on the differences and developments in architectures for different data types and representations. Section 2.2 discuss methods based on images, while 2.3 discusses methods that use multiple images to incorporate a notion of three-dimensionality. Methods that operate on meshes by partitioning the space are discussed in Section 2.4. Finally we discuss methods that process meshes by transforming them to the frequency domain in Section 2.5.

2.1 An introduction to Artificial Neural Networks

The traditional way of solving problems in computer science is by manually defining rules or algorithms. So, if the goal is to create a program that distinguishes between cats and dogs, the first step is to define rules that determine when an image depicts a cat, and when it depicts a dog. Rules could define the shape of the animals, the differences in color, and the texture of their fur. These distinctive characteristics are examples of features. However, manually designing features is tedious and does not scale well to more classes and complex features. If the program would have to distinguish between fish too, the programmer needs to create an entire new set of features.

Machine learning techniques such as **Artificial neural networks (ANNs)** can be trained to automatically capture abstract features that would otherwise be too complex to model by hand. Neural networks can be adapted to solve different tasks such as estimating relations between variables (regression), grouping data (classification), finding a bounding box for an object in an image (object detection) or finding all the pixels in an image that belong to a class (semantic segmentation). Neural networks have been adapted to many data types including raw numbers, text, images, sound, video, 3D models and point clouds. They have become one of the most powerful tools for machine learning.

As their name might suggest, artificial neural networks were originally inspired by biological neural networks, such as the brain. The key principle is the composition of a large number of highly interconnected processing units that solve a specific task together. Artificial neural networks have since diverged from the biological neural networks by introducing more mathematical, statistical and computational elements tailored to solve specific computation tasks. While for modern artificial neural networks the analogy to the brain only loosely holds, it still offers an intuitive introduction.

A biological neuron receives input signals through its dendrites. Outputs signals travel along the axon, which branches into synapses that are connected to the dendrites of other neurons. Figure 2.1 illustrates a biological neuron. In 1943 the **McCulloch-Pitts (MCP) model** [1] was presented as a mathematical simplification of the neuron. Many of these artificial neurons wired together in a network, would theoretically be able to reproduce the highly complex patterns as they occur in the brain. A McCulloch-Pitts neuron computes a sum of its inputs, where each input is weighted equally, and compares it to a threshold. The threshold is similar to how a biological neuron only fires a signal if its action potential reaches a certain threshold. Adjusting the weights is analogue

to the Hebbian learning principle; "When an axon of cell A is near enough to excite a cell B and repeatedly or persistently takes part in firing it, some growth process or metabolic change takes place in one or both cells such that A's efficiency, as one of the cells firing B, is increased." [2]. This is summarized by Löwel as: "neurons wire together if they fire together" [3]. In 1958 the **Perceptron** was introduced by Rosenblatt [4], which extended the MCP model with more flexible computational features. The Perceptron introduced separate weights for each input and a learning rule to optimize the ways analytically. The weights of a network are also called parameters. Figure 2.2 illustrates an artificial neuron.

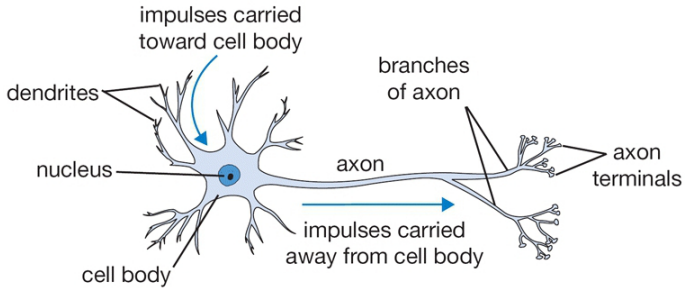


FIGURE 2.1: A biological neuron receives input signals through its dendrites. Output signals are carried away from the cell body along the axon, which branches into synapses that are connected to the dendrites of other neurons. A neuron only fires a signal if its action potential reaches a threshold.

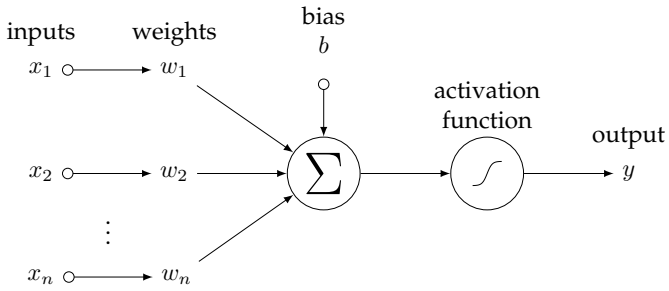


FIGURE 2.2: An artificial neuron computes a weighted sum of its inputs. A bias can be added which, together with the activation function, simulates a thresholding or scaling operation. The artificial neuron loosely resembles a biological neuron.

In 1962 Hubel and Wiesel showed that the functional architecture of the cat's visual cortex is highly hierarchical, where each layer acts on a receptive field in the previous layer [5]. Fukushima created the **Neocognitron** [6] in 1980 as the first neural network with such a hierarchical architecture. A **convolution** neuron computes for each pixel in the input image a weighted sum with its neighbors, mimicking the receptive fields in the visual Cortex. The collection of weights for a receptive field is called a convolution kernel or a filter. A convolutional neuron is depicted in Figure 2.3. Since the same convolution kernel is used for each pixel, the network is invariant to translations. The hierarchical architecture of convolution neurons enables higher level neurons to have effectively bigger receptive fields over the image. Lower level neurons learn features that correspond to edges, while higher level neurons correspond to more abstract features such as eyes, noses, mouths, or entire faces. This way, the hierarchical architecture also mimics the hierarchical structure of the world around us. Since the information from many neurons is combined in the higher levels of the hierarchy, the network becomes more invariant to noise, deformations and color changes. The architecture of the neocognitron is depicted in Figure 2.4. The features that neurons at different level in the hierarchy of a CNN correspond to are depicted in 2.5. The success of the neocognitron inspired further development of **convolutional neural networks (CNNs)**.

In 1998 Yann LeCun kept a similar architecture and layout with the **LeNet** network [8]. Instead of using a threshold, the network used a non-linear continuous activation function. The weighted

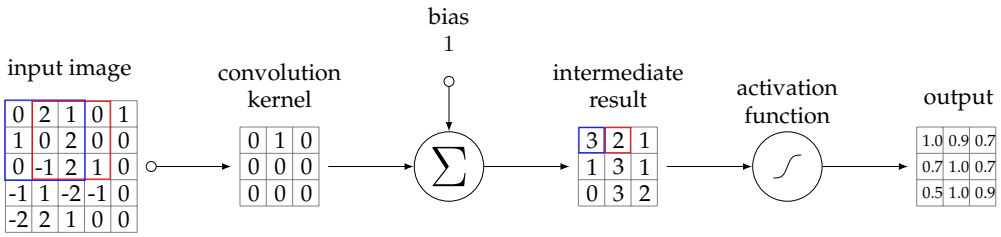


FIGURE 2.3: A convolution neuron computes a weighted sum of a local region of pixel values. For each region, it computes a single output value. The neuron considers all regions in a sliding-window fashion. In this example a sliding window size of 3×3 is used, resulting in 9 sliding window locations. The blue region in the input image corresponds to the single blue value in the intermediate result. The value has been computed as follows. First the sum of the input values is computed, while each value is weighted with the corresponding value in the convolution kernel: $(0 \times 0) + (2 \times 1) + (1 \times 0) + (1 \times 0) + (0 \times 0) + (2 \times 0) + (0 \times 0) + (-1 \times 0) + (2 \times 0) = 2$. Then the bias is added. $2 + 1 = 3$. The intermediate result is scaled by the activation function to compute the final output. The activation function used in this example is the sigmoid activation function: $y = \frac{1}{1+e^{-x}}$.

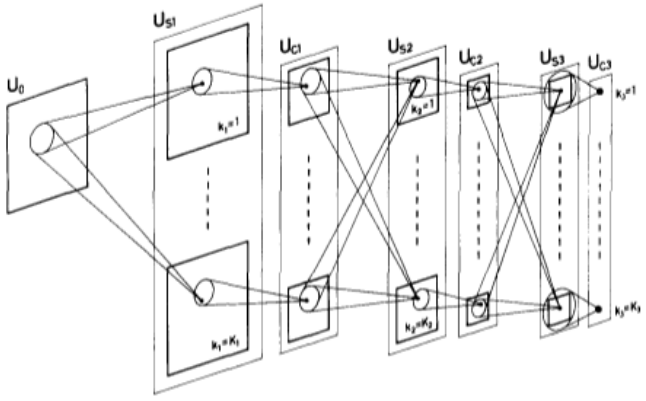


FIGURE 2.4: The architecture of the neocognitron. The hierarchical architecture mimics the visual cortex. Neurons in higher levels of the architecture have a bigger effective receptive field and are more invariant to noise, deformations and color differences.

sum of the inputs, which is a linear combination, is multiplied by the activation function to create a non-linear output. Using this non-linearity enables the network to define more complex features. A technique called **backpropagation** can be used that automatically optimizes all the weights in the network. This is called **end-to-end learning**. To train a network, training samples should be provided to illustrate the desired output. A loss function is defined to measure the difference between the desired and the predicted output. For every training sample, the partial derivatives of all weights with respect to the loss function are computed. The weights can then be automatically tweaked with **gradient descent** techniques that use the partial derivative to know whether the weights should be increased or decreased. Backpropagation is essentially the application of the chain rule for derivatives, applied on the many components of the neural network. The weights are incrementally optimized. The more training samples are available, the closer the weights will come to their optimal value. If the training data is representative of the whole data domain, then these weights will also generalize well to unseen data. However, too much training is harmful, as the network might overreact to fluctuations in the training data, disabling it to generalize to unseen data. This is called **overfitting**.

Due to powerful computers, new techniques, and the accessibility of large image databases through

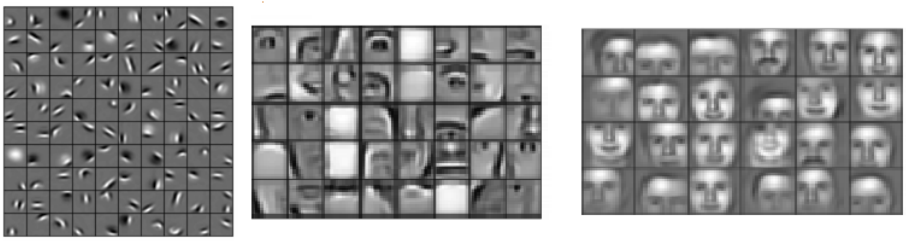


FIGURE 2.5: Hierarchical features as learned by a CNN. Features learned by neurons in the lower layers of the hierarchy correspond to edges and contrasts. Higher level features are more complex and respond to body parts such as eyes, mouth or nose. The features in the highest layers of the network correspond to faces. Image adapted from [7].

the internet, neural networks could already outperform hand-designed features at this point. Features can be interpreted as complex mathematical formulas, and neural networks can approximate any unknown formula, as long as the network has enough parameters, and appropriate training data is available. Modern CNNs rival the accuracy of the Primate IT Cortex, and even function similarly [9]. The challenge is now to define neural network architectures for different problems and different types of data that can be trained with realistic amounts of data. A general theory to tackle such challenges has, however, not been well-established. We are no longer limited by computation power or data, but rather by ideas.

2.2 Image-based methods

In this section we discuss image-based localization methods since they have been studied extensively and many image-based techniques can be transferred to other types of data and representations easily. The localization of objects in images can be approached in several ways. In both **object detection** and **instance segmentation** the goal is to classify and localize object instances. In object detection however, instances are localized using bounding boxes, while in instance segmentation the set of pixels that belongs to an instance is computed. In this section we discuss both approaches and how they are optimized for efficiency.

The general process of object detection involves the extraction of fixed-size regions from an image. The regions are then classified separately. The region that has the highest class score then defines the position of the bounding box that fits the object. If a neural network is trained to detect i.e. a cat in an image patch of 10×10 pixels, while the test image is bigger than that, all the potential regions are exhaustively considered. First the 10×10 region at the top-left position is considered. Then the window of interest "slides" from left to right, from top to bottom, to exhaustively consider all potential regions. This is called a **sliding-window** approach. The number of potential regions increases rapidly with the size of an image. However, region proposal can be generated using criteria such as color, texture or position, to limit this problem. Selective Search [10] is one method that uses a diverse set of complementary and hierarchical grouping strategies to propose regions that are likely to contain an object. This way, instead of considering all regions in an exhaustive search, only a subset has to be processed.

For object detection, Girshick et al. propose **Region-based CNNs (RCNNs)** [11]. An RCNN relies on category-independent region proposals on the image. The proposed regions are likely to contain objects of interest. A CNN is then invoked to extract a feature vector for each region independently. The feature vectors are finally classified using a category-specific linear SVM. The method is agnostic to the particular region proposal method that is used. The proposed regions can later be refined by training a per class bounding-box regressor that uses the CNN features. The architecture of an RCNN is depicted in Figure 2.6.

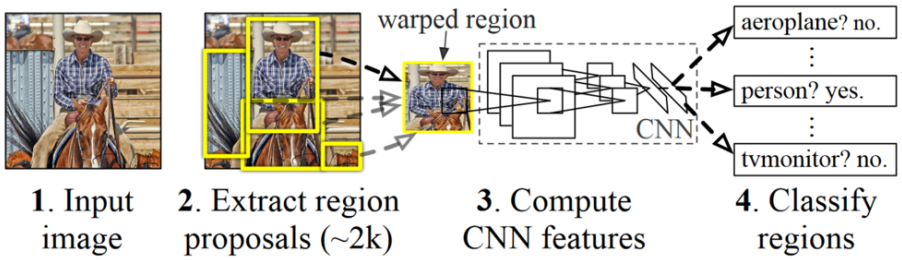


FIGURE 2.6: The RCNN architecture. Regions are first proposed by a separate method. For each region individually, the pixels are extracted from the image and then scaled to a fixed size. The CNN then computes and classifies the region features.

The RCNN however, has some notable drawbacks such as its multi-stage pipeline, which has high memory requirements and slow computation time, due to the fact that each region is processed individually, while some of this computation could be shared. The network is effectively split in two, where one part deals with generating region proposals and the other part deals with classifying each region. This inefficiency can be reduced by sharing features among the two parts of the network.

Trying to decrease region proposals as a bottleneck, Girshick later proposed the ***Fast* RCNN** [12] which contains a single stage training process that jointly learns to classify object proposals and refine their spatial locations by sharing features, improving both speed and accuracy. The Fast RCNN requires an image and a set of region proposals as input. Instead of extracting the regions directly from the image, the network first computes full-image features, and extracts the regions from these features. Instead of computing the feature for each region independently the computation is now shared over the entire image. A subnetwork then computes a class probability distribution and a set of candidate bounding boxes for each region. Non-maximum suppression is then applied independently for each class to find the best bounding box. The architecture of a Fast RCNN is depicted in Figure 2.7.

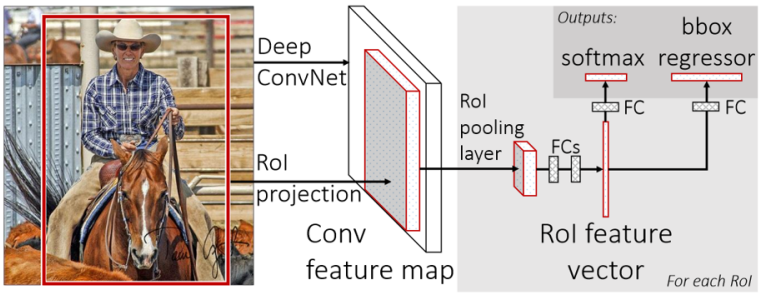


FIGURE 2.7: The Fast RCNN architecture. First, full-image features are computed. The regions are then extracted from these features. Finally, the regions are classified by the CNN and a regressor further refines the bounding box.

Later, the ***Faster* RCNN** was proposed by Ren et al. [13] to further decrease the region proposal bottleneck. While region proposals can be computed quite fast already, the fact that they are computed using different features than those used for detection makes them impractical. This can be solved by introducing a deep convolutional **Region Proposal Network (RPN)**. The RPN takes the full-image features as an input, effectively merging it with the detection network. This merged architecture is depicted in Figure 2.8. An RPN takes an image as input and outputs a set of rectangular regions, with an associated "objectness" score which estimates whether the region contains an object or not. To generate region proposals the RPN considers a sliding window over the full-image features. At each sliding window location multiple region proposals are generated using a fixed set of boxes with different scales and aspect ratios, which are later refined using

regression. Through training, the network then learns which regions are most likely to contain objects. This is similar to a neural network version of selective search.

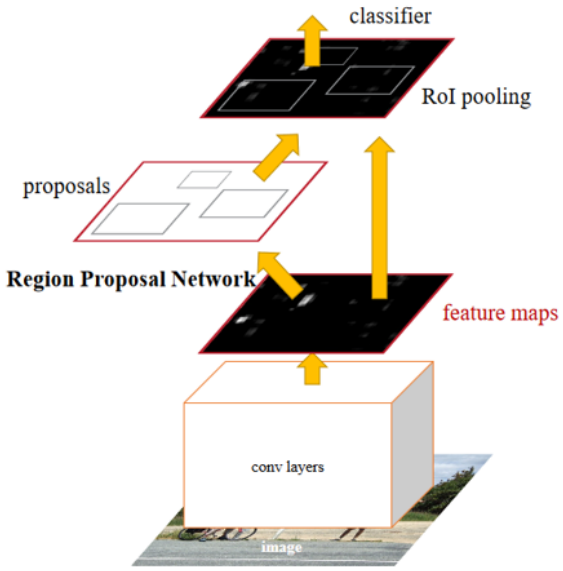


FIGURE 2.8: The Faster RCNN architecture. Instead of relying on a separate region proposal method, a neural network is trained to propose regions. The region proposal network takes full-image features as input and is therefore effectively merged with the classification network.

While in the classification task a class label is assigned to an image as whole, the semantic segmentation task consist of computing a class label for every pixel in the image individually. **Fully Convolutional Networks (FCN)** [14] have been a successful approach for semantic image segmentation, where a score map stores a class likelihood distribution for each pixel. In such maps however, different instances with the same semantic value are not recognized as separate objects as depicted in Figure 2.9. Furthermore, FCNs have been designed for high translation-*invariance* in order to find objects at different positions in the image. However, when localizing individual instances, the relative positioning of objects to each other, and thus translation-*variance* is important. The task that deals with semantic segmentation while separating class instances, is called **instance segmentation**. We will see that neural networks for both the object detection and segmentation tasks, have become highly similar.

To separate object instances from the same class, and determine which set of pixels belongs to each object, Dai et al. propose an **Instance-sensitive FCN (InstanceFCN)** [15]. The InstanceFCN stores for each pixel a classifier of relative positions to an object instance. Multiple score-maps encode these relative positions. For example, the "top-center-sensitive" score map contains high scores for features that are similar to the top center position of a class instance. This means, that instead of a single classifier that scores whether a region depicts a human, there are multiple classifiers that each score whether a sub-region depicts a specific part of a human. Each region is then subdivided into sub-regions. By copying the values from each sub-region in the corresponding score-map an instance mask proposal is assembled. The region classification is performed separately. This process is depicted in Figure 2.10.

For the classification of regions Fast and Faster RCNN, however, still apply a sub-network for each region individually. Therefore Dai et al. proposed a **Region-based FCN (R-FCN)** [16] that shares almost all computation on the entire image, eliminating the need for expensive per-region computations. The approach is very similar to the InstanceFCN. This time however, the position-sensitive score-maps are used to estimate a bounding box instead of finding a pixel mask. The only per-region computation that is required then is average voting over the score-maps for each proposed region to finalize the bounding box. The bounding box can be even further refined with

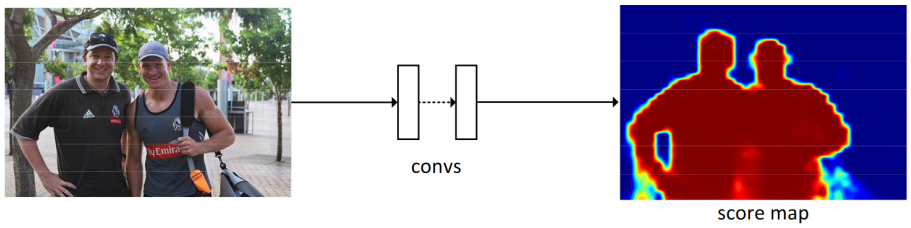


FIGURE 2.9: An input image and its score map as computed by an FCN for semantic segmentation. The network does not distinguish between class instances.

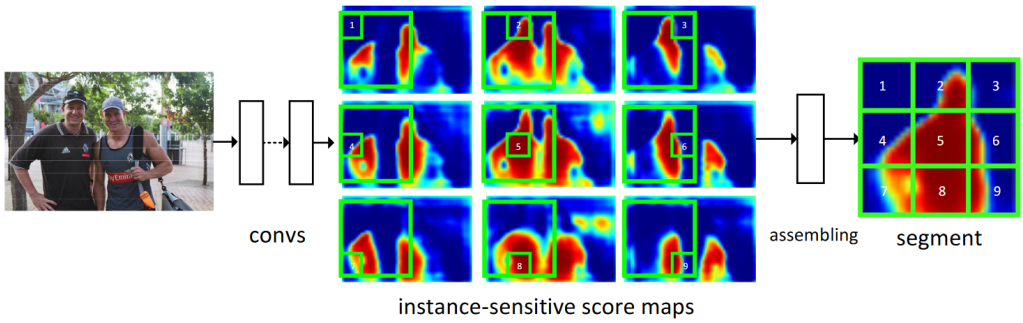


FIGURE 2.10: An InstanceFCN computes position-sensitive score-maps. For example, the "top-center-sensitive" score map contains high scores that are similar to the top center position of a class instance. A region is then subdivided into sub-regions. By copying the values from the same sub-region in the corresponding score-map an instance mask proposal is assembled.

a regressor.

Li et al. presented the first **FCN for Instance Segmentation (FCIS)** [17] that jointly performs instance mask prediction and classification. In the RFCN the position-sensitive score-maps score how much a pixel is similar to a particular region of an object, assuming it is *inside* the object. The FCIS adds another set of score-maps that score how much a pixel is similar to a particular region *outside* an object. This provides the network with important context information such as whether the object is in the air, on the ground or underwater. An object is more likely a fish when it is underwater, but less so when it is in the air. Classification is then performed by computing a max-operation over inside and outside score-maps. By applying a per-pixel softmax operation on the score-maps a foreground probability is computed which is used to create the instance mask.

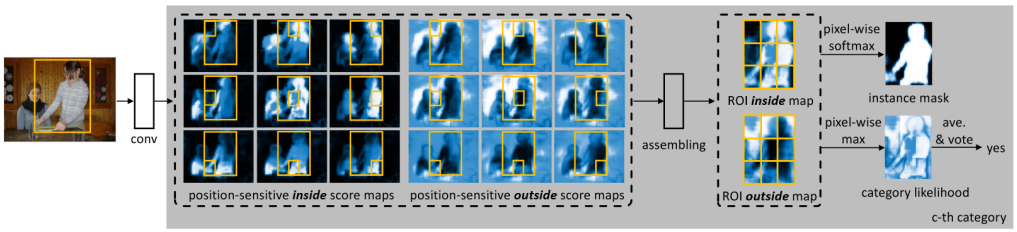


FIGURE 2.11: An FCIS computes position-sensitive score maps for regions both inside and outside class instances. The outside score map captures important context information.

We have seen that recent approaches for object detection and instance segmentation apply similar methods for region proposals and classification. The difference is then whether the region is refined to a bounding box for object detection, or if an instance mask is created for instance segmentation. In order to optimize efficiency, sub-networks with different purposes such as region proposals, classification and segmentation, are merged by sharing features. This has also resulted

in moving away from separate region-proposal methods, and towards fully convolutional sliding window approaches. FCNs have enabled sharing of full-image features but make detecting separate instances harder due to their translation-invariance. In order to overcome this issue, position-sensitive score-maps can be applied. Challenges that remain primarily relate to accuracy of the methods. The accuracy can be improved by creating deeper networks that capture more complex features. However, deeper networks are harder to train, simply because they have more parameters to optimize. A challenge is therefore to find new ways to achieve similar performance with fewer parameters. Invariance to intra-class variations and deformations can be achieved by providing appropriate training data. It is however, still difficult to achieve such invariance without artificially modifying the data. Finally, image-based methods can be made more reliable by incorporating more information, such as depth, as we will see in the next chapter.

2.3 Image-based methods with depth information

While images do not directly store how pixels relate to three-dimensional structure, data representations such as RGBD images, meshes and point clouds do contain some notion of three-dimensionality. We therefore call them 3D representations. Features that can be learned from such representations include geometric edges and curvature. The localization of objects in 3D representations can be inspired by object localization techniques for images as seen in Section 2.2. In this section we discuss how image-based techniques can be applied to 3D representations.

Perhaps the most naive approach to classify a mesh is to render it to a **single image**, and apply image-based methods on that. Su et al. have shown that such an approach can even outperform neural networks that operate on meshes directly. They rendered a mesh to an image, and processed it using a VGG-M network [18]. The reason this simplistic approach works so well is that many features, such as the texture of the material for example, are constant over all sides of an object.

Gupta et al. modify the RCNN as discussed in Section 2.2 to support RGBD images, which store a depth value for each pixel. The network is called **RCNN-depth** [19]. The depth in the RGBD images is computed using two images of the same object, with each image created from a slightly different position, similar to how our eyes are positioned apart. The depth information is used to detect contours, generate better region proposals and boost performance in both object detection and segmentation. The depth information is encoded into regular RGB images using a special feature encoding that emphasizes complementary discontinuities in the image. The channels in this encoding represent the height above ground, angle between the normal of the surface and the direction of gravity, and the horizontal disparity. The disparity refers to the distance between two points in the two images that depict the same three-dimensional point. This encoding helps the network to learn more complex features, without having to learn basic features first, which can be difficult when little training data is available, increasing performance. Furthermore, it was shown that the encoded depth images have similar structural characteristics as regular RGB images. As an example, edges in the encoded depth image correspond to edges in the original image. Networks trained on regular images can therefore be used as an initialization of networks for depth images.

By including more views, Su et al. further increase the accuracy of their CNN for object classification [20]. The resulting network is called a **Multi-View CNN (MVCNN)**. MVCNNs typically involve two steps. First the 3D object is projected to multiple views that capture different sides of the object. To create these images, the object is rendered from different viewpoints, with virtual cameras positioned in a ring or a sphere around the object. During the second step, the views are classified using well-established image-based CNNs. The final class is the one with the most votes across all images. The network can be further optimized and aggregated by combining the images into one image using a max-operation. A final CNN then performs classification on this combined image. A similar idea was discussed earlier by LeCun et al. A small network is trained on stereo-vision images, which are images taken from two viewpoints a small distance apart. In contrast to the virtual views used in MVCNN, the stereo-vision images are recorded in the real

world. Since the information from different viewpoints is considered, the network becomes invariant to pose [21].

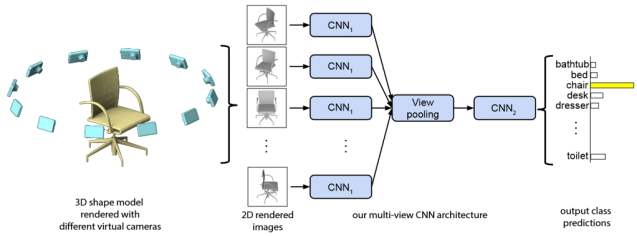


FIGURE 2.12: The Multi-View CNN architecture. A multi-view representation is created by rendering the scene from different viewpoints. Features are extracted from each view independently, combined and then jointly used for classification.

Instead of classifying each view independently, Johns et al. propose a **pairwise** approach [22]. First, every possible pair of views in the set is generated. Each pair, combined with the relative camera change between the views is then classified independently. The final class probability distribution is finally computed as the weighted average over all pairs. This way the transformations between viewpoints are captured, and the network becomes more invariant to viewpoint differences. If such data is available the method can be extended with depth images for additional discriminative features. This pairwise method outperforms MVCNN.

While it is intuitive to create images from multiple viewpoints, the entire scene can be captured in one image. Shi et al. introduce **DeepPano**, which is a CNN trained directly on panoramic images of 3D models [23]. The panoramic projection is illustrated in Figure 2.13. However, when an object is rotated, this results in a shift in the panoramic image. To make the network invariant to these changes, a row-wise max pooling layer which takes the maximum value of each row in the convolutional feature maps, is introduced. The panoramic mapping is parallel to one of the principle axes of the object, and orthogonal to the other two. This disables the projection to capture information from some sides of the object, resulting in a representation that is less information rich than multi-view representations.

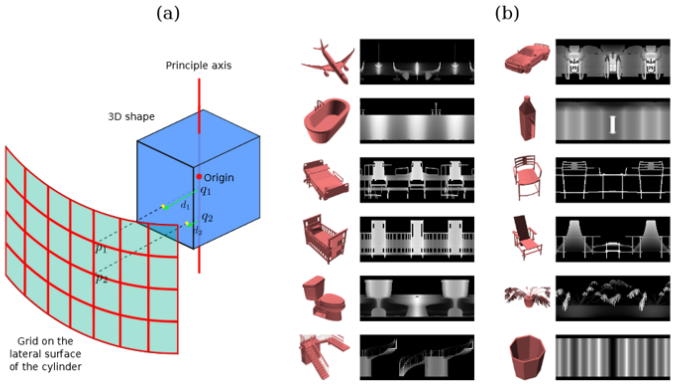
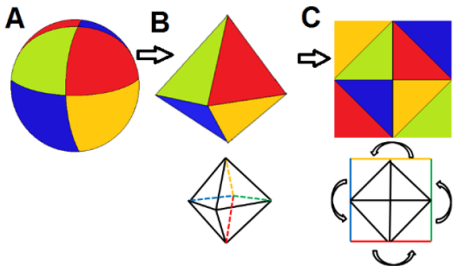


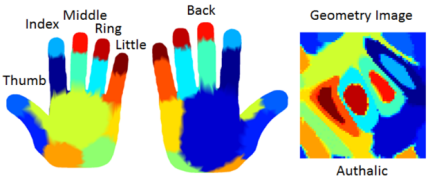
FIGURE 2.13: The DeepPano architecture uses a mapping that creates a panoramic view, which is a cylindrical projection around the object's principle axis.

Sinha et al. propose another way to map a 3D shape to an image [24]. The goal is to generate a single image which provides a highly descriptive representation of all sides of the 3D shape. This can be done by encoding the surface properties of the object in the image. The shape is spherically parameterized, mapped to an octahedron and then cut along the edges to unfold it as a square and produce a so-called **geometry image**. The position of a pixel in a geometry image then defines how the point is related to the three-dimensional surface of the object. This representation

captures enough information to reconstruct the original 3D model from the image with relatively high accuracy. Since geometry images encode surface properties, they are more invariant to inter-class variations and deformations than other methods.



(A) The process of mapping a 3D model to a geometry image.



(B) An example 3D model and the corresponding geometry image

We have seen that intuitive approaches exist that enable the use of image-based methods on 3D data, with good performance. Adding depth information to images improves performance, while only requiring little changes to the network. While using a single image already works well, using multiple views increases performance and even outperforms smart mappings such as panoramic or geometry images.

2.4 Volumetric methods

While images capture only the visible parts of the surfaces in the scene, meshes and point clouds represent the three-dimensional scene in its entirety. Before a neural network can be applied on meshes or point clouds, however, the data has to be mapped to a different representation on which the neural network can efficiently operate. In this chapter methods based on the volumetric properties of meshes and point clouds are discussed.

A commonly used representation, which captures the three-dimensional information of the entire scene, is the voxel grid. The space is first partitioned with a regular grid of cubes. Each grid cell is called a voxel. Voxels are usually binary, storing a 1 when they are occupied, a 0 otherwise. To create a voxel representation of a 3D model every voxel that intersects the model surface is considered occupied. In a similar manner a voxel can be created from a point cloud. Every voxel that contains (a thresholded value of) samples of the point cloud, is considered occupied. Alternatively the point cloud can first be reconstructed as mesh. The number of voxels used in the grid is also called the resolution. For example a grid of $30 \times 30 \times 30$ voxels is said to have a resolution of 30. Voxel representations do have some inconvenient properties. For example, a 3D voxel volume with small resolution, has the same memory requirements as an image with relatively high resolution.

Wu et al. train a voxel-based CNNs, called **3D ShapeNets**, for object classification, using voxel representations with a resolution of 30 [25]. After training the network using voxel representations of full 3D models, it is able to reconstruct missing data in voxel representations of RGBD images. First each voxel is marked as free, surface or occluded. The voxels that correspond to pixels in the original RGBD image are observed to be either free or surface segments, while occluded voxels are considered missing data. Object classification can be performed using only the observed data. When reconstructing a voxel representation from an RGBD image, the missing data is randomly initialized to see how the class distributions change and the voxels corresponding to the highest scoring label are clamped. A number of such iterations is performed on a number of instances in parallel. The most frequently sampled class then determines the final classification label. Although the reconstruction abilities of the method are convenient, it does not reach state of the art performance in classification. This performance difference could be explained by the small resolution of the volumetric representation.

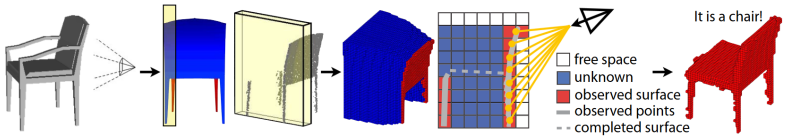


FIGURE 2.15: The process of reconstructing missing data in voxel models created from RGBD images. One slice is used for visualization. The unkown voxels are iteratively clamped to maximize the score of the most likely class.

Maturana and Schere propose another 3D CNN, called **VoxNet** [26]. VoxNet is similar to ShapeNet in its data representation but uses a different network which contains fewer parameters. VoxNet performs slightly better than ShapeNet, which can be explained by the slightly higher voxel resolution of 32. Another factor in the performance difference is that VoxNet suffers less from overfitting due to fewer parameters. Additional experiments show that rotation invariance can be improved by adding rotated variations of meshes to the training data. The alteration of data to improve invariance is generally called **data augmentation**.

Different representations capture different features. The representations can therefore be used complementary to each other to obtain more discriminative features. Hegde and Zadeh concluded that most methods use either voxels or multi-views, although a combination of both would probably perform best. Therefore, they propose **FusionNet**, which fuses three networks, which rely on different representations [27]. The first network is an MVCNN as discussed in Section 2.3, which describes strong features through its image-based representation. The second network is a regular volumetric CNN which captures spatially local features through its voxel-based representation. The final network is another volumetric CNN which contains a few alterations inspired by **GoogLeNet** [28]. The **inception-module** originally occurred in GoogLeNet and introduced a new level of organization. A network consist of many different types of layers. However, it is not obvious in which cases 1×1 convolution, 3×3 convolution, 5×5 convolution or pooling operations are preferred. Instead of choosing the layers manually, they could all be computed in parallel. The results can then be summed and let the network can learn their respective relevance. Regular convolution layers could then be combined with 1×1 convolutions, that reduce feature depth to decrease memory consumption, while keeping height and width the same. This allows for very deep networks that capture more abstract information and encode higher representational power without much additional computation cost. Experiments show that while each network performs reasonably well individually, the combination of all three does indeed make a stronger classifier.

Qi. et al. argue that volumetric methods should capture at least as much information as an MVCNN does [29]. To illustrate how the low resolution of a voxel model would influence the performance of a multi-view method, the voxel model is rendered using multiple views and processed by a regular multi-view CNN. The voxels are rendered as spheres because those are view-invariant. This comparison is further illustrated in Figure 2.16. The results indicate that even with the lower resolution, the multi-view method performs better than the volumetric approach, although worse than when normal rendering is done. From this, it is concluded that the input resolution amounts to a small performance difference between volumetric and multi-view methods. The majority of the performance difference is explained by differences in network architecture. The performance differences are illustrated in Figure 2.17.

Qi et al. apply an approach similar to that of the MVCNN, where the 3D model is first projected to a 2D representation, after which the image representation is classified. The voxel-based representation is first mapped to an image by a CNN which they call the Anisotropic probing module. The idea is that the anisotropic probing module learns a rendering function that captures the global structure of the 3D volume, similar to an X-ray. A 2D CNN then learns to classify these representations. By extracting 3D features from multiple orientations the network becomes orientation invariant. The resulting network is like a voxel-based adaptation of the originally image-based MVCNN and is called a **Multi-Orientation Volumetric CNN (MOVCNN)**. This way Qi et al. successfully close the gap in performance between volumetric and 2D CNNs when using a resolution of 30. However, further increasing the resolution quickly results in intractable memory

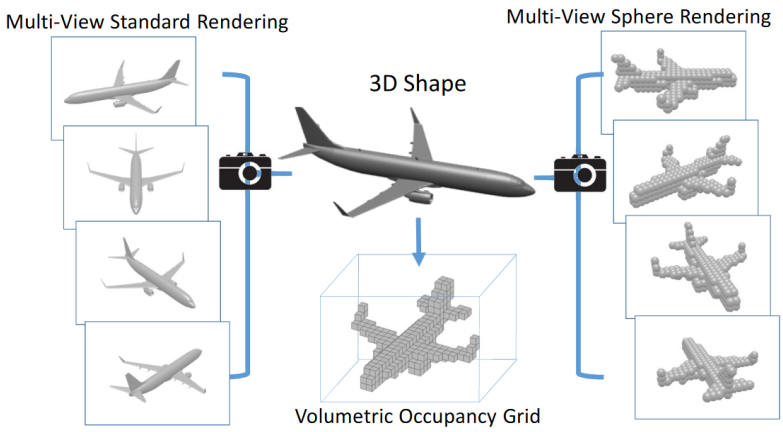


FIGURE 2.16: Voxel models are rendered to multiple views using sphere-rendering. The performance difference between such images and images using a standard rendering, explains how much of the performance difference between volumetric and multi-view methods is accountable to input resolution.

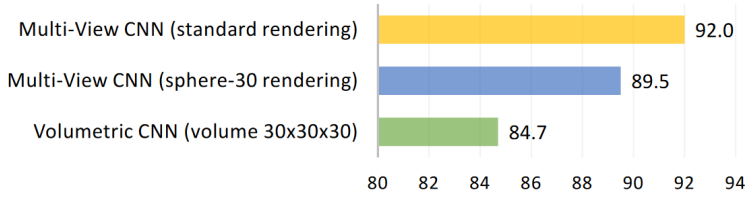


FIGURE 2.17: Performance differences between methods using different data representations. The performance difference between MVCNNs using standard rendering and sphere rendering explains how much of the performance difference between multi-view and volumetric methods is accountable to input resolution. The performance difference between the MVCNN using sphere rendering and the volumetric CNN explains how much of the performance difference between multi-view and volumetric methods is accountable to network architecture.

requirements. Therefore, new techniques are required that allow for efficient scaling of volumetric methods.

Recently Riegler et al. propose **OctNet** as a network that allows for higher resolution volumetric data [30]. The space is partitioned with a hybrid grid-octree data structure, which focuses computation and memory allocation on the relevant regions. The data structure can be stored efficiently using a compressed bit-string representation. It is also shown how 3D convolution and pooling operations can be implemented efficiently on this data structure. OctNet has been tested with a resolution up to 256. To analyze the influence of increased resolution on performance, the number of parameters in the network is kept constant. Therefore some convolution layers are removed when testing higher resolutions. Results indicate that for classification there are diminishing returns beyond a voxel resolution of 32. For orientation estimation the performance increases are negligible for resolutions higher than 64. However, if the network is extended with more convolution layers, introducing more parameters, the increased resolution does improve orientation estimation performance. Lower resolutions seem sufficient for simple classification tasks. The reason for this is that the classes that could not be classified correctly with a lower resolution, remain ambiguous with higher resolutions. An example of such an ambiguous case is the distinction between a dresser and a night stand since both objects have very similar shape. he primary benefit of using OctNet is that it requires significantly less memory than other methods. However, Riegler et al. did not test how the performance of their network would compare to others, when increasing parameters and resolution while keeping memory requirements similar.

Brock et al. combine ideas from multiple successful and influential networks into one single

architecture for volumetric data called a **VoxCeption-ResNet (VRN)** [31]. Their **voxception-module** is a 3D adaptation of the inception-module from GoogLeNet, as previously discussed. The voxception-module is then altered to incorporate concepts first proposed in **ResNet** [32]. Deep neural networks suffer from the **vanishing gradient problem**, where the gradient decreases exponentially with the depth of a network, resulting in slow weight optimization and therefore limiting network depth. ResNet effectively solves the vanishing gradient problem by introducing "shortcut connections" that skip one or more layers, and directly add their result to the results of a higher level layer. When the network must apply an identity function, it is easier to use a shortcut connection than to approximate the identity function through multiple non-linear layers. Such connections do not add any parameters to the network and are therefore a cheap and simple improvement. Through these residual connections, deep networks become easier to optimize. **InceptionNetv4** by Szegedy et al. [33] added the shortcut connections from ResNets as an extra path in an inception module. Brock et al. then applied that idea to voxel data (with a resolution of 32), resulting in their VRN architecture, which outperforms all other volumetric approaches. The success of the VRN relies on the increased expressive power resulting from its significantly increased depth. A VRN has 45 layers, while ShapeNet has only 5. The best results are obtained through a simple ensemble by summing predictions from five VRNs and one Voxception network consisting of four Voxception modules and three Voxception-Downsample modules.

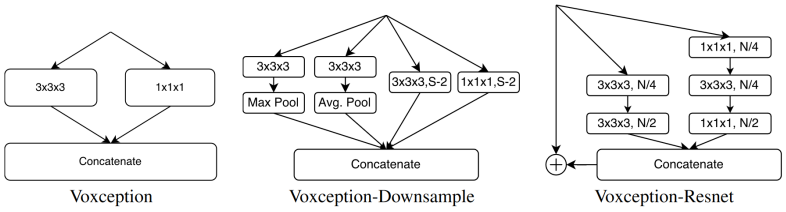


FIGURE 2.18: VRN modules

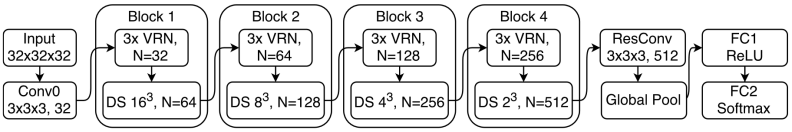


FIGURE 2.19: The VRN architecture. VRN refers to the VoxCeption-Resnet module and DS refers to the Voxception-Downsample module as depicted in Figure 2.18

We have discussed the major volumetric methods. To summarize, volumetric representations can easily be processed using the same concepts as used for images. However, volumetric methods have not been able to perform as well as image-based methods for a long time. The differences in performance are caused by low input resolution and differences in architecture. Recent methods support higher resolutions through efficient space partitioning and memory models. The increased resolution helps for orientation estimation, but does not significantly improve performance for classification. The architecture can be improved by taking more orientations into account, making the model more invariant to rotation. Other methods improve performance by combining the discriminative power of volumetric features with that of image features. Although resolution and architecture are significant factors in performance, the most powerful volumetric network contains little architectural innovation, and instead focuses on network depth. Even though the deeper network does not support higher resolutions or complex representation mappings, it enjoys higher expressive power, outperforming all other volumetric methods. To our knowledge, no volumetric methods have been combined with color information yet.

2.5 Intrinsic methods

Instead of encoding the occupied volume, a three-dimensional data representation can also encode surface properties. The latter approach has some significant advantages over the former, as

will be discussed in this section.

The previous sections discuss methods that work on 3D geometry by mapping it to a representation that can be easily processed by neural networks. Approaches as seen in Section 2.3 which map 3D data to a single image, or multiple images from different views, are primarily used because the success of image-based CNNs has already been illustrated. Volumetric approaches as seen in Section 2.4 rely on voxels because they allow for efficient convolutions that are similar to those in convolutional networks working on images. These convolutions are highly efficient as they make use of the locality in the data; pixels or voxels that are close to each other are correlated. However, these approaches consider geometric data as extrinsic Euclidean structures, which are not invariant to deformations, unless complex network architectures and large datasets are used. In contrast, intrinsic approaches try to find representations in non-Euclidean domains, which are invariant to deformations. This difference is illustrated in Figure 2.20. Note that intra-class variations can also be considered small deformations.

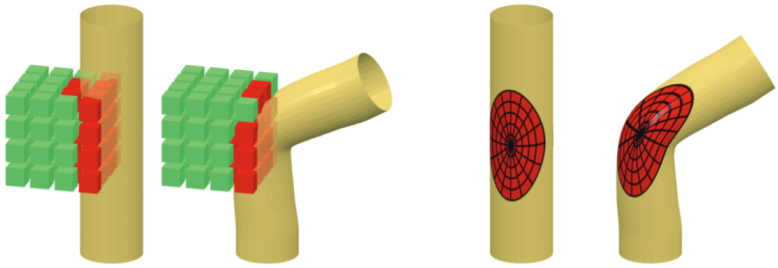


FIGURE 2.20: Differences between extrinsic and intrinsic representations. Left: Extrinsic methods rely on Euclidean representations, such as volumetric occupancy, which are not invariant to deformations. Right: Intrinsic methods use non-Euclidean representations, encoding data of the surface, which are invariant to deformations. Image adapted from Boscaini et al. [34]

In the same way that images can be interpreted as signals, meshes can be considered signals. It is however not obvious how convolution can be performed on a mesh. The convolution theorem states that the convolution of signals in the primal domain is identical to point-wise multiplication in the spectral domain. The spectral domain is a higher-dimensional equivalent to the Fourier domain, also called the frequency domain. Convolution via the spectral domain is usually mathematically simpler and computationally cheaper than in the primal domain. **Spectral mesh processing** is a process in which meshes are transformed to the spectral domain for processing using appropriately defined operators. A mesh is first mapped to a matrix representation that captures the pairwise relations between mesh elements, such as vertices that are connected by edges. This encoding of pairwise relations is similar to a graph interpretation of the mesh. An example of such a representation is the graph Laplacian as depicted in Figure 2.21. Similarly, images and arrays of numbers can be interpreted as graphs. The graph interpretation is related to shape, since graph connections only exist between samples that are close in Euclidean space. Consecutively the eigenvectors and eigenvalues of the matrix representation are computed to construct a basis in the spectral domain. When meshes have a different number of vertices, a fixed number of eigenvectors corresponding to the lowest frequencies are kept, similar to a low-pass filter. The spectral transform is the process of transforming the original signal to this new basis. This transformation is a generalization of the Fourier-transformation to non-Euclidean domains. The basis then defines a feature space which can be used in a problem-specific manner. For an extensive introduction to spectral mesh processing we refer the reader to the paper by Lévy et al. [35].

Bruna et al. use the graph Laplacian, and define operators that capture important properties such as smoothness. The Fourier transformation of the graph Laplacian also exhibits highly correlated local statistics and can thus be processed by generalizing convolution operators. This results in a new type of CNN on graphs, called **Spectral Networks** [36].

$$D = \begin{pmatrix} 2 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad A = \begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} \quad L = \begin{pmatrix} 2 & -1 & -1 & 0 \\ -1 & 3 & -1 & -1 \\ -1 & -1 & 2 & 0 \\ 0 & -1 & 0 & 1 \end{pmatrix}$$

FIGURE 2.21: The combinatorial graph Laplacian can be computed as $L = D - A$ where D and A are the Degree and Adjacency matrices respectively.

The spectral networks as proposed by Bruna et al. make use of prior knowledge about the graph structure. It is known, in advantage, which samples are connected to each other. When the exact structure of the graph is unknown, these connections could be estimated. This is similar to reconstructing a mesh from a point cloud. Henaff et al. extend spectral Networks with a **Graph Estimation** procedure [37]. Instead of applying measures that have been manually designed by humans, like Euclidean distance, Z-score or square-correlation, a neural network could be trained to optimize for this particular task. First a fully connected network is trained for a classification task. Then the first-layer features are extracted. The similarity between two samples is then computed as the Euclidean distance between their first-layer features. When the graph structure is not known a priori, the neural network for graph estimation performs significantly better than the manually designed measures. However, the architecture remains sensitive to graph estimation errors. When a graph is estimated incorrectly this is equivalent to introducing a deformation to the model. Such an error can result in misclassification, mistaking i.e. a bed for a couch. The network has been applied on regression problems and image classification.

Ravanbaksh et al. define a network with **set-invariant layers** that directly takes a point cloud as input [38]. In contrast to the graph interpretations which encode pairwise relations, this methods operates on the point cloud as a set. Structural composition, pooling and set convolution are defined using associative and commutative operators based on set-theory. Each neuron then responds to particular 3D constellations of points. The success of the approach is illustrated for classification of images and point clouds, and the prediction of galaxy distances as a regression problem. The method performance is similar to that of a MVCNN.

Yi et al. propose a **Synchronized Spectral CNN (SyncSpecCNN)** [39], which is similar to an FCN, but performs graph convolutions via the spectral domain. Information sharing for different types of objects is difficult since their graph Laplacians are different. Therefore the eigenbases are aligned in the spectral domain by a **Spectral Transformer Network (SpecTN)**. Alignment is done efficiently by focusing only on the low-frequency part of the domain. Experiments validate that the SpecTN does improve performance and makes the network more invariant to input resolution.

Qi et al. propose **PointNet** [40], a neural network which takes point clouds directly as input, and can output class labels either for the entire point cloud or for each point individually. Similarly to Ravanbaksh et al. PointNet interprets a point cloud first as an unordered set defined in a Euclidean space. First a function that is invariant to the order of its arguments is learned by a multi-layer perceptron network to construct a global signature of the input point cloud. The global features are then concatenated with the per point information, in order to compute stronger per-point features such as accurate normal information or labels. Small transformation networks learn transformation matrices that align the input point clouds and align the feature point clouds, in order to make them invariant to transformations. The features learned by the network are similar to some conventional hand-crafted features and keypoints that capture local neighborhood. Further experiments indicate that the network is significantly more robust to missing data than VoxNet. This is an important property when processing point clouds as they only provide partial and sparse data.

ShapeNet (not to be confused with the volumetric 3D ShapeNets) was proposed by Masci et al. [41] Similar to PointNet, the network learns feature descriptors that capture local curvature. The network uses sets of geodesics (shortest paths along the curved space), cast in different directions using local polar coordinates, to define surface patches used for convolution. This idea was first

presented by Kokkinos et al. [42] and is further illustrated in Figure 2.22. Each patch is then convolved with geodesic patch filters, which can be represented using sparse matrices that scale well to larger meshes. The output of the network is a feature vector which could be used in a task-specific manner.

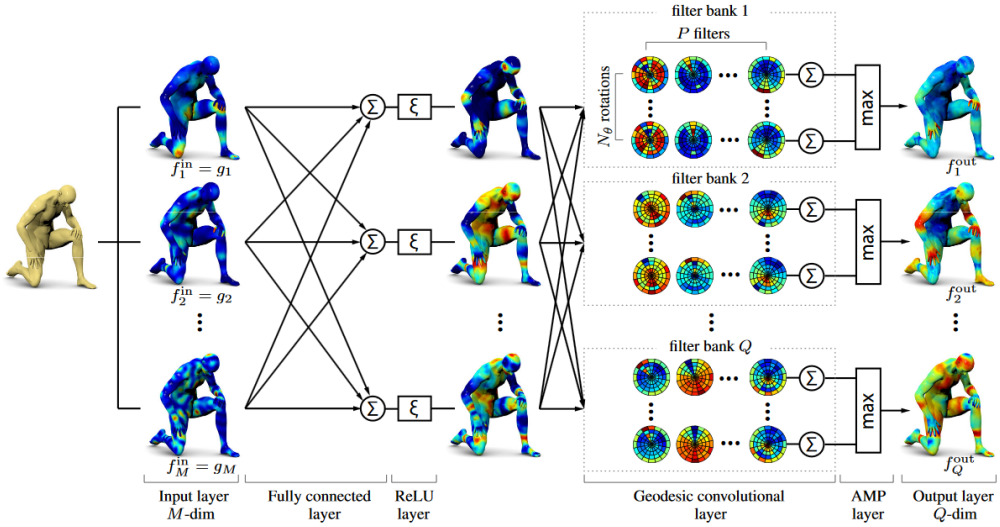


FIGURE 2.22: The ShapeNet architecture. Convolution is applied on local geodesic patches. The convolution filters are applied in all possible rotations. Angular max pooling (AMP) is applied to obtain rotational invariance. The colors on the figures represent the response to specific features.

Intrinsic methods show promising results due to their invariance to isometric transformations. Neural networks using spectral mesh processing can be made efficient by performing convolutions via the spectral domain. However, these methods have been primarily compared to each other, and have not been extensively compared to image based or volumetric methods on 3D shapes. Additionally, intrinsic methods are hard to implement due to their mathematical complexity. An interesting property of intrinsic methods is that they generalize to many different types of data; arrays, images, models and point clouds can all be interpreted as graphs or sets.

Chapter 3

Methodology

This chapter introduces the novel pipeline which combines information from images and point clouds to localize objects. First, Section 3.1 discusses the considerations that led to the proposed architecture. Section 3.2 discusses the single-view pipeline, where a segmentation network processes images individually. Section 3.4 discusses the multi-view pipeline, which utilizes a second neural network that aggregates multiple segmentations from different recording locations to improve segmentation accuracy.

3.1 Motivation and high-level considerations

In Chapter 2 multiple neural network architectures have been discussed that incorporate depth information in different ways. However, not all of these methods perform localization or incorporate color information. The goal of this work is to combine the color information from panoramic images with the depth information from Lidar point clouds. This section discusses how the properties of the discussed methods are weighed against each other and against our requirements.

While intrinsic methods offer high mathematical flexibility, they are also mathematically complex, and their performance has not yet been illustrated on problems similar to ours. The flexibility of intrinsic methods with regard to the type of input data is not interesting for our application, as we only require support for a single representation.

Volumetric methods offer an intuitive approach to convert 3D models into a structure that can easily be convolved. However volumetric approaches have only been illustrated on single objects and are difficult to adapt to outdoor scenes due to their high memory requirements and low resolution. These properties are especially wasteful considering that outdoor scenes typically contain more empty space in some areas, while being more cluttered in others. Voxel-based methods have also not been used together with color. Assigning a color to each voxel would increase memory requirements further, and would again suffer from the low resolution.

Some images-based methods incorporate depth information through additional image channels. The resulting images can be processed by neural networks similar to any other type of image. Depth can then be encoded in multiple ways, and even information from multiple sides of an object can be captured in a single image. Similarly, multi-view CNNs capture depth information from different angles of a mesh or a point cloud. While a virtual camera could be positioned anywhere within the reconstructed meshes, the quality of both the point clouds and the cycloramas is highest at the original recording locations. While the recording locations might not be as densely distributed as is commonly simulated in a Multi-view CNN, the recording interval of 5 meters is high enough to ensure high correlation between images, which means that the same objects are visible from similar sides, resulting in an overlap of information. We propose to use the depth cycloramas in a neural network similar to a Multi-view CNN. Since MVCNNs generally only deal with fixed camera poses, we will introduce a novel method that supports varying camera poses. With this approach we can rely on established methods such as RCNN-depth and MVCNN as stepping stones, while making optimal use of the properties that are specific to our data.

3.2 The single-view pipeline

We first describe a single-view pipeline that performs image-segmentation, than reconstructs the segmentations as labeled point clouds, and applies three-dimensional point cloud clustering to extract the world position for each detected object. The architecture of our single-view pipeline is depicted in Figure 3.1. An example of the pipeline output is illustrated in Figure 3.2.

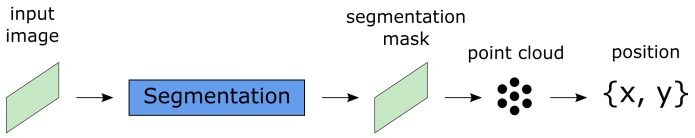


FIGURE 3.1: The single-view pipeline. An image is segmented by the segmentation-network. Since we know for each pixel the corresponding depth value, the resulting segmentation can be reconstructed as a point cloud, with a class label for each point. A clustering algorithm is then applied to extract the clusters of points belonging to the light pole class. The centroids of all clusters are computed to predict the positions of the light poles.

First a neural network is considered that only operates on individual RGBD images. A segmentation performs pixel-wise classification and computed per pixel a probability for each class. The final segmentation is created by picking for each pixel the class with the highest probability. After the neural network has predicted an object of interest within the image, the next step is to estimate a location in world coordinates. Many object detection methods, like the RCNN family [11] [12] [13], and the RFCN [16], predict a set of bounding boxes within each view. With the available depth information, these bounding boxes can then be projected into world-space and possibly combined. However, it is not obvious how this projection to world-space should be implemented. The depth of the bounding box can be computed as the average of all depth values of the pixels within the bounding box, or alternatively, the depth value of the center pixel of the bounding box could be chosen. Since light poles are thin objects with a protruding head on the top, it is common that many pixels within the bounding box are not relevant and that the center of the bounding box does not even coincide with the light pole itself. To minimize the number of irrelevant pixels we opt for a **segmentation network** that computes a per-pixel classification, instead of a detection network. The design of the pipeline does not rely on any specific segmentation network implementation, and thus any segmentation network can be used.

In a segmented image, each pixel is assigned a class label. Until this point no distinction has been made between different instances of the same class. As a consequence, instances still have to be separated in an additional phase. Instance separation can also be achieved with an instance-level segmentation network. However, these networks are often complex, while regular point cloud clustering algorithms are relatively simple. We propose to make optimal use of the available depth information by first recreating a point cloud from the segmentation and then applying three-dimensional point cloud clustering. To create a point cloud, each pixel that is classified as "light pole" is reconstructed as three-dimensional point. As illustrated in Figure 1.3 each pixel in an image can be associated with a direction vector if the camera parameters are known. By setting the length of the direction vector equal to the corresponding depth value, the original three-dimensional point is effectively reconstructed. Note that if a pixel has no depth-value, or if the depth value is zero, as is the case for pixels corresponding to the sky, a point can not be reconstructed. This means that pixels belonging to the sky that are incorrectly labeled as "light pole" are automatically filtered when a segmentation is reconstructed as a point cloud.

A clustering algorithm is applied to determine the clusters in the point cloud and distinguish between object instances [43]. For every point in the point cloud the following steps are performed: The point is first added to an empty queue. For each unprocessed point in the queue the set of

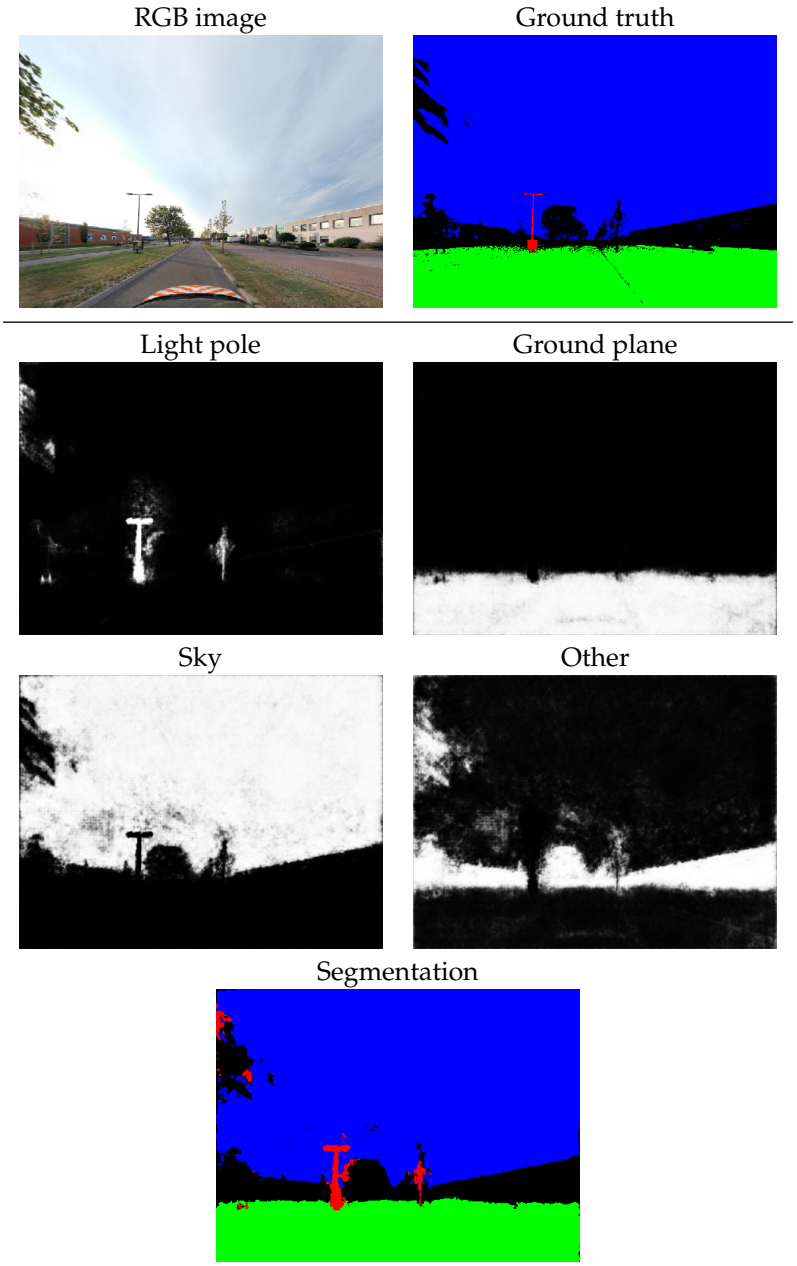


FIGURE 3.2: An example of the output of our segmentation-network. The RGB image is provided as input. The corresponding ground truth segmentation is considered the perfect output for this image. The network for each pixel a probability per class. The probabilities for all pixels are illustrated per class. The final segmentation is constructed by picking, for each pixel, the class with the highest probability.

point neighbors within a sphere with radius r is computed. Each neighbor that has not been processed already is added to the current queue. When all points in the queue have been processed, the queue contains all points that form a cluster together. This process terminates when all points in the point cloud have been processed and have thus been assigned to a cluster. The centroids of these clusters are the potential locations of our objects of interest.

As we are only interested in the coordinates of the objects within the horizontal plane, the z-coordinate of the centroids is ignored. The three-dimensional clustering is expected to work better than image-space clustering since incorrectly classified pixels will be located further away from the actual object of interest, and correctly classified pixels will be more densely packed if the overall classification is correct. The risk with instance-separation methods like clustering is that if the

instances are not separated correctly, the computed positions will be affected. Another risk is that the centroid of the cluster might not coincide with the actual object if the object is concave. The larger the concave protrusions are, the further the computed position might be off. While taking these risks into account, the clustering method can be generalized to any type of object as long as instances consist of a single geometrical component within the point cloud.

3.3 Alternative depth-derived features

In the previous sections, depth cycloramas were used to provide additional information to the segmentation network. The choice for depth information is obvious and intuitive as the output from the Lidar laser sensor consist of depth values. However, the representation of the data affects the performance of the neural network. If the data is not provided in an optimal representation, the network might first have to learn to transform the data into something useful, or might be unable to use the data at all. The expectation is that depth information is useful to increase performance, but that other derivative representations might perform even better.

Depth information on its own does not provide a network with directly useful information. Only by combining depth information of many pixels, the relative positioning and orientation of objects can be determined. An algorithmic approach to compute these features explicitly is well-defined. Therefore, it is undesirable to let a neural network estimate these features, as it might be unprecise. We therefore compute these features explicitly and pass them to the neural network as additional information. The benefit of this is that the neural network no longer has to learn these specific features, and can use the spare weights for estimation of other features.

We propose to directly incorporate orientation information derived from depth information in the images. The orientation of a surface can be defined by the normal vector, which is the unit vector perpendicular to the surface. While normals are not directly available, they can be estimated within the point clouds. The problem of determining the normal to a point on a surface can be approximated by estimating the normal to a plane tangent to the surface [43]. To fit a plane to the point on a surface we therefore first find all neighbors within a radius, construct a covariance matrix from those, and apply Principle Component Analysis to see how a plane would best fit the set. The sign of the normal can, however, not be solved since PCA can not mathematically determine the front or back of the plane. This can result in inconsistent normal orientation throughout a point cloud. However, since we know the position of the camera, we can orient all normals towards the camera in a consistent manner.

The normal vector can be defined by its x, y and z components. However, these values are not directly meaningful. We therefore propose the encoding of normal information using azimuth and elevation, which are commonly used to define the position of an object in the sky, relative to an observation point. The relation between a normal, azimuth and elevation is illustrated in Figure 3.3. We adapt the terminology to define the orientation of a normal vector, relative to the camera orientation. The **azimuth** is commonly defined as an angle in the horizontal plane, in a range of $-\pi$ to π, relative to the observer's forward facing direction. However, since all normals in our scene point toward the camera, the azimuth values would then commonly occur around the boundary between $-\pi$ and π, which might confuse the neural network. To ensure that azimuth values change continuously we define them relative to the backward facing direction of the camera. The **elevation** then defines the angle between the normal and the horizon where $\frac{1}{2}\pi$ corresponds to straight up, and $-\frac{1}{2}\pi$ corresponds to straight down. The resulting azimuth and elevation values can be stored per pixel in a 2-channel image. We abbreviate this feature encoding as **AzEl-encoding**. Examples of AzEl-encoded images are illustrated in Figure 3.4.

As discussed in Section 2.3, Gupta et al. defined a feature for RCNN-depth consisting of height above ground, angle with the gravity and horizontal disparity. We propose a similar feature that replaces horizontal disparity with the distance along the camera direction. We argue that objects in the real world are most commonly oriented vertically. This means that objects commonly have a top and bottom, but can be orientated at different yaws. Since azimuth values are different when

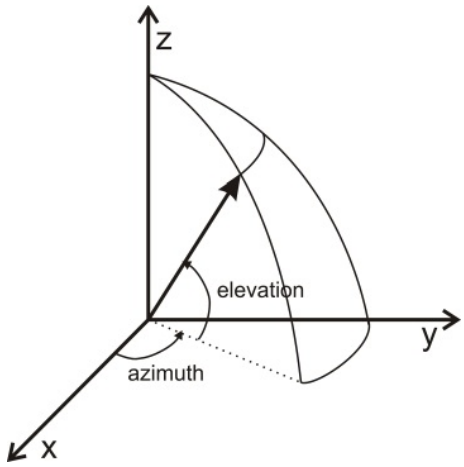


FIGURE 3.3: A direction vector which is usually defined through its x, y and z components can alternatively be defined using two values, for azimuth and elevation. Azimuth is defined as the angle in the horizontal plane relative to a fixed direction. Elevation is defined as the angle between the direction and the horizon.

an object is oriented differently, we expect that azimuth is not very useful to distinguish between classes. However, some objects such as buildings are commonly observed from the side, while others such as roads and pavements are commonly observed from the top. It is therefore expected that elevation is useful to distinguish between classes. To define a value similar to elevation independent from azimuth we compute the angle between the normal and the gravity-direction. Since the camera might be positioned on a slope, the gravity direction within the local coordinate space of the camera might not always correspond to straight down. In addition to this orientation information, positioning information can be provided in a more useful way than depth alone. Because the depth values are defined as the distance to the camera, flat objects have non-linearly changing depth values. To make it easier for the network to distinguish straight objects, we measure the distance of a point relative to the imaging plane, parallel to the camera direction, so that depth features change linearly over flat surfaces. Finally, some objects have different features at different heights; e.g. a leafy tree top in comparison with its trunk. In addition, some objects are more common at specific heights; e.g. bicycles commonly occur on the road, but are uncommon in tree tops. While height can be inferred from a pixel's location and the corresponding depth value, we propose to include height information in the feature encoding. In short the feature encoding consists of distance, height and angle with the gravity, encoded in a 3-channel image. We abbreviate this feature encoding as **HDA-encoding**. Examples of HDA-encoded images are illustrated in Figure 3.4.

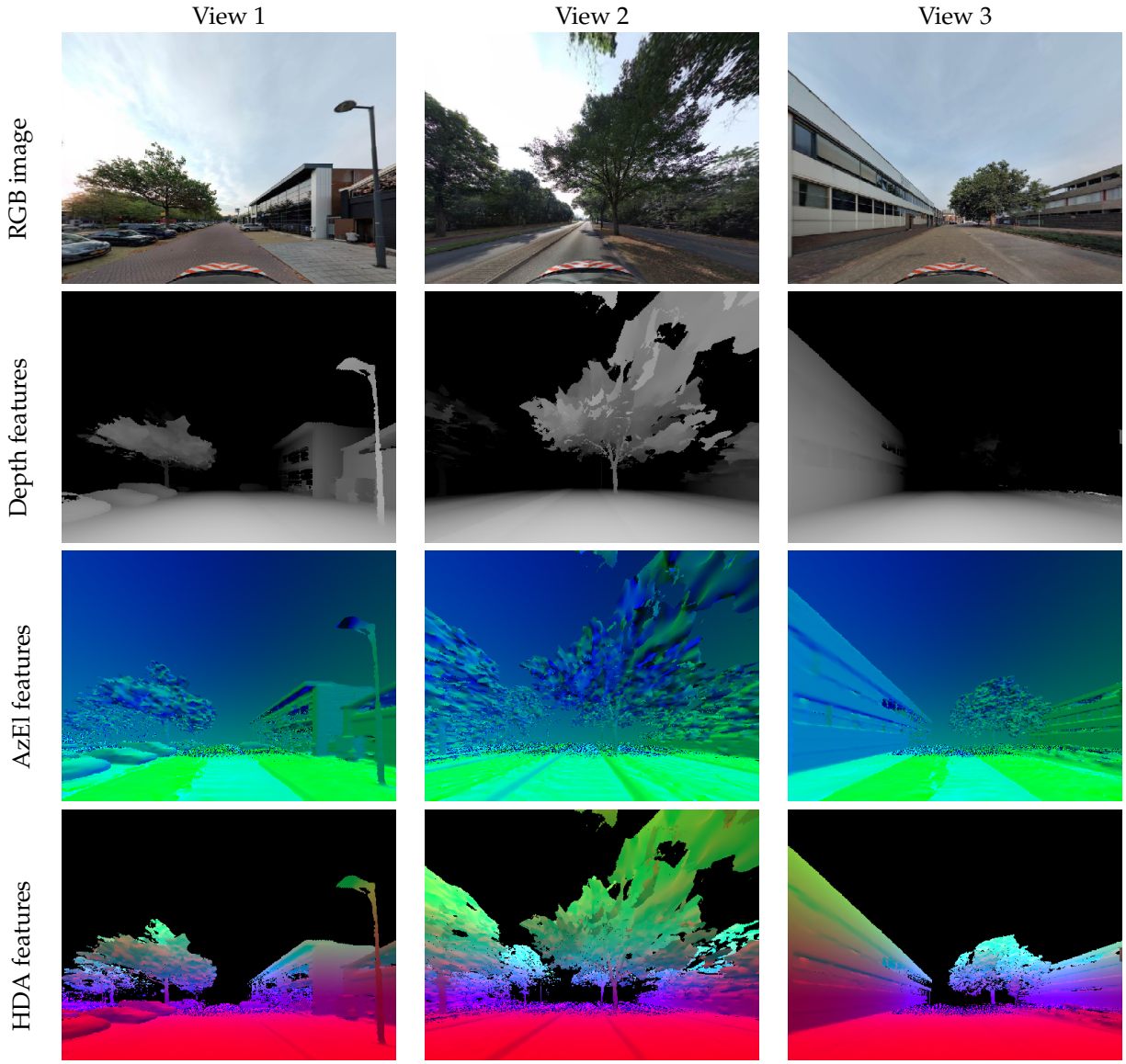


FIGURE 3.4: The original RGB images together with different depth features. The second row illustrates single-channel depth information. The second tow illustrates depth information encoded as 2-channel AzEl features. The green and blue color channels correspond to azimuth and elevation respectively. The third row illustrates depth information encoded as 3-channel HDA features. The green, blue and red color channels correspond to height, distance and angle with the gravity respectively.

3.4 The multi-view pipeline

The pipeline as described in Section 3.2 invokes a segmentation network for each view independently. To make the network more invariant to the inaccuracies of a single recording location, segmentations from multiple views could be combined with a second neural network, which we call an **aggregation network**. The aggregation network is similar to max-pooling across views followed by convolution as in an MVCNN, but more elaborate as the network learns aggregation over multiple layers. The benefit of a multi-view approach is that inaccuracies of a single view due to occlusions could be corrected with information from other views. In addition, confusion between classes or uncertainty of features is alleviated. The architecture then becomes more invariant to view-point changes, and the importance of each view can be learned.

The overall process is as follows. First a recording location is processed by the single-view pipeline. If the single-view pipeline detects an object, the multi-view pipeline is invoked to improve the detection. The original image from the single-view pipeline is refocused so that the predicted location is centered in the view. Images from neighboring recording locations are then also refocused to look at the same location. The refocused images are then all segmented and reprojected to the base image. The base image is also segmented and concatenated together with the segmentations from the neighboring images. Concatenation is performed in a layer-wise fashion, which means that concatenating m images of n channels results in a single $(m \cdot n)$-channel image. Through this concatenation multiple images can easily be fed to the aggregation network. The aggregation network then outputs a new single-channel segmentation. Similar to the single-view pipeline, the pixels in the segmentation corresponding to our class of interest are then reconstructed as a point cloud, and clustered to predict the final object location. The architecture of our multi-view pipeline is illustrated in Figure 3.5.

As discussed in Section 2.3, MVCNNs define a set of virtual camera poses at fixed locations around a single object. Since these locations are fixed, the network can learn how the different views are correlated and how information from different views can be combined. However, the relative positions of our recording locations with respect to the objects of interest are not fixed, as the trajectories of the recording cars vary. Sometimes an object is approach in a straight trajectory, while other times an object is approached in a curved trajectory because the recording makes a turn. One could train a network to deal with this type of variance and correlate the images. However, while the combination of information from multiple views can result in powerful new features, we expect that it is very difficult for the network adapt to the viewpoint differences. We therefore propose to first calculate a segmentation for each view individually and then apply an **image reprojection**, which transforms all pixels from a segmentation from one camera pose to another, using the corresponding depth information. The reprojected segmentations then all correspond to the same camera pose, and although we disable the network to combine features from different images, correlation no longer has to be inferred since a pixel in one segmentation, directly corresponds to the same pixel in the other segmentation. While reprojection of RGB images can be complex, the implementation becomes more straightforward when depth information is available.

The image that we want to reproject *from* is called the **source image**. Similar to how a point in world-coordinates can be computed for a set of pixel coordinates, a point can be mapped back to pixel coordinates by calculating the vector from the 3D point to the center of projection, and computing where the vector intersects the virtual imaging plane. For reprojection, first the world coordinates of a pixel from the target image are reconstructed. Then the three-dimensional point is projected into the source image. The geometric relationship between a target pixel, its reconstructed three-dimensional point, and the corresponding two-dimensional projection onto the source image, is illustrated in Figure 3.6. By relating each pixel in the target image, to a pixel in the source image, a geometric image transformation can be performed, which maps and interpolates the pixels from the source image, onto the target image, to create a new reprojected image. It is often the case that a pixel in the target image has no corresponding pixel in the source image. In that case the pixel is assigned a special value to indicate this. These pixels can be considered a new class. This approach only works if the positioning data is highly accurate. To make sure that no big inaccuracies are propagated, the distance of the reconstructed point from the source

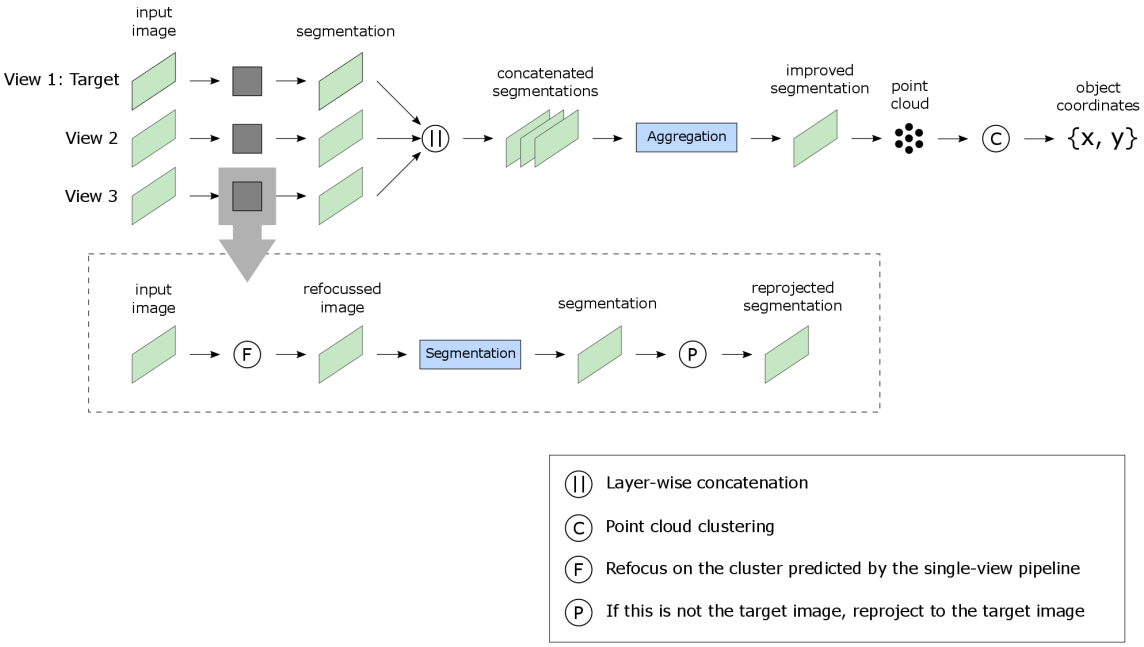


FIGURE 3.5: The multi-view pipeline. Before the multi-view pipeline is invoked, the single-view pipeline generates candidates. For each potential object location found by the single-view pipeline, the image the object is found int is refocused so that the potential object is in the center of the image. This image is now considered the target image. Similar images are created by refocusing the images from the two recording locations that are nearest to the recording location of the target image. The three images are then segmented individually by the segmentation network. The segmentations that do not belong to the target image are then reprojected to the target image, so that the segmentations have a shared camera pose. The three segmentations are than concatenated in a layer-wise fashion. This means that concatenating m images of n channels, results in a single $(m \cdot n)$-channel image. This concatenated image is then aggregated into a single-channel segmentation by the aggregation-network. The new segmentation is reconstructed as point cloud after which a clustering algorithm extracts the positions of the predicted objects.

image to the target image, is compared to the depth value that was stored in the source image. If the difference is below a certain threshold the correspondence is accepted. The threshold can be set dependent on the positioning accuracy of the original point clouds, although a very high positioning accuracy is required for this method to create high quality reprojections. Examples of reprojected RGB-images and segmentations are illustrated in Figure 3.8.

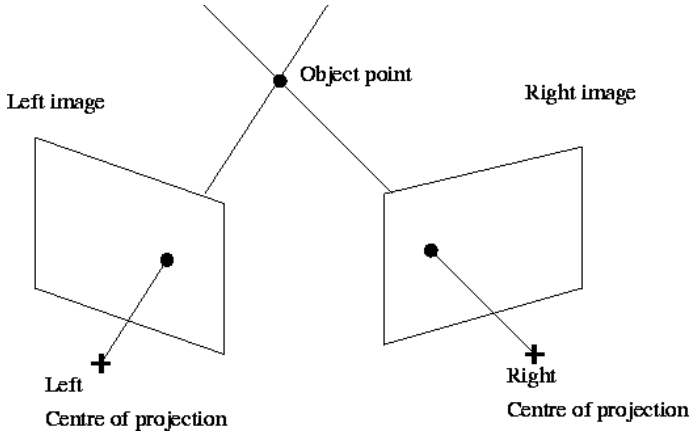


FIGURE 3.6: This image illustrates the geometric relationship of a pixel in a source image, its reconstructed three-dimensional point, and the two-dimensional reprojected point on the imaging plane of a target camera. Whether the point was occluded in the target image can be verified by comparing the original depth value in the target image, with the depth value of the reprojected point. If the difference is higher than a user-specified threshold, the point is disregarded. Some points of the source might not be visible in the target image, while some pixels in the target image might not have a point reprojected onto them. Therefore, the reprojected image does not contain the exact same parts of the scene as the original image.

We expect that segmentations works best when a camera directly focuses on the object of interest. Therefore, all involved cameras are refocused at the same object. **Refocusing** is in essence another reprojection, where only the camera orientation differs between source and target poses, while the position remains the same. Figure 3.9 illustrates how refocusing an image can results in a significant improvement of object visibility. Figure 3.10 provides an example of an image that has been refocused on an incorrect object location.

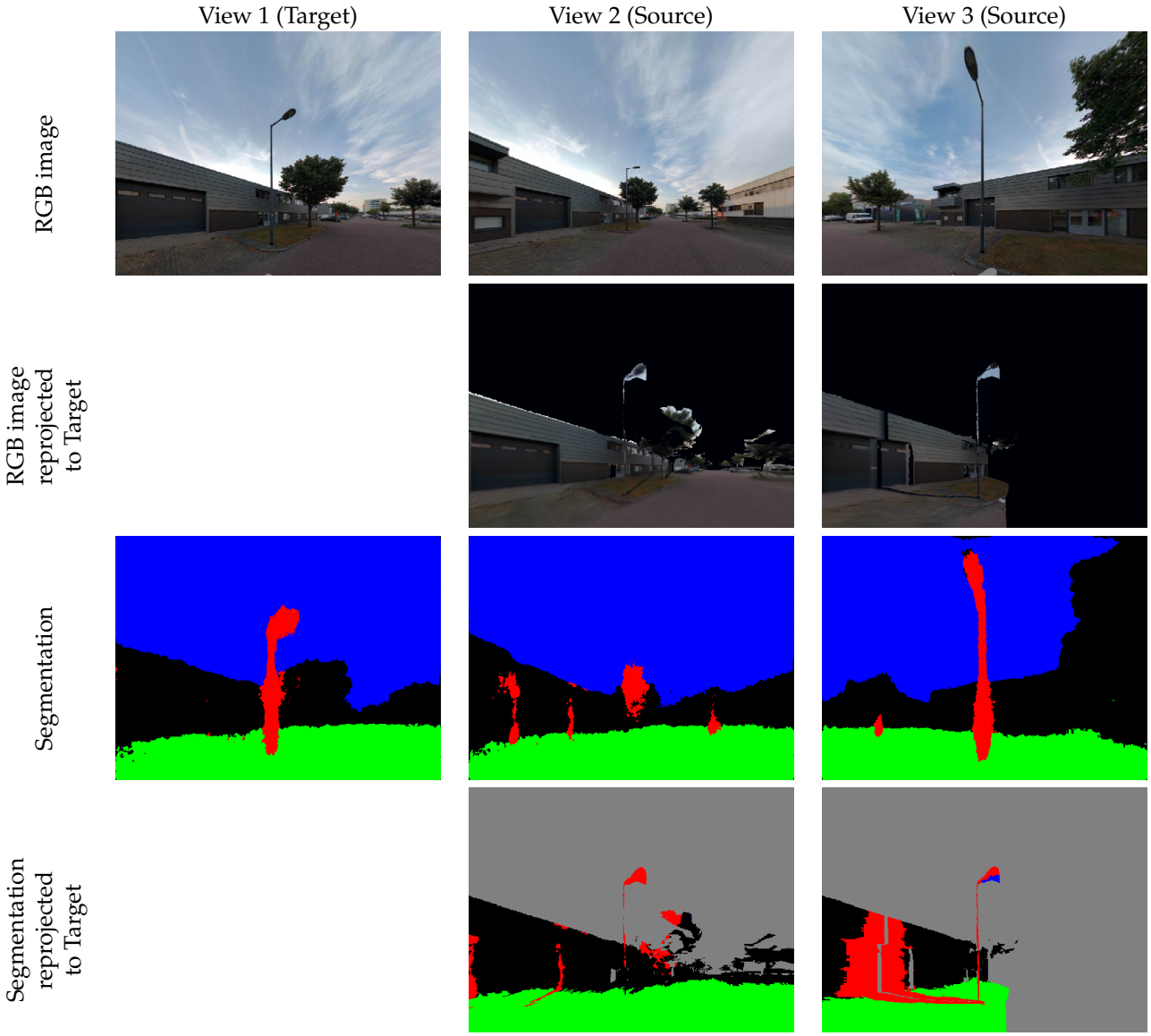


FIGURE 3.7: A reprojection example. View 1 is the target image to which views 2 and 3 are reprojected. With reprojection, the parts of the world that are visible in one view are reprojected to another. The result is an image that contains the parts of the scene that are visible in both views, while the rest of the image is empty. Since reprojection makes use of the depth information for each pixel which might contain inaccuracies, the reprojected images might introduce artefacts. The reprojected RGB images give an intuitive example of what parts of an image are transferred with reprojection. However, the RGB images are not actually reprojected in our pipeline. Instead, only the segmentations are reprojected. The segmentation from the target view, and the reprojected segmentations from the other views are combined in the multi-view pipeline.

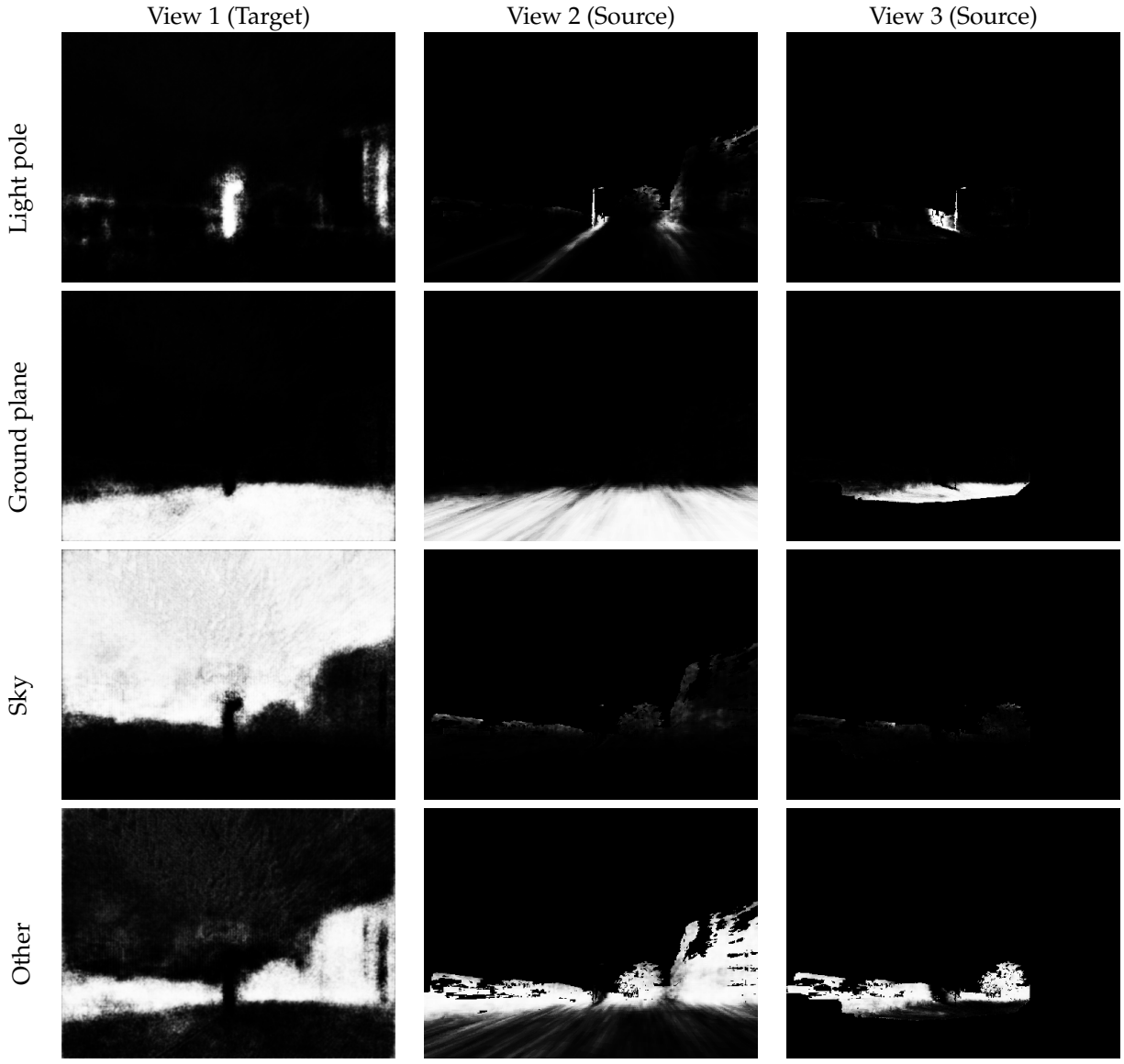


FIGURE 3.8: These images represent the probabilities computed for every class per view. The probabilities of view 2 and view 3 have been reprojected to view 1 so that they can directly be correlated. The 12 images are concatenated in a layer-wise fashion and used as input for the aggregation-network. The aggregation network is expected to learn to combine these probabilities and reduce inaccuracies. It should be noted that probabilities of the sky class can generally not be reprojected since pixels corresponding to sky don't have corresponding depth values. The probabilities that are left after reprojecting the sky probabilities stem from pixels that have incorrectly been labeled as "sky" but do have a corresponding depth value.

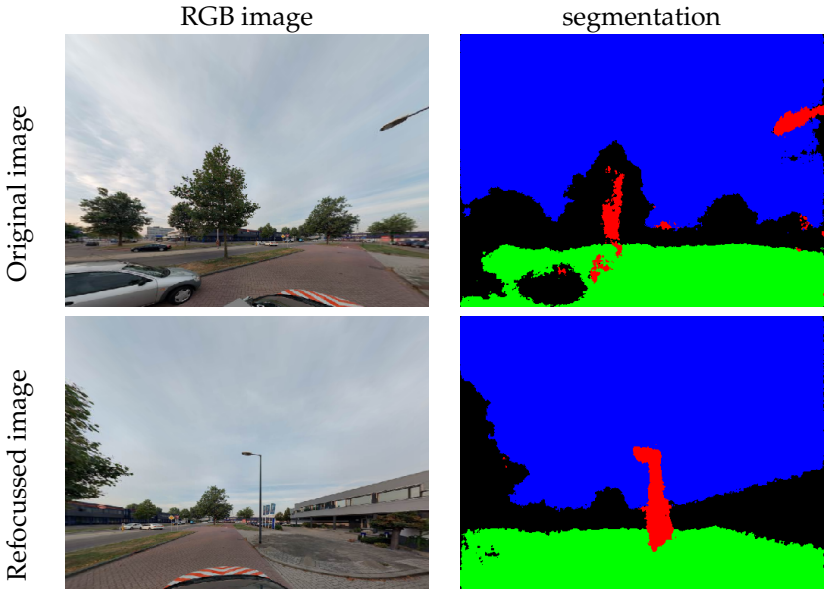


FIGURE 3.9: An example of a refocused image. Even though the light pole is only partially visible on the right side of the original RGB image, the position could be extracted quite accurately by the single-view pipeline. The refocused image therefore has a clear view of the light pole, resulting in a clean segmentation. Even though the segmentation of the original RGB image predicted that parts of the tree in the center of the image belong to the light pole class, these false positives are filtered by the clustering algorithm when the segmentation is reconstructed as a point cloud, since the tree's geometry spreads the samples out too far.

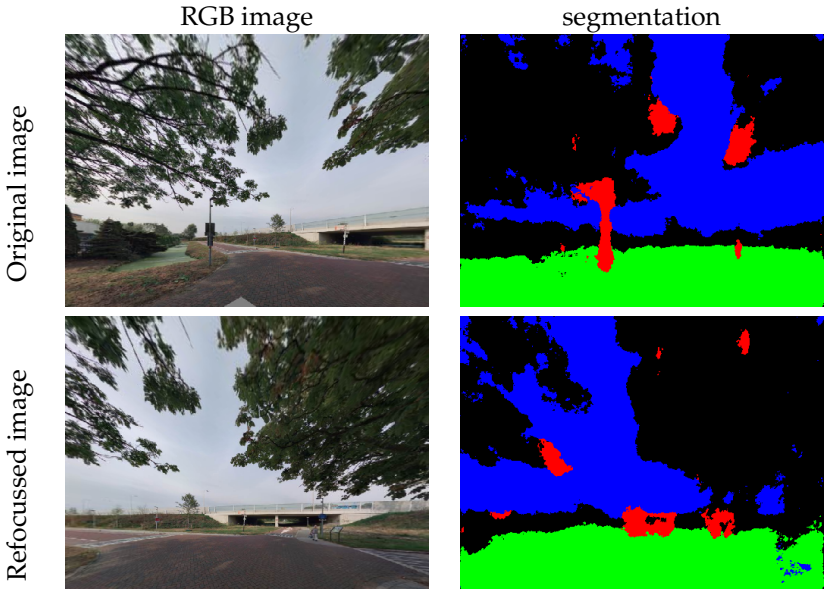


FIGURE 3.10: An example of an image that is refocused at the wrong location. The segmentation of the original image has predicted parts of the trees as belonging to the light pole class. In this case, the false positives have formed a cluster at which the image is refocused. This is however no problem, as the aggregation-network will combine multiple segmentation that are refocused at this location. Since these segmentations do not contain an actual light pole the probability of the false-positives overlapping is low, especially within the point clouds, and the aggregation network will likely generate a segmentation that does not predict a light pole in this location.

Chapter 4

Experiments

This chapter discusses how the performance of our method is measured. First, section 4.1 discusses the implementation of our neural networks. Then, Section 4.2 provides an overview of the data that is available to us. Section 4.3 discusses how this data is preprocessed to provide datasets for training and evaluation. Section 4.4 discusses how the data and ground truth are used to train the neural networks. Finally, Section 4.5 discusses what metrics are used to evaluate the performance of our method.

4.1 Implementation

The pipeline is designed in such a way that it does not dictate a specific implementation of the segmentation and aggregation networks. The choice of network is therefore considered a hyper-parameter. In our implementation we applied the SegNet architecture for both segmentation and aggregation [44]. SegNet uses an encoder to first performs classification, after which a decoder upsamples the low resolution feature maps to fill input resolution feature maps. Instead of learning the parameters of the upsample layers, the decoder takes the pooling indices computed by the corresponding max-pooling layers in the encoder. This is done in a memory conservative way by storing the max-pooling indices only, instead of entire feature maps. The sparse upsampled feature maps are improved by trainable convolution layers. The architecture of SegNet is depicted in Figure 4.1.

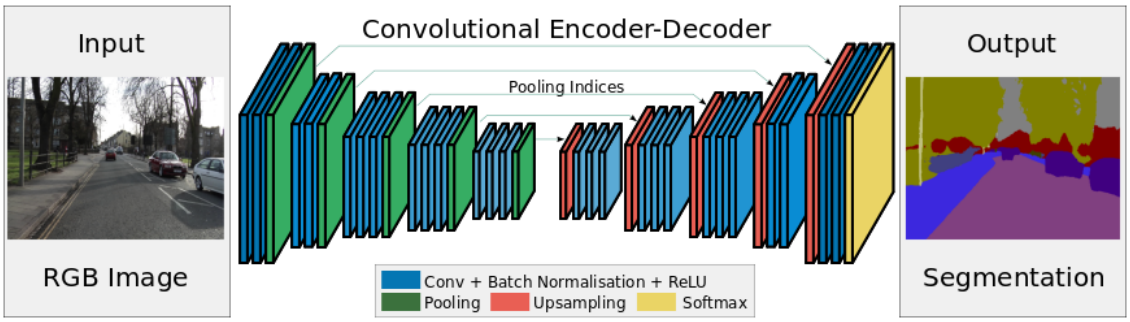


FIGURE 4.1: The SegNet architecture. The encoder part performs classification, resulting in a low-resolution feature map. The decoder map then applies upsampling to scale the feature maps back to input resolution. The upsample layers are not trained but rather the pooling indices from the corresponding max-pooling layers in the encoder part are used. The resulting upsampled feature-maps are improved with trainable convolution layers. The convolution layers in both the encoder and decoder are followed by batch normalization and a ReLU activation function. A final SoftMax layer takes for each pixel the class with the highest probability to generate the final segmentation.

One of the benefits of SegNet is that the encoder part consist of the same layers as the first 13 layers of the VGG16 network, designed for object classification [45]. Therefore, the weights of the trained VGG model can be used as an initialization of the weights of SegNet. The fully connected layers of VGG are discarded in the encoder to retain higher resolution feature-maps as the final encoder output. This reduces the number of parameters from 134M to 14.7M resulting in a

network that requires relatively little memory, requires little computation time during inference and is relatively easy to train. VGG is typically trained on ImageNet [46], which is a common benchmark dataset consisting of hundreds of object categories and millions of images for object recognition. Initial experiments indicate that using pretrained weights does improve the initial loss during training, but does not result in improved performance after convergence. This indicates that our own dataset has a feature representation as rich as that of ImageNet.

Another reason for choosing SegNet is that the original SegNet paper illustrates the performance of the network on the CamVid dataset [47]. CamVid is a high quality database of street-level videos with corresponding pixel-level annotations. The reason that SegNet has had good performance on the CamVid database indicates that it should be easy to adapt to our own street-level imagery.

For the aggregation network it might be computationally cheaper to use a simple network consisting of a few convolution layers for aggregation. However, since the SegNet architecture is already embedded for segmentation, it is easier to employ the same architecture once more for aggregation, than to define an additional neural network architecture. Any segmentation network, or neural network whose output size is the same as the input size could be used for aggregation, as it only needs to compute some sort of dimensionality reduction over the input. In fact, a single convolution layer, as in a MVCNN, might be enough [20]. However, SegNet's relatively complex architecture will allow the aggregation network to learn features that could compensate for erroneous input. We therefore use the SegNet as our aggregation-network architecture. Note that since the aggregation-network receives more data as input, and the number of parameters in the first layer depends on the input size, the aggregation-network has more parameters than the segmentation-network, even though the same architecture is used.

The overall pipeline is implemented in C++. The segmentation and aggregation networks are embedded using the C++ API of the Caffe deep learning framework [48]. Operations on point clouds are implemented with PCL (Point Cloud Library) [49]. Operations on images are implemented using OpenCV (Open Source Computer Vision Library) [50].

First, we experiment with the single-view pipeline to explore whether a combination of depth and color information allows a network to outperform networks that have been trained on only one of these components. Then, the different depth-derived features are evaluated to see whether they offer an improvement. Finally, the multi-view pipeline is evaluated to see if performance could be improved by combining information form multiple recording locations.

4.2 Dataset

The data that used in our experiments is from the city of Schiedam in the Netherlands. The image data consists of street-level 360 degree spherical panoramic images, called cycloramas. Every road of Schiedam has been recorded with a 5 meter interval, resulting in 34320 recording locations. In addition, Lidar point clouds were obtained continuously with a HDL-32E Velodyne Lidar while driving. The point clouds have been reconstructed as meshes. The information from the point clouds and images has been combined resulting in a set of 34320 depth cycloramas. The goal is to find the coordinates of all light poles visible in these images.

Due to the perspective projection in cycloramas, straight lines in the real world are not straight in the cycloramas. In addition, cycloramas are large images. Since the segmentation network architecture is originally meant for images of 480×360 pixels, and because we expect that straight lines can be recognized more easily, the cycloramas are divided into 8 different views with a rectilinear projection. The first image is directed straight along the nose of the recording car, and the other images are obtained with 45 degree yaw increments. To make sure that objects at the edges of these views can still be segmented correctly, the views are created with 120 degree horizontal field of views, introducing overlap between the different views.

The new images have a size of 512×512 pixels. However, initial experiments have shown that this image resolution is too big to be successfully processed by the segmentation network. as the same amount of information is spread over a larger number of pixels, the network does not have enough convolution layers to gain a receptive field over the image that is large enough to recognize features over larger patches and extract more abstract features. For comparison; the feature maps after the final layer of the encoder-part of the SegNet architecture have a size of 16×16 for images of size 512×512, while they have a size of 15×12 for images of the original 480×360 resolution. To solve this issue, more convolution layers can be introduced to the network. However simply resizing the images to the desired 480×360 pixels offers a simpler solution. Linear interpolation is used for color information. Nearest-neighbor interpolation is used for depth values, since interpolating between depth values could result in depth values that do not correspond to any object. For example, interpolating between a nearby wall and a distant wall could result in depth values that incorrectly indicate the existence of a wall in between. Due to the interpolation performed while resizing, some detail in the images will be lost. As a result, some features might be estimated less accurately, although the actual effect is expected to be small.

4.3 Ground truth

In order to train and evaluate models a "golden standard" dataset, also called the **ground truth**, is required to compare to. The ground truth does not need to be perfect but its quality should be as good as possible. As the pipeline is optimized to approximate the quality of the ground truth, the ground truth dictates the upper limit of the quality of the trained model. Different ground truth is required for the different aspects of the pipeline. To train and evaluate the quality of the segmentations produced by the segmentation and aggregation networks, ground truth segmentations are required. To evaluate the quality of the final world positions predicted by the pipeline, ground truth positions are required.

To obtain the locations of all light poles in Schiedam, a team from Cyclomedia has manually inspected all images in the dataset and annotated the light poles. The world positions of the light poles have been determined by calculating where the vectors through the annotated positions from multiple images intersect. The light pole positions are stored as xy-coordinates, ignoring the height of the terrain. The resulting set of 7683 **ground truth positions** can be used to compare with the light pole locations computed by the pipeline. The ground truth positions are used directly to evaluate the world positions predicted by our pipelines.

The main class of interest in this work is the light pole class. In order to see what the effect of the dataset is on the learning of other classes, a few additional classes are included in the ground truth. A second reason to do so is to counter the class imbalance. The number of pixels that occurs for each class within an image differs. This results in a tendency of the network to optimize for the class that occurs more frequently. The class imbalance when dealing with the classes light poles versus the rest can be made smaller by splitting the rest class into multiple other classes. We distinguish between the following four classes: "sky", "ground plane", "light poles" and "other". The "other" class consists of all objects that do not belong to the other three classes. The "other" class mainly consists of buildings, cars and vegetation. Ground truth segmentations are created as follows.

First, all pixels in the image that have a depth value which is too high, are labeled as sky. The HDL-32E Velodyne Lidar scanner has a range of approximately 80 to 100 meters. However since the point clouds are obtained while moving, a slightly higher range is deemed accurate enough for our purposes. The maximum depth at which the point cloud is considered precise enough for more accurate classification is 150 meters.

For all pixels with a depth smaller then 150 meters, the corresponding 3D points are reconstructed. This results in a new point cloud in which we can label different types of objects. A region growing algorithm is applied to estimate the planes in the scene. Initial results have indicated that while the most prominent planes in the scene can easily be extracted, the quality of the estimations is not high enough to properly include all planar objects as a new class. However, the ground

plane is consistently extracted using this method, and is therefore included as an extra class in the ground truth segmentations.

For each ground truth light pole position we define a cylinder with a radius of 1 meter in the xy-plane, and which stretches infinitely upward and downward. For each pixel in an image the corresponding 3D point is reconstructed. If the 3D point lies within one of the cylinders, it is considered to be a light pole, and the corresponding pixels are labeled as such.

Finally, all points that do not adhere to our definitions of the classes "sky", "light pole" or "ground plane", are grouped together in a single class named "unlabeled". The resulting ground truth segmentations therefore contain four classes.

Note that the resulting ground truth segmentations are only a noisy approximation to a perfect per-pixel classification. Since tree often appear near light poles, and their branches might reach within the cylinder defined around light poles, some parts of trees are incorrectly labeled as "light pole". However, neural networks can learn some invariance to this noise. For example, it is common to train face-detection networks, using bounding boxes around the faces as ground truth. These bounding boxes however, also contain some pixels that belong to other classes. A neural network can be invariant to such inaccuracies as long as they are not too frequent. As neural networks first optimize towards the features that are most common in the training set, a network becoming affected by low-frequency inaccuracies can be recognized as overfitting.

4.4 Training Methodology

While the original Schiedam dataset consists of 34320 recording locations, a smaller selection is made to keep computation time and memory requirements tractable. Recording locations are selected and processed randomly until to create a dataset of 5000 images for the single-view pipeline. Note that as panoramic images are split in multiple images, and images without a properly visible light pole are filtered, the number of usable images extracted differs per recording location. These datasetsare divided into a training set, a test set and a validation set using a $3000 : 1000 : 1000$ ratio. Since the multi-view pipeline is trained on segmentations from the single-view pipeline, a new dataset is produced, processed by the single-view pipeline and refocussed, to create a set of 3000 images for the multi-view pipeline. This dataset is split into a training, validation and testing set using a $1000 : 1000 : 1000$ ratio. To tune the training parameters, the validation set is used. Since the model has not seen any of the images from the validation set during training, the performance on the validation set gives an impression of how well the model generalizes to unseen data. While the loss of the model on the training data will generally decrease, the loss of the model on the validation set will increase when the model starts overfitting. At this point the model adapts to the details that are specific to the training set, while those are not representative of the features in a generalized setting. The validation set is therefore used to determine when the training process should be stopped. Since the validation set is used to optimize the training process, the model is also adjusted to the validation data and a separate test set is used to measure the final performance of the model.

Automatic optimization algorithms exist that adaptively change the learning rate of the solver over time. Caffe supports for example AdaDelta [51], AdaGrad [52], Adam [53], Nesterov's accelerated gradient [54] and RMSprop [55]. However, choosing the rate at which the learning rate should degrade is another hyperparameter that requires tweaking to find the right value. Instead, we opt for a more manually controlled training approach. First we experiment with multiple learning rates, to find a the highest learning rate at which the network still converges. The high learning rate allows for faster training. The network is trained until it converges. Then the learning rate is decreased manually to see if the network converges further. This process is iterated until altering the learning rate no longer improves convergence. Training is performed using a Tesla K40M GPU card with 12GB GDDR5 RAM. With a validation and testing set enabled, the maximum batch size without running out of memory is 3 for most of our networks. Even for the networks that did allow larger batch sizes, a batch size of 3 was used, so that the learning rates

did not have to be adjusted.

The classes in our data are not distributed equally. A larger part of the pixels in an image correspond to sky, while relatively few pixels belong to the light pole class. The exact percentage of pixels for each class and the corresponding weight have been illustrated in Figure 4.2. Refocusing of images in the multi-view pipeline has little impact on the class percentages, and the same weights are used in the single-view pipeline and the multi-view pipeline. As a result, the easiest way for the network to optimize the loss, the steepest gradient, corresponds to optimizing for the sky class first. However, we want all classes to be optimized equally, while a slight advantage for the light pole class could even prove beneficial as it is our main class of interest. To ensure the network optimizes all classes, we weight each class in the loss function so that the number of pixels belonging to the class, multiplied by the weight is the same for each class. Weights are defined so that the weight for the light pole class is exactly 1 and that the other weights are defined relative to that. Class weighting is not not ideal, as the class distributions vary in each image. Since we rather include more false positives in our results that could be filtered manually, than have false negatives that can not be corrected, a small bias towards light poles was intentionally created by increasing the weight for the light pole class to 10.

Class	Percentage	Weight
Light poles	0.4%	10.00
Ground plane	17.4%	0.023
Sky	51.3%	0.008
Other	30.9%	0.013

FIGURE 4.2: This table depicts for each class the percentage of corresponding pixels. Percentages have been computed over a dataset of 3000 ground truth segmentations. Each class is weighted so that the percentage multiplied by the weight is equal for all classes. The weight for the light poles class was originally set to 1, while the other weights were defined relative to that. Initial experiments indicate that increasing the light pole class weight by a factor of 10 yields slightly better results.

Note that as the number of values in the different image formats differs, the number of parameters in the convolution kernels in the first layer of the network also differs. The different networks therefore have a slightly different number of parameters, and should learn different features. Because of this, each network requires a different number of training iterations. Therefore, it is uninformative to compare the exact number of training iterations for each network. During training each network is trained until it converges. A network has converged when the loss when the loss on the validation set has stopped decreasing. If the loss on the validation set starts increasing, the network starts overfitting, and the training process should be stopped, or resumed with a lower learning rate.

4.5 Metrics

For evaluation we distinguish between several conventional types of test outcomes. Predictions can be either positive or negative. Whether a prediction matches with the ground truth defines if it is true or false. When a class is predicted and it matches the ground truth, this is called a **true positive (TP)**. When a class is predicted while the ground truth does not specify so, it is called a **false positive (FP)**. When a class is not predicted in a location, where it should have been, it is called a **false negative (FN)**. Finally, when a class is not predicted in a location where the class does indeed not occur, it is called a **true negative (TN)**. We will refer to these different test outcomes as events. This distinction between events is illustrated in Figure 4.3. A table can be used to count the various types of events for a specific class. A **confusion matrix** is a similar table that visualizes how predictions are distributed over multiple classes. It should be noted that not all types of events are equally relevant in different settings. True negatives can easily be counted when considering the pixels of an image, infinite true negatives exist when considering positions

in a continuous space. For the evaluation of segmentations we will therefore include true negatives, while we will ignore true negatives when evaluating world positioning performance.

		Predicted	
		Yes	No
Actual	Yes	True Positive	False Negative
	No	False Positive	True Negative

FIGURE 4.3: This table illustrates the different types of events that can occur when comparing a prediction to the ground truth. A prediction can be either positive or negative for a specific class. Whether the prediction matches the ground truth defines whether the prediction is true or false.

Common metrics that incorporate the different types of events are the precision and recall metrics. The **precision** metric is the fraction of predictions that is correct. A higher precision means fewer false positives. The **recall** metric is the fraction of all actual instances that has been correctly predicted. A higher recall means fewer false negatives. Neither precision or recall requires a count of false negatives, and both can therefore be used for evaluation of both segmentation and world positioning. Precision and recall are computed as follows.

$$presision = \frac{true\ positives}{true\ positives + false\ positives} \qquad recall = \frac{true\ positives}{true\ positives + false\ negatives}$$

Typically, precision and recall are inversely-related to one another. As precision increases, recall decreases and vice-versa. This trade-off can be visualized using a **Precision-Recall-curve (PR-curve)**. To create a PR-curve, a threshold is applied on specific parameters of the predictions, such as the probability or uncertainty, or the blob size of class. Precision and recall are then computed using many different values for the threshold, and these points are plotted with recall on the horizontal axis, and precision on the vertical axis of the graph. The **area under the curve (AUC)** can be used as another metric, which summarizes the PR-curve in a single value. AUC measures the performance of the model over the whole range of threshold values. A perfect model will have an AUC of 1.0, while a model that is influenced by the parameter will have a lower AUC. To summarize the performance of multiple classes, **the mean of the AUCs per class (mAUC)** is used.

Conventional metrics that summarize the trade-off between precision and recall are the break-even point between precision and recall, or the F-score, which is a weighted average of precision and recall. However, these metrics are less meaningful to us. The precision is of less importance than recall, since false positives can easily be filtered out through post-processing steps such as manual verification. However, stronger requirements apply for the recall of the method, since there is no scalable approach to correct false negatives.

The computed positions can be evaluated in multiple ways. In a **per-segmentation evaluation** we evaluate how many of the predicted positions extracted for each view are correct. If a detection is missing where an actual object exists, it is considered a false negative. A prediction that is within a 1.5 meters to the ground truth is considered a true positive, while considered a false positive otherwise. The per-image segmentation evaluation indicates how well segmentations are on a pixel level. In a **per-object evaluation** we evaluate how many of the actual objects have been found, when the predictions over all images are combined. In this scenario, when at least one detection is within 1.5 meters to the ground truth position, it is considered a single true positive. Each detection that does not match a ground truth position is still considered a false positive. While the per-segmentation provides valuable information regarding segmentation quality, the per-object evaluation provides insight in the performance of the pipeline overall. Since we are most interested in the recall rate of light poles, the per-object evaluation is the most important metric.

Chapter 5

Results and discussion

In this chapter we evaluate our pipeline on different types of data and evaluate the performance with regard to segmentation and world-positioning. The segmentation performance of our regular single-view pipelines is evaluated in Section 5.1. The world positioning performance of the regular single-view pipeline is evaluated in Section 5.2. Then, the segmentation and world positioning performance of the single-view pipelines with alternative depth-derived features are evaluated in Section 5.3 and Section 5.4 respectively. Finally the segmentation performance of the multi-view pipeline is evaluated in Section 5.5, while its world positioning performance is evaluated in Section 5.6.

5.1 Segmentation performance of the single-view pipeline

The first task of the proposed pipeline is to solve a segmentation problem. In this section the segmentation results of the single-view pipeline using color-information, depth-information, and a combination both are evaluated. The networks operating on the different types of images are referred to as the RGB-network, D-network and RGBD-network. It is expected that some classes are easier to segment than others due to difference complexity of geometry, color and frequency. To provide a qualitative estimate of the performance, some examples of good segmentations are shown in Figure 5.1, and some examples of bad segmentations are shown in Figure 5.2.

The resulting segmentations have been evaluated quantitatively by comparing the estimated labels to the ground truth. Figure 5.3 shows per class how the precision and recall rates change when only probabilities above a certain threshold accepted. All three networks perform well with regard to the "ground plane", "sky" and "other" classes. The D-network and the RGBD-network score consistently above 0.95 AUC for these classes. The RGB-network, however, performs slightly worse with an AUC of 0.94 for the "ground plane" class and a 0.85 AUC for the "other" class. The plots have a consistent ordering to the networks, with the RGB-network scoring the lowest and the RGBD-network scoring the lowest. This ordering is confirmed by the mAUC scores. It is interesting to see that the D-network outperforms the RGB-network, which indicates that depth information on its owns is more useful for our purposes than color information on its own. A combination of both types of information performs even better. This trend holds for all classes, except for the sky class where the D-network and RGBD-network perform equally well. The biggest improvement of depth information over color information occurs at the most important and most difficult class; light poles. The difference in performance between the RGB-network and the D-network is larger than the difference in performance between the D-network and the RGBD-network. This is shown both in the per class AUC scores and the overall mAUC scores.

Figure 5.4 shows for each network a confusion matrix summarizing how predictions of the segmentations are distributed over classes. Generally, the percentage of correctly predicted labels is highest in the RGBD-network, and lowest in the RGB-network, while confusion is lowest in the RGBD-network, and highest in the RGB-network. The confusion matrices therefore further support the conclusion that the RGBD-network performs best and RGB-network performs worst. Remarkable is that while the D-network performs better than the RGB-network for every other type of event, the confusion is slightly higher for pixels belonging to the "other" class that are misclassified as the "light pole" class. This difference is however relatively small. In all three networks, the most confusion occurs due to light pole pixels being misclassified as belonging to the

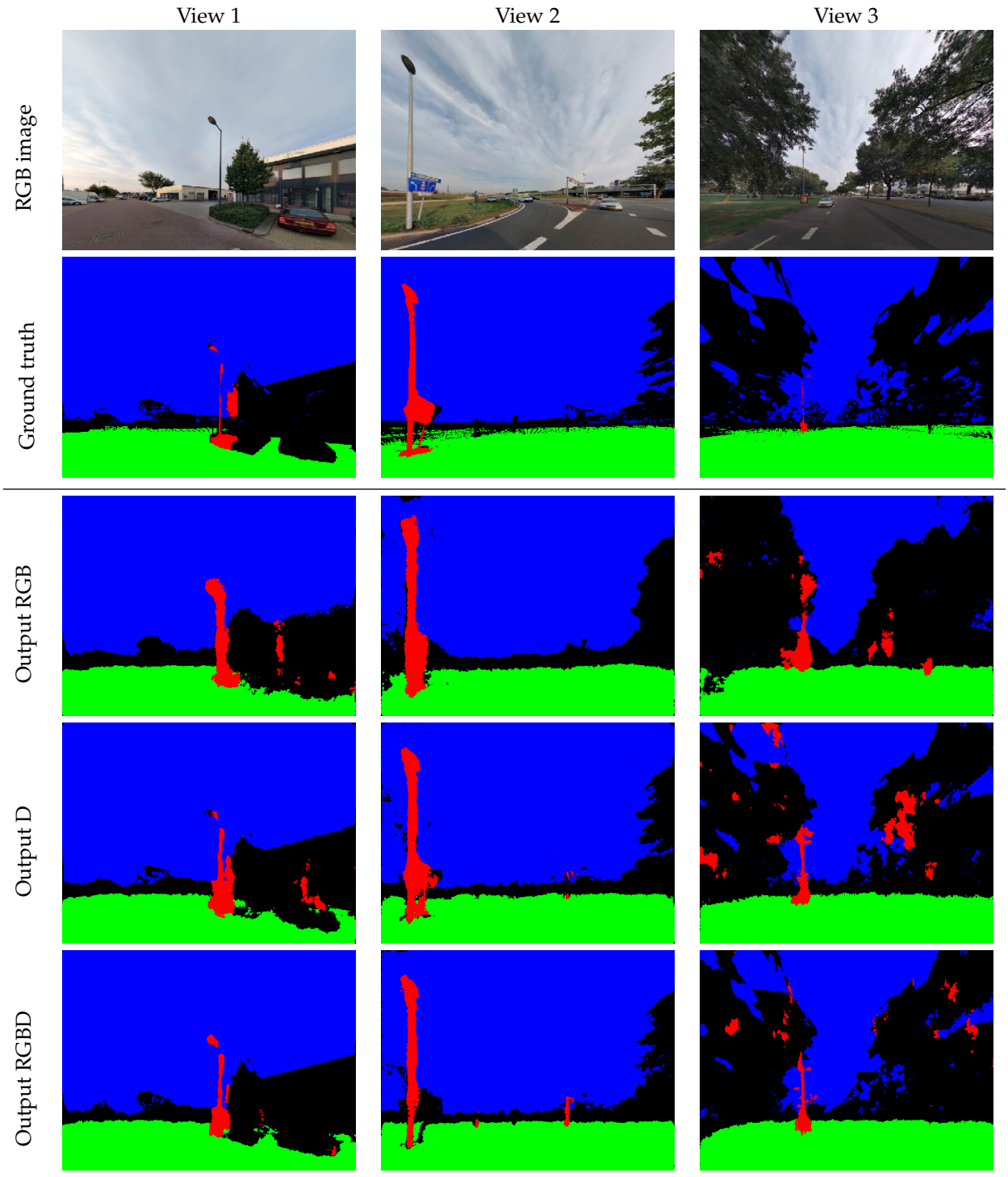


FIGURE 5.1: These are some examples of good segmentations resulting from the single-view pipeline. The quality of the ground truth is not perfect. As a result the ground truth of view 1 has incorrectly labeled parts of the tree and bushes as light poles. In the ground truth of view 2, the traffic sign next to the light pole is also labeled as a light pole. In the ground truth of view 3 the light pole is located quite far from the camera, making it expectedly harder for the network to correctly segment the light pole correctly. However, in the resulting segmentations, the light poles are segmented quite clearly. The overall trend seems to be that segmentations of RGB-images are quite generous, and include many false positives around light poles, while segmentations of D-images and RGBD-images are cleaner, with RGBD-images being segmented the clearest. Unexpectedly, The segmentation of the RGBD-image of view 2 is even cleaner than the ground truth, and even correctly includes a light pole that was not present in the ground truth because of its distance. The segmentations of view 3 include a few false positives for the light pole class. However, the fact that the light pole has been segmented correctly, while it seems nearly invisible in the original RGB image is promising. Light poles are segmented especially accurate when the background is sky.

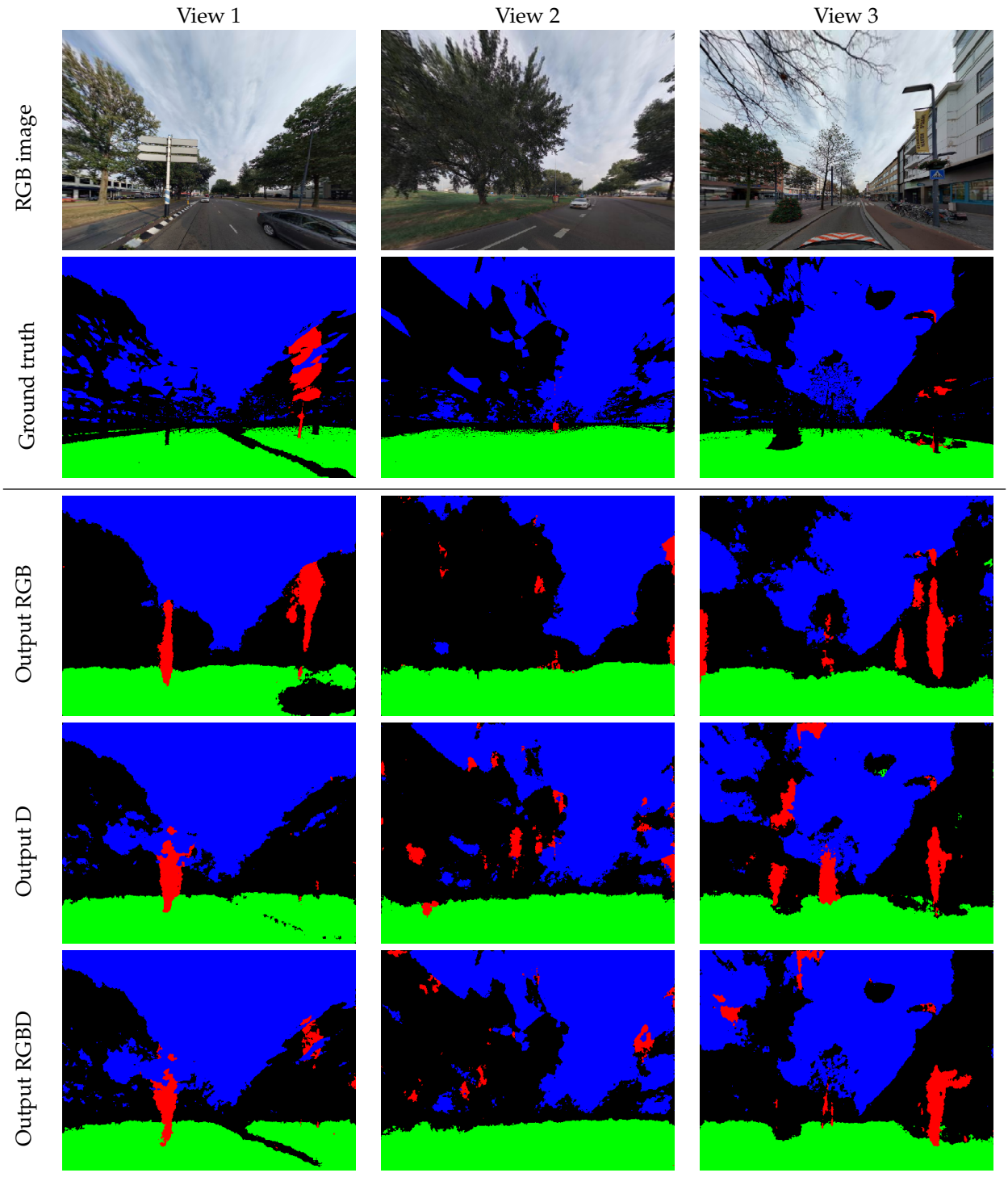
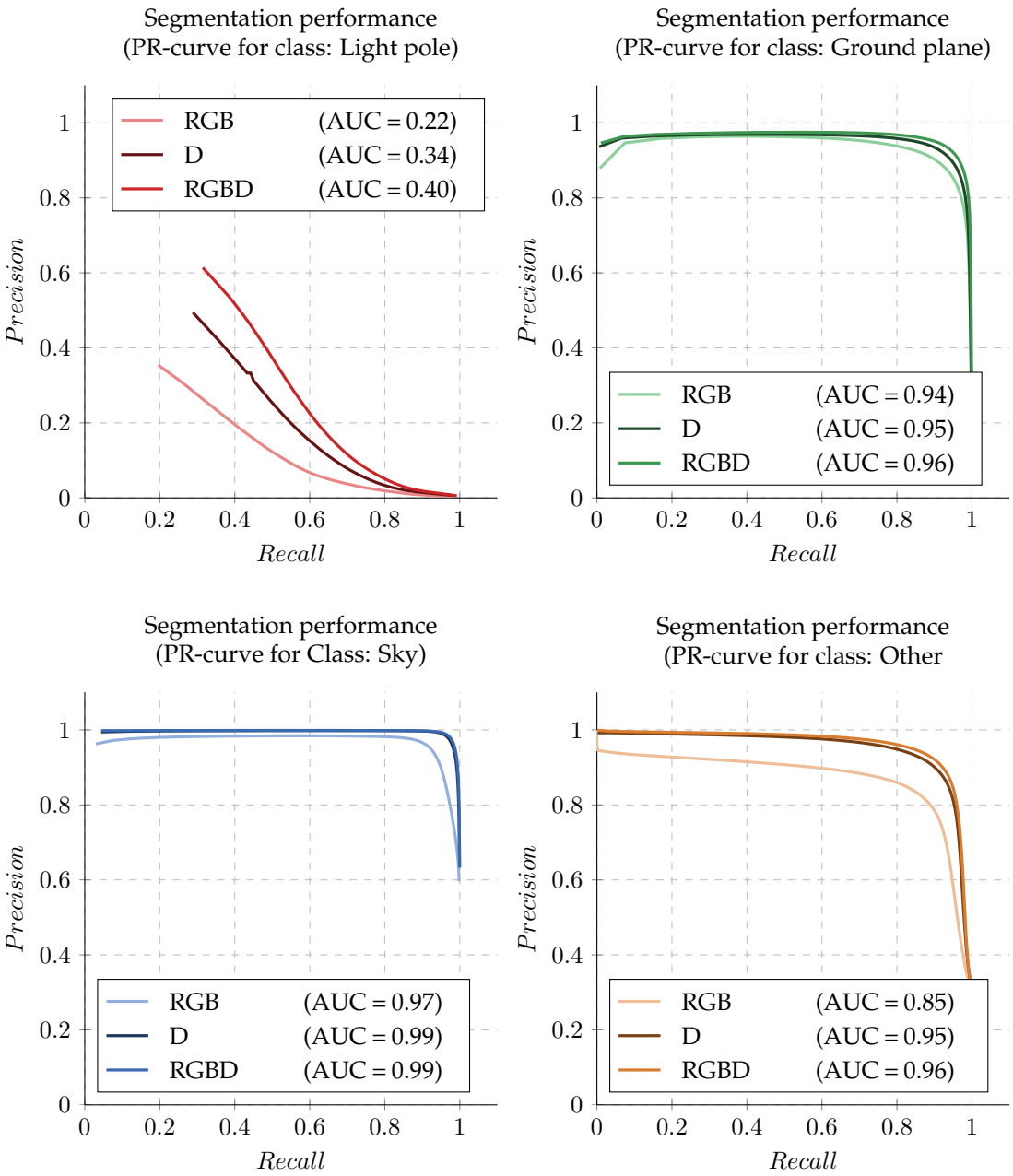


FIGURE 5.2: These are some examples of bad segmentations resulting from the single-view pipeline. The ground truth of view 1 indicates the existence of a light pole but has incorrectly labeled many pixels that correspond to the tree, as light pole class. The corresponding segmentations incorrectly label the sign post on the left as a light pole. The segmentation of the RGB-image is the only segmentation that does include the actual light pole. The ground truth of view 2 has marked a light pole that is nearly impossible to see in the image. The corresponding segmentations have not labeled significant blobs of pixels around that area and miss the light pole. Many false positives for light poles occur in the trees. This can be explained by the fact that trees have also been present in many ground truth segmentations. The ground truth of view 3 has not clearly labeled the light pole. This can occur when the manually determined light pole position is off. The corresponding segmentations are not very clean but do include the light pole to some extend. However, because the pixels do not have the same values as the ground truth, they are still considered incorrect.



Network	mAUC
RGB	0.75
D	0.81
RGBD	0.83

FIGURE 5.3: These are Precision-Recall curves for the RGB, D and RGBD-networks, compared per class. The results are thresholded on class probabilities. When only predictions with high probabilities are accepted, lower precision and higher recall is achieved. When lower probabilities are accepted too, recall increases while precision decreases. A higher recall corresponds to fewer false negatives. A higher precision corresponds to fewer false positives. The trend for each class is that the RGBD-network has the most accurate predictions, while the RGB-network performs worst.

Network: RGB

		Predicted			
		light pole	ground plane	sky	other
Actual	light pole	410883 **0.53**	45498 **0.06**	22527 **0.03**	289112 **0.38**
	ground plane	311816 **0.01**	27065883 **0.94**	142567 **0.00**	1373085 **0.05**
	sky	1071574 **0.01**	10905 **0.00**	81511560 **0.92**	5892594 **0.07**
	other	2285452 **0.04**	3937512 **0.07**	4064899 **0.07**	44709733 **0.81**

Network: D

		Predicted			
		light pole	ground plane	sky	other
Actual	light pole	512572 **0.63**	26598 **0.03**	6773 **0.01**	265322 **0.33**
	ground plane	307996 **0.01**	26911075 **0.95**	27416 **0.00**	1178013 **0.04**
	sky	543004 **0.01**	548 **0.00**	85118110 **0.97**	2361646 **0.03**
	other	2736526 **0.05**	3060022 **0.05**	1753661 **0.03**	48336318 **0.86**

Network: RGBD

		Predicted			
		light pole	ground plane	sky	other
Actual	light pole	506863 **0.66**	24926 **0.03**	4717 **0.01**	234003 **0.30**
	ground plane	231289 **0.01**	27719226 **0.96**	16736 **0.00**	764950 **0.03**
	sky	387795 **0.00**	572 **0.00**	86595926 **0.97**	2205841 **0.02**
	other	2154215 **0.04**	3013223 **0.06**	1530631 **0.03**	47754687 **0.88**

FIGURE 5.4: These are the confusion matrices of the segmentation performance of the RGB-network, D-network and the RGBD-network. The rows indicate the actual class a pixel belongs to according to the ground truth. The columns indicate what class the pixel was predicted as. The predicted class is the class with the highest probability. The numbers in each cell represent the number of pixels, and what percentage this number is of all pixels of that class in the ground truth. E.g. 30% of all pixels that should have been predicted as the "light pole" class, were actually labeled by the RGBD-network as the "other" class. Keep in mind that, since class distributions are not equal, percentages from different rows can not be directly compared to each other. The RGBD-network seems to produce the highest quality segmentations, with the highest true-positive rights, and the least confusion between classes.

"other" class. This confusion is smallest in the RGBD images and largest in the RGB images. It is likely that this confusion is partially caused by the inaccuracies in the ground truth. In many images, trees are very close to the light poles, and due to the approach in which the "light pole" class is labeled, some parts of those trees are incorrectly labeled as light poles. Since trees are mainly included in the "other" class, the neural networks could be confused between the tree features present in the "light pole" class and the tree features included in the "other" class. Another reason is that "sky" and "ground plane" classes are relatively easy to learn, since the sky has a relatively constant color and no corresponding depth values, and the ground plane has relatively constant color and flat geometry, while the "light pole" class has more complex features and the "other" class consists by definition of even more complex features of many different kinds of objects. The final segmentations follow the same trend as the probabilities per class; The RGB-network performs worst, while the RGBD-network performs best. The "light pole" class is the class that is estimated worst, and the "other" class causes the most confusion.

5.2 World positioning performance of the single-view pipeline

The second task of the proposed pipeline is to solve a world positioning problem. In this section the quality of the extracted light pole positions is evaluated. The pipeline including segmentation, clustering of the segmentation-derived point clouds, and world positioning is evaluated for color information, depth information, and the combination of both. The pipelines will be referred to as the RGB-pipeline, D-pipeline, and the RGBD-pipeline. Note that the test set consists of 1000 images that together contain 725 unique light poles.

As neural networks learn to approximate the features in the ground truth, they cannot be expected to produced output with a higher quality than that of the the ground truth. By evaluating the world positioning performance of our pipeline while using ground truth segmentations, we can compute the maximal performance that can be expected of our pipeline. The ground truth segmentations are not perfect, as they contain many pixels that are incorrectly labeled as "light pole". However, only pixels within a one meter radius of a ground truth light pole position are labeled as light pole. During testing we consider every cluster within a 1.5 meter radius to a ground truth light pole position correct. The incorrectly labeled clusters therefore all fall within the acceptance radius and the precision of the positions derived from ground truth segmentations is therefore exactly 1. The recall rate of the ground truth segmentations is, however, not optimal. Some light poles are partially occluded within the segmentations and therefore consist of multiple geometrical components that are smaller than 100 points each. This is not checked during ground truth creation. Since the clustering performed in the segmentation-derived point clouds only accepts clusters larger than 100 points, the ground truth segmentations introduce false negatives. Experiments indicate that the maximal recall score of the ground truth on a per-view basis is 0.74 and that the recall limit on a per-object basis is 0.79. By comparing the recall scores of our pipelines relative to these limits, we can measure how close the score is to the maximum achievable score.

The world positioning performance, both on a per-view basis and on a per-object basis is shown in Figure 5.5. The performance is measured both absolute and relative to the recall limits. The AUC scores indicate that the RGBD-pipeline generates the best world positioning score over all threshold values. The RGBD-pipeline also achieves the highest recall scores in all cases. The world positioning performance ordering is the same as for segmentation performance; The RGBD-pipeline achieves the best results, while The RGB-pipeline performs worst. Since the order of performance is the same for world positioning as for segmentation performance, we can conclude that better segmentations do result in better world-positioning performance. The RGBD-network reaches 56% recall rate on a per-view basis, which is 75% of the limit, with a precision of 17%. On a per-object basis the RGBD-pipeline achieves a 62% recall rate, which is 78% of the limit, with a precision of 13%.

The difference between the recall score of the pipeline when using ground truth segmentations and the recall score of the pipeline when using predicted segmentations, indicates the performance improvements that can be achieved through better segmentations. This means that with the results of the RGBD network, 17% recall can still be achieved through better segmentations. Better segmentations can be achieved using different segmentation-network architectures, and finding optimal data representations for the segmentation-network to operate on.

The difference between the recall score of the pipeline when using ground truth segmentations and a perfect recall score of 1, indicates what performance improvements could still be achieved through other pipeline changes, such as improved segmentation-derived point cloud clustering, better ground truth and tweaking of hyperparameters such as the minimal cluster size. The current results achieved with ground truth segmentations indicate that 21% recall could still be achieved through pipeline changes.

The fact that world positioning performance is better on a per-object basis than on a per-view bases indicates that if a light pole is missed in one image, it can still be found in another. This suggests that information from multiple views could be combined for improved detections and confirms our reasoning for creating the multi-view pipeline.

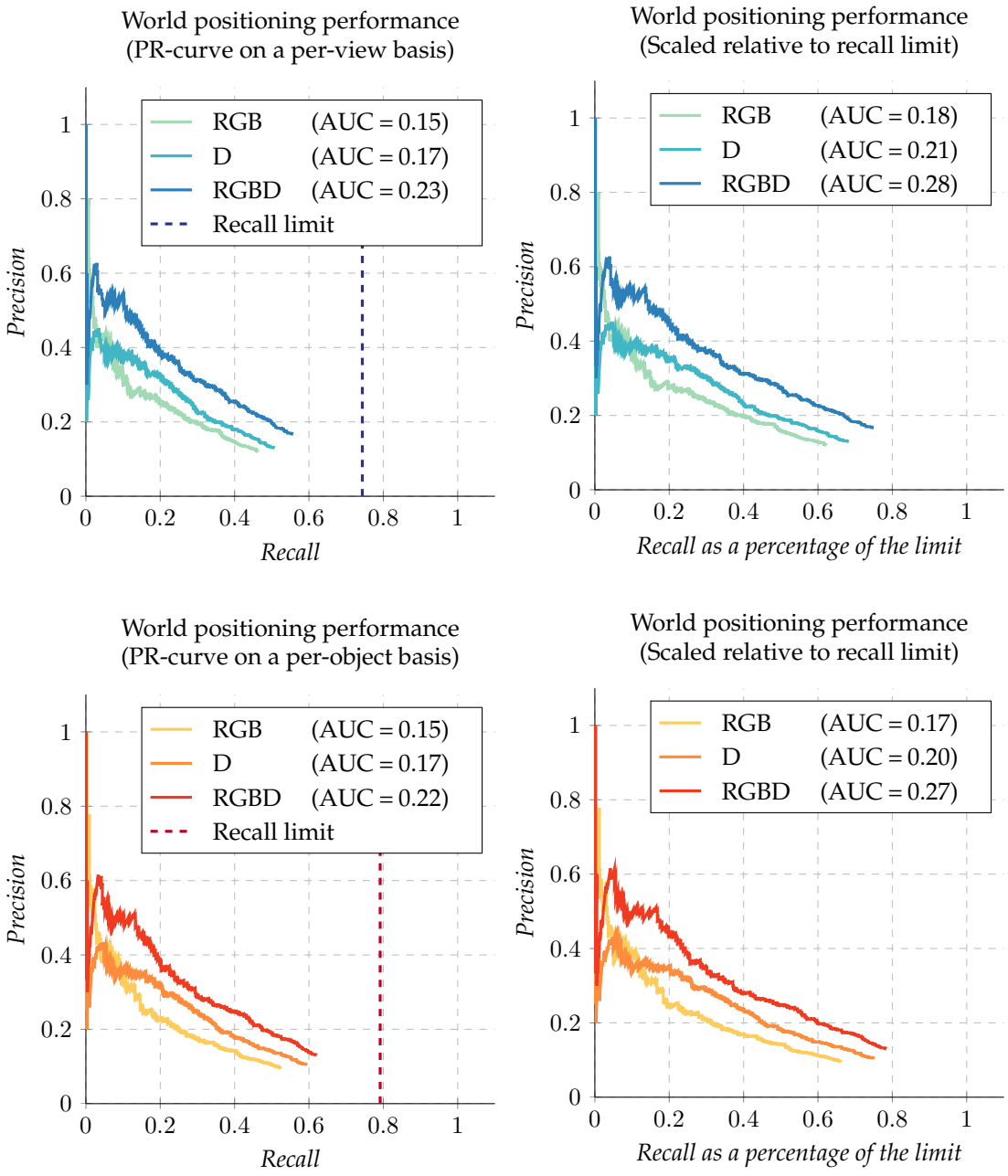


FIGURE 5.5: The PR-curves for the world positioning performance of our single-view pipelines. Predicted positions are considered correct if they are within 1.5 meters to a ground truth light pole position. The graphs on the top show the performance on a per-view basis; if a light pole is missed in an image it counts as a false negative. The graphs on the bottom show the performance on a per-object basis; if an object is missed in one view, but is found in another, it still counts as a true positive. The graphs on the left show the absolute precision and recall values. Evaluating the world positioning performance with ground truth segmentations as input has illustrated that even when using theoretically perfect segmentations, the clustering approach does not yield a perfect recall score. This indicates that there is an upper limit to the attainable recall score. This is primarily a result of the clustering process, which only accepts clusters larger than 100 points. The graphs on the right show the PR-curves, where recall is measured relative to the maximally attainable recall limit. The precision reached when using perfect segmentations is 1. This means that all blobs in the ground truth, larger than 100 points, are within 1.5 meters of an actual light pole position. The precision is therefore not scaled.

5.3 Segmentation performance of depth-derived features

In this section the segmentation performance of segmentation-network operating on images with color-information in combination with alternative depth-derived features is evaluated. The networks using AzEl and HDA features will be referred to as the RGBAzEl-network and the RGBHDA-network respectively. Figure 5.6 shows a few examples of images processed by the RGBAzEl-network and the RGBHDA-network. Surprisingly, the segmentations resulting from the RGBAzEl-pipeline and the RGBHDA-pipeline seem qualitatively worse than those of the pipelines without the depth-derived features. Interestingly, the networks seem to have a high response to pole-like objects in general, instead of to light poles specifically.

The confusion matrices summarizing how class predictions are distributed are shown in Figure 5.7. The conclusion that the pipelines with the alternative depth-derived features perform generally worse than the pipelines without, is quantitatively supported by these confusion matrices. Compared to the RGB-pipeline, the worst performing pipeline without depth-derived features, the RGBAzEl-pipeline and RGBHDA-pipeline have fewer "light pole" pixels misclassified as "other", but have more pixels of the "sky" and "other" classes being misclassified. Since the "sky" and "other" classes, correspond to more pixels than the "light pole" class, the overall number of correctly classified pixels is smaller than that of the RGB-pipeline. This effect is caused by the segmentation-network being very generous towards the "light pole" class. As the segmentations show, predictions for the "light pole" class consist of overly large blobs of pixels around the objects. Due tot his, the number of true positives is very high for the "light pole" class, but the number of false positives is too, resulting in lower true positive rates for the other three classes.

The depth-derived features performing worse than any other single-view pipeline is surprising. Theoretically, neural networks are able to learn to ignore irrelevant data. If the networks would set the weights in the first layer corresponding to the channels of the depth-derived features to zero, then they would be equal to the RGB-network. Therefore, the networks should theoretically never perform worse than the RGB-pipeline. The fact that the network has been unable to learn this, indicates that there are architectural limitations that limit how the gradients to the first layers are able to propagate during training. Since the introduction of the network used in our implementation, SegNet, and the development of this work, there have been large improvements in deep neural network architecture. One of the problems that have been solved since is the degradation problem. The general trend has been to increase neural network depth to allow for stronger features. However, as depth increases, accuracy saturates and then degrades rapidly [32]. This is a result of the network being unable to preserve accuracy after saturation, as identity mappings are difficult to approximate with a set of non-linear layers. As a solution, He et al. proposed additional "residual connections" that simply perform identity mapping, skip one or multiple layers, and sum their output with the output of the stacked layers. As a result, identity mappings do not have to be approximated, and gradients can propagate over multiple layers, without being limited by the layers in between. Residual connections do not contain weights and therefore do not add any complexity to the network. Inspired by residual networks, dense networks were proposed by Huang et al. that connect each layer to every other layer in a feed forward fashion, alleviating the vanishing gradient problem, strengthening feature propagation further and encouraging feature reuse [56]. In contrast to residual networks, the additional connections in dense networks are not summed but concatenated, resulting in more parameters. Our conclusion is that it is likely that the performance of our pipelines with depth-derived features is limited by the current neural network architecture. The depth-derived features might still prove beneficial when using more complex segmentation-network architectures.

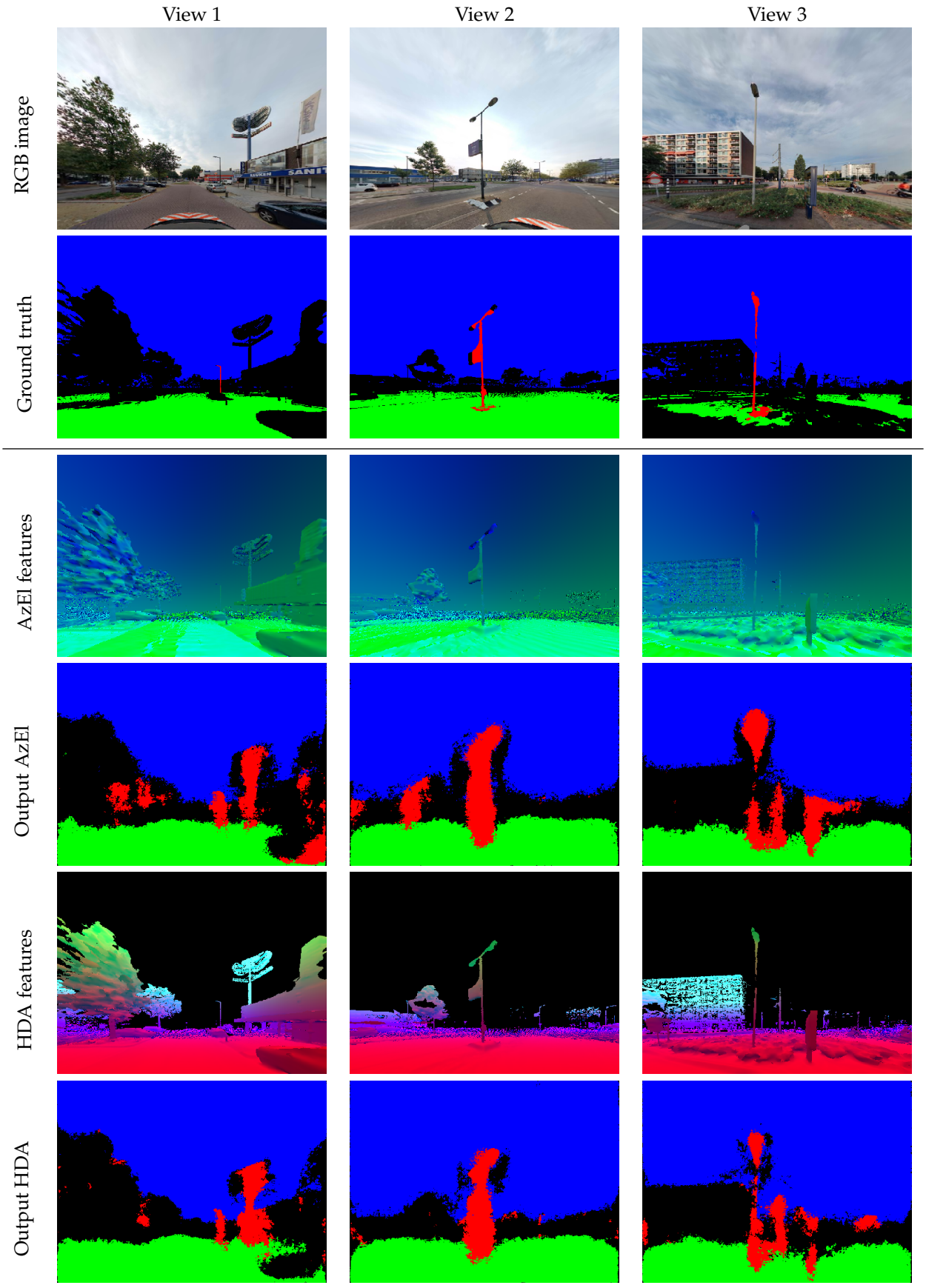


FIGURE 5.6: Some examples of segmentations resulting from the RGBAzEl-pipeline and the RGBHDA-pipeline. Surprisingly, the segmentations seem qualitatively worse than those of the pipelines without the depth-derived features. The segmentations close less narrowly to the contours of objects and large areas of false positives are predicted around pole-like objects in general.

Network: RGB		Predicted			
		light pole	ground plane	sky	other
Actual	light pole	410883 **0.53**	45498 **0.06**	22527 **0.03**	289112 **0.38**
	ground plane	311816 **0.01**	27065883 **0.94**	142567 **0.00**	1373085 **0.05**
	sky	1071574 **0.01**	10905 **0.00**	81511560 **0.92**	5892594 **0.07**
	other	2285452 **0.04**	3937512 **0.07**	4064899 **0.07**	44709733 **0.81**

Network: RGBAzEl		Predicted			
		light pole	ground plane	sky	other
Actual	light pole	455617 **0.60**	41276 **0.05**	26123 **0.03**	238036 **0.31**
	ground plane	837280 **0.03**	26327735 **0.92**	53309 **0.00**	1458610 **0.05**
	sky	2861586 **0.03**	12075 **0.00**	76803316 **0.87**	8357512 **0.09**
	other	7174984 **0.13**	4453465 **0.08**	6174717 **0.11**	37524359 **0.69**

Network: RGBHDA		Predicted			
		light pole	ground plane	sky	other
Actual	light pole	419108 **0.55**	51354 **0.07**	29951 **0.04**	262306 **0.34**
	ground plane	638015 **0.02**	26583876 **0.92**	99939 **0.00**	1486275 **0.05**
	sky	2356820 **0.03**	8642 **0.00**	78246935 **0.87**	7616521 **0.09**
	other	5377781 **0.10**	4501222 **0.08**	6154083 **0.11**	38967172 **0.71**

FIGURE 5.7: These are the confusion matrices of the segmentation performance of the RGB network. Since the quality of the segmentations from the pipelines using depth-derived features should theoretically never be worse than those of the pipeline using RGB-images only, the confusion matrix of the RGB-network is repeated here for comparison. While the RGBAzEl-network and the RGBHDA-network have higher true positive rates for the "light pole" class, and fewer pixels from the "light pole" class that are mislabeled as the "other" class, more pixels from the other three classes have been incorrectly labeled.

5.4 World positioning performance of depth-derived features

In this section we evaluate the world positioning performance of our single-view pipeline based on the RGBAzEl-network and RGBHDA-network. The pipeline will be referred to as the RGBAzel-pipeline and the RGBHDA-pipeline respectively. Again, the test consists of 1000 images that together contain 725 unique light poles. The world positioning performance of these pipelines is shown in Figure 5.8.

As discussed in Section 5.3, the segmentations resulting from these pipelines are worse in quality than the segmentations from the other pipelines. As the segmentation-network includes more false positives for the light pole class in the segmentations, more false positives are introduced in the world positioning. It is therefore unsurprising that the world positioning performances of the RGBAzEl-pipeline and the RGBHDA-pipeline are also worse than that of the RGB-pipeline. Even though the segmentations of the RGBAzEl-network and the RGBHDA-network contain more true positives for the "light pole" class than the RGB-network, the recall rate in world positioning is lower. This is likely caused by the additional false-positives, introducing inaccuracies when they are included in a cluster. This effect is worse for the RGBAzEl-pipeline than for the RGBHDA-pipeline. The performance of the RGBHDA-pipeline is slightly better than that of the RGBAzEl pipeline. It has both a higher AUC score and reaches a higher recall score. However, as concluded in Section 5.3, the performance is currently limited by the architecture of the segmentation-network and not by the features themselves. No conclusions can therefore be drawn regarding the usability of the depth-derived features. The performance might still improve with a more complex segmentation-network architecture, and that with the improved segmentations, the world-positioning also improves.

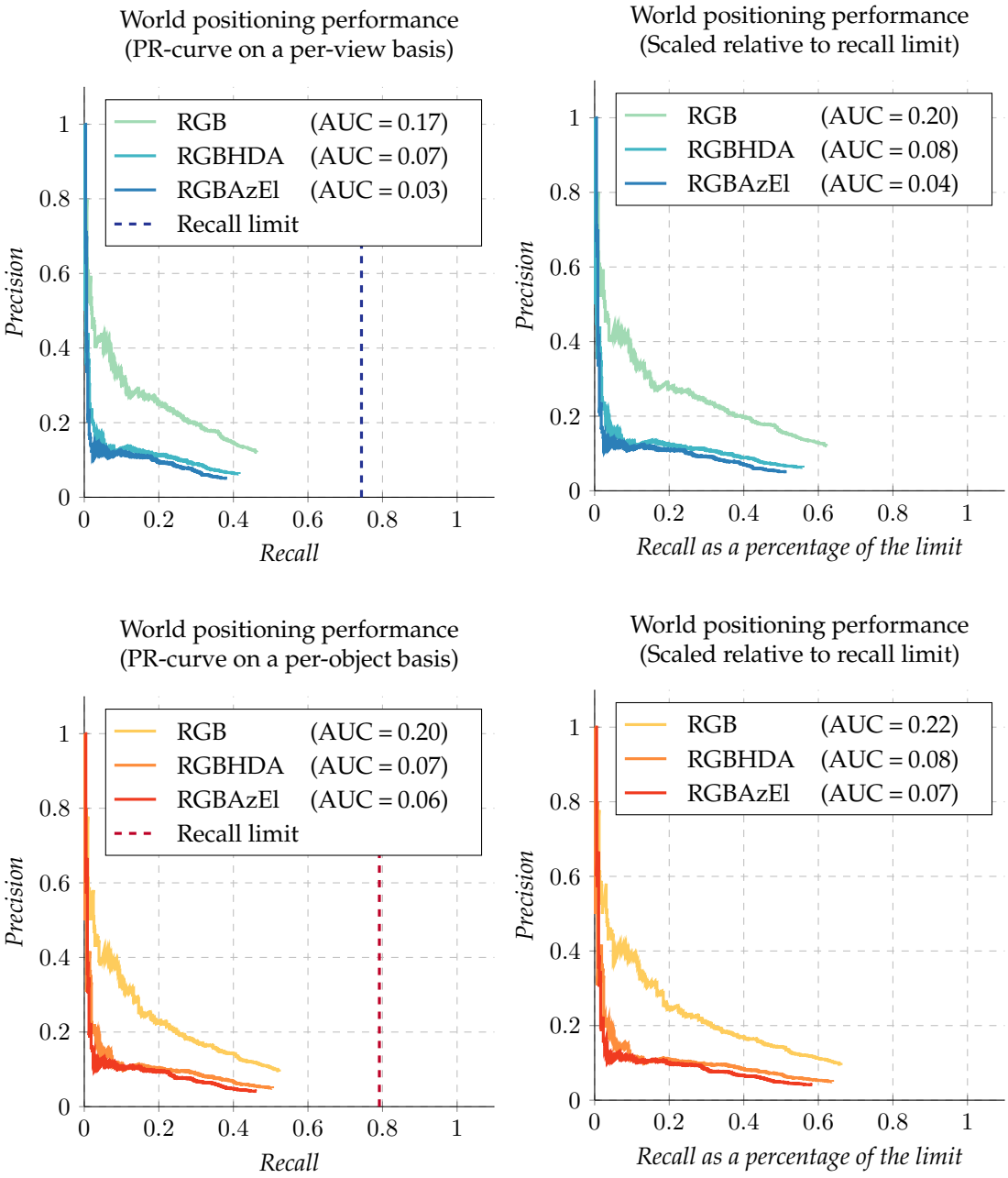


FIGURE 5.8: The world positioning performance of the RGBHDA-pipeline and the RGBAzEl pipeline. Since the quality of the corresponding segmentations should theoretically never be worse than that of the segmentations of the RGB-pipeline, the word-positioning performance of the RGB-pipeline is repeated in these graphs for comparison. Unsurprisingly, worse segmentations result in worse world-positioning performance. Although the performance of the RGBHDA-pipeline seems better than that of the RGBAzEl-pipeline, no definitive conclusion can be made with regard to usability of the features, as the performance is limited by the current segmentation-network architecture.

5.5 Segmentation performance of the multi-view pipeline

In this section the segmentations from the multi-view pipelines are evaluated. Two variants of the multi-view pipeline are considered that both operate on RGB-images. The first pipeline aggregates segmentations with an aggregation-network and is referred to as the MV-Agg pipeline. In the second pipeline segmentations are combined by simple per-class probability summation. The second pipeline is referred to as the MV-Sum pipeline. Figure 5.9 shows some segmentations produced with both pipelines, and the differences in the corresponding per-class probabilities that the aggregation methods produce. Figure 5.10 shows a few more segmentation examples, without the corresponding probabilities.

It seems that the segmentations produced by the MV-RGB pipeline still contain many false-positives for the "light pole" class. However, the segmentations have the interesting property that light poles in the center of the image are segmented quite generously, with many false positives clustering around the true positives within the images. Relatively few false positives occur in other areas of the images. The segmentation quality of the MV-Sum pipeline is even better, with very accurate light pole detection where the boundaries of the prediction narrowly fit the boundaries of the object in the ground truth. Relatively few false positives for the "light pole" class are predicted by the MV-Sum pipeline.
The segmentations are evaluated quantitatively in Figure 5.11. Because both the MV-Agg and the MV-Sum networks use RGB-images for segmentation, the confusion matrix for the RGB-network is repeated for comparison. Even though the segmentations look quite good qualitatively, the performance on a pixel-level is less conclusive. While the MV-Agg and has a slightly higher true positive rate for the "light pole" class, the MV-Sum has a lower true positive rate, compared to the RGB-network. The MV-Agg network has a lower confusion rate of pixels from the "light pole" class being incorrectly labeled as "other" pixels. However, the MV-Agg network has more pixels of the "other" class that are confused with the "light pole" and "ground plane" classes. This illustrates that the trade-off between the "light pole" and "other" classes is just slightly tilted in the multi-view networks. The actual segmentation quality however has not decisively improved.

The performance of the aggregation-network could have been limited through the neural network architecture and the data it is provided with. As the segmentation performance ha snot significantly improved, the network apparently is not able to correct inaccuracies in the predictions through supplemental information. However, this would have been very difficult given the data provided. Since the aggregation-network operates on probabilities without the original data, all color information and features extracted by the segmentation-network are lost when the probabilities are passed to the aggregation-network. The aggregation-network cannot possibly reconstruct this information from the probabilities alone. It might be better to provide color information and depth information to the aggregation-network, in addition to the probabilities. Even better would be to merge the single-view pipeline with the segmentation-network and the aggregation-network into a single large neural network, where all features are shared and reprojection and refocusing are implemented as network layers. This network would however be harder to train due to its increased depth.

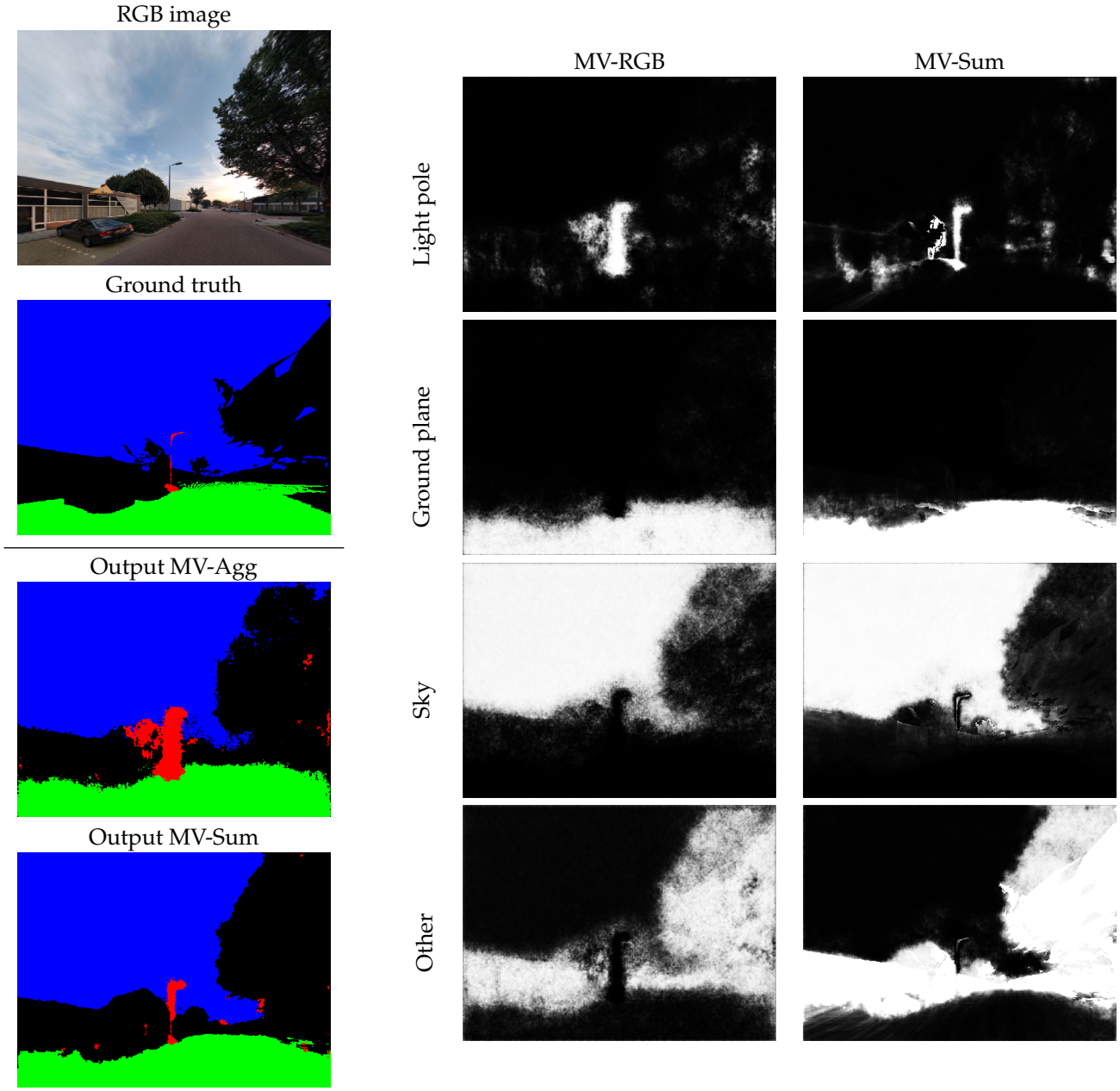


FIGURE 5.9: These are examples of segmentations produced by the MV-RGB pipeline and the MV-Sum pipeline. The segmentation from the MV-RGB pipeline contains many false positives for the "light pole" class around the actual object. However, it should be noted that very few false positives for the "light pole" class occur in other areas of the image. The tree on the right side of the image is even better represented in the segmentation than in the ground truth. The segmentation from the MV-Sum pipeline is even cleaner. Very few false positives a present overall and the boundary of the light pole in the segmentation fits the object in the ground truth quite narrowly. We can see that same patterns apply for the corresponding probabilities. The boundaries between classes are much sharper in the probabilities from the MV-Sum pipeline.

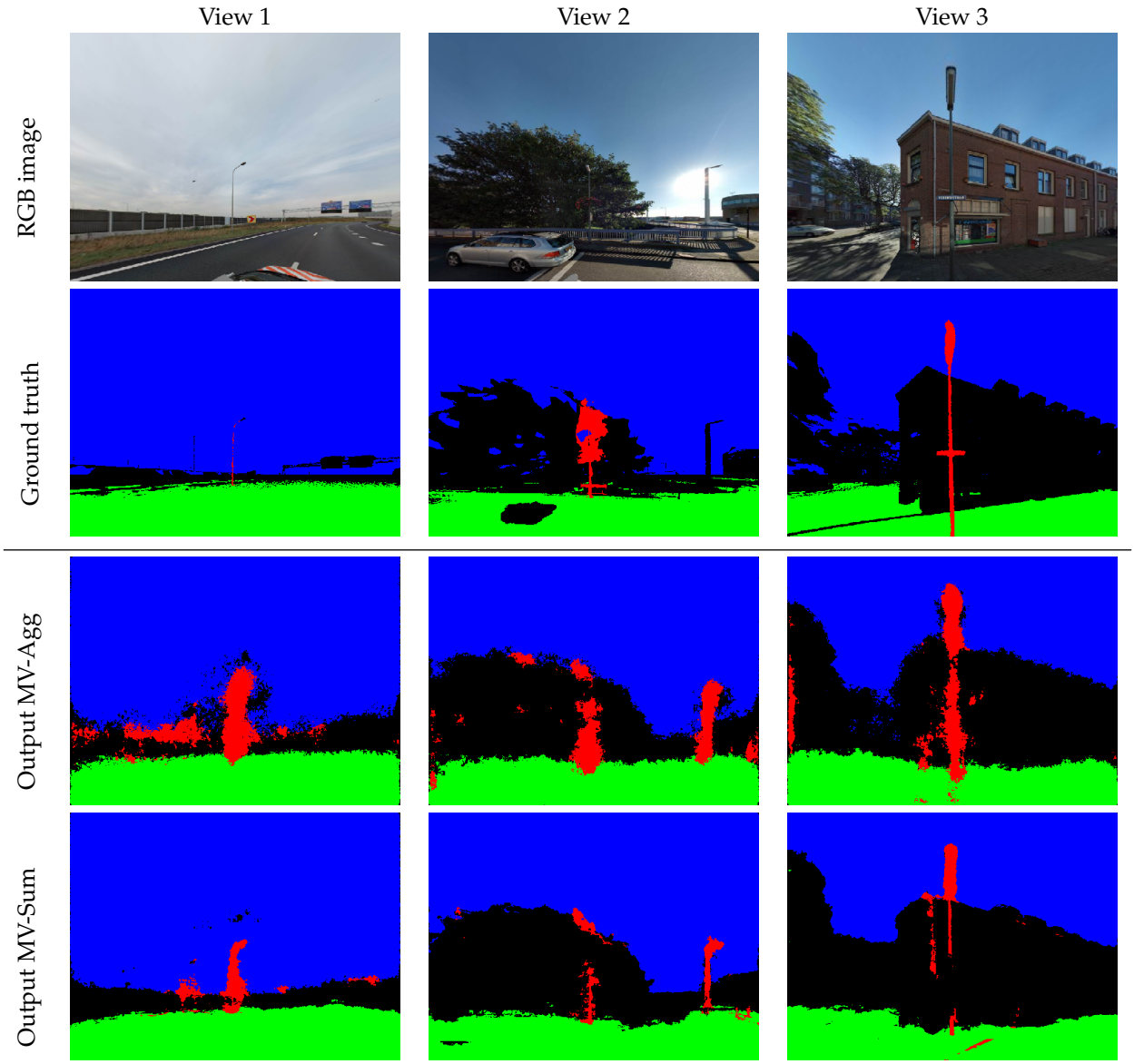


FIGURE 5.10: These are more examples of segmentations produced by the MV-RGB pipeline and the MV-Sum pipeline. The MV-RGB pipeline has many false positives for the "light pole" class. It is clear that the segmentations from the MV-Sum pipeline are much cleaner, with less noise around the boundaries between classes and details captured more accurately. Even very thin structures can correctly be captured by the RGB-Sum pipeline.

Network: RGB

		Predicted			
		light pole	ground plane	sky	other
Actual	light pole	410883 **0.53**	45498 **0.06**	22527 **0.03**	289112 **0.38**
	ground plane	311816 **0.01**	27065883 **0.94**	142567 **0.00**	1373085 **0.05**
	sky	1071574 **0.01**	10905 **0.00**	81511560 **0.92**	5892594 **0.07**
	other	2285452 **0.04**	3937512 **0.07**	4064899 **0.07**	44709733 **0.81**

Network: MV-Agg

		Predicted			
		light pole	ground plane	sky	other
Actual	light pole	398946 **0.54**	68812 **0.09**	17939 **0.02**	247196 **0.34**
	ground plane	472273 **0.02**	26597842 **0.93**	57557 **0.00**	764950 **0.05**
	sky	2248700 **0.02**	2987 **0.00**	78873593 **0.89**	7341259 **0.08**
	other	3771008 **0.07**	4329292 **0.08**	3606200 **0.07**	43314424 **0.79**

Network: MV-Sum

		Predicted			
		light pole	ground plane	sky	other
Actual	light pole	355806 **0.49**	65917 **0.09**	12912 **0.02**	298258 **0.40**
	ground plane	213453 **0.01**	27169377 **0.95**	51956 **0.00**	1144858 **0.04**
	sky	1088193 **0.01**	9143 **0.00**	81333771 **0.92**	6035432 **0.07**
	other	1435641 **0.03**	3686046 **0.07**	2152892 **0.04**	47746345 **0.87**

FIGURE 5.11: These are the confusion matrices of the segmentation performance of the multi-view networks. Because the multi-view pipelines operate on RGB-images the confusion matrix of the single-view RGB-pipeline is repeated here for comparison. Although the segmentations visually look good, there are many discrepancies between the segmentations and the ground truth. There are multiple reasons for this. First, objects in the center of the images are segmented well because those parts of the image are best supplemented by the additional recording locations through reprojection due to high overlap. Because only objects in the center of images are segmented well, the other areas of the segmentation have more misclassification, resulting in a worse quality overall. Secondly, the resulting segmentations are in some cases qualitatively better than the ground truth. This means that when a segmentation is correct, and the ground truth is not, the network is still punished for it during training.

5.6 World positioning performance of the multi-view pipeline

In this section we evaluate the world positioning performance of the multi-view pipelines. The pipeline using an aggregation network is referred to as the MV-Agg pipeline, while the pipeline that performs simple probability summation is referred to as the MV-Sum pipeline. Again, the test set consists of 1000 images, however, because multiple images are refocused on the same position, the set consist of only 245 light poles. The corresponding PR-curves are shown in Figure 5.12.

In the multi-view pipeline, all positions predicted by the single-view pipeline are further investigated by first refocusing the images at the positions, and then aggregating the information from multiple nearby recording locations through reprojection. In doing so, a new set of images is created with slightly different characteristics from the original images. To measure how refocusing impacts the performance of the pipeline, new ground truth segmentations are generated for the refocused images, and used to calculate the new recall limits achievable by the pipeline. The recall limit has moved from 74% to 79% on a per-view basis and has moved from 79% to 83% on a per-object basis, illustrating that refocusing at every cluster position predicted by the single-view pipeline does indeed produce images with better object visibility, resulting in improved world-positioning recall performance. However, since more images are introduced that are not segmented perfectly, more false positives are also introduced, resulting in a lower precision.

As established in Section 5.5, the segmentations for these pipelines are not very accurate but have some desirable properties whose usefulness is confirmed in the PR-curves. While many false negatives for the "light pole" class result in missing objects during positioning on a per-view basis, the missing objects are generally found within other views, resulting in a higher recall score both on a per-view and on a per-object basis.
While the segmentations from the RGB-Sum pipeline are slightly worse with respect to the "light pole" class overall, the recall rate is slightly higher than that of the RGB-Agg pipeline on a per-view basis. However, when considering all images, the MV-Agg can compensate better for missed light poles than the MV-Sum pipeline, resulting in a higher recall rate for the MV-Agg pipeline on a per-object basis. The MV-Agg reaches 51% recall rate on a per-view basis, which is 64% of the limit, with a precision of 9%. On a per-object basis the MV-Agg pipeline achieves a 66% recall rate, which is 79% of the limit, with a precision of 3%.

These results indicate that on a per-object basis, approximately 17% recall could still be achieved through better segmentations, while another 17% recall could be obtained through other pipeline changes such as improved clustering and better ground truth.

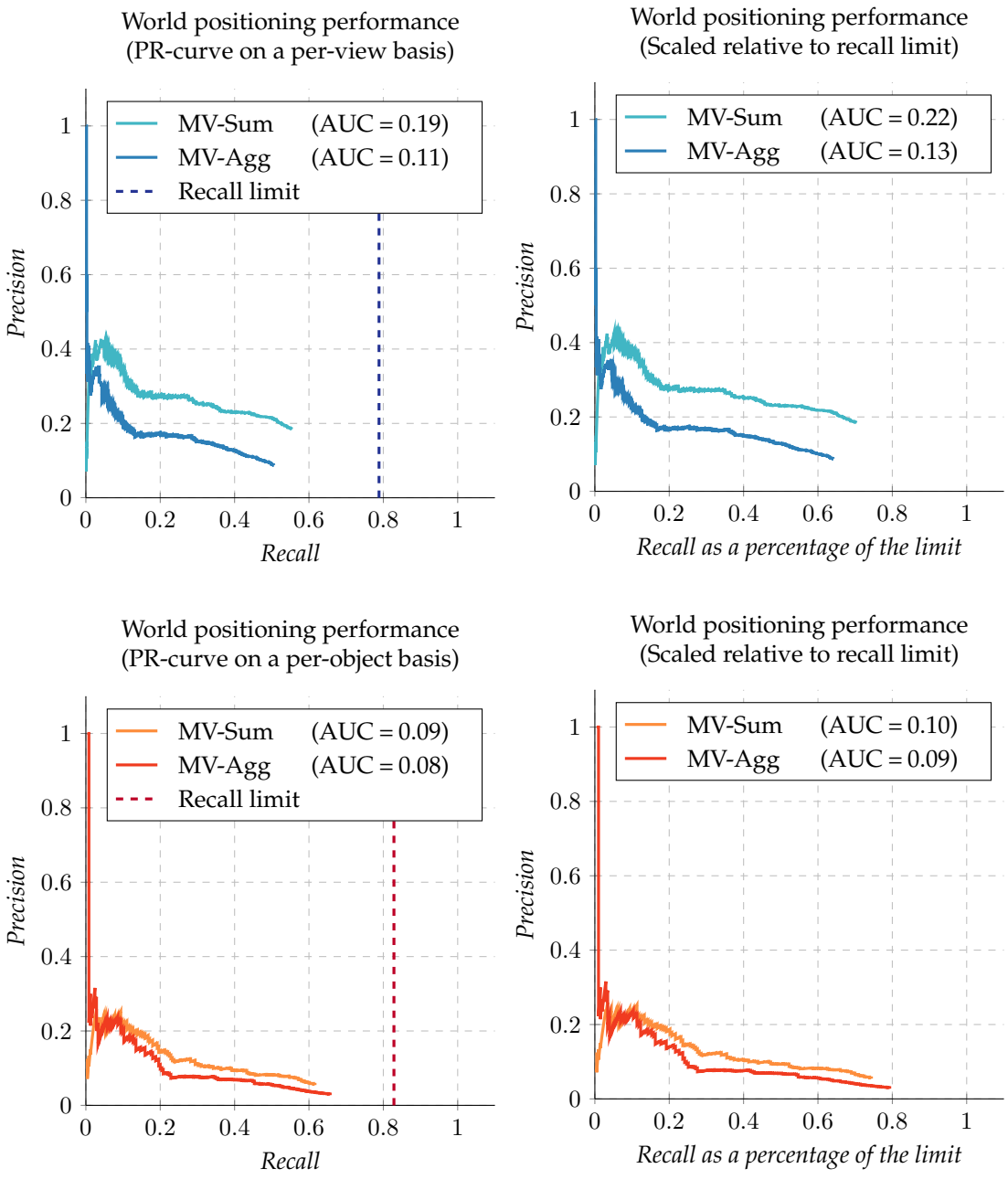


FIGURE 5.12: The world positioning performance of the multi-view pipelines. While the performance on a per-view basis seem to indicate that the MV-Sum pipeline performs better, the performance on a per-object shows that the MV-Agg pipeline reaches a higher recall. These results can be explained by the fact that especially the objects in the center of the images are segmented well because those parts of the image are best supplemented by the additional recording locations through reprojection. This effect is apparently stronger for the MV-Agg pipeline.

5.7 Overall comparison

A comparison has been made between pipelines operating RGB-images, D-images, and RGBD-images, where the depth-information is derived from Lidar point clouds. As Section 5.2 and Section 5.1 discuss, the depth information on its own is more useful for achieving a high light pole recall rate than color information alone. The combination of both has been shown to work best. Since the data-representation affects the capabilities of a neural network to learn specific features, different depth-derived features are explored. As Section 5.3 and Section 5.4 discuss, the performance of the pipeline with the new features is however limited by the current segmentation-network architecture and performs worse than a pipeline without depth information.

As discussed in Section 5.2 the world positioning performance of the pipeline with ground truth segmentations provides an indication of the performance that can maximally be expected from the pipeline. The fact that the recall rate on a per-object basis is higher than at a per-view basis indicates that information from multiple views could be combined for improved segmentation.

A multi-view pipeline has been introduced which incorporates several techniques for improved segmentation. The multi-view pipeline reconsiders every position predicted by the single-view pipeline. From the panoramic images, an image patch is extracted and resampled to focus on the predicted position for optimal visibility. As discussed in Section 5.6 this already improves the recall rate of light poles. Information from multiple recording locations is then reprojected to a target camera pose so that correlation between images is then explicitly defined, and no longer has to be inferred. The reprojected images can then be combined into a single representation. As discussed in Section 5.5 the resulting segmentations are not necessarily better on a pixel level but do have some beneficial properties. The position that the image has been refocused at is segmented accurately while the surroundings are less accurately segmented. As a result, the recall on a per-view basis has improved slightly, and the recall on a per-object basis has improved considerably compared to the single-view pipeline.

To answer **sub-question 1** we conclude as follows. Our pipeline operating on the combination of color information and depth information outperforms pipelines that operate on information from only one of these sources. While, theoretically, alternative depth-derived features could result in an even better performance, the current segmentation-network architecture has limited capabilities to learn the correct features. The optimal data representation for the current pipeline therefore consists of RGBD-images, where the depth values directly correspond to the distance from the recording location to the mesh reconstructed from the Lidar point clouds.

To answer **sub-question 2** we conclude as follows. By reprojecting information from multiple recording locations to a target camera pose,the information can be combined into a single representation. Reprojection can be achieved by utilizing the depth information derived form Lidar point clouds. While these representations do not result in better segmentations at a pixel-level, the objects in the center of the image are generally segmented more accurately, resulting in higher recall rates.

Chapter 6

Conclusion

In this chapter we conclude this work. A summary is provided in Section 6.1. Finally, Section 6.2 discusses future work.

6.1 Summary

We discussed how color information from panoramic images and depth information derived from Lidar point clouds could be combined for improved street furniture detection. In contrast to many related methods, our method deals with real world environments consisting of many complex objects. A novel method is proposed that enables multi-view deep convolutional neural networks to combine data from recording locations that do not have fixed relative positions. The focus is on the recall rate of light poles.

A single-view pipeline is introduced which embeds a deep convolutional neural network for segmentation, using both images and Lidar point clouds. The resulting segmentations are then reconstructed as labeled point clouds, after which a three-dimensional clustering method extracts the world positions for the segmented objects. Results show that a network trained on depth-images outperforms a network trained on regular images. A network trained on representation that combines both types of images creates the highest quality segmentations. It is also shown that better segmentations generally result in better world position estimates. On a per-object basis the pipeline that combines color and depth information achieves a 62% recall rate, which is 78% of the limit, with a precision of 13%.

Several depth-derived features are explored to see whether these are more informative to our deep neural networks than depth information alone. Results indicate that with the current network architecture, the additional data confuses the network, with worse segmentations as a result. Due to network architecture limitations, the network does not learn to ignore irrelevant data and both segmentation and world positioning performance is worse than that of a network without depth information.

Finally, a multi-view pipeline is introduced which combines the segmentations of several recording locations into a single better-informed segmentation through image reprojection. While the quality of the resulting segmentations is not necessarily better, due to a double check on every cluster found by the single-view pipeline and segmentations which are more accurate towards the position they have been refocused at, the recall is increased. The multi-view pipeline which uses an additional neural network for segmentation-aggregation achieves a 66% recall rate, which is 79% of the limit, with a precision of 3%.

6.2 Future work

The main limitations of the pipeline are caused by the limitations of the SegNet architecture for segmentation, the data passed to the aggregation network, the clustering step and the quality of the ground truth.

To improve segmentation quality multiple alterations to the architecture could be made. There have been significant improvements in network architecture after the introduction of SegNet and during the creation of this work. In addition to improvements such as residual connections and inception modules as discussed in Section 5.3, other significant improvements have been made to make deep neural networks in more powerful in general. Inspired by residual networks, densely connected networks were proposed by Huang et al. that connect each layer to every other layer in a feed forward fashion, alleviating the vanishing gradient problem, strengthening feature propagation en encouraging feature reuse [56]. In contrast to residual networks, the additional connections in dense networks are not summed but concatenated, resulting in more parameters. Another recent development in deep convolutional neural networks was the incorporation of attention modules which learn attention-aware features [57] as proposed by Wang et al. An attention module consists of an encoder-decoder architecture that learns in which areas of an image a feature is relevant. This means that internally, the network learns a mask for each feature so that the corresponding network output is scaled during inference, and the corresponding gradients are scaled during training. Recently, residual attention networks have surpassed state-of-the-art image classification methods. We expect that incorporating residual connections, dense connections, inception modules and attention modules in a new pipeline could further improve the performance of our pipeline. Finally, he performance of the aggregation is currently limited by the information it is provided with. Since the aggregation-network only receives probability information, all features extracted by the segmentation-network are lost when information is passed to the aggregation-network. The best solution for this would be to merge the segmentation and aggregation networks in a single larger network, where are features are shared. Such a network would however contain more parameters and be would be harder to train.

Besides network architecture there are other aspects that can be improved. The performance of the ground truth is currently not optimal, with a recall limit of 0.79 on a per-view basis, and a recall limit of 0.83 on a per-object basis. Improving the ground truth could therefore make a significant improvement in performance. The ground truth for the "light pole" class incorrectly includes some parts of nearby trees. These points are however not geometrically attached to the light pole. Additionally, some light poles are occluded. While only ground truth with at least 100 light pole pixels is accepted, these pixels are not guaranteed to belong to a single geometrical cluster. Therefore, an additional clustering step at the time of ground truth creation could be used to remove the irrelevant clusters. It could be further explored what the exact number of occluded light poles is, and how both recall and precision change if smaller clusters are accepted. To minimize confusion between the "light pole" class and the "other" class, buildings and trees could be added to the ground truth as new classes. How these object could be annotated automatically, however, is not obvious. Alternatively, the ground truth could be improved through manual inspection. If the quality of the ground truth is better, than the segmentation-networks will automatically learn better features.

There are also a few other metric parameters that can still be further explored to evaluate our pipeline. For example the acceptance radius of 1.5 meters could be changed to see how this affects performance. Additionally, the number of neighboring recording locations incorporated in the multi-view pipeline could be increased to 4 or 5 to see if this has benefit. It is however expected that the information from these additional recording locations decreases in importance as the distance to the original recording location increases, because the images will have smaller overlap.

The pipeline currently contains many inaccuracies introduced by point cloud positioning, refocusing, reprojection and interpolation. Although the effect of each of these steps is expected to be small, the combination of all these effects could be significant. To deal with images that do not have the correct resolution, the network could operate on image-pyramids, consisting of differently scaled versions of the same image. Instead of regular linear interpolation for colors and nearest neighbor interpolation for depth values improved methods could be used. Interpolating depth values might be more accurate when the nearest depth value is chosen, instead of the depth value of the nearest pixel. Even better, instead of interpolation over pixels, depth values could be resampled from the original Lidar point cloud reconstructed as a mesh.

The pipelines have been illustrated with light poles as the main class of interest. The clustering within the segmentation-derived point clouds could work on any type of object that consists of a single geometrical component as long as concave protrusions are taken into account. The method will be productized by Cyclomedia to obtain data for remote inspection and inventory purposes of street furniture. A next step would therefore be to see how these pipelines perform when a different main class of interest such as traffic lights, traffic signs, garbage bins of road markings is chosen. The only thing that is necessary to train new networks is the availability of a new dataset and ground truth. However the ability of the networks to generalize to other classes is not obvious. When similar classes are introduced to the model, it could result in more misclassification. For example, if a second class of pole-like objects is introduced, that is different from light poles, the network might suffer form confusion. Additionally, in the current pipeline it is assumed that instances of the same class never touch. However, if an object is misclassified, the probability of it connecting to an object with the same label increases, and clustering in the segmentation-derive point clouds might not be enough. It might be beneficial at that point to perform instance level segmentation instead of regular semantic segmentation to distinguish between objects of the same class.

Bibliography

[1] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity", *The bulletin of mathematical biophysics*, vol. 5, no. 4, pp. 115–133, 1943.

[2] D. O. Hebb, *The organization of behavior: A neuropsychological theory*. Psychology Press, 2005.

[3] S. Lowel and W. Singer, "Selection of intrinsic horizontal connections in the visual cortex by correlated neuronal activity", *Science*, vol. 255, no. 5041, p. 209, 1992.

[4] F. Rosenblatt, "The perceptron: A probabilistic model for information storage and organization in the brain.", *Psychological review*, vol. 65, no. 6, p. 386, 1958.

[5] D. H. Hubel and T. N. Wiesel, "Receptive fields, binocular interaction and functional architecture in the cat's visual cortex", *The Journal of physiology*, vol. 160, no. 1, pp. 106–154, 1962.

[6] K. Fukushima and S. Miyake, "Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition", in *Competition and cooperation in neural nets*, Springer, 1982, pp. 267–285.

[7] H. Lee, R. Grosse, R. Ranganath, and A. Y. Ng, "Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations", in *Proceedings of the 26th annual international conference on machine learning*, ACM, 2009, pp. 609–616.

[8] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition", *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[9] C. F. Cadieu, H. Hong, D. L. Yamins, N. Pinto, D. Ardila, E. A. Solomon, N. J. Majaj, and J. J. DiCarlo, "Deep neural networks rival the representation of primate it cortex for core visual object recognition", *PLoS Comput Biol*, vol. 10, no. 12, e1003963, 2014.

[10] J. R. Uijlings, K. E. Van De Sande, T. Gevers, and A. W. Smeulders, "Selective search for object recognition", *International journal of computer vision*, vol. 104, no. 2, pp. 154–171, 2013.

[11] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation", in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 580–587.

[12] R. Girshick, "Fast r-cnn", in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 1440–1448.

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

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

[15] J. Dai, K. He, Y. Li, S. Ren, and J. Sun, "Instance-sensitive fully convolutional networks", in *European Conference on Computer Vision*, Springer, 2016, pp. 534–549.

[16] Y. Li, K. He, J. Sun, *et al.*, "R-fcn: Object detection via region-based fully convolutional networks", in *Advances in Neural Information Processing Systems*, 2016, pp. 379–387.

[17] Y. Li, H. Qi, J. Dai, X. Ji, and Y. Wei, "Fully convolutional instance-aware semantic segmentation", *ArXiv preprint arXiv:1611.07709*, 2016.

[18] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman, "Return of the devil in the details: Delving deep into convolutional nets", *ArXiv preprint arXiv:1405.3531*, 2014.

[19] S. Gupta, R. Girshick, P. Arbeláez, and J. Malik, "Learning rich features from rgb-d images for object detection and segmentation", in *European Conference on Computer Vision*, Springer, 2014, pp. 345–360.

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

[21] Y. LeCun, F. J. Huang, and L. Bottou, "Learning methods for generic object recognition with invariance to pose and lighting", in *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, IEEE, vol. 2, 2004, pp. II–104.

[22] E. Johns, S. Leutenegger, and A. J. Davison, "Pairwise decomposition of image sequences for active multi-view recognition", in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 3813–3822.

[23] B. Shi, S. Bai, Z. Zhou, and X. Bai, "Deeppano: Deep panoramic representation for 3-d shape recognition", *IEEE Signal Processing Letters*, vol. 22, no. 12, pp. 2339–2343, 2015.

[24] A. Sinha, J. Bai, and K. Ramani, "Deep learning 3d shape surfaces using geometry images", in *European Conference on Computer Vision*, Springer, 2016, pp. 223–240.

[25] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao, "3d shapenets: A deep representation for volumetric shapes", in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 1912–1920.

[26] D. Maturana and S. Scherer, "Voxnet: A 3d convolutional neural network for real-time object recognition", in *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, IEEE, 2015, pp. 922–928.

[27] V. Hegde and R. Zadeh, "Fusionnet: 3d object classification using multiple data representations", *ArXiv preprint arXiv:1607.05695*, 2016.

[28] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions", in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 1–9.

[29] C. R. Qi, H. Su, M. Nießner, A. Dai, M. Yan, and L. J. Guibas, "Volumetric and multi-view cnns for object classification on 3d data", in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 5648–5656.

[30] G. Riegler, A. O. Ulusoys, and A. Geiger, "Octnet: Learning deep 3d representations at high resolutions", *ArXiv preprint arXiv:1611.05009*, 2016.

[31] A. Brock, T. Lim, J. Ritchie, and N. Weston, "Generative and discriminative voxel modeling with convolutional neural networks", *ArXiv preprint arXiv:1608.04236*, 2016.

[32] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition", in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.

[33] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning", *ArXiv preprint arXiv:1602.07261*, 2016.

[34] D. Boscaini, J. Masci, E. Rodolà, and M. Bronstein, "Learning shape correspondence with anisotropic convolutional neural networks", in *Advances in Neural Information Processing Systems*, 2016, pp. 3189–3197.

[35] B. Lévy and H. R. Zhang, "Spectral mesh processing", in *ACM SIGGRAPH 2010 Courses*, ACM, 2010, p. 8.

[36] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, "Spectral networks and locally connected networks on graphs", *ArXiv preprint arXiv:1312.6203*, 2013.

[37] M. Henaff, J. Bruna, and Y. LeCun, "Deep convolutional networks on graph-structured data", *ArXiv preprint arXiv:1506.05163*, 2015.

[38] S. Ravanbakhsh, J. Schneider, and B. Poczos, "Deep learning with sets and point clouds", *ArXiv preprint arXiv:1611.04500*, 2016.

[39] L. Yi, H. Su, X. Guo, and L. Guibas, "Syncspeccnn: Synchronized spectral cnn for 3d shape segmentation", *ArXiv preprint arXiv:1612.00606*, 2016.

[40] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation", *ArXiv preprint arXiv:1612.00593*, 2016.

[41] J. Masci, D. Boscaini, M. M. Bronstein, and P. Vandergheynst, "Shapenet: Convolutional neural networks on non-euclidean manifolds", *CoRR*, vol. abs/1501.06297, 2015.

[42] I. Kokkinos, M. M. Bronstein, R. Litman, and A. M. Bronstein, "Intrinsic shape context descriptors for deformable shapes", in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, IEEE, 2012, pp. 159–166.

[43] R. B. Rusu, "Semantic 3d object maps for everyday manipulation in human living environments", *KI-Künstliche Intelligenz*, vol. 24, no. 4, pp. 345–348, 2010.

[44] V. Badrinarayanan, A. Kendall, and R. Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for image segmentation", *ArXiv preprint arXiv:1511.00561*, 2015.

[45] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition", *ArXiv preprint arXiv:1409.1556*, 2014.

[46] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, *et al.*, "Imagenet large scale visual recognition challenge", *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.

[47] G. J. Brostow, J. Fauqueur, and R. Cipolla, "Semantic object classes in video: A high-definition ground truth database", *Pattern Recognition Letters*, vol. 30, no. 2, pp. 88–97, 2009.

[48] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding", *ArXiv preprint arXiv:1408.5093*, 2014.

[49] R. B. Rusu and S. Cousins, "3d is here: Point cloud library (pcl)", in *Robotics and automation (ICRA), 2011 IEEE International Conference on*, IEEE, 2011, pp. 1–4.

[50] G. Bradski, "The opencv library.", *Dr. Dobb's Journal: Software Tools for the Professional Programmer*, vol. 25, no. 11, pp. 120–123, 2000.

[51] M. D. Zeiler, "Adadelta: An adaptive learning rate method", *ArXiv preprint arXiv:1212.5701*, 2012.

[52] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization", *Journal of Machine Learning Research*, vol. 12, no. Jul, pp. 2121–2159, 2011.

[53] D. Kingma and J. Ba, "Adam: A method for stochastic optimization", *ArXiv preprint arXiv:1412.6980*, 2014.

[54] I. Sutskever, J. Martens, G. Dahl, and G. Hinton, "On the importance of initialization and momentum in deep learning", in *International conference on machine learning*, 2013, pp. 1139–1147.

[55] G Hinton, N Srivastava, and K Swersky, "Rmsprop: Divide the gradient by a running average of its recent magnitude", *Neural networks for machine learning, Coursera lecture 6e*, 2012.

[56] G. Huang, Z. Liu, K. Q. Weinberger, and L. van der Maaten, "Densely connected convolutional networks", *ArXiv preprint arXiv:1608.06993*, 2016.

[57] F. Wang, M. Jiang, C. Qian, S. Yang, C. Li, H. Zhang, X. Wang, and X. Tang, "Residual attention network for image classification", *ArXiv preprint arXiv:1704.06904*, 2017.