Geodesic pixel neighborhoods for 2D and 3D scene understanding

Vladimir Haltakov[a,b], Christian Unger[a,b], Slobodan Ilic[b]

[a] *BMW Group, Knorrstr. 147, 80807 Munich, Germany*
[b] *Technical University Munich, Boltzmannstr. 3, 85748 Garching, Germany*

Abstract

Scene understanding is an important class of computer vision problems that is an enabler for a wide variety of applications such as advanced driver assistance systems, autonomous vehicles or mobile assistive robots. Semantic segmentation is one of the common ways to address this problem. Unlike the more standard approaches based on a probabilistic graphical model, in this paper we present a two stage classification framework based on the concept of *pixel neighborhoods*. In the first stage, every pixel is classified based on its appearance. The output of the first classifier in a specific region around every pixel, which we call the *pixel neighborhood*, is summarized by a novel voting histogram feature and given as input to a second classifier. We show how to define the *pixel neighborhood* by using the geodesic distance in a way that it is able to capture both local image context as well as more global object relations.

We perform a quantitative and qualitative evaluation on six well-known and challenging datasets and show that our model is able to natively handle both 2D and 3D data. We compare our method to several baselines and multiple closely related methods and show state-of-the-art performance. We also present a real world application of our method in a system that automatically detects parking spaces from a moving vehicle in real time.

Keywords: semantic segmentation, geodesic distance, classification, context

Email addresses: vladimir.haltakov@bmw.de (Vladimir Haltakov), christian.unger@bmw.de (Christian Unger), slobodan.ilic@in.tum.de (Slobodan Ilic)

1. Introduction

Many assistive systems and applications like mobile robots, advanced driver assistance systems and autonomous vehicles need to interpret the environment they operate in. Such systems are usually equipped with one or more cameras as their main sensor and, therefore, strongly rely on computer vision methods for the perception of the environment, which needs to be fast, robust and accurate.

Traditional object detection systems can usually detect objects that have a well-defined shape like vehicles, traffic signs, people and household objects. However, for a system to get a complete understanding of the scene it also needs to recognize other parts of the surrounding environment that do not always have clearly defined appearance like roads, sidewalks, buildings, floors and walls. Semantic segmentation is one of the fundamental computer vision problems that enables such environment interpretation and understanding from camera images that provides more context information about the environment than sparse object detection systems and can be of great benefit for many applications that need to operate in complex, uncontrolled environments.

In recent years, depth sensors like stereo cameras and the Microsoft Kinect have become increasingly accurate and affordable and they are now a standard part of assistive robots, autonomous vehicles or as part of driver assistance packages. Therefore, many modern assistive systems have access not only to 2D camera images, but also to depth data that offers richer information about the scene and can be used to greatly improve accuracy and robustness.

Semantic segmentation has been a very active research field in recent years and different works focus on different aspects of the problem: designing new and more discriminative features, building stronger and more general classifiers or modeling the relations between different parts of the image. In this paper, we focus on the latter. We build our method around the concept of the *pixel neighborhood*, which represents a set of pixels related in some way to the pixel of interest. We present a two-stage classification framework in which the output of a unary classifier based on image features is summarized by a novel voting

histogram feature and given as input to a second classifier. We show that this concept allows us to model both local and global context relations.

In this paper, we explore different ways to define neighborhoods and introduce a novel neighborhood type based on the geodesic distance transform. We also show that this concept can naturally be applied to both 2D and 3D images and that the combination of both modalities allows us to deal with problems, which are difficult to solve by either one separately. We compare our method to multiple state-of-the art approaches and show similar or increased performance, while keeping the runtime of our method low.

Our main contributions are the two-stage segmentation framework based on local and global geodesic neighborhoods and the voting histogram features that provide a compact representation of context information. Furthermore, we show how to combine 2D and 3D data in the unary features and in the geodesic neighborhood by a novel use of the 2D Walsh-Hadamard transform and a combined distance measure. Additionally, we show how to use the *geodesic neighborhood* to efficiently perform smoothing of the final segmentation. The concept of the *geodesic neighborhood* was initially introduced in [1].

2. Related work

There is a huge amount of work on semantic segmentation, but here we focus on the methods that have similar goals or use similar approaches. We cluster the related works in three main categories: random fields based methods, classification based methods and methods that explicitly handle 3D scenes.

2.1. Random fields based methods

One of the most popular approaches to multi-class image labeling is to use a Markov Random Field (MRF) or a Conditional Random Field (CRF) [2] as the underlying model and an energy minimization framework for inference. The interaction between different parts of the image are modeled as potential functions defined over one, two or more pixels. The relatively simple pairwise

models already provide good results [3, 4] even though they explicitly model relations only between two adjacent pixels. Due to the simple form of the resulting energy function, such models allow for fast training and inference [5].

Modeling more complex pixel or region interactions requires augmenting the pairwise models with higher-order potential functions. Although, more difficult to optimize efficiently, they lead to improved segmentation performance [6, 7]. The Robust P^n model of [7] presents a special type of higher-order potentials based on large pixel segments generated by unsupervised segmentation. While the resulting energy function can be efficiently optimized by graph-cuts based methods, the parameter estimation in the training phase is difficult in practice and requires exhaustive search on a validation dataset. Various other works extend the Robust P^n model to explicitly handle 3D information [8], to include information from object detectors [9] or to build region hierarchies [10, 11].

More complex potential functions enable random fields to better model object relations and to improve the segmentation performance, but this usually leads to an increased number of model parameters. This in turn makes parameter estimation and training more difficult and requires significant approximations in order to be tractable. Another problem with the standard CRF and MRF models is that the parameters, e.g. the weighting between the unary and the pairwise terms, depends strictly on the training dataset for which it is optimized and is fixed for all other input data. Therefore, the weighting may differ for the input data and may in general not be suitable. To overcome this problem, methods like the decision tree fields [12] and the regression tree fields [13] make the parameters dependent on the input image and use a trained tree model to find the best parameters for the given input. While powerful, these models introduce additional complexity on top of the already complex CRF models and are usually difficult and computationally expensive to train. An alternative approach is the inference machines method [14], which employs a classifier to learn the messages passed during belief propagation, used as the inference method for a CRF model. The authors of [15] present an extension of this idea specialized for 3D point clouds, which is able to learn spatial se-

mantic context from several source regions defined for each point. The source regions cover bigger parts of the space and are in some ways similar to our rays neighborhood, but they do not adapt to the input structure and are integrated differently into the model.

Instead of a CRF model, our method relies on a classification framework. All pixel and region relations are modeled implicitly by the classifier and are therefore naturally dependent on the input. Furthermore, the classifier handles both the training and the inference phase, while CRF models require different methods during training and evaluation. Overall, our approach is very simple and fast to train and use and has a low number of meta parameters.

2.2. Classification based methods

A different approach to the semantic segmentation problem is to use a classifier or a sequence of classifiers to directly predict the class of each pixel.

The semantic texton forests model [16] employs two random forest classifiers with the first one applied directly on features computed from the image. Local rectangular features computed on the output of the first classifier serve as an input to the second one enabling it to learn local context relations.

The auto-context method [17] trains a chain of classifiers where each one has access to both the image features and the predictions of the previous classifier at fixed positions around each pixel. This allows the classifiers to learn both local and global context relations, but since the look-up positions in the probability map of the previous classifier are fixed for each pixel, the learning method does not take into account the structure of the scene. By contrast, our proposed method is able to align well to object boundaries as shown in Section 4.2.

Similar to auto-context, the stacked hierarchical learning method of [18] employs a series of classifiers based on the concept of stacking [19], but in a hierarchical way. The image is segmented at different levels with different parameters such that on the top level there is only one segment, while on the bottom level the segments are in the form of superpixels. A classifier is trained for each level to predict the proportion of the labels in each segment.

Another approach using the auto-context idea are the Iterative Context Forests (ICF) of [20, 21]. They are based on the well-known Random Decision Forests, but in each level of the decision trees that class probability maps from the previous level are made available to the classifier in the form of additional feature maps. This allows the classifier to iteratively learn context by choosing features operating on the class probability maps. On one side, the ICF method is able to express much finer context relations then our model, by comparing regions at arbitrary positions in the image, while the classifier in our approach receives only the summarized information from the local neighborhoods and the rays of the global neighborhood. On the other side, the regions used by our method have a shape that aligns well to the object boundaries guided by the geodesic distance. The voting histogram features computed over those regions are a much more compact representation that still manages to capture well the context information for the region, enabling the neighborhood classifier to learn context relations between the pixels and their neighborhoods.

The entangled forest model [22] separately trains several decision trees as is usually done in a random forest classifiers, but at several levels of the tree, the predictions of the parent nodes are pooled together with the image features. The GeoF model [23] extends the entangled forests by employing geodesic smoothing of the predictions of the parent nodes. Even though the authors also employ the geodesic distance transform, their method differs in several ways from ours. While we use the geodesic distance to compute a set of pixels comprising the neighborhood of each pixel in the image, the GeoF model uses it to smooth big probabilistic regions and adapt them to the image gradients.

2.3. Models specialized for 3D scenes

Several methods in the literature explicitly make use of 3D information in order to improve the segmentation performance. The authors of [24] segment the image into regions based on stixels [25] computed form dense disparity maps, which are then classified into one of several semantic classes. An extension of this model, called Stixmantics [26], introduces a method for enforcing temporal

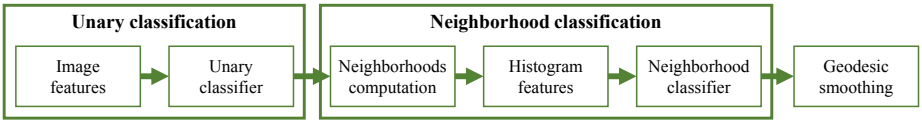


Figure 1: The pipeline of our two-stage classification method.

consistency and a CRF model to achieve spatial smoothness. While the stixel representation is well suited for road scenes, it can be used only when 3D information is available and only to recognize classes that stay on the ground plane and have 3D structure. Therefore, this method cannot be used to detect objects like traffic signs, traffic lights or lane markings. Furthermore, since the stixels are a hard constraint on the shape of the segments, if an error occurs while computing the stixel regions, it cannot be corrected at a later stage. In contrast, our method is applicable to all types of objects and also in cases where only 2D camera images are available. We use the neighborhoods only as soft constraints and the final decisions are taken on the pixel level.

The authors of [27] model the semantic label and the stereo disparity of each pixels jointly in a CRF framework and show that both tasks solved jointly benefit from each other. Another method [28] models the semantic class and the depth jointly, only using a mono camera image. The goal is to achieve a 3D normalization of the scene so that the classifier does not need to learn the appearance of the objects at different scales, but only at a certain canonical distance. In this paper, we focus only on the problem of semantic segmentation.

3. Method

The general pipeline of our two-stage classification method is illustrated in Fig. 1. We first compute simple image features from the input camera image (and depth map if available) and we use a standard multi-class classifier to get the probability distribution of every pixel over the possible classes. This process if fairly standard for many semantic segmentation methods. Next, we compute one or more neighborhoods for each pixel. Based on the neighborhoods and the predictions of the first classifier we compute a new type of features, called

voting histograms, which serve as the input of the second classifier. The resulting segmentation is then geodesically smoothed to get the final segmentation.

3.1. Neighborhood classification framework

Our goal is to model the conditional probability distribution $P(\mathbf{y}|\mathbf{x})$ of the pixel labels $\mathbf{y}$ given the input image $\mathbf{x}$. With y_i we denote the label variable for pixel i, which takes values from a predefined set of classes $\mathcal{L}$. Under the assumption that the labels of all pixels are independent of each other the probability distribution can be written as:

$$P(\mathbf{y}|\mathbf{x}) = \prod_i P(y_i|\mathbf{x}), \tag{1}$$

where $P(y_i|\mathbf{x})$ is the probability of pixel i taking the label y_i. The distribution for one pixel can be estimated by a standard multi-class classifier trained on multiple pixel samples and the corresponding ground truth labels. In practice, the classifier does not operate on the image pixels $\mathbf{x}$ directly, but on some features extracted from the image. We denote the feature vector computed for pixel i as $f_U(x_i)$. Training such classifier is a very common first step in, many segmentation methods, especially for CRF, which typically use similar classifiers to define the unary potentials [6, 7, 3, 11, 29]. We use this unary probability distribution as an input to our second classification stage.

The assumption that all pixel labels are independent of each other leads to simple factorization of the conditional probability distribution, but it rarely holds true in reality. Therefore, the segmentation resulting from the model above is usually very noisy and contains a lot of errors. To overcome this problem we make each label dependent on a set of labels that are related to it in a defined way. For a pixel i we call the set of pixels related to it *pixel neighborhood* and denote it as N_i. The choice of the neighborhood is of critical importance for the performance of the method. In Section 3.2 we analyze different ways to define the neighborhood and introduce a novel method based on geodesic distance.

Under the assumption that each label is independent of the other labels

given its neighborhood, the conditional probability distribution factorizes as:

$$P_N(\mathbf{y}|\mathbf{x}) = \prod_i P(y_i|\mathbf{x}, N_i). \tag{2}$$

We again model this probability distribution for pixel i as the output of a classifier operating on a feature vector $f_N(N_i, P(y_{N_i}|\mathbf{x}))$ computed from the unary probability distribution of the pixels in the neighborhood N_i. For the computation of this feature vector f_N we propose a new compact feature, called voting histogram, described in detail in Section 3.4.2.

3.2. Pixel neighborhoods

As described above, the neighborhood N_i of pixel i is defined as a set of pixels that are related in some defined way to the pixel of interest i. The choice of the neighborhood is important, because the neighborhood classifier determines the class of the pixel depending only on the pixels in its neighborhood. In this paper, we divide the possible neighborhoods in two variants: local and global neighborhoods. The local neighborhoods cover an area in the vicinity of the pixel i, while the global neighborhoods may include pixels throughout the whole image. Below we show several ways to define local and global neighborhoods and in Section 4.2 we compare their performance.

Euclidean neighborhood. The most intuitive way to define a local neighborhood is just to take all the pixels in a predefined radius around the pixel i. This neighborhood has the disadvantage that it is independent of the input image and therefore cannot adapt to the image structure.

Superpixel neighborhood. Segmenting the image in superpixels is another natural way to define a local pixel neighborhood. We define the set N_i as all pixels belonging to the same superpixel as the pixel i. In this case, all the pixels in one superpixel have the same neighborhood. The fact that all superpixels have approximately the same size means that they may be too big for some small objects like signs or poles, while at the same time covering only very small part of bigger areas like the street or the sky. To compensate for this we generate

Superpixels Mean-shift

Figure 2: Unsupervised segmentation with superpixels and mean-shift for different parameters.

the *superpixels neighborhood* for 3 different value of the size parameter to get superpixels with different sizes (see Figure 2). For the computation of the superpixels, we use the state-of-the-art SLICO method [30] - a version of SLIC [30], which automatically adapts the compactness parameter.

Mean-shift neighborhood. Mean-shift segmentation [31] is another unsupervised segmentation approach, in which the segments are not constrained to have approximately the same size. We define the *mean-shift neighborhood* in a way similar to the *superpixel neighborhood*: N_i is the set of all pixels belonging to the same segment as pixel i. Since it is very unlikely that all segments perfectly align to all object boundaries at the same time, we follow the approach presented in [7] to generate 3 segmentations with different parameters ranging from oversegmentation to undersegmentation (see Figure 2).

Geodesic neighborhood. The goal of the *geodesic neighborhood* is to define a local neighborhood that covers pixels in the vicinity of the pixel of interest that only belong to the same object. In this way, if a pixel is wrongly classified in the first stage, it can get support from other pixels nearby that were classified correctly. We define the *geodesic neighborhood* N_i as the set of the n pixels with the lowest geodesic distance to the pixel i. The geodesic distance is an extension of the Euclidean distance that also considers the image intensities. Therefore, two points in the image that have a high gradient between them will have a bigger distance than two points at the same Euclidean distance, but without strong edge between them. Formally, the geodesic distance is defined as:

$$D(i,j) = \inf_{\mathbf{G}\in\mathcal{P}_{i,j}} \int_0^l \sqrt{1+\gamma^2(\nabla I \cdot \mathbf{G}'(\mathbf{s}))^2}ds, \tag{3}$$

a) Local $\gamma = 0$ b) Local $\gamma = 50$ c) Local $\gamma = 1000$ d) Global $\gamma = 1000$

Figure 3: Visualization of the shapes of the presented neighborhoods for selected pixels (marked in black). The first 3 images show the shape of the local *geodesic neighborhood* for different values of γ, while the last image shows the *global geodesic neighborhood*, consisting of 8 separate ray neighborhoods.

with $\mathcal{P}_{i,j}$ denoting all possible paths between two pixels i and j, $\mathbf{G}$ and $\mathbf{G}'$ a path of length l and its spatial derivative correspondingly. Here, we denote the image as I, but it can be both a texture image or a depth image. The parameter γ regulates the weight between the Euclidean and the geodesic term such that for $\gamma = 0$ the geodesic distance is equivalent to the Euclidean distance and for big values of γ the geodesic term dominates. In Figure 3 a), b) and c) we show how the parameter γ influences the shape of the geodesic neighborhood. We see that with the increase of the value of γ, the neighborhoods align better to the image structure and therefore provide more consistent evidence of the object that the pixel of interest belongs to. We present an efficient algorithm to compute the geodesic neighborhoods of all pixels in the image in Section 3.3.

Global geodesic neighborhood. One drawback of all of the neighborhoods described above is that they only cover the area in the vicinity of the pixel of interest and therefore can only provide local context. In many cases, however, global context relations can be a very important queue to resolve ambiguities. Therefore, we introduce another type of neighborhood that is able to provide global context coming from other parts of the image, which again makes use of the geodesic distance to increase the robustness of the method.

The neighborhood is formed by first shooting 8 rays equally spaced at an angle of 45° from the pixel of interest to the borders of the image. Then, we take the local geodesic neighborhood at each point along the ray and create

the union of all those neighborhoods separately for each ray. In this way, we get a set of 8 different neighborhoods, one for each ray, capturing the context information in a certain direction. While the shape of the neighborhood follows the ray, it can adapt to the image structure around it guided by the geodesic distance. The shape of the global neighborhood is shown on Figure 3 d).

3.3. Computing the geodesic neighborhood

While equation 3 is the formal definition of the geodesic distance, computing it in such a way is very inefficient. While there are several methods for fast and approximate computation of the geodesic distance transform [32, 33, 23] they are suitable for the computation of the geodesic distance to large image regions. Since we need to compute the distance from each pixel to the n closest points in the image, these methods cannot be applied in our case.

We propose an algorithm for the computation of the geodesic distance based on the Dijkstra algorithm for finding the shortest path from a point to all other points in the image. We define a 4-connected graph over the image such that every pixel is a node in the graph and each pixel is connected to its 4 adjacent pixels with an edge. In practice, we connect each pixel with 4 pixels that are several pixels away from it in order to allow the neighborhood to quickly cover bigger parts of the image. The weight of an edge connecting two pixels i and j is defined as the geodesic distance between them:

$$w(i,j) = \sqrt{1 + \gamma^2 d(i,j)^2}, \tag{4}$$

where $d(i,j)$ is some distance measure depending on the type of image that is used. For a color image I we define $d(i,j)$ as the distance between the pixels in the RGB color space $d_{rgb}(i,j) = ||I(i) - I(j)||_2$, while for a depth image we use the metric distance between the 3D points $d_{3D}(i,j) = ||P(i) - P(j)||_2$. If both a color image and a depth image are present, the distance measure can be defined as a combination of both with the help of a parameter α to balance their contribution $d_{combined}(i,j) = \max(\alpha d_{rgb}(i,j), d_{3D}(i,j))$. Because the two

distance measures are in different domains, RGB and metric space respectively, we choose $\alpha = 10$ such that edges in both modalities have an equal strength.

The computation of the geodesic neighborhood itself is done by performing N iterations of the Dijkstra algorithm on the graph defined above and the neighborhood consists of the nodes chosen at each iteration. For a more formal description see Algorithm 1 in Appendix A. The discrete version D^* of the geodesic distance computed by our method can be formalized as:

$$D^*(i,j) = \inf_{\mathbf{G} \in \mathcal{P}_{i,j}} \sum_{k \in G} w(k, k-1), \tag{5}$$

where G denotes a discrete path from the set of all possible paths $\mathcal{P}_{i,j}$ between i and j, k iterates over the nodes of the path and $w(k, k-1)$ denotes a weight of the edge between the node k and its predecessor in the path $k-1$.

3.4. Feature vectors

We use different types of features for the two classification stages of our method. The features for the unary classifier of the first stage are extracted directly from the texture image and the depth image (if available), while the features for the neighborhood classifier in the second stage are computed from the pixel neighborhood and the class probability maps of the unary classifier.

3.4.1. Unary classification features

For the unary features two well-known and efficient methods are used: histograms of oriented gradients (HOG) [34] and the 2D Walsh-Hadamard transform [35]. The HOG features are computed only for texture images in a window of size five around every pixel and divided into 16 directional bins, resulting in a 16 dimensional feature vector, containing the gradients for each direction.

The 2D Walsh-Hadamard transform [35] is a discrete approximation of the cosine transform and is very fast to compute, since it involves only addition and subtraction operations (see Figure 4). In the case of texture images, we use a configuration similar to the one proposed by [6]. First, the image is converted in the Lab color space and then the first 16 coefficients of the Walsh-Hadamard

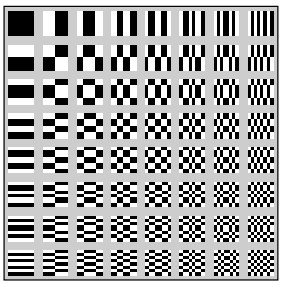


Figure 4: The basis vectors of the 2D Walsh-Hadamard transform of order 8. The black squares denote the coefficient -1 and the white squares the coefficient 1.

transform are computed in a window around every pixel separately for each color channel. We use windows of 6 different sizes: 2, 4, 8, 16, 32 and 64 pixels. In this way, we have 48 features for each scale, except for the window of size 2, where we have 4 instead of 16 coefficients. In the case of a color image, the feature vector has a size of 252, while in the case of grayscale images, we only have one channel and therefore the feature vector has a size of 84.

For depth images, we propose a novel way to use the 2D Walsh-Hadaramd transform. Computing the 3D coordinates of each pixel in the image results in structure that can be interpreted as a 3-channel float image (one channel for each 3D coordinate) and we compute the 2D Walsh-Hadamard transform in the same way as for the 3-channel color images described above. In this way, we get an approximation of the spatial frequencies of the image structure along the 3 coordinate axes in a very efficient way. This approximation works well for street scenarios, since most objects, like roads and buildings, tend to be aligned to the same axes. Note, that in this case a 3D Walsh-Hadamard transform is not suitable, because we only have information derived from a 2D image and not a dense 3D volume like in a MRI scan for example.

Finally, we add the 2D coordinates of each pixel and if available the 3D coordinates as well, feature in order to encode location context. The final feature vector is constructed by stacking all of the available features.

3.4.2. Neighborhood classification features

For the neighborhood classification stage, we introduce a new voting histogram feature. The goal is to create a descriptor for the content of each pixel's neighborhood, based on the class probability maps from the unary classifier.

First, we compute a normalized histogram of each pixel i over unary classifier responses of the pixels in its neighborhood N_i:

$$h_i(c) = \frac{\sum_{j \in N_i} [c = v_i]}{|N_i|}, \tag{6}$$

where $c \in \mathcal{L}$ is a class label and $v_i = \underset{c}{\operatorname{argmax}}\, P(y_j = c|\mathbf{x})$ is the most probable label for the pixel j according to the response of the unary classifier. The so computed voting histogram can be directly used as a feature vector for the neighborhood classifier. Furthermore, we can compute multiple different neighborhoods $N_i^1, N_i^2, \ldots, N_i^k$ for each pixel and stack the histograms $h_i^1, h_i^2, \ldots, h_i^k$ for the different neighborhoods into one large feature vector. Additionally, we also add the responses of the unary classifier for the pixel of interest i. The final neighborhood feature vector for pixel i becomes:

$$f_N(i, N_i) = \begin{pmatrix} h_i^1 \\ \vdots \\ h_i \\ P(y_i|\mathbf{x}) \end{pmatrix}. \tag{7}$$

The dimensionality of this feature vector is $|\mathcal{L}| \cdot (k+1)$. Directly taking the unary responses of all pixels in the neighborhoods as features, would result in a much bigger feature vector of size $|\mathcal{L}| \cdot \sum_{j=1}^{k} |N_i^j|$. Taking as an example a typical configuration from our experiments, consisting of 3 geodesic neighborhoods with sizes of 10, 50 and 200 for an 11 class problem, results in 2860 features when taking all pixels and only 44 when using the voting histogram.

It is important to note that all pixels are considered when building the histogram. Therefore, the histogram represents a summary of all pixels and while the information about individual pixels is not available anymore, every individual pixel still plays its role into building the features and, therefore, in

practice there is no significant difference between the results of the two variants. Using the summarized information, the classifier cannot model relations between individual pixels, but between regions of different sizes, which are more robust to individual pixel errors. The histogram feature also has the advantage of being much smaller, which leads to much faster training and evaluation.

We would like to point out that the neighborhood features depend only on unary classifier output, but not on the image itself. While this may be seen as a limitation of the expressive power of the neighborhood classifier, this formulation allows us to use compact and efficient features and leads to very short training and evaluation times as we show in Section 4.5. Furthermore, in our experiments we do not observe big performance loss, since the unary classifier seems to be able to fully exploit the image information. The authors of auto-context [17] report similar findings in their multi-stage classification framework.

3.5. Geodesic smoothing

The output of the neighborhood classifier already provides very good quantitative results, but still every pixel is classified individually. Therefore, in some cases there are individual wrong pixels which look like noise (see Figure 5 d)). Because this is a common problem with most classifier-based methods, many rely on a post-processing step, like for example by a pairwise CRF, to get smooth segments. However, such post-processing stages tend to be computationally expensive and require a lot of parameter tuning or a separate training step.

We propose an alternative and very efficient approach that is again based on the geodesic distance. We compute the voting histogram again, but this time using the output of the neighborhood classifier incited of the unary classifier and we denote it as $\hat{h}_i$. We use the same definition as in Equation 6, but now $v_i = \underset{c}{\operatorname{argmax}}\, P(y_j = c|\mathbf{x}, N_i))$. This time, however, instead of interpreting the voting histogram as a feature vector, we interpret it directly as the final probability distribution for pixel i, which can then be written as $P(y_i|\mathbf{x}, N_i) = \hat{h}_i$. This method has a positive impact on the quantitative and especially the qualitative results, with smoother regions that align well to the image structure (see Figure

5 e)). However, note that the size of the neighborhood used for smoothing should be small, because otherwise smaller objects may be smoothed over. Refer to Section 4.4 for detailed evaluation of the size of the smoothing neighborhood.

4. Evaluation

In this section, we present an extensive evaluation of our method based on six well-known and challenging datasets. First, we analyze how different parts of our method deal with different problems in segmenting the image. We then demonstrate how 3D information can be easily integrated in our framework. We also present more implementation details and discuss the model parameters in order to give more insight of how our method works. Finally, we evaluate the runtime of the method and its individual stages.

In order to measure the performance of different methods and configurations we rely on three widely used evaluation measures:

- Global pixel-wise accuracy - the percentage of correctly classifier pixels.
- Average per class accuracy - the average of the pixel-wise accuracies of each class calculated separately.
- Average per class intersection over union measure as used in the Pascal VOC challenge (Pascal accuracy) [36] - a measure that is a combination of the global and average per class accuracies.

4.1. Datasets

Our evaluation is based on six widely used datasets, three of which stem from the automotive domain whereas the other three contain more generic scenes. Additionally, in Section 6 we present an evaluation on our own dataset for detection of parking spaces on the side of the road. A systematic overview of the six public datasets that we use is presented in Table 1.

Dataset	Scene type	Camera images	Depth data	Image resolution	Labeled images	Classes count
CamVid [37, 38]	driving	color	no	320×240	601	11
MSRC-21 [3]	general objects	color	no	320×240	591	21
Stanford Background [39]	outdoor scenes	color	no	320×240	715	8
eTRIMS [40]	building facades	color	no	768×512	60	8
Daimler Urban [24]	driving	grayscale	stereo	976×360	500	6
KITTI Segmentation [28]	driving	color	stereo	1240×376	60	9

Table 1: Overview of the datasets used for evaluation.

CamVid. The CamVid dataset [37, 38] contains sequences of color images recorded from a moving car at daytime and at dusk. Most of the sequences, however, are recorded at a slow frame rate of roughly 1 frame per second. While computing structure-from-motion information from those sequences is principally possible [8], here we only rely on single color images. We use the same evaluation protocol as in [37, 8, 9] by downscaling the images to a resolution of 320×240 and taking the same 367 images for training and 233 for testing. Similar to [37] we consider only the 11 classes with most instances.

MSRC-21. The MSRC-21 dataset [3] is also a well-known and very challenging dataset, which contains nature and indoor images of which 276 are used for training and 256 for testing [3]. We consider the most common 21 classes (the classes MOUNTAIN and HORSE are commonly ignored) which range from trees, grass and sky to people, buildings and bicycles.

Stanford Background. The Stanford Background dataset [39] focuses on outdoor scenes that have some particular objects in the foreground. The commonly used testing procedure for this dataset [39] is to perform a 5 fold cross validation on a split of 572 training and 143 testing images.

eTRIMS. The eTRIMS image database [40] contains images of building facades divided in 8 semantic classes. Since the dataset contains only 60 images, we use the same testing procedure as in [20] by performing the evaluation on 10 different random splits of 40 images for training and 20 for testing.

a) Image b) Unary c) Local d) Global e) Geo. Smoothing

Figure 5: Visualization of the results of the different stages of our method.

Daimler Urban Segmentation. The recent Daimler Urban Segmentation dataset [24] contains 500 grayscale stereo image pairs with corresponding dense disparity maps. The images have a resolution of 1024×440 pixels and are labeled in the classes: GROUND, VEHICLE, PEDESTRIAN, BUILDING, SKY and BACKGROUND. For evaluation we use the same split of 300 training images and 200 testing images as suggested by the authors.

KITTI. The KITTI dataset [41] is a comprehensive benchmark for various computer vision problems related to autonomous driving like stereo and optical flow computation, object detection, visual odometry and others. Recently, 60 images from the stereo benchmark have been annotated with pixel-wise semantic labels [28] which form another dataset for which high quality depth information can be computed. The images have a resolution of approximately 1240×376 and are equally split into a training and a testing set.

4.2. Analyzing the method

In this section, we analyze how the different steps of our method deal with different segmentation problems and how they contribute to the final performance. In Figure 5 we visualize the results at the different method stages on one example image from the Stanford Background dataset and use it in order to explain different effects through the section. In parallel, we also present a full evaluation of the same method steps on the whole CamVid, MSRC-21 and Stanford Background datasets in order to measure the same effects quantitatively. Those results are summarized in Table 2, while in Figure 11, Figure 15 and Figure 12 we show results for qualitative evaluation.

4.2.1. Unary classification

The unary classification step of our method uses relatively simple, but very fast features, and therefore the segmentation results are quite noisy and with big error regions. Looking at Figure 5 a) the windscreen of the car is wrongly classified as WATER, the bumper of the car as ROAD and there are multiple wrong patches on the ground as well. While employing more powerful features would lead to better unary classification performance at the cost of increased runtime, in this paper we focus on developing a universal higher-level method that is able to correct errors by integrating context from multiple image regions.

4.2.2. Local neighborhood

By using local neighborhoods, our method is able to correct many errors from the unary classifier by using context information from the vicinity of each pixel. Looking at Figure 5 c), we see that many of the smaller errors are filtered and the shape of the car is much better defined. This is also confirmed by the quantitative evaluation (see Table 2), with all of the local neighborhoods delivering significantly better results than the unary classifier. Comparing the different local neighborhoods between each other we see that the *local geodesic neighborhood* achieves the best results. While the difference in some cases may seem small, the *geodesic neighborhood* has the important property of aligning the segmentation borders to the object boundaries that have a strong gradient.

In order to better understand why this is the case, we visualize the voting histogram features computed over different neighborhood variants in Figure 6. Here, we show the normalized values of the histogram bins for the classes TREE, ROAD and OBJECT. Note that even though the images can be interpreted as a probability distribution for each class, these are actually the features that serve as the input to the neighborhood classifier as described in detail in Section 3.4.2. We see that there are some significant differences in the histogram features for the different neighborhoods and therefore also in the resulting segmented images shown in the last row of Figure 6. The *Euclidean neighborhood* simply covers all pixels in a certain radius and therefore pixels close to object edges get responses

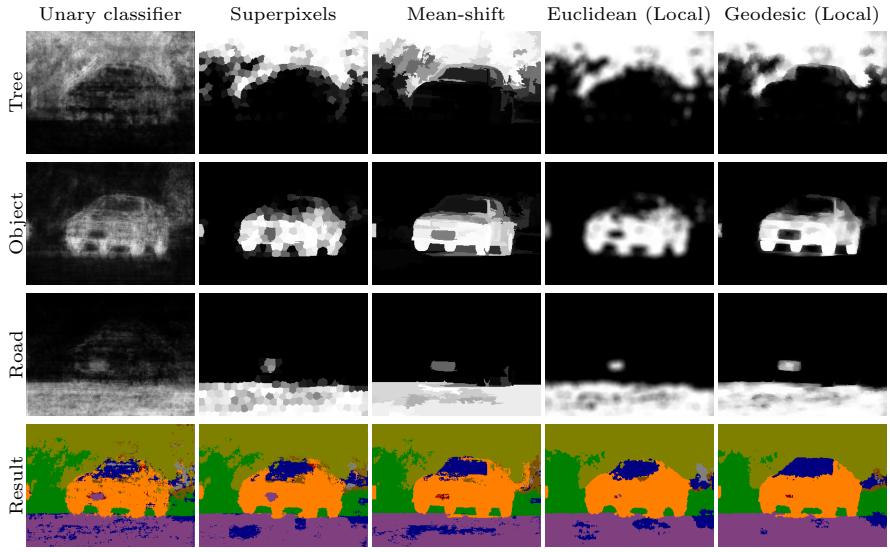


Figure 6: Comparison between the voting histogram features computed on different local neighborhoods. The images show the normalized values of the histogram bins for the corresponding classes and the resulting segmentation.

from other pixels on both side of the edge. Therefore, the features in those regions are blurred, resulting in segmentation borders that do not align well to the object edges. The superpixels can better align to the image structure, but they are either too big to fit to the small details of the car, or too small to capture enough local context. The mean-shift segmentation does not have the size constraints of the superpixels and can therefore adopt better to the objects. However, as with any unsupervised segmentation method, it is extremely difficult to create a disjoint image segmentation that captures all object edges. This problem is somewhat mitigated by creating multiple segmentations with multiple parameters as described in Section 3.2, but we can see that the results are still not as good as with the *geodesic neighborhood*. The main advantage of the *geodesic neighborhood* is that it generates a neighborhood with a different shape for each individual pixel, which allows it to adapt well to small structure details, while in the same time being big enough to capture enough local context.

Method	CamVid			MSRC			Stanford Background		
	Global	Average	Pascal	Global	Average	Pascal	Global	Average	Pascal
Unary	70.6	56.4	35.7	61.4	50.4	33.4	66.6	64.2	46.0
Superpixels	74.5	60.6	39.6	67.2	59.2	40.7	69.5	66.8	48.8
Mean-shift	75.6	61.7	40.5	71.7	63.7	46.0	71.1	68.2	50.4
Euclidean	76.0	62.3	41.2	73.0	67.5	49.1	70.4	68.0	49.5
Geodesic (local)	76.5	63.1	41.9	71.9	66.2	47.8	71.5	68.4	50.5
Geodesic (global)	**77.9**	**64.0**	**43.6**	**76.3**	**70.7**	**54.5**	**73.2**	**70.1**	**52.4**
Auto-context [17]	74.5	61.7	40.5	72.5	67.3	49.4	72.0	69.4	51.5
Robust P^n [7]	**77.9**	56.0	40.5	73.4	65.0	47.7	71.9	67.9	50.8

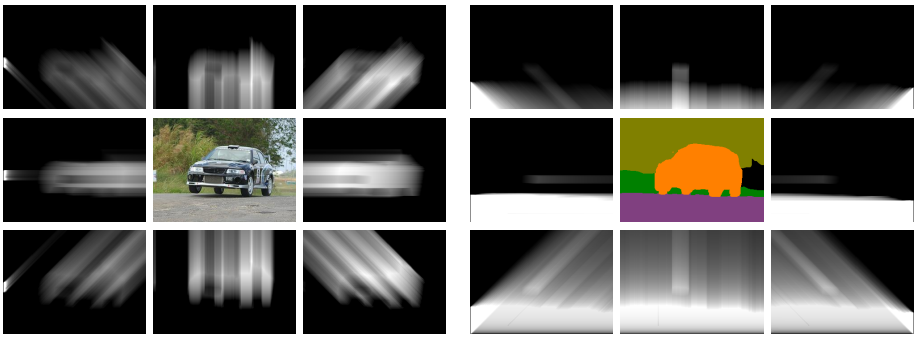
Table 2: Quantitative evaluation on CamVid, MSRC-21 and Stanford Background.

4.2.3. Global neighborhood

While the local neighborhoods are able to correct smaller errors, their limited range makes it more difficult to handle bigger errors like the bumper of the car or the "water" patches on the ground in Figure 5 c). Adding the global neighborhood to the neighborhood classifier feature vector, allows it to learn higher-level context information that resolves most of these problems.

To explain this, in Figure 7 we visualize the features of the eight rays of the *global geodesic neighborhood* for the classes ROAD and OBJECT. The images can be interpreted in the following way: a high value for a pixel in the image for the ray pointing to the left for the class OBJECT means a high probability that there is a region of the class OBJECT on the left of the chosen pixel. If we take as example one of the pixels on the ground that were wrongly classified as WATER, it will get very high responses for the class ground from the rays pointed downwards and sideways, while from the rays pointed up it will get responses for the classes OBJECT, TREE and SKY. Based on this information of the objects in the image surrounding the pixel, the classifier is able to learn the respective context relations and correct most of the errors (see Figure 5 d)).

From the quantitative evaluation in Table 2 we can see that the usage of the *global geodesic neighborhood* gives a significant advantage over the local variants. It is interesting to note that the increase in performance is much bigger for the MSRC-21 dataset. Analyzing the images in more detail shows



Rays for class tree Rays for class object

Figure 7: Visualization of the 8 rays of the *global geodesic neighborhood* for the classes tree and object.

that in the Stanford Background and in the CamVid dataset most of the images contain instances of all classes together. In this case, the global neighborhood learns spatial context relations between objects, e.g. cars are on top of the road and the sky is above the buildings and the cars. In MSRC-21, however, the images show a variety of different situations that usually contain only 2 or 3 of the 21 classes. In this case, the global neighborhood is able to also learn the co-occurrence between the classes, e.g. cows and trees appear in images with grass, while birds and airplanes appear in images with sky. Further examples of this can be seen in Figure 15.

4.2.4. Geodesic smoothing

While the segmentation based on the global neighborhood is already good, we can see that there are many small pixel errors everywhere in the image. This is due to the fact that the neighborhood classifier takes the decisions about the classes of the pixels independently of each other. By using our geodesic smoothing method (see Section 3.5) we get smoother segment (see Figure 5 e)) and an improvement in the overall classification performance.

4.3. Integrating 3D data

Our framework is not limited to handling 2D images, but it can also naturally handle 3D data in all stages: feature computation, neighborhood com-

Method	**Global**	**Average**	**Pascal (all)**	**Pascal (dyn)**	Ground	Vehicle	Person	Building	Sky
Unary Texture	60.6	64.8	50.6	33.6	68.8	41.1	26.1	68.0	48.9
Unary Depth	61.1	64.2	44.2	37.7	71.0	45.2	30.2	25.3	49.1
Unary Combined	66.2	70.6	57.7	46.6	72.6	54.9	38.3	68.9	53.6
Geodesic Texture (global)	70.2	**76.4**	65.2	62.0	73.8	**71.0**	53.0	72.2	55.8
Geodesic Depth (global)	70.2	75.5	64.3	60.1	73.2	69.0	51.2	71.3	56.9
Geodesic Combined (global)	**71.5**	76.2	66.1	61.9	75.5	70.9	52.8	**74.2**	56.9
Multi-cue segmentation [24]			56.6	58.8	82.8	63.9	53.6	29.0	53.8
Stixmantics [26]			66.7	**64.0**	**87.6**	68.9	**59.0**	57.6	60.2
Depth-enabled ICF [26]			52.7	44.2	86.2	53.5	34.9	35.1	53.9

Table 3: Quantitative evaluation on the Daimler Urban segmentation dataset.

putation and geodesic smoothing. The Daimler Urban dataset and the KITTI segmentation dataset contain stereo image pairs from which depth data can be computed. While the Daimler Urban dataset already comes with precomputed dense disparity maps of high quality, for the KITTI dataset, we compute the disparity maps by using the efficient method of [42] which is one of the top ranked methods on the KITTI stereo benchmark. For performance reasons, we do not classify each pixel individually, but we use cells of 4×4 pixels. Note, however, that this is not equivalent to downscaling the images 4 times, because we still compute the features at full resolution and compare the result to the ground truth data on the pixel level.

For both datasets, we use the same error measures as for the 2D datasets. The authors of the Daimler Urban dataset use a somewhat different measure, which is based on the average intersection over union, but excludes the class BACKGROUND. Furthermore, they also report the average only on the two dynamic classes VEHICLE and PERSON. We report the same measures as in [26] in addition to the others in order to allow for comparison.

4.3.1. Unary classification

For the unary classification step we evaluate three different unary features sets (as described in Section 3.4): only texture features, only depth features

Method	**Global**	**Average**	**Pascal**	Road	Building	Sky	Tree	Sidewalk	Car	Grass	Column	Fence
Unary Texture	66.1	60.4	38.8	74.2	66.7	92.1	64.7	59.8	57.0	69.9	7.8	**51.8**
Unary Depth	51.7	53.5	31.2	79.4	54.3	85.4	30.6	57.5	65.7	49.8	**14.2**	44.4
Unary Combined	72.4	65.5	45.3	79.3	75.4	92.3	69.6	66.1	71.5	77.9	10.5	47.3
Geodesic Texture (global)	75.2	**68.4**	48.7	**84.8**	**85.0**	**93.6**	66.8	64.8	**79.6**	**84.9**	13.2	42.7
Geodesic Depth (global)	74.0	66.0	46.5	84.1	81.1	88.0	67.8	63.2	74.5	83.3	1.4	50.5
Geodesic Combined (global)	**76.1**	68.3	**49.3**	84.2	82.7	93.4	**70.9**	**67.5**	78.8	82.5	11.5	43.4

Table 4: Quantitative evaluation on the KITTI segmentation dataset.

or both together. From the quantitative evaluation in Table 3 and Table 4 we can see that for both datasets using each modality separately already gives meaningful results. This also shows that our novel formulation of the 2D Walsh-Hadamard transform in 3D space is meaningful. Combining both modalities in the classifier results in a noticeable improvement. Therefore, all of the following evaluations of the neighborhood classifier are based on the output of the combined texture and depth unary classifier.

4.3.2. Neighborhood classification

Having access to the depth data, gives us the possibility to define the geodesic distance not only on the image gradients but also in the 3D space (see Section 3.3). Therefore, we can compare the performance of the *geodesic neighborhood* when defined for different image modalities. From the quantitative evaluation on both of the datasets we see that the two variants perform similarly well, which shows that our method can be naturally adapted to different image modalities (in fact we are even using the same unary features).

However, looking at the shapes of the neighborhoods in more detail (see Figure 8) we can see that they can be quite different in some regions. The geodesic neighborhood defined in the image space can overflow from the dark area on the cars to the dark areas on the building because of the lack of contrast. While this is not the case for the 3D *geodesic neighborhood*, because there is a big difference in the depth between the cars and the building, the geodesic

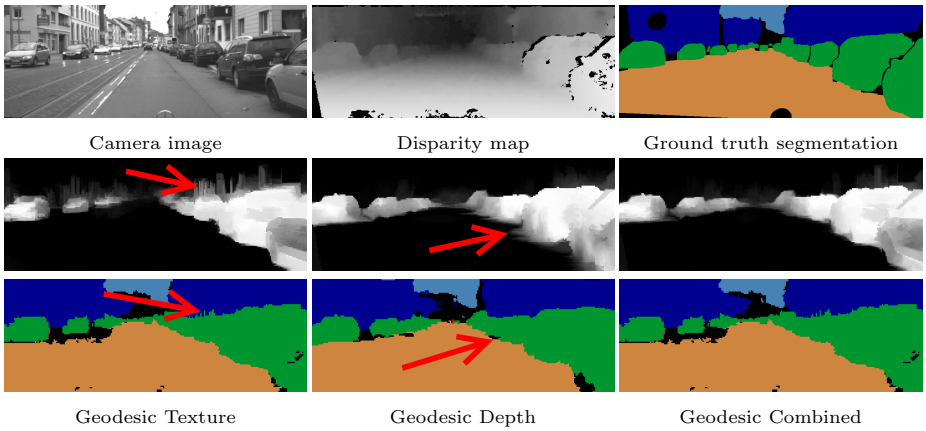


Figure 8: Visualization of the *geodesic neighborhood* defined in the 2D texture space, the 3D metric space and combined. The red arrows point to some typical problems of the *geodesic neighborhood* when defined in the texture or depth information alone that can be resolved when both are combined.

neighborhoods can flow out of one object into another one if they are touching. This can be observed around the tires of the cars touching the street. Combining both distances in the neighborhood definition as described in Section 3.3 allows us to use the advantages of both modalities, which leads not only to big qualitative, but also to significant quantitative improvements. This is also the case for the *global geodesic neighborhood*, where the rays can better adapt to the image structure if they use the edge information in both image modalities.

4.4. Implementation details and parameters evaluation

In this section, we give more details about setting the model parameters. Our method generalizes well enough so that we can use the same parameters for the computation of the features and for the classifier training over all datasets.

4.4.1. Classifier parameters

For both the unary and the neighborhood classifiers, we employ the JointBoost method [43] that can efficiently handle multi-class classification problems. We use some standard techniques [44, 11] to speed up the training by not taking the entire image, but sub sampling it on a grid of size 5 instead. Furthermore,

at each iteration step we test a random 30% of the features by comparing them to 100 thresholds that are sampled uniformly over the values of each feature. In fact, the only difference between the unary and the neighborhood classifiers is in the number of iterations, which is only for performance reasons. While for the unary classifier we need 5000 iterations in order to get the maximum out of the relatively big feature space (see Section 3.4), the neighborhood classifier, which operates on our compact voting histogram features, saturates after 1000 iterations. The parameters of the classifiers have influence mainly on the training time of the method.

4.4.2. Neighborhood computation

Here we evaluate the influence of the geodesic neighborhoods computation parameters on the performance of our method. Since the results are comparable for all datasets, here we show detailed numbers only from the CamVid dataset.

Neighborhood size and geodesic weight. The size of the neighborhood and the weighting parameter γ are the most important parameters for the segmentation results, because they influence the size of the region taken into account and its shape. In Figure 9 we show an evaluation of our method using the *local geodesic neighborhood* for different combination of the values. We see that making the neighborhood bigger also gives better results, but the effect saturates after a size of 100. Smaller neighborhoods work better for smaller objects, because they do not go over the object boundary, but for bigger objects, they do not cover enough pixels to capture enough local context. Therefore, a combination of several neighborhoods is beneficial. In our case, we use three neighborhoods of sizes 10, 50 and 200 (denoted as 3x). Note that the three neighborhoods can be computed efficiently, by just computing the neighborhood of size 200 and taking the first 10 and 50 pixels for the others. As discussed above, the *local geodesic neighborhood* achieves better results than the *local Euclidean neighborhood*, which can also be seen in Figure 9. The effect saturates at high values of γ, where the effect of the Euclidean distance becomes negligible. For our experiments, we use $\gamma = 5000$ in order to maximize the effect of the geodesic term.

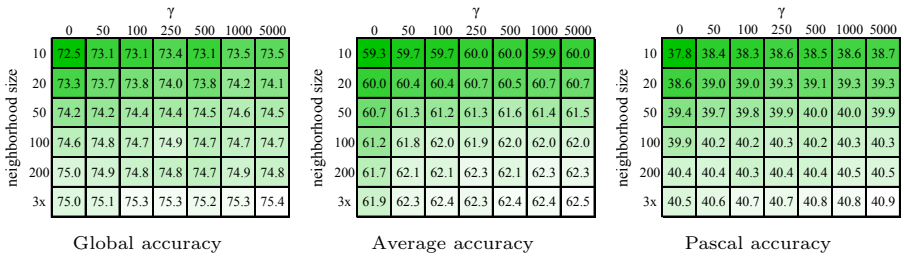


Figure 9: Influence of the two most important parameters controlling the size and the shape of the geodesic neighborhood on the performance of our method. The neighborhood size 3x indicates the usage of 3 neighborhoods of size 10, 50 and 200 pixels.

Neighboring pixels grid size. Another parameter that influences the shape of the neighborhood is the grid size at which neighboring pixels are sampled by the Dijkstra algorithm (see Section 3.3 for details). Higher values of the parameter allow the neighborhoods to cover bigger parts of the image and lead to some improvement in performance especially for the average accuracy (see Figure 10). However, this effect quickly saturates for after a grid size of 3.

Rays count. For the *global geodesic neighborhood*, the number of rays determines from which directions information from the image is gathered. As can be see in Figure 10, more directions give the neighborhood classifier more information about the scene and lead to higher accuracy. Adding more than 8 rays, however, does not lead to any further improvement.

Geodesic smoothing. As discussed in Section 3.5, the size of the neighborhood used for geodesic smoothing should be small in order to not smooth over smaller objects. This is confirmed by the results in Figure 10, where the global accuracy increases with bigger neighborhood sizes, but for larger values, the average accuracy starts to decrease. This is due to the fact, that large neighborhoods flow out of smaller objects like poles or signs and consequently they tend to be lost in the image resulting in a drop of the accuracy for the smaller classes and therefore also the average per class accuracy.

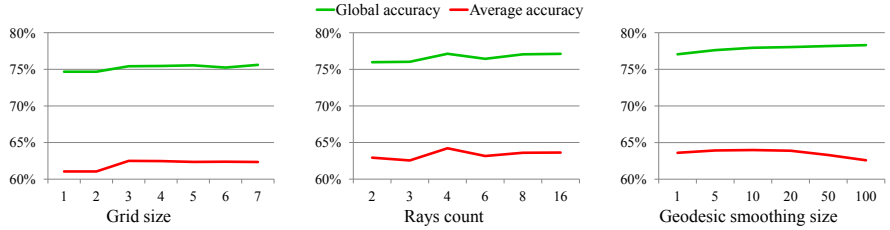


Figure 10: Evaluation of the parameters for the sampling grid size, the number of rays in the *global geodesic neighborhood* and the size of the neighborhood used for geodesic smoothing.

Method stage	**CamVid** 1 × 1 pixel	**Daimler Urban** 4 × 4 pixels	**Daimler Urban** 8 × 8 pixels
Unary features	544 ms	159 ms	90 ms
Unary classifier	1106 ms	53 ms	11 ms
Geodesic Neighborhood	2016 ms	402 ms	52 ms
Voting Histogram Features	137 ms	149 ms	15 ms
Neighborhood classifier	290 ms	187 ms	35 ms
Geodesic Smoothing	13 ms	5 ms	1 ms
Total	**4106 ms**	**955 ms**	**204 ms**

Table 5: Runtime of our method on two of the datasets used in the evaluation.

4.5. Runtime evaluation

A lot of computer vision systems like driver assistant systems, autonomous vehicles or mobile robots need to operate and react in the real world. This means that the runtime of such methods is an important factor to consider. Furthermore, using specialized hardware like a GPU or a FPGA may not always be possible. In this section, we evaluate the runtime of our method and show that we can adjust the trade-off between speed and accuracy.

In Table 5 we show the runtime of our method for two of the datasets used for evaluation - one using camera images only (CamVid) and one with both 2D and 3D data (Daimler Urban). For the Daimler Urban dataset we evaluate two variants of our method using cells of size 4×4 pixels, like in the quantitative evaluation presented above and 8×8 pixels. All tests are performed on a desktop machine equipped with two Intel Xeon X5690 processors with 6 cores each running at 3.46 GHz. All parts of our method are parallelized using OpenMP,

but no other specific optimizations, like usage of SSE instructions, have been performed. All of the code is running on the CPU only and no GPU is used.

We can see that the performance of our method is highly dependent on the pixels or cells our method has to classify and compute the features for. While for the CamVid dataset, we have 76,800 pixels per image, when we use cells of size 4×4 pixels for the Daimler Urban dataset we need to process only 21,960 cells and this explains the big difference in the runtime between the two datasets. Around 40% of the runtime is spent computing the geodesic neighborhoods for every pixel. This can be further optimized by taking into account the nodes found for the neighboring pixels, because adjacent pixels tend to have very similar neighborhoods. We plan to address this in our future work.

One useful property of our method is that one can easily control the trade-off between segmentation accuracy and runtime by adjusting the size of the cells. We show this on the Daimler Urban dataset by increasing the cell size to 8×8 pixels and by this effectively making the method more than 4 times faster. As expected, the accuracy decreases, but not by very much - the faster method is still able to achieve 65.2% average Pascal accuracy for all classes instead of 66.1% and 59.5% instead of 61.9% for the dynamic classes.

Since in our method one and the same operation is applied to multiple pixels simultaneously, the code can be easily speeded up further by running vectorizing it on a suitable hardware (like a GPU or DSP) or by using SSE instructions. Another option for optimization would be to adapt the computation of the geodesic neighborhood for a given pixel to use the results of neighboring pixels that have already been computed.

In Section 6 we show how our method is used in a real-time application as a part of a system for the detection of parking spaces from a driving car.

5. Comparison to other methods

We perform a detailed comparison of our method using the *global geodesic neighborhood* to several closely related state-of-the-art methods. Because most

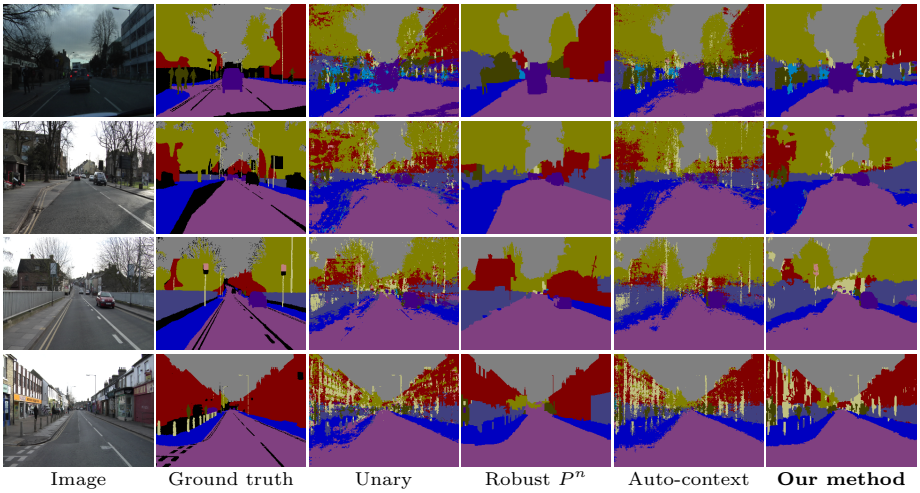


Figure 11: Result images on the CamVid dataset.

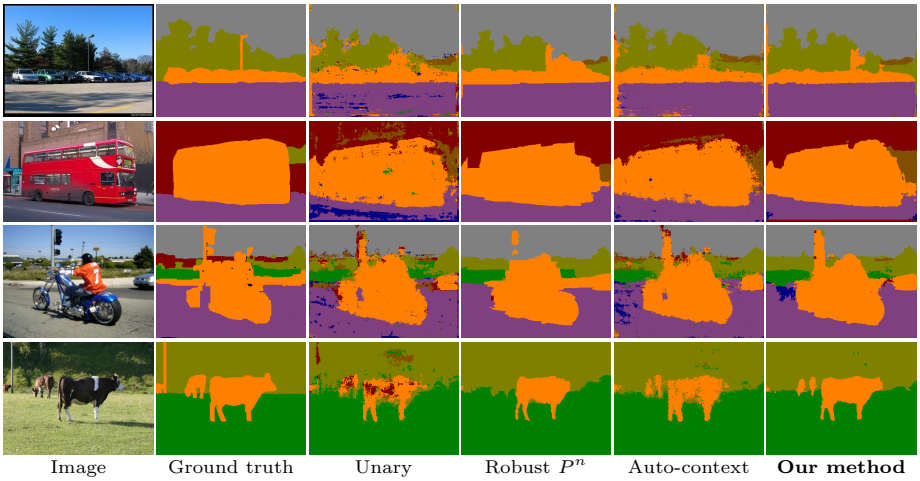


Figure 12: Result images on the Stanford Background dataset.

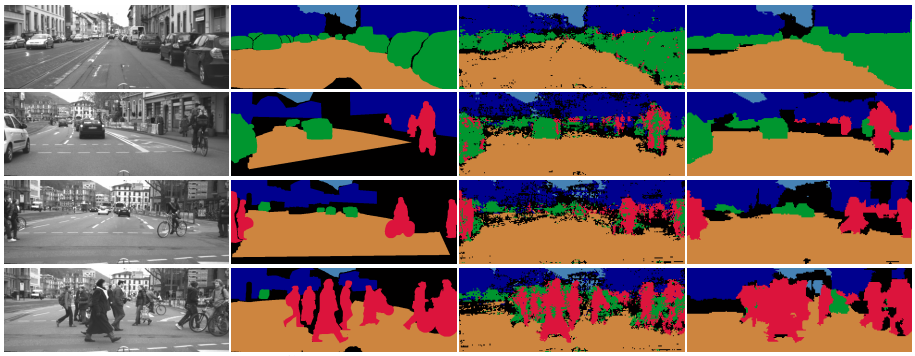


Figure 13: Result images on the Daimler Urban Segmentation dataset.

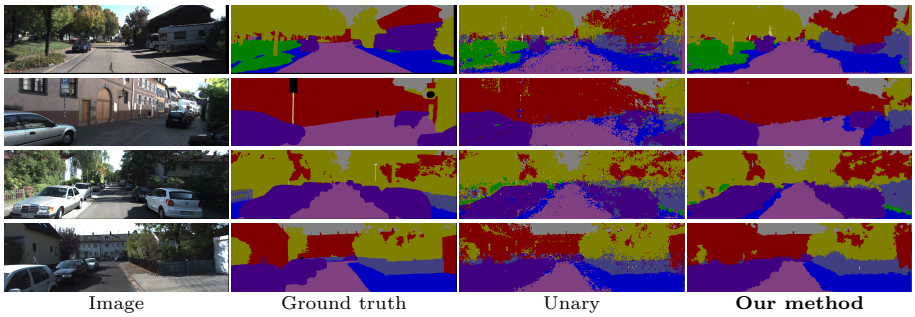


Figure 14: Result images on the KIITI dataset.

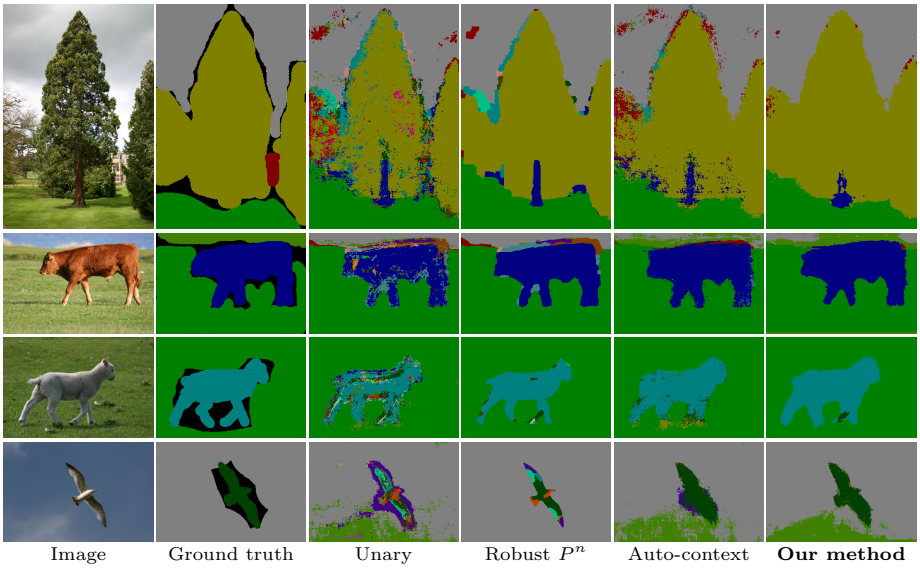


Figure 15: Result images on the MSRC-21 dataset.

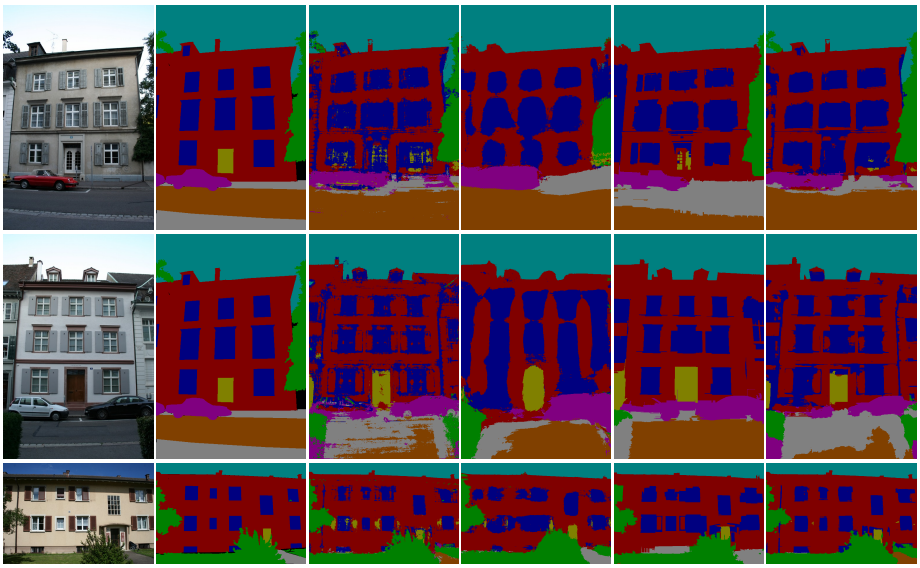


Figure 16: Result images on the eTRIMS image database.

semantic segmentation methods are strongly dependent on the features that are computed from the images, a meaningful comparison between methods focused on higher-level reasoning can be done only if the methods use the similar features. For the CRF based Robust P^n model [7] and for the classifier based auto-context method of [17], we use the same image features and the same unary classifier as in our method. For the state-of-the-art method of [11] we use the opposite approach and we adopt the features and the unary classifier from the implementation that is published online by the authors. The overall accuracies with those features are much higher because the authors [11] use more sophisticated, but also much slower image features.

Other strongly related methods are the multi-cue segmentation method of [24] and the Stixmantics method [26], which combine texture and depth data to create a fast method for real-time semantic segmentation. Both methods are evaluated on the Daimler Urban dataset, but since the authors do not provide access to their code, we cannot use the same features during evaluation.

The Iterative Context Forests (ICF) method [20, 21] is also related to our

method since it aims on learning context relations. We compare to a version of ICF that was extended by the authors of [26] to handle depth data and was also evaluated on the Daimler Urban dataset.

5.1. Robust P^n model

The Robust P^n model is a powerful higher level CRF model that relies on multiple segmentations (based on mean-shift) of the input image in order to better align the labeled segments to the image gradients. While our idea is similar, we take a completely different approach for modeling the relationships between pixels and segments. While the authors of [7] rely on a CRF to model these relations, in our model they are encapsulated in the voting histogram features and learned by the classifier. For our experiments we use the inference implementation provided by the authors, but using our image features and classifier for the unary potentials of the CRF.

The Robust P^n model performs very well with regards to the global accuracy on the CamVid dataset, but is outperformed by our method on the other datasets and for the other evaluation measures. If we look at the result images in Figure 11, 15 and 12 we can see that this method produces well-defined label segments that align good to image structures due to the mean-shift segmentation, but they may be wrong or smooth over smaller objects and therefore get worse results on the average per class accuracy and the Pascal accuracy.

5.2. Auto-context

The auto-context method is related to our model, because it uses a chain of classifiers, but with a different idea behind the higher-level features. While we use the adaptive geodesic neighborhoods to summarize the output of the unary classifier around the pixels of interest and along 8 rays that provide more global context, auto-context uses fixed points along 8 similar rays to sample the image features and the responses of the previous classifier. Therefore, the feature vectors for the classifiers in auto-context are much bigger than those in our method and the sampling is not adapted to the image structure, but is

Method	CamVid			MSRC		
	Global	Average	Pascal	Global	Average	Pascal
Unary from [11]	77.7	60.8	42.4	84.3	77.7	65.7
Geodesic (global)	81.0	**61.9**	44.8	84.5	**79.5**	66.4
Associative Hierarchical RF [11]	**85.1**	59.9	**50.4**	**87.8**	77.5	**69.7**

Table 6: Quantitative evaluation on the CamVid and MSRC-21 datasets based on the unary potentials from [11].

always done at the same offsets. However, we can easily implement auto-context as a use-case of our method by using the same sampling structure as in [17] and regarding each sampling point as a separate neighborhood containing only one pixel. We perform two iterations of auto-context because we also train two classifiers for our method and the analysis in [17] shows that the performance of auto-context quickly saturates after the second iteration.

From the qualitative results, we can see that the sampling structure of auto-context is able to capture more context than the local version of our method on the MSRC-21 dataset. However, our global method is still able to provide better results because it can better adapt to the image structure. On the other datasets, where the global context relations are not that strong even the local version of the *geodesic neighborhood* performs better.

5.3. Associative Hierarchical Random Fields

The associative hierarchical random fields of [11] are originally based on the Robust P^n model, but extend it by adding more complex higher level reasoning based on additional classifiers acting on higher level nodes in the graph that are organized in a hierarchical structure. We use the implementation and parameters provided by the authors to run their method on the CamVid and MSRC-21 dataset. For our method, we substitute the output of our unary classifier with the output of the unary classifier of [11].

As we can see from the results presented in Table 6, the unary classifier already provides very good segmentation due to the powerful image features used in [11]. Note, however, that those features are much slower to compute.

While the 2D Walsh-Hadamard features can be computed in 544 ms per image for the CamVid data set (see Table 5), the features used in [11] take 1627 ms per image on the same machine and with a similar level of code optimization.

The results show that the method of [11] delivers better results according, to the global and intersection-over-union measure, while our method is better at the average accuracy measure. Overall, even though our results are slightly worse than [11], our method runs twice as fast. Both our method and the method of [11] are optimized to take advantage of parallel processing on multiple processors and cores, but the parallelization is done on different levels. In our method multiple pixels or cells are processed in parallel so that the system can be used in an online setup, while in the publicly available code of [11], multiple images are processed in parallel. In order to eliminate the effects of the different parallelization approaches, we also performed the timing experiments on only one processor core, with comparable results.

5.4. Stixmantics

The multi-cue segmentation method of [24] and the Stixmantics method [26] are evaluated on the Daimler Urban dataset (see Table 3), with both being real-time systems with the same goal as our paper - combine fast texture and 3D features for semantic segmentation. Our *global geodesic neighborhood* method is able to clearly outperform the method of [24]. In comparison to the Stixmantics method, our results are very similar for the evaluation measure of [26]. The method of [26] performs slightly better for dynamic objects, however, it is worth noting, that we process each image separately and do not use temporal information as done in [26], which is especially helpful for dynamic objects.

5.5. Iterative Context Forests

In order to compare to the Iterative Context Forests (ICF) method of [20, 21] we run our method on the eTRIMS image database [40] which is also used for evaluation by the authors of ICF. The authors of [20] make use of the powerful Geometric Context method of [45] to create a set of base features. In order to

Method	**Global**	**Average**	**Pascal**	Building	Car	Door	Pavement	Road	Sky	Vegetation	Window
Unary	75.6	71.6	51.7	70.3	**59.9**	**64.7**	48.5	72.8	97.2	88.6	71.1
Geodesic (global)	**79.4**	**73.5**	**55.8**	**75.9**	57.5	61.7	**52.8**	**74.5**	**97.6**	**89.4**	**78.5**
ICF [21]	70.8	68.6									
ICF (best) [20]	77.2	72.2									

Table 7: Quantitative evaluation on the eTRIMS image database.

allow for a better comparison we integrate the geometric context features in the feature vector of the unary classifier, by taking the probability of each of the 8 geometric classes for each pixel directly as a feature.

The quantitative results of our method are shown in Table 7 and some example result images are shown in Figure 16. Our base unary features cannot compete with the ICF method delivering inferior results. By running our complete pipeline, however, we get clearly better results than those reported in [20, 21] showing that our approach is indeed able to capture context relations.

Using the Daimler Urban dataset we can compare to ICF as well. The original method is extended by [26] to also handle depth data, so that it can take advantage of the disparity maps in the Daimler Urban dataset. In Table 3 we show the numbers reported in [26]. Even though, the ICF is more flexible in learning context relations, our local and global neighborhoods are able to summarize the context information well enough and again deliver better results.

6. Application

In this section, we present a real world application of the proposed segmentation method for the detection of free parking spaces on the side of the road using a side-viewing camera. Fisheye cameras in the vehicle's mirrors are already widely used for parking systems on a wide range of production vehicles. We use a camera in the right mirror to observe the side of the vehicle while driving in order to detect free parking spaces and parked cars. This information

780 can then be used by various advanced driver assistant systems like for example automated parking. This system is similar to the one originally presented in [46], but it uses only camera images and a simple pairwise CRF model, while here we use our *geodesic neighborhood* method and both texture and depth data.

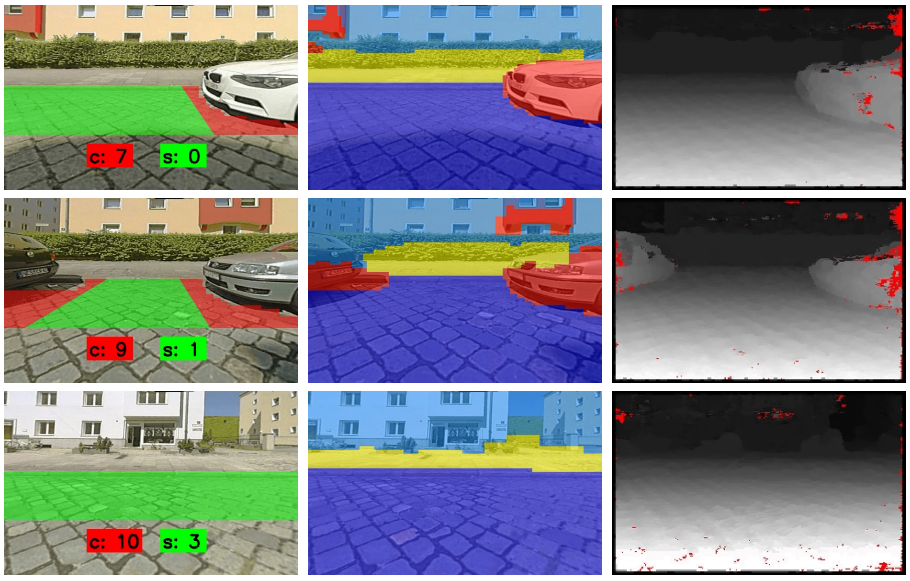
6.1. Dataset

785 For training and evaluation of the system, we recorded a dataset of around 30 km of driving in small or averagely big streets in the city under different environment conditions: cloudy weather, sunny weather and rain. All images are rectified, cropped to the relevant region of interest and scaled down to a resolution of 320×240 (see Figure 17 for example images). We labeled 220 of 790 the images pixel-wise in 6 semantic classes: ROAD, VEHICLE, LANE MARKING, HORIZONTAL BACKGROUND (like sidewalk, grass areas) and VERTICAL BACKGROUND (like buildings and sky). We use 148 images for training and 72 for testing. Because the environment on the side of the car in smaller streets is predominantly static, we can use structure-from-motion to extract 3D information, 795 which we can use in the segmentation along with the texture camera images. Since the camera is calibrated and the images are rectified, we can employ the stereo fusion method of [47] that uses multiple frames in order to generate robust disparity maps even in challenging outdoor conditions (see Figure 17).

6.2. Detection of parking spaces and parked vehicles

800 We segment each camera frame semantically and extract the border of the ground plane (the class ROAD). Since we calibrated the camera to the ground plane, we can now compute the 3D position of each point belonging to the ground under the assumption that it is flat. This allows us to measure the distance of each point on the border of the ground plane to the vehicle. Fur-805 thermore, from the semantic segmentation we also know the type of object that is beyond that border. We split the space next to the car in 20 cm wide sections that can be one of the following 3 classes:

- PARKED CAR - if the segment is limited by another segment of class vehicle,



Detected parking spaces | Semantic Segmentation | Motion stereo disparity map

Figure 17: Parking space detection application build on the proposed semantic segmentation method.

- FREE - if there are at least 2 m to the ground plane border,
- NOT FREE - if there are less than 2 m to the ground plane border.

Using the vehicle movement data, we can fuse the information from multiple frames in order to find the number of free parking spaces and the number of parked cars, assuming that a space is free if it is at least 6 m long.

In order to achieve real-time performance of the system we make several optimizations to speed up the segmentation method. As for the Daimler Urban dataset and the KITTI dataset in the previous section, we do not classify each pixel, but cells of 8×8 pixels. Furthermore, we compute the *geodesic neighborhood* for a smaller number of pixels - 100 instead of 200. With this configuration, we are able to segment one image in 38 ms, which is enough for the method to run in real-time in our test vehicle.

Method	**Global**	**Average**	**Pascal**	Road	Lane Marking	Car	Back. (hor.)	Back. (vert.)
Unary Texture	78.3	73.8	50.3	84.8	**92.4**	82.8	27.4	81.7
Unary Depth	78.2	56.0	46.2	**95.5**	1.2	73.7	36.7	73.1
Unary Combined	80.2	75.7	53.2	87.5	90.4	83.7	35.3	81.3
Geodesic Texture (global)	81.7	80.0	56.2	80.7	90.7	**90.8**	53.6	84.5
Geodesic Depth (global)	79.9	75.9	53.9	79.5	76.2	86.4	**53.9**	83.7
Geodesic Combined (global)	**82.0**	**80.0**	**56.5**	81.7	90.0	90.7	53.1	**84.5**

Table 8: Quantitative evaluation on the Parking Space Detection dataset.

6.3. System performance

We evaluate the performance of the system on two levels: segmentation accuracy and application accuracy. For the segmentation accuracy we use the same measures as for the evaluation of the datasets in Section 4.2. The quantitative results are summarized in Table 8. Here, we can again observe the same effects as for the other datasets: combining texture and depth information gives a significant boost over each of the modalities used separately.

Additional to the segmentation accuracy, we also evaluate the ability of the system to recognize parking spaces, which is directly related to the performance of the segmentation method. We compare the output of our detection system to a human counting the parking spaces and the parked cars over several sequences of total length of approximately 2.5 km. The average recall rate for parked cars is 98.2% at 1.6 false positive detections per 1000 m, while the average detection rate of the free parking spaces is 83.6% at 0.4 false positives per 1000 m. The parking space detection system is slightly biased towards detecting cars, because the classifier tends to detect unknown objects like trash cans or bus stops as cars. This problem can be resolved by adding more training images to the dataset.

7. Conclusion

In this paper, we presented a two-stage classification framework for semantic image segmentation of both 2D and 3D images. We showed how to define local and global pixel neighborhoods based on geodesic distance that are able to model

local and global context relation between image regions. We also introduced a compact histogram feature, which summarizes context information in those neighborhoods which allows for fast training and evaluation.

We evaluated our method on six challenging datasets containing both 2D and 3D information and gave detailed insights how different part of our model work. Our method achieves results comparable to the state-of-the-art, while at the same time being very fast. We demonstrated how our work is used in a vehicle to detect free parking spaces on the side of the road in real time.

In the future, we aim to further speed up the process of computing the geodesic neighborhoods and to design more neighborhoods that can capture not only spatial but also temporal information.

Appendix A. Geodesic neighborhood computation algorithm

In Algorithm 1 we present a formal description of the algorithm used for the computation of the geodesic neighborhoods.

Acknowledgments

The support from the BMW Group and the Technical University of Munich is gratefully acknowledged.

References

[1] V. Haltakov, C. Unger, S. Ilic, Geodesic pixel neighborhoods for multi-class image segmentation, in: BMVC, 2014.

[2] J. D. Lafferty, A. McCallum, F. C. N. Pereira, Conditional random fields: Probabilistic models for segmenting and labeling sequence data, in: ICML, 2001.

[3] J. Shotton, J. Winn, C. Rother, A. Criminisi, TextonBoost: Joint appearance, shape and context modeling for multi-class object recognition and segmentation, in: ECCV, 2006.

Algorithm 1: Computation of the geodesic neighborhood of a pixel.

Input: I - texture or depth image, d(i,j) - distance measure between two pixels, (x_s, y_s) - coordinates of the pixel of interest s, n - neighborhood size, g - step size, γ - geodesic term weight

Output: N_s - set of neighbors of the pixel s

$N_s \leftarrow \{\}$

foreach $p \in I$ **do**

  $dist[p] \leftarrow \infty$

  $dist_{geo}[p] \leftarrow 0$

end

$dist[s] \leftarrow 0$

for $i \leftarrow 1$ **to** n **do**

  $p \leftarrow$ pixel with minimum $dist$

  $N_s \leftarrow N_s \cup \{p\}$

  $P \leftarrow \{(x_p + g, y_p), (x_p, y_p + g), (x_p - g, y_p), (x_p, y_p - g)\}$

  for $t \in P$ **do**

    if $t \notin N_s$ **then**

      $d_e \leftarrow \sqrt{(x_t - x_s)^2 + (y_t - y_s)^2}$

      $d_g \leftarrow dist_{geo}[p] + d(t, p)$

      $d \leftarrow \sqrt{d_e^2 + \gamma^2 d_g^2}$

      if $d < dist[t]$ **then**

        $dist[t] \leftarrow d$

        $dist_{geo}[t] \leftarrow d_g$

      end

    end

  end

end

[4] S. Kumar, M. Hebert, Discriminative fields for modeling spatial dependencies in natural images, in: NIPS, 2003.

[5] Y. Boykov, O. Veksler, R. Zabih, Fast approximate energy minimization via graph cuts, PAMI.

[6] C. Wojek, B. Schiele, A dynamic conditional random field model for joint labeling of object and scene classes, in: ECCV, 2008.

[7] P. Kohli, L. Ladicky, P. H. S. Torr, Robust higher order potentials for enforcing label consistency., in: IJCV, 2009.

[8] P. Sturgess, K. Alahari, L. Ladicky, P. H. S. Torr, Combining appearance and structure from motion features for road scene understanding, in: BMVC, 2009.

[9] L. Ladický, P. Sturgess, K. Alahari, C. Russell, P. H. S. Torr, What, where and how many? Combining object detectors and CRFs, in: ECCV, 2010.

[10] L. Ladický, C. Russell, P. Kohli, P. H. S. Torr, Associative hierarchical crfs for object class image segmentation, in: ICCV, 2009.

[11] L. Ladický, C. Russell, P. Kohli, P. H. S. Torr, Associative hierarchical random fields, in: PAMI, 2013.

[12] S. Nowozin, C. Rother, S. Bagon, T. Sharp, B. Yao, P. Kohli, Decision tree fields, in: ICCV, 2011.

[13] J. Jancsary, S. Nowozin, T. Sharp, C. Rother, Regression tree fields - an efficient, non-parametric approach to image labeling problems, in: CVPR, 2012.

[14] S. Ross, D. Munoz, M. Hebert, J. A. Bagnell, Learning message-passing inference machines for structured prediction, in: CVPR, 2011.

[15] R. Shapovalov, D. Vetrov, P. Kohli, Spatial inference machines, in: CVPR, 2013.

[16] J. Shotton, M. Johnson, R. Cipolla, Semantic texton forests for image categorization and segmentation, in: CVPR, 2008.

[17] Z. Tu, X. Bai, Auto-context and its application to high-level vision tasks and 3D brain image segmentation, in: PAMI, 2010.

[18] D. Munoz, J. A. Bagnell, M. Hebert, Stacked hierarchical labeling, in: ECCV, 2010.

[19] W. W. Cohen, V. R. Carvalho, Stacked sequential learning, in: IJCAI, 2005.

[20] B. Fröhlich, E. Rodner, J. Denzler, Semantic segmentation with millions of features: Integrating multiple cues in a combined random forest approach, in: ACCV, 2012.

[21] B. Fröhlich, E. Rodner, J. Denzler, As time goes by - anytime semantic segmentation with iterative context forests, in: DAGM, 2012.

[22] A. Montillo, J. Shotton, J. Winn, J. E. Iglesias, D. Metaxas, A. Criminisi, Entangled decision forests and their application for semantic segmentation of CT images, in: Information Processing in Medical Imaging, 2011.

[23] P. Kontschieder, P. Kohli, J. Shotton, A. Criminisi, GeoF: Geodesic forests for learning coupled predictors, in: CVPR, 2013.

[24] T. Scharwächter, M. Enzweiler, U. Franke, S. Roth, Efficient multi-cue scene segmentation, in: GCPR, 2013.

[25] D. Pfeiffer, U. Franke, Towards a global optimal multi-layer stixel representation of dense 3d data, in: BMVC, 2011.

[26] T. Scharwächter, M. Enzweiler, U. Franke, S. Roth, Stixmantics: A medium-level model for real-time semantic scene understanding, in: ECCV, 2014.

[27] L. Ladický, P. Sturgess, C. Russell, S. Sengupta, Y. Bastanlar, W. Clocksin, P. H. S. Torr, Joint optimisation for object class segmentation and dense stereo reconstruction, in: BMVC, 2010.

[28] L. Ladický, J. Shi, M. Pollefeys, Pulling things out of perspective, in: CVPR, 2014.

[29] Z. Tu, Auto-context and its application to high-level vision tasks, in: CVPR, 2008.

[30] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, S. Susstrunk, Slic superpixels compated to state-of-the-art superpixel methods, in: PAMI, 2012.

[31] D. Comaniciu, P. Meer, S. Member, Mean shift: A robust approach toward feature space analysis, in: PAMI, 2002.

[32] P. J. Toivanen, New geodesic distance transforms for gray-scale images, in: Pattern Recognition Letters 17, 1996.

[33] L. Yatziv, A. Bartesaghi, G. Sapiro, O(N) implementation of the fast marching algorithm, in: Journal of Computational Physics 212, 2006.

[34] N. Dalal, B. Triggs, Histograms of oriented gradients for human detection, in: CVPR, 2005.

[35] Y. Hel-Or, H. Hel-Or, Real time pattern matching using projection kernels, ICCV.

[36] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, A. Zisserman, The Pascal Visual Object Classes (VOC) challenge, IJCV (2).

[37] G. J. Brostow, J. Shotton, J. Fauqueur, R. Cipolla, Segmentation and recognition using structure from motion point clouds, in: ECCV, 2008.

[38] G. J. Brostow, J. Fauqueur, R. Cipolla, Semantic object classes in video: A high-definition ground truth database, Pattern Recognition Letters.

[39] S. Gould, R. Fulton, D. Koller, Decomposing a scene into geometric and semantically consistent regions, in: ICCV, 2009.

[40] F. Korč, W. Förstner, eTRIMS Image Database for interpreting images of man-made scenes, Tech. rep., University of Bonn (2009).

[41] A. Geiger, P. Lenz, R. Urtasun, Are we ready for autonomous driving? The KITTI vision benchmark suite, in: CVPR, 2012.

[42] K. Yamaguchi, D. McAllester, R. Urtasun, Efficient joint segmentation, occlusion labeling, stereo and flow estimation, in: ECCV, 2014.

[43] A. Torralba, K. P. Murphy, W. T. Freeman, Sharing visual features for multiclass and multiview object detection, in: PAMI, 2007.

[44] J. Shotton, J. Winn, C. Rother, A. Criminisi, TextonBoost for image understanding: Multi-class object recognition and segmentation by jointly modeling texture, layout, and context, IJCV (2007).

[45] D. Hoiem, A. Efros, M. Hebert, Geometric context from a single image, in: ICCV, 2005.

[46] V. Haltakov, H. Belzner, S. Ilic, Scene understanding from a moving camera for object detection and free space estimation, in: IV, 2012.

[47] C. Unger, E. Wahl, P. Sturm, S. Ilic, Stereo fusion from multiple viewpoints, in: DAGM, 2012.