A Probabilistic Framework for Real-time 3D Segmentation using Spatial, Temporal, and Semantic Cues

David Held, Devin Guillory, Brice Rebsamen, Sebastian Thrun, Silvio Savarese
Computer Science Department, Stanford University
{davheld, deving, thrun, ssilvio}@cs.stanford.edu

***Abstract*—In order to track dynamic objects in a robot's environment, one must first segment the scene into a collection of separate objects. Most real-time robotic vision systems today rely on simple spatial relations to segment the scene into separate objects. However, such methods fail under a variety of real-world situations such as occlusions or crowds of closely-packed objects. We propose a probabilistic 3D segmentation method that combines spatial, temporal, and semantic information to make better-informed decisions about how to segment a scene. We begin with a coarse initial segmentation. We then compute the probability that a given segment should be split into multiple segments or that multiple segments should be merged into a single segment, using spatial, semantic, and temporal cues. Our probabilistic segmentation framework enables us to significantly reduce both undersegmentations and oversegmentations on the KITTI dataset [3, 4, 5] while still running in real-time. By combining spatial, temporal, and semantic information, we are able to create a more robust 3D segmentation system that leads to better overall perception in crowded dynamic environments.**

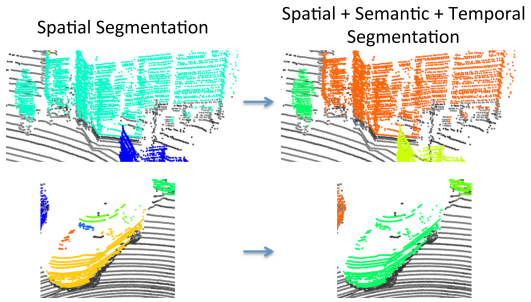

Fig. 1. Top: Using only spatial cues, a person is segmented together with a nearby building. By tracking this person over time, we segment it as a separate object. Bottom: Using only spatial cues, a car is oversegmented into multiple pieces. Using semantic and temporal information, we determine that these multiple pieces all belong to a single object. Each color denotes a separate object instance, estimated by the segmentation algorithm. (Best viewed in color)

I. Introduction

A robot operating in a dynamic environment must identify obstacles in the world and track them to avoid collisions. Robotic vision systems often begin by segmenting a scene into a separate component for each object in the environment [2, 11, 24]. Each component can then be tracked over time to estimate its velocity and predict where each object will move in the future. The robot can use these predictions to plan its own trajectory and avoid colliding into any static or dynamic obstacles.

Due to the real-time constraint on these systems, many 3D robotic perception systems rely on simple spatial relationships to find the objects in a scene [2, 11, 14, 24]. For instance, a common strategy is to use proximity to group points together: if two 3D points are sufficiently close, they are assumed to be part of the same object, and if the points are far apart and disconnected, they are assumed to belong to different objects.

However, spatial relations alone are not sufficient for distinguishing between objects in many real-world cases. For example, in autonomous driving, it is common for a pedestrian to get undersegmented together with a neighboring object, such as a tree, a nearby car, or a building, as shown in Figure 1 (top). If an autonomous vehicle is not aware of a nearby pedestrian, the vehicle will have difficulty anticipating the pedestrian's actions, which could lead to a disastrous collision. It is also common for a nearby object such as a car to get oversegmented into multiple pieces, with the front, side, and interior of the object being considered as separate segments (Figure 1, bottom). This oversegmentation can lead to poor tracking estimates, causing the autonomous vehicle to incorrectly predict the future position of nearby objects.

We present the first real-time 3D segmentation algorithm that combines spatial, temporal, and semantic information in a coherent probabilistic framework. A single frame of an environment is often ambiguous, as it can be difficult to determine if a segment consists of one object or multiple objects. The temporal cues in our framework allow us to resolve such ambiguities. By watching objects over time, we obtain more information that can help us determine how to segment the scene.

If we can recognize an object, then we can also incorporate the object class into our framework to assist our segmentation. At the same time, because we are using a probabilistic framework, the semantic cues are optional, enabling us to segment unknown objects: if we do not recognize an object, then our semantic probabilities will be uninformative and our framework will automatically rely on the spatial and temporal information for segmentation.

Our method outputs an "instance segmentation", i.e. a partition of a collection of 3D points into a set of clusters, with

one cluster corresponding to each object instance in the scene. These objects can then be individually tracked to estimate their future positions and avoid collisions.

We apply this method to the KITTI dataset, which was collected from a moving car on crowded city streets for the purpose of evaluating perception for autonomous driving [3, 4, 5]. We show that our framework significantly reduces both oversegmentation and undersegmentation errors, with the total number of errors decreasing by 47%. Our method runs in only 34 milliseconds per frame and is thus suitable for real-time use in robotic applications. By combining spatial, semantic, and temporal information, we are able to create a more robust perception pipeline that should lead to more reliable autonomous driving. More information about our approach can be found on our website: http://davheld.github.io/segmentation3D/segmentation3D.html

II. Related Work

Spatial segmentation. Most real-time 3D segmentation methods only use the spatial distance between points in the current frame for clustering [2, 11, 24], ignoring semantic information as well as temporal information from previous frames. However, because these methods only use the spatial distance between points, they cannot resolve ambiguous cases where an occlusion causes an object to become oversegmented or when multiple nearby objects become segmented together.

Temporal segmentation. Some methods focus on segmenting objects that are currently moving [1, 17, 27]. For example, Petrovskaya and Thrun [17] use "motion evidence" to detect moving vehicles. After filtering out static objects, clustering the remaining moving objects becomes simpler. Another approach is to estimate the motion for each point in the scene and then to cluster these motions [22, 8]. However, these methods would not be able to segment a stopped car or other objects that are currently stationary but may move in the future.

An additional approach is to cluster an image into coherent superpixels by enforcing spatial and temporal consistency [9, 28, 29]. However, these superpixels do not represent whole objects, so a post-processing method must be used to find the objects in the scene so that each object can be tracked for autonomous driving or other robotics applications. Further, most of these methods are applied to dense RGB-D images from indoor scenes [9, 28, 29, 22, 8] and would not directly apply to large outdoor scenes with sparse 3D sensors.

Semantic segmentation. Other papers have attempted to incorporate semantic information for 3D segmentation. Many of these methods can only segment objects of particular classes that can be recognized by a detector [21, 13, 6]. Using our probabilistic framework, semantic information is an optional cue; we are able to use spatial and temporal information to segment novel objects that we cannot identify, which is important for robust autonomous driving in the real world.

Wang et al. [26] use a foreground/background classifier to detect and segment objects. We compare to this method and demonstrate that we achieve much better performance by also incorporating temporal information. Further, their method assumes that foreground objects are well-separated from each other, which does not hold for crowded urban environments.

Point-wise labeling. Some papers apply a semantic label to each individual point in the scene [12, 20]. For example, these methods will give a label "person" to every point in a crowd of pedestrians or "car" to every point in a group of cars, without locating the individual objects in this group. The output of these methods are less useful for autonomous driving, since we need to be able to segment and track each object instance in our environment so we can predict where the object will go and avoid collisions.

Our approach. We are able to segment both known and unknown objects, as well as objects that are either moving or stationary. Each segment output by our method represents a single object instance. Our real-time segmentation method is thus suitable for autonomous driving and other mobile robot applications, in which all objects in our environment must be segmented so we can track each one and avoid collisions.

III. System Overview

Given a stream of input data, we first perform ground-detection, i.e. we determine which points belong to the ground and which points belong to objects. Next, we apply a fast initial segmentation method to cluster the non-ground points into separate segments.

We modify these initial segments using our probabilistic framework, splitting segments apart or merging segments together using spatial, temporal, and semantic information. Our framework computes the probability that a set of points represents a single object or more than object. If we estimate that a segment consists of more than one object, then we split this segment apart; if we estimate that a group of segments represents a single object, then we merge these segments together. Our goal is to obtain a final set of segments where each segment is estimated to represent a single object.

After obtaining the final segmentation for a given frame, each segment is tracked and classified to further improve the robot's understanding of its environment. Various methods for the ground-detection, initial segmentation, tracking, and classification may be used; the methods used in our pipeline are listed in Section VI-D.

IV. Probabilistic Framework

A. Segmentation Model

Our probabilistic framework estimates whether a segment represents a single object or multiple objects, incorporating the current observation, the estimated data association, and the estimated segmentation of the previous frame. We also optionally incorporate semantic information if we can recognize the type of object being segmented, as explained in Section IV-C.

Let s_t be any set of 3D points observed at time t, and let z_t be a set of features (defined in Section IV-B) computed based on the points in s_t. In our segmentation framework, s_t will be formed by the union of one or more of our initial segments. We want to estimate whether the points in s_t belong to a single

object or more than one object, and based on this estimate, we will merge the segments in s_t together or split s_t apart, as explained in Section V.

To determine whether the points in s_t belong to a single object or more than one object, we compute a probability $p(N_t^{s_t}|z_t, S_{1..t-1})$ where $N_t^{s_t} = true$ if the points in s_t belong to only one object, and $N_t^{s_t} = false$ if the points belong to more than one object (we abbreviate this variable as N_t when it is clear which segment it refers to). We also incorporate information from previous frames, given by $S_{1..t-1}$, where $S_i = \{s_{i,1} \cdots s_{i,n}\}$ represents the estimated segmentation of frame i, i.e. a partitioning of the points in frame i into disjoint subsets. Based on the computation of $p(N_t|z_t, S_{1..t-1})$, we decide whether to split our initial segments apart or to merge segments together, as explained in Section V.

We would like to accumulate segmentation evidence across time to better estimate this probability. To do so, we incorporate the data association $a_t^{s_t}$, which represents a matching between a collection of points s_t in the current frame and a collection of points in the previous frame (we abbreviate this variable as a_t). Suppose that the estimated final segmentation at time $t-1$, given by S_{t-1}, is some partition of the points observed at time $t-1$ into n segments, $S_{t-1} = \{s_{t-1,1} \cdots s_{t-1,n}\}$. Then a_t is an n-dimensional vector where the i^{th} element is 1 if a subset of the points in s_t match to a subset of the points in $s_{t-1,i}$ and 0 otherwise. The data association indicates that these points belong to the same object or group of objects viewed at different points in time.

We assume that the points in the previous frame have already been segmented and that the most likely data association has already been computed. For previous frames, we only allow the data association a_{t-k} (for $k \geq 1$) to take on 2 possible values: either all elements are 0, or the element corresponding to the single most likely data association is equal to 1 (and the rest are 0). Thus the segment s_{t-1} is associated with an entire history of segments $s_{1..t-1}$, given by the most likely data association at each time step.

We do not know the true data association for the current frame, so we sum over different possible data associations:

$$p(N_t|z_t, S_{1..t-1}) = \eta_1 \sum_{a_t} p(z_t|N_t, a_t, S_{1..t-1})\, p(N_t, a_t|S_{1..t-1}) \tag{1}$$

where we have expanded the probability using Bayes' rule, with the normalization constant $\eta_1 = 1/p(z_t|S_{1..t-1})$. Although a_t can take on 2^n possible values, we show a tractable way to compute this in Section VI-A.

If we condition on the data association a_t, then we only need to be concerned with the matching segments from the previous frames $s_{1..t-1}$ and their features $z_{1..t-1}$ rather than the entire previous segmentation $S_{1..t-1}$. Thus we have that

$$p(z_t|N_t, a_t, S_{1..t-1}) = p(z_t|N_t, a_t, z_{1..t-1}). \tag{2}$$

The probability $p(z_t|N_t, a_t, z_{1..t-1})$ incorporates the evidence from our current observation z_t and will be explained in Section IV-B.

We can expand the final term of equation 1 using the previous segmentation estimate as

$$p(N_t, a_t|S_{1..t-1}) = \eta_2 \sum_{N_{t-1}} p(N_t|a_t, N_{t-1})\, p(N_{t-1}|S_{1..t-1}) \tag{3}$$

where we treat $p(a_t|N_{t-1}, S_{1..t-1}) = p(a_t)$ as a constant η_2, and we have simplified the equation using the conditional independence assumption

$$p(N_t|a_t, N_{t-1}, S_{1..t-1}) = p(N_t|a_t, N_{t-1}). \tag{4}$$

The variable $N_{t-1} = true$ if segment s_{t-1} is only one object and false otherwise. Thus N_{t-1} is independent of all segments from $t-1$ other than segment s_{t-1} (and its features z_{t-1}), so we can rewrite the final term from equation 3 as

$$p(N_{t-1}|S_{1..t-1}) = p(N_{t-1}|z_{t-1}, S_{1..t-2}) \tag{5}$$

The expression on the right of equation 5 is similar to the expression on the left side of equation 1 for the previous time step, so we compute this recursively, allowing us to accumulate segmentation evidence across time.

The term $p(N_t|a_t, N_{t-1})$ in equation 3 represents the probability that a segmentation remains consistent across time. Although a segmentation usually remains consistent, sometimes a single object can split to become more than one object. For example, a person might dismount from a bicycle; in this case, a single object (person-riding-bicycle) now becomes two separate objects as the person moves away from the bicycle. Conversely, a person can enter into a car, in which case two objects (person and car) are now treated as a single object. We treat this probability as a constant, determined using cross-validation on our training set.

The overall probability can now be written as:

$$p(N_t|z_t, S_{1..t-1}) = \eta \sum_{a_t} p(z_t|N_t, a_t, z_{1..t-1}) \sum_{N_{t-1}} p(N_t|a_t, N_{t-1})\, p(N_{t-1}|z_{t-1}, S_{1..t-2}) \tag{6}$$

where $\eta = \eta_1 \eta_2$. The three terms in equation 6 represent our current measurement model, temporal consistency, and our previous estimate, respectively. Note that these terms correspond to those in the standard update equations for a Bayes filter [25].

B. Measurement Model

The term $p(z_t|N_t, a_t, z_{1..t-1})$ of equation 6 represents the probability of our current observation's features z_t, known as a "measurement model." For segment s_t, we define $z_t = \{z_t^p, z_t^s\}$ as a set of features for this segment, where z_t^p is the position of the segment's centroid and z_t^s are a set of features that represent the segment's shape (defined below).

Assuming that the position and shape are independent, we can now write this as

$$\begin{aligned} p(z_t|N_t, a_t, z_{1..t-1}) &= p_P(z_t^p|N_t, a_t, z_{1..t-1})\, p_S(z_t^s|N_t, a_t, z_{1..t-1}) \\ &= p_P(z_t^p|a_t, z_{1..t-1})\, p_S(z_t^s|N_t). \end{aligned} \tag{7}$$

We have eliminated the dependency of the first term on N_t since the centroid's position is not dependent on the number of objects. We have further simplified our shape probability computation by assuming that $p_S(z_t^s|N_t, a_t, z_{1..t-1}) = p_S(z_t^s|N_t)$. Although in reality our current shape features z_t^s depend on the previous shape features $z_{1..t-1}^s$, this conditional independence assumption will greatly simplify our calculations. These two terms will be explained in more detail below. The overall probability can now be written as:

$$p(N_t|z_t, S_{1..t-1}) = \eta \sum_{a_t} p_P(z_t^p|a_t, z_{1..t-1})\, p_S(z_t^s|N_t) \sum_{N_{t-1}} p(N_t|a_t, N_{t-1})\, p(N_{t-1}|z_{t-1}, S_{1..t-2}). \tag{8}$$

1) Position Probability: In equation 8, we sum over the different possible data associations a_t. Each data association represents a matching between the set of points s_t and a previous set of points, s_{t-1} (where s_{t-1} might be a union of one or more of our previous segments). However, some of the data associations are more likely than others. The probability $p_P(z_t^p|a_t, z_{1..t-1})$ in equation 8 will allow us to weight more highly the terms in the summation corresponding to more likely data associations and down-weight the terms corresponding to less likely data assocations.

The position probability, $p_P(z_t^p|a_t, z_{1..t-1})$, is computed from the position of the centroid z_t^p compared to its predicted position, based on the estimated object velocity. To compute the position probability, we integrate over the different object velocities x_t:

$$p_P(z_t^p|a_t, z_{1..t-1}) = \int_{x_t} p(z_t^p|x_t)\, p(x_t|a_t, z_{1..t-1})\, \mathrm{d}x_t. \tag{9}$$

Note that this computation is related to the Bayesian update of a Kalman Filter [25]. To compute this, we assume that we have modeled the previous distribution over object velocities $p(x_{t-1}|z_{1..t-1})$ as a Gaussian with mean μ and covariance Σ.

To compute this integral, we first perform the Kalman Filter prediction step to get a prediction of the object's current position and velocity (see [25] for details). The resulting prediction is given by a Gaussian with mean $\bar{\mu} = [\bar{\mu}_v, \bar{\mu}_x]$ and covariance $\bar{\Sigma}$, with the predicted mean velocity and position given by $\bar{\mu}_v$ and $\bar{\mu}_x$ respectively. If we model the measurement probability $p(z_t^p|x_t)$ by a Gaussian with measurement noise Q, then equation 9 is the convolution of two Gaussians, which results in another Gaussian. The probability is thus given by

$$p_P(z_t^p|a_t, z_{1..t-1}) = \mathcal{N}(z_t^p; \bar{\mu}, \bar{\Sigma} + Q). \tag{10}$$

To compute this probability, we first shift the point cloud from segment s_{t-1} to its predicted position $\bar{\mu}_x$ to get the predicted point cloud $\bar{s}_t$. Next, we find the point $\bar{p}_i \in \bar{s}_t$ that is nearest to the centroid z_t^p. We then compute the probability as

$$p_P(z_t^p|a_t, z_{1..t-1}) = C_1 \exp\Big(-\frac{1}{2}(z_t^p - \bar{p}_i)^T \Sigma^{-1} (z_t^p - \bar{p}_i) \Big) \tag{11}$$

where $C_1 = \eta_3 |\Sigma|^{-1/2}$ and $\Sigma = \bar{\Sigma} + Q$ (and η_3 is a normalization constant). We find that using the entire point cloud $\bar{s}_t$ is more robust than simply using its centroid.

2) Shape Probability: The second term of equation 8, $p_S(z_t^s|N_t)$, is the probability of observing a measurement with shape features z_t^s (defined below) conditioned on the number of objects in the points s_t. This probability is the main term in equation 8 that directly relates to the number of objects; the other terms are used to accumulate segmentation evidence across time. Without any semantic information, this probability is based only on the distance between points, as is normally used in spatial clustering [2, 11, 24]. However, in our framework, this probability is accumulated across time to give a more accurate segmentation, as described in Section IV-A.

To compute this probability, we define z_t^s to be the largest gap d within the points s_t. Formally, we define d to be the largest distance such that the points in s_t can be partitioned into two disjoint subsets, $s_{t,1}$ and $s_{t,2}$, where

$$||p_i, p_j|| \geq d, \quad \forall p_i \in s_{t,1}, p_j \in s_{t,2}, \tag{12}$$

where $||\cdot||$ is the Euclidean distance. A large value for the distance d implies that s_t is more likely to consist of more than one object, whereas a smaller value of d could be caused by an oversegmentation of a single object. In practice, we only compute d when we are considering merging two segments together (see Section V-B), in which case d is the distance between the nearest points between the two segments (computed with a kd-tree); otherwise, we use a constant value for d, which is chosen using cross-validation.

Using statistics from our training set, we find that the distance d between pairs of different objects varies according to an exponential distribution

$$p_D(d|\neg N_t) = \frac{1}{\mu_D} \exp(-d/\mu_D), \tag{13}$$

where μ_D is a parameter determined by our training set. On the other hand, occlusions or sensor noise can cause a single object to be oversegmented into multiple pieces that may have a gap of d between the pieces. We model the probability of observing an oversegmentation of a single object with a gap of d between two pieces by an exponential distribution

$$p_D(d|N_t) = \frac{1}{\mu_S} \exp(-d/\mu_S), \tag{14}$$

where μ_S is determined using cross-validation on our training set. Typically we have that $\mu_D > \mu_S$ based on the intuition described above: a larger gap between points is indicative that the points may belong to multiple objects. The shape probability is then set to be equal to this distribution:

$$p_S(z_t^s|N_t) = p_D(d|N_t). \tag{15}$$

C. Semantics

In the previous sections we have used spatial and temporal information (accumulating segmentation evidence across time) to estimate the segmentation. In this section we show how to incorporate semantic information: if we can recognize the type

of object for a given segment, this should give us additional information about how to segment the scene. On the other hand, if we cannot recognize the object, then the probabilistic model naturally reverts to the model of Sections IV-A and IV-B that incorporates spatial and temporal information.

We assume that we have a segment classifier $p(c|s_{1..t})$ that computes the probability that the set of segments $s_{1..t}$ is some class c. To use the classifier, we must sum over class labels. Equation 1 can now be rewritten as

$$p(N_t|z_t, S_{1..t-1}) = \eta_1 \sum_{a_t} \sum_{c} p(z_t|N_t, a_t, c, S_{1..t-1}) \; p(N_t, a_t, c|S_{1..t-1}). \quad (16)$$

The second term of equation 16 can be expanded as

$$p(N_t, a_t, c|S_{1..t-1}) = p(N_t, a_t|S_{1..t-1})\, p(c|N_t, a_t, S_{1..t-1}). \quad (17)$$

For the second term of equation 17, if the data association is given then the class label c is only dependent on the corresponding matching segments $s_{1..t-1}$, so we compute this term as $p(c|a_t, s_{1..t-1})$ using the segment classifier.

We now compute a more accurate measurement probability, since we are also conditioning on the class label c. For example, we now compute equation 14 as

$$p_D(d|N_t, c) = \frac{1}{\mu_{S,c}} \exp(-d/\mu_{S,c}), \quad (18)$$

where the parameter $\mu_{S,c}$ can vary based on the class label. For example, for a car, an oversegmentation might reasonably cause a 15 cm gap between points, whereas for a single person, only a 3 cm gap might be considered reasonable. We choose $\mu_{S,c}$ for each class using cross-validation on the training set.

We also incorporate more information about the object shape into the computation of $p_S(z_t^s|N_t, c)$, since we are conditioning on the object class. Although a complex shape model can be used, for sake of efficiency we use the volumetric size of the segment s_t. We first compute the length and width (l_1, l_2) of a bounding box fit to the segment s_t. We now define z_t^s to be the set $\{d, l_1, l_2\}$, where d was defined in Section IV-B2. For each class, we model the size of an average segment for that class by a Gaussian with parameters $\mu_{i,c}$ and $\sigma_{i,c}$ along each dimension i. However, a partially occluded segment may appear smaller than this; thus, we compute the probability along each dimension as

$$p_V(l_i|N_t, c) = \eta_4 \begin{cases} \exp\left(\frac{-(l_i-\mu_{i,c})^2}{2\sigma_{i,c}^2}\right), & \text{if } l_i \geq \mu_{i,c} \\ 1, & \text{otherwise.} \end{cases} \quad (19)$$

where η_4 is a normalization constant. If $N_t = false$ (i.e. the points in s_t consist of more than one object), we expect the segment to be bigger than average with a large variance. We thus compute this probability as

$$p_V(l_i|\neg N_t, c) = \mathcal{N}(l_i; \mu'_{i,c}, \sigma'_{i,c}) \quad (20)$$

with $\mu'_{i,c} = k_1\mu_{i,c}$ and $\sigma'_{i,c} = k_2\sigma_{i,c}$ for constants $k_1, k_2 > 1$ chosen using cross-validation on our training set. We can now replace the shape probability from equation 15 with

$$p_S(z_t^s|N_t, c) = p_D(d|N_t, c) \prod_i p_V(l_i|N_t, c). \quad (21)$$

Note that one of the classes may be "unknown." When we condition on the "unknown" class, then the size probability $p_V(l_i|N_t, c)$ will be uninformative; however, the probabilistic models from Sections IV-A and IV-B are still used. Thus, our segmentation model benefits when we recognize the object class, but we can still segment the scene even for unknown objects.

V. Segmentation Algorithm

We now describe how we use the probabilistic model described above, combining spatial, temporal, and (optionally) semantic information to improve our scene segmentation.

A. Splitting

After obtaining an initial coarse segmentation, we iterate over all of our initial segments s_t and compute the probability $p(N_t|z_t, S_{1..t-1})$ using equation 8 to determine whether the segment should be further split. Although the initial segmentation might only use spatial information, the computation of $p(N_t|z_t, S_{1..t-1})$ also incorporates semantic and temporal information, allowing us to make a more informed decision about how to segment the points in s_t. We defined $N_t = false$ if the points in s_t belong to more than one object. Thus if $p(\neg N_t|z_t, S_{1..t-1}) > 0.5$, we conclude that the segment s_t represents more than one object and should be split into smaller segments. The details of this splitting operation are explained in Section VI-C.

B. Merging

After all of our segments have been split using spatial, semantic, and temporal information, some objects may have become oversegmented. We now use our probabilistic model to determine when to merge some of these segments back together. To do this, we iterate over pairs of segments $s_{t,i}$ and $s_{t,j}$, and we propose a merge of these segments to create a single merged segment $s_{t,ij} = s_{t,i} \cup s_{t,j}$ with features $z_{t,ij}$. We then compute the probability $p(N_t|z_{t,ij}, S_{1..t-1})$ for this proposed merged segment, again using equation 8. If $p(N_t|z_{t,ij}, S_{1..t-1}) > 0.5$, then we conclude that these two segments are actually part of one object and we merge them together; otherwise, we keep these two segments separate. We continue to merge segments until no more segments can be merged, meaning that our segmentation has converged.

VI. Implementation Details

A. Approximations

In order to make our method run in real-time, we have to perform a number of approximations. For example, in equation 1, we sum over all data associations for segment s_t. However, segment s_t can match to any number of segments

from the previous frame; if there are n segments in the previous frame, then there are 2^n possible data associations.

However, it is not necessary to compute all of these associations, since we only want to know whether segment s_t consists of one object or more than object. Therefore, we simplify the problem by only considering matches to either single segments, to pairs of segments, or to no segments from the previous frame. Formally, we only consider values of a_t where 0, 1, or 2 elements are equal to 1 and the rest are equal to 0. Thus we only need to consider $n^2 + n + 1$ potential data associations, making this summation much faster to compute. Further, if a segment from the previous frame is sufficiently far away from the current segment, then it will likely not contribute very much to this probability, so we do not need to consider this segment for data association.

Additionally, when we match to a single segment in the previous frame, we use the prior probability in the recursive computation, i.e. $p_1(N_{t-1}|z_{t-1}, S_{1..t-2}) \approx p_1(N_{t-1})$, which speeds up our method at little cost in accuracy. We still compute the probability $p(N_{t-1}|z_{t-1}, S_{1..t-2})$ recursively when we consider a data association to multiple prior segments.

We precompute the class probability $p(c|a_t, s_{1..t-1})$ for each individual segment s_{t-1}. For efficiency reasons, we therefore do not use semantics when conditioning on a value of a_t such that s_t matches to more than one previous segment.

B. Lazy computation

Normally, when using a Bayes filter, we take advantage of the Markov property: when computing frame t, we only need information from frame $t-1$, so we do not need to keep any prior information. However, in our segmentation framework, we only need a subset of the information from the previous frame. Thus, we instead perform a lazy computation: we keep the last T frames of data, and we make use of the prior information only as needed for our recursive computation $p(N_{t-1}|z_{t-1}, S_{1..t-2})$. This allows our method to be much more efficient, because we do not need to compute probabilities for pairs of segments until needed by our framework.

C. Splitting Operation

As described above, we use our probabilistic framework to decide which segments s_t represent more than one object and therefore must be split into smaller segments. Once the decision to split segment s_t is made, the segment can be split using a variety of methods. We split the segment using spatial and temporal information as described below.

1) Temporal Splitting: First, we attempt to split segment s_t by matching it to segments from the previous frame. If more than one segment from the previous frame matches to segment s_t with high probability, then we split s_t based on the shape of those previous segments.

First, we find the segments $s_{t-1,1} \cdots s_{t-1,n}$ from the previous frame that have a strong match to the segment s_t in the current frame according to equation 11. Next, we compute whether these segments from the previous frame are likely to represent different objects; they may have merely been the result of a previous oversegmentation. To do this, we take all pairs of these matching segments $s_{t-1,i}$ and $s_{t-1,j}$ and we merge each pair into a single segment $s_{t-1,ij}$. We then compute $p(N_{t-1}|z_{t-1,ij}, S_{1..t-2})$ for this merged segment using equation 8. If $p(\neg N_{t-1}|z_{t-1,ij}, S_{1..t-2}) > 0.5$, then we assume that $s_{t-1,i}$ and $s_{t-1,j}$ belong to two different objects and can be used for temporal splitting. We then find the set of segments $s_{t-1,1} \cdots s_{t-1,m}$ with a strong match to s_t that are all estimated to represent different objects.

To perform the split, for each point in the current frame $p_i \in s_t$, we compute the probability $p_P(p_i|a_t, z_{1..t-1,j})$ from equation 11 to match this point to each segment $s_{t-1,j}$ from the previous frame. We assign point p_i to the segment with the highest probability, and then the point assignments define a set of m new segments that get created. Using this approach, we split segment s_t into m new segments by matching the points in s_t to segments from the previous frame.

2) Spatial Splitting: The temporal splitting process described above may not lead to the segment s_t being split into multiple segments, for various reasons. The segment s_t may not match to any previous segment with high probability, or these previous segments could have resulted from an oversegmentation.

In such a case, we will split segment s_t using spatial information, using a smaller threshold than was used in our initial segmentation. Note that we would not want to apply such a small threshold to all segments; however, we have determined using semantic and temporal information that segment s_t is too large, so a smaller threshold may be used. In our implementation, we split segment s_t using the Euclidean clustering method in PCL [18, 19], with a clustering threshold based on the sensor resolution r, ensuring that all points in each segment are at least $k_3\, r$ from all points in a neighboring segment, for some constant k_3 chosen using our training set.

D. Pre- and Post-processing

As a pre-processing step, we remove the points that belong to the ground using the method of Montemerlo et al. [15]. The initial coarse segmentation for our method is obtained using the Flood-fill method of Teichman et al. [24], extended to use a 3D obstacle grid. After segmentation, we perform data association using the position probability of Section IV-B1. We next estimate a velocity for each object using the tracker of Held et al. [7]. Finally, we classify these segments using the boosting framework of Teichman and Thrun [23], and we use the same classes as in that reference: pedestrians, bicyclists, cars, and background (which we refer to as unknown). The velocity and classification estimates are used to provide temporal and semantic cues to improve the segmentation, as we have described. We find the components listed here to be both fast and accurate; however, any method for ground-detection, tracking, and 3D classification may be used.

VII. Evaluation

Below we discuss our quantitative evaluation. A video demonstrating our segmentation performance can be found on the project website.

Dataset: We evaluate our segmentation method on the KITTI tracking dataset [3, 4, 5]. The KITTI dataset was collected from a moving car on city streets in Karlsruhe, Germany for the purpose of evaluating perception for autonomous driving. The 3D point clouds are recorded with a Velodyne HDL-64E 64-beam LIDAR sensor, which is often used for autonomous driving [15, 16, 26]. This dataset is publicly available and consists of over 42,000 3D bounding box labels on almost 7,000 frames across 21 sequences. Although the bounding boxes do not create point-wise accurate segmentation labels, the segments produced by these bounding boxes are sufficiently accurate to evaluate our method.

We use 2 of these sequences to train our method and select parameters, and the remaining 19 sequences are used for testing and evaluation. The specific parameter values chosen using our training set are listed in the project website.

Although the KITTI tracking dataset has been made publicly available, the dataset has typically been used to evaluate only tracking and object detection rather than evaluating segmentation directly. However, segmentation is an important step of a 3D perception pipeline, and errors in segmentation can cause subsequent problems for other components of the system. Because the KITTI dataset is publicly available, we encourage other researchers to evaluate their 3D segmentation methods on this dataset using the procedure that we describe here.

Evaluation Metric: The output of our method is a partitioning of the points in each frame into disjoint subsets ("segments"), where each segment is intended to represent a single object instance. The KITTI dataset has labeled a subset of objects with a ground-truth bounding box, indicating the correct segmentation. We wish to evaluate how well our segmentation matches the ground-truth for the labeled objects.

To evaluate our segmentation, we assign a best-matching segment to each ground-truth bounding box. For each ground-truth bounding box gt, we find the set of non-ground points within this box, C_{gt}. For each segment s, let C_s be the set of points that belong to this segment. We then find the best-matching segment to this ground-truth bounding box by computing

$$s = \arg\max_{s'} |C_{s'} \cap C_{gt}| \tag{22}$$

The best-matching segment is then assigned to this ground-truth bounding box for the evaluation metrics described below.

We describe on the project website how the intersection-over-union metric on 3D points [26] is non-ideal for autonomous driving because this score penalizes undersegmentation errors more than oversegmentation errors. Instead, we propose to count the number of oversegmentation and undersegmentation errors directly. Roughly speaking, an undersegmentation error occurs when an object is segmented together with a nearby object, and an oversegmentation error occurs when a single object is segmented into multiple pieces. More formally, we count the fraction of undersegmentation errors as

$$U = \frac{1}{N} \sum_{gt} \mathbb{1}\Big(\frac{|C_s \cap C_{gt}|}{|C_s|} < \tau_u\Big) \tag{23}$$

where $\mathbb{1}$ is an indicator function that is equal to 1 if the input is true and 0 otherwise and where τ_u is a constant threshold. We count the fraction of oversegmentation errors as

$$O = \frac{1}{N} \sum_{gt} \mathbb{1}\Big(\frac{|C_s \cap C_{gt}|}{|C_{gt}|} < \tau_o\Big), \tag{24}$$

where τ_o is a constant threshold. In our experiments, we choose $\tau_u = 0.5$ to allow for minor undersegmentation errors as well as errors in the ground-truth labeling. We use $\tau_o = 1$, since even a minor oversegmentation error causes a new (false) object to be created. We do not evaluate oversegmentations or undersegmentations when two ground-truth bounding boxes overlap. In such cases, it is difficult to tell whether the segmentation result is correct without more accurate ground-truth segmentation annotations (i.e. point-wise labeling instead of bounding boxes). Examples of undersegmentation and oversegmentation errors are shown in Figure 2.

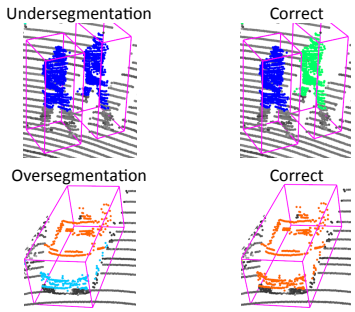

Fig. 2. Examples of an undersegmentation error (top) and an oversegmentation error (bottom). Each color denotes a single segment, and the ground-truth annotations are shown with a purple box, where each box represents a single object instance. (Best viewed in color)

We then compute an overall error rate based on the total number of undersegmentation and oversegmentation errors, as

$$E = U + \lambda_c O \tag{25}$$

where λ_c is a class-specific weighting parameter that penalizes oversegmentation errors relative to undersegmentation errors. For our experiments we simply choose $\lambda_c = 1$ for all classes, but λ_c can also be chosen for each application based on the effect of oversegmentation and undersegmetation errors for each class on the final performance.

VIII. Results

Baselines: We compare our segmentation method to a number of baseline methods. First, we compare to the Flood-fill method of Teichman et al. [24], which is also used as a pre-processing step for our method. We also compare to the Euclidean Clustering (EC) method of Rusu [18], which was applied to segmentation of urban scenes in Ioannou et al. [10]. We evaluate the method of Rusu [18] using cluster tolerances of 0.1 m, 0.2 m, or 1 m (EC 0.1, EC 0.2, EC 1).

Finally, we compare to the method of Wang et al. [26], which also incorporates semantic information into the segmentation. Because the code of Wang et al. [26] was not available,

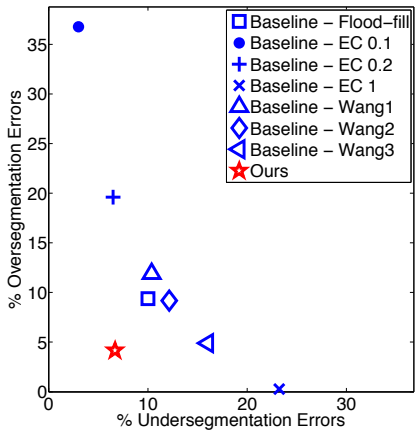

Fig. 3. Oversegmentation vs Undersegmentation errors for a variety of segmentation methods. Best-performing methods appear in the lower-left corner of the graph.

we re-implemented their method. We classify segments using the method of Teichman and Thrun [23]. The initial segmentation for this method is obtained using Flood-fill [24], and the post-processing step is obtained using Euclidean Clustering with a dynamic cluster tolerance chosen based on the sensor resolution. This setup was the best-scoring implementation of their method that we were able to achieve. We experiment with three different parameter settings for this method (Wang1, Wang2, Wang3).

Baseline Comparison: Our method operates on the full point-cloud in real-time, and we improve the segmentation at all distances. However, our analysis focuses on segments within 15 m because these are the objects that our autonomous vehicle will most immediately encounter.

The segmentation evaluation is shown in Figure 3, for segments within 15 m. Combining oversegmentation and undersegentation errors, our method has the lowest error rate of 10.2%, compared to the Flood-fill baseline [23] with 19.4%. Our method thus reduces the total number of oversegmentation and undersegmentation errors by 47%, thus demonstrating the benefit of combining spatial, temporal, and semantic information for 3D segmentation. Our method runs in real-time, taking an average of 33.7 ms per frame, or 29.7 Hz, not including the initial coarse segmentation of [24] which requires an additional 19.6 ms.

The Euclidean Clustering baselines achieved error rates of 23%, 26%, or 40% (for different parameter settings). The method of Wang et al. [26] achieved error rates of 20.9%, 21.3% or 22.3% (for different parameter settings). Although this method incorporates semantic information, they make assumptions that do not hold in crowded environments. Their method assumes that, after removing the ground points and background objects, the remaining objects can be segmented using a spatial clustering approach. Their underlying assumption is that foreground objects tend to remain separated from each other in 3D space. Although this assumption may hold true for cars, it does not hold true for pedestrians, which often stand in crowds. Further, pedestrians and bicyclists may move close to parked cars, or they may move past cars at a crosswalk, causing the assumptions of this method to fail to hold.

Ablative analysis: To understand the cause for the improvement in performance, we separately analyze the benefit of using only temporal information and only semantic information, shown in Table I. As can be seen, adding either temporal or semantic information alone results in small improvements in performance. However, the largest gains are achieved by combining spatial, semantic, and temporal information, thus showing that all three components are important for the accuracy of our system.

Method	% Errors (< 15 m)	% Errors (all)
Spatial only [24]	19.4	11.6
Spatial + Temporal	16.4	10.8
Spatial + Semantic	15.4	10.8
Spatial + Semantic + Temporal	**10.2**	**8.2**

TABLE I
SEGMENTATION ACCURACIES FOR VARIOUS VERSIONS OF OUR METHOD.

Performance Analysis: We evaluate the effect of classification on segmentation by separately evaluating our segmentation performance on objects that we classify correctly compared to objects that we classify incorrectly. For objects that we classify correctly, we reduce the segmentation errors by 48.4%, compared to 5.8% for objects that we classify incorrectly (for objects at all distances). Thus, improving the classification accuracy can lead to significant gains in our segmentation performance, when using our method which combines spatial, temporal, and semantic information.

Effect on Tracking: Finally, we show the effect of segmentation accuracy on object tracking. In the KITTI tracking dataset, the ground-truth bounding boxes are also annotated with object IDs, so we are able to compute the number of ID switches of our system. Using our segmentation method, we get 25% fewer ID switches compared to the baseline [24] Thus, accurate segmentation is also important for accurate tracking. Because segmentation occurs at the beginning of the perception system, errors in segmentation can propagate throughout the rest of the system, and improving segmentation will improve other aspects of the system such as tracking, as we have shown.

IX. CONCLUSION

To perform accurate 3D segmentation, it is not sufficient to only use the spatial distance between points, as there will be many ambiguous cases caused by occlusions or neighboring objects. Our method is able to incorporate spatial, semantic, and temporal cues in a coherent probabilistic framework for 3D segmentation. We are able to significantly reduce the number of segmentation errors while running in only 34 ms. Robotic systems should thus combine spatial, temporal, and semantic information to achieve accurate real-time segmentation and more reliable perception in dynamic environments.

References

[1] Asma Azim and Olivier Aycard. Detection, classification and tracking of moving objects in a 3d environment. In *Intelligent Vehicles Symposium (IV), 2012 IEEE*, pages 802–807. IEEE, 2012.

[2] Bertrand Douillard, James Underwood, Noah Kuntz, Vsevolod Vlaskine, Alastair Quadros, Peter Morton, and Alon Frenkel. On the segmentation of 3d lidar point clouds. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 2798–2805. IEEE, 2011.

[3] Jannik Fritsch, Tobias Kuehnl, and Andreas Geiger. A new performance measure and evaluation benchmark for road detection algorithms. In *International Conference on Intelligent Transportation Systems (ITSC)*, 2013.

[4] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[5] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *International Journal of Robotics Research (IJRR)*, 2013.

[6] Saurabh Gupta, Ross Girshick, Pablo Arbeláez, and Jitendra Malik. Learning rich features from rgb-d images for object detection and segmentation. In *Computer Vision–ECCV 2014*, pages 345–360. Springer, 2014.

[7] David Held, Jesse Levinson, Sebastian Thrun, and Silvio Savarese. Combining 3d shape, color, and motion for robust anytime tracking. In *Proceedings of Robotics: Science and Systems*, Berkeley, USA, July 2014.

[8] Evan Herbst, Xiaofeng Ren, and Dieter Fox. Rgb-d flow: Dense 3-d motion estimation using color and depth. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 2276–2282. IEEE, 2013.

[9] Steven Hickson, Stan Birchfield, Irfan Essa, and Helen Christensen. Efficient hierarchical graph-based segmentation of rgbd videos. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 344–351. IEEE, 2014.

[10] Yani Ioannou, Babak Taati, Robin Harrap, and Michael Greenspan. Difference of normals as a multi-scale operator in unorganized point clouds. In *3D Imaging, Modeling, Processing, Visualization and Transmission (3DIMPVT), 2012 Second International Conference on*, pages 501–508. IEEE, 2012.

[11] Klaas Klasing, Dirk Wollherr, and Martin Buss. A clustering method for efficient segmentation of 3d laser data. In *ICRA*, pages 4043–4048, 2008.

[12] Abhijit Kundu, Yin Li, Frank Dellaert, Fuxin Li, and James M Rehg. Joint semantic segmentation and 3d reconstruction from monocular video. In *Computer Vision–ECCV 2014*, pages 703–718. Springer, 2014.

[13] Kevin Lai, Liefeng Bo, Xiaofeng Ren, and Dieter Fox. Detection-based object labeling in 3d scenes. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 1330–1337. IEEE, 2012.

[14] John Leonard, Jonathan How, Seth Teller, Mitch Berger, Stefan Campbell, Gaston Fiore, Luke Fletcher, Emilio Frazzoli, Albert Huang, Sertac Karaman, et al. A perception-driven autonomous urban vehicle. *Journal of Field Robotics*, 25(10):727–774, 2008.

[15] Michael Montemerlo, Jan Becker, Suhrid Bhat, Hendrik Dahlkamp, Dmitri Dolgov, Scott Ettinger, Dirk Haehnel, Tim Hilden, Gabe Hoffmann, Burkhard Huhnke, et al. Junior: The stanford entry in the urban challenge. *Journal of field Robotics*, 25(9):569–597, 2008.

[16] Frank Moosmann and Christoph Stiller. Joint self-localization and tracking of generic objects in 3d range data. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 1146–1152. IEEE, 2013.

[17] Anna Petrovskaya and Sebastian Thrun. Model based vehicle detection and tracking for autonomous urban driving. *Autonomous Robots*, 26(2-3):123–139, 2009.

[18] Radu Bogdan Rusu. *Semantic 3D Object Maps for Everyday Manipulation in Human Living Environments*. PhD thesis, Computer Science department, Technische Universitaet Muenchen, Germany, October 2009.

[19] Radu Bogdan Rusu and Steve Cousins. 3D is here: Point Cloud Library (PCL). In *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, May 9-13 2011.

[20] Sabyasachi Sengupta, Eric Greveson, Ali Shahrokni, and Philip HS Torr. Urban 3d semantic modelling using stereo vision. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 580–585. IEEE, 2013.

[21] Luciano Spinello, Kai Oliver Arras, Rudolph Triebel, and Roland Siegwart. A layered approach to people detection in 3d range data. In *AAAI*, 2010.

[22] Jörg Stückler and Sven Behnke. Efficient dense rigid-body motion segmentation and estimation in rgb-d video. *International Journal of Computer Vision*, pages 1–13, 2015.

[23] Alex Teichman and Sebastian Thrun. Group induction. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pages 2757–2763. IEEE, 2013.

[24] Alex Teichman, Jesse Levinson, and Sebastian Thrun. Towards 3d object recognition via classification of arbitrary object tracks. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 4034–4041. IEEE, 2011.

[25] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic robotics*. MIT press, 2005.

[26] Dominic Zeng Wang, Ingmar Posner, and Paul Newman. What could move? finding cars, pedestrians and bicyclists in 3d laser data. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 4038–4044. IEEE, 2012.

[27] Nicolai Wojke and M Haselich. Moving vehicle detection and tracking in unstructured environments. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 3082–3087. IEEE, 2012.

[28] Chenliang Xu, Caiming Xiong, and Jason J Corso. Streaming hierarchical video segmentation. In *Computer Vision–ECCV 2012*, pages 626–639. Springer, 2012.

[29] Chenliang Xu, Spencer Whitt, and Jason J Corso. Flattening supervoxel hierarchies by the uniform entropy slice. In *Computer Vision (ICCV), 2013 IEEE International Conference on*, pages 2240–2247. IEEE, 2013.