Spatial Semantic Embedding Network: Fast 3D Instance Segmentation with Deep Metric Learning

Dongsu Zhang
Seoul National University
96lives@snu.ac.kr

Junha Chun
Seoul National University
nikriz@snu.ac.kr

Sang Kyun Cha
Seoul National University
chask@snu.ac.kr

Young Min Kim
Seoul National University
youngmin.kim@snu.ac.kr

Abstract—We propose spatial semantic embedding network (SSEN), a simple, yet efficient algorithm for 3D instance segmentation using deep metric learning. The raw 3D reconstruction of an indoor environment suffers from occlusions, noise, and is produced without any meaningful distinction between individual entities. For high-level intelligent tasks from a large scale scene, 3D instance segmentation recognizes individual instances of objects. We approach the instance segmentation by simply learning the correct embedding space that maps individual instances of objects into distinct clusters that reflect both spatial and semantic information. Unlike previous approaches that require complex pre-processing or post-processing, our implementation is compact and fast with competitive performance, maintaining scalability on large scenes with high resolution voxels. We demonstrate the state-of-the-art performance of our algorithm in the ScanNet 3D instance segmentation benchmark on AP score.

1 We plan on releasing the code for reproducibility. The code will be released on https://github.com/96lives/ssen

Fig. 1: Sample results of our method. Given the input scene, our method outputs labels for individual object instances. Our pipeline transforms the problem of instance segmentation into clustering in the embedded space. The bottom right image visualizes the distance to the cluster mean in the embedded space. The black points represent embedding farther from the cluster mean, and we can observe that black points are physically located when nearby objects are in close contact with each other. This shows us that our embedding captures not only semantic information of the scene but also spatial information.

I. INTRODUCTION

Scene understanding provides information on layout of multiple elements in the scene in terms of their locations, shapes, and poses in addition to possible relationship between them. The extracted information is crucial to perform subsequent stages for intelligent applications, including high-level interaction or context-aware service [39] in addition to autonomous driving, navigation, localization, or grasping. Especially, 3D instance segmentation directly labels the measurement points in 3D as individual instances of objects, providing the exact location and extent of individual objects in the physical space.

3D instance segmentation started to gain attention relatively recently compared to the 2D instance segmentation, as the available deep learning architecture or datasets in 3D have been introduced in the past couple of years. The first 3D instance segmentation approach was suggested by [34] using the PointNet [29] architecture on a rather small scale. Larger-scale 3D scene understanding regresses bounding boxes of individual objects [37] or the centers of individual objects [28]. For 3D instance segmentation, many recent works take advantage of 3D measurements for post-processing, mainly using the actual 3D shape of the proposed objects [38] or finding spatially connected components [20]. On the other hand, our framework does not require additional post-processing steps demonstrating that learning the correct embedding is sufficient for instance segmentation.

We address the task of 3D semantic instance segmentation by training a deep neural network [2], [11] for metric learning, called Spatial Semantic Embedding Network (SSEN). The network is trained to embed each voxel to a feature space where embeddings from the same instance are closely located, while those from different instances are placed further away [5]. As our algorithm effectively learns both semantic and spatial information, the instance segmentation of complex geometric shapes can be converted as a problem of simple clustering in the embedding space. We adapt to the different sizes of individual instances by applying the hierarchical density-based clustering [1] in the embedded space. There is no post-processing in 3D Euclidean space, which is common in previous works, for
example, iterations of multiple hypotheses, non-maximum suppression, or adaptive thresholds [20]. As a consequence, our algorithm is more than two times faster than the state-of-the-art instance segmentation approach [37].

Our embedding space contains semantic information, robust to physical contact between objects, complex 3D topology or noise including occlusions while maintaining the overall spatial structure of the 3D scene, Fig. [1] In short, our contribution can be summarized as the following: (1) We suggest a highly scalable deep metric learning approach that turns an instance segmentation problem into a simple clustering in the embedding space which is the fastest method to our knowledge; and (2) we demonstrate that our algorithm, without any fine-tuning or explicit manipulation in the original 3D space, can achieve the state-of-the-art results in the ScanNet dataset on AP [4].

II. Related Works

We first review relevant previous works on 2D object recognition with a special focus on metric learning and instance segmentation. For an extensive list of related works, we would like to refer to recent surveys on semantic segmentation [21], [9] or metric learning [16], [19]. Then we describe neural network architecture for 3D data and the recent extensions into 3D instance segmentation.

A. 2D Object Recognition via Bounding Box Regression

The mainstream method of object detection is to first create region proposals with bounding boxes and then to refine them while classifying the object inside the bounding box [10], [32], [31], [25]. For instance segmentation, He et al. [14] propose a method to perform pixel-wise masking from obtaining the bounding box of the object. While these methods result in the state-of-the-art performance for object recognition, nearby objects are sometimes misclassified as a single instance of an object, due to the overlapping of bounding boxes. These cases raise a question of whether methods based on the bounding box detection and refinement are suitable for object recognition [33].

B. 2D Instance Segmentation via Metric Learning

The approach by Fathi et al. [8] is one of the first papers to perform instance segmentation with deep metric learning. They map each pixel to an embedding space where the pixels from the same instance are closely located while pixels from different instances are far from each other. The neural network is trained extensively calculating the similarity for pairs of sampled pixels. A more scalable way to perform metric learning for instance segmentation is proposed by Brabandere et al. [5]. They propose a discriminative loss function for deep metric learning that consists of inter-class loss and intra-class loss. The inter-class loss pulls the embeddings of pixels of the same instances towards the means of the corresponding instance embeddings, or cluster centers. The intra-class loss pushes the embedding centers of different instances far from each other. This work is more efficient in terms of both memory and time compared to [8], since no pairwise similarity between the pixels needs to be computed. Also, this induces a far richer gradient using all the pixels in the image.

C. Deep Neural Networks on Point Cloud

While most of the deep learning approaches using 2D images are based on CNN-based architectures [22], [18], 3D deep neural networks process different formats of 3D data and are limited in resolution. Starting from PointNet [29], networks that directly process 3D point cloud measurements have been proposed. The networks demonstrate good performance in segmentation or recognition in small-scale scenes with relatively simple structure [30], [36], [23]. However, recent works on sparse convolution [12], [2] on voxelized point clouds have shown that neural networks in 3D data can be efficiently computed allowing the network to go deeper. Not only is this method efficient in both memory and time, but sparse convolution [12], [2] has also outperformed all other point cloud neural networks on ScanNet semantic segmentation benchmark [4].

D. 3D Point Cloud Instance Segmentation

Compared to the recognition or segmentation for images, the instance segmentation in 3D data has acquired its attention lately with the recent advance of 3D deep learning architecture. The work by Wang et al. [34] is the first deep learning algorithm to perform instance segmentation on point cloud by grouping the points based on the similarity matrix. Wang et al. [35] and Pham et al. [27] are the first to apply deep metric learning on 3D indoor instance segmentation. However, both of them divided a large scene of point cloud into small overlapping windows to apply post-processings (such as k-means, CRF or mean-shift), which required a considerable time in merging the small windows. Lahoud et al. [20] also try deep metric learning for 3D instance segmentation as a mean to obtain object proposals. In addition to object proposals obtained by feature embedding, they also obtain object proposals using connected components and center predicting networks, followed by expensive non-maximum suppression. Our work is closely related to the work by DeBrabandere et al. [5], [20]. However, we show that only obtaining embedding space is enough to perform instance segmentation outperforming all other methods on the ScanNet benchmark [4] on AP while being simple and fast.

III. Spatial Semantic Embedding Network

Our system consists of two sparse convolution networks [12], [2], one for semantic segmentation, and the other for obtaining the embedding of voxels, named Spatial Semantic Embedding Network (SSEN). The input to our system is voxels converted from a large-scale raw point cloud measurements of 3D scene. The individual voxels have features \( x_k \in \mathbb{R}^f \) such as 3-D coordinates or colors. We first use semantic segmentation network to obtain the labels of each voxels. Since the backgrounds (floors and walls) are not of interest, we filter the points that are not predicted as
The overall pipeline of our method. Given the set of voxelized input $X$, the embedding network outputs the embedding of points $E$. The embedded points are clustered using HDBSCAN algorithm, where the resulting clusters correspond to points that belong to individual instances.

![Overall pipeline of our method.](image)

**TABLE I:** Average precision (AP) with different $p$ weighting for the loss function (Equation (3)) on validation set after training 50k steps. The objects with large scale such as bathtub, bookshelf, curtain, sofa have significantly higher AP when $p = 1$. However, objects with a smaller number of points such as chair, sink, and windows tend to have higher AP when $p = 0$.

| $p$ value | AP | bathtub | bed | bookshelf | cabinet | chair | counter | curtain | desk | door | other | picture | refrig | showerCart | sink | sofa | table | toilet | window |
|-----------|----|---------|-----|-----------|---------|-------|---------|---------|------|------|-------|---------|-------|------------|------|------|------|--------|-------|
| $p = 0$   | 0.292 | 0.303 | 0.299 | 0.223 | 0.191 | 0.694 | 0.010 | 0.131 | 0.055 | 0.298 | 0.279 | 0.188 | 0.368 | 0.292 | 0.281 | 0.342 | 0.812 | 0.223 |
| $p = 0.5$ | 0.309 | 0.337 | 0.317 | 0.252 | 0.174 | 0.682 | 0.004 | 0.152 | 0.087 | 0.272 | 0.304 | 0.174 | 0.426 | 0.258 | 0.279 | 0.255 | 0.399 | 0.209 |

On the other hand, the inter-instance loss pulls the points of the same instance together.

$$L_{inter} = \frac{1}{|I|} \sum_{i \in I} \left( \sum_{j \in I, j \neq i} |E_i|^p l_i \right)$$

where $l_i$ represents the average of inter-instance loss for the $i$-th instance.

$$l_i = \frac{1}{|E_i|} \sum_{e_k \in E_i} \left( \frac{||\mu_i - e_k||}{\delta_{inter}} \right)$$

Similar to the hinge function in Equation (2), the average loss of an individual instance $l_i$ is considered only for the elements that deviate farther than a threshold $\delta_{inter}$ from the cluster mean $\mu_i$. Taking advantage of the mean instead of sampled pair-wise instances [5], the calculation is efficient in terms of required computation and memory. We did not use the regularization loss used in [5], [20], and empirically observed that the embeddings did not diverge. The regularization loss limits the magnitude of instance means, but in spirit conflicts the intuition of metric learning that generates embedding with distant, isolated clusters.

While our loss function is based on the combination of attractive and repulsive forces as other metric learning approaches [5], [20], we generalize the formulation by adjusting weights for different sizes of instances. Specifically, the average of inter-instance loss $l_i$ is weighted by $|E_i|^p$ in Equation (3), where $|E_i|$ represents the number of points that belong to the instance $i$ and $0 \leq p \leq 1$. By controlling $p$, we can stabilize the training for objects of different 3D sizes, or equivalently, clusters with different numbers of points.

If $p = 0$, our formulation is equivalent to previous works [5], [20], where $l_i$ is summed up weighted equally. This formulation is widely used in the image domain, where
the number of pixels of an instance does not represent the actual scale of the physical instance, and therefore scale invariance is desired. In the 3D setting, however, the number voxels reflects the actual size of an object. We observed that with $p = 0$, the attractive force fails to condense embedded points that belong to the instances with larger sizes, while instances with a small number of points are well condensed. On the other hand, if $p = 1$, $L_{inter}$ is weighted equally for all points.

The argument is verified by comparing the performance of the instance segmentation with $p = 0$, $p = 0.5$ and $p = 1$ (Table II). The AP is the highest at $p = 1$ for large objects, such as bathtub, bookshelf, curtain, and sofa while $p = 0$ performs the best for small instances or objects that often present occluded, for example, cabinet, chair, or counter. For all of our presented results, we use $p = 1$ which produces the best averaged performance for object instances of various scales.

B. Clustering in the Embedded Space

After successful training, the same instance points lie close to each other separated by other instances. The instance segmentation is mere clustering in the embedded space. There are several criteria that need to be considered for the choice of the clustering algorithm:

- **Clustering metric space induced by our loss function.** If our network is trained perfectly, all the embeddings of the same instances must be inside a sphere centered at $\mu_i$ with a radius of $\delta_{inter}$. The distances between means of different instances should be greater than or equal to $2\delta_{intra}$, and with triangular inequality, the distances between elements of different instances are at least $2\delta_{intra} - 2\delta_{inter} > 0$. Intuitively, the resulted embedding space is dense around the mean within the pre-defined distance threshold, with varying density depending on the sizes of objects.

- **Clustering undetermined number of objects.** Since the number of instances is undetermined in the testing stage, the clustering algorithm should be able to predict the number of instances from the embedding space. This criterion makes $k$-means clustering [13] not applicable to our set-up.

- **Efficient and robust to outliers.** The clustering algorithm should be efficient and able to robustly handle outliers in case the trained network fails to learn the proper embedding for a subset of points. Although the example in Fig. 2 outputs embeddings that are dense near the mean of instance centers, outliers exist between nearby instances. The clustering algorithm should be able to handle that flexibly.

Out of various conventional clustering algorithms [13], [3], [1], [7], we select HDBSCAN [1] as our clustering algorithm. HDBSCAN is an extension of DBSCAN [7] that uses hierarchical clustering. HDBSCAN is very efficient and can robustly handle varying density clusters, and there is no need to fix the number of clusters. On the other hand, mean-shift [3] was not scalable because multiple iterations are required, and DBSCAN performed poorly since it could not adapt to different densities.

IV. EXPERIMENTAL RESULTS

Implementation Details. The input scan is converted into voxels of size 2cm. The conversion uses the hash function implemented by Choy et al. [2] with almost no overhead. Individual voxels have the 3D coordinates and the RGB color value ($f = 6$). The output embedding dimension $d = 8$, and we set the parameters of the loss function (Section III-A) as $\gamma_{intra} = 10$, $\delta_{inter} = 0.1$, $\delta_{intra} = 0.5$. We use the implementation of the Minkowski UNet34 [2] for both semantic segmentation network and SSEN. We train our network using the ADAM optimizer [17], with the learning rate of 1e-4 for the batch size of 8. The learning rate is decayed by 0.8 for every 10k iterations. We use pretrained MinkowskiNet [2] for semantic segmentation.

We used the ScanNet [4] to train, test and show various aspects of our algorithm. For data augmentation in the training phase, we randomly jitter the color of voxels with 0.03 scaled Gaussian noise. We also randomly scale the scene by $[0.8, 1.2]$. The scene is arbitrarily rotated around the z-axis, and around the x-axis with 5°scaled Gaussian noise, with maximum 10°.

Evaluation on ScanNet Benchmark Challenge. ScanNet contains 3D reconstructed meshes of 1513 labeled indoor scenes for instance segmentation. The benchmark challenge for 3D instance segmentation is then evaluated with unlabeled 100 test scans. The performance is evaluated in terms of the average precision score. The AP score averages scores obtained with IoU thresholds ranging from 50% to 95% with a step size of 5.

The quantitative results in Table II shows that we achieve the highest score for the average precision compared to other state-of-the-art methods, namely: MTML [20], 3D-BoNet [37], PanopticFusion [26], MASC [24], 3D-SIS [15], and DPC [6]. This is remarkable considering that our algorithm does not use additional 3D information once the network maps the points into the embedding space, whereas other competitive results include post-processing stages. In comparison, we outperform MTML (FE) by a large margin, which is the version of MTML [20] that uses only feature embedding without any post-processing using connected components.

We also carefully mention that the performance measure is partially degraded by the fact that our embedding can sometimes even learn more fine-grained instances than the instance labels are provided in the ground truth. For example, in Fig. 4, we show that our algorithm separately recognizes several instances of connected couches with little distinctive feature, which are grouped as a single instance in the ground truth. While we can discuss this as a failure case, at the same, the result shows the discriminating power of our metric learning segmentation that can distinguish objects in complex configurations with interaction, occlusion or physical contact between objects.
FIG. 3: Computational time for various methods on the validation split (312 scenes) on ScanNet [4] (in seconds). The numbers except ours are excerpted from BoNet [37]. We are at least two times faster than BoNet [37] and about 50 times faster than ASIS [35]. Our run time has been averaged from five runs, and the standard deviation is within 0.8%.

FIG. 4: Sample failure cases of our method. The input scenes (first column) have two round couches that are composed of little couches. Although the ground truth is labeled them as connected (second column), our network predicts as if it were partitioned (third column). The last column visualizes the distance from the mean of the corresponding instance center with a color gradient from the cluster color (close to the cluster center) to black (far from the cluster center).

As our formulation is compact and efficient, it has high potential to be employed in large-scale scenes for high-level tasks. Figure 3 compares the computational time of the entire pipeline for other compatible methods, excerpted from reported number on [37]. We are at least two times faster than that of BoNet which is the fastest algorithm reported to our knowledge and our method outperforms all of the methods stated in performance. As mentioned before, our algorithm does not add additional processing including group merging or block merging, which often consumes excessive power. Also, our clustering algorithm (HDBSCAN [1]) is fast compared to the mean-shift [3] used for clustering in ASIS [35], which dominates the computation. While our method only took 5 minutes to obtain results from ScanNet validation set (composed of 312 scans), replacing HDBSCAN [1] with mean-shift [3] took one day with highly degraded performance.

Figure 5 presents the qualitative results of instance segmentation compared to BoNet [37] and GSPN [38]. Our su-

---

TABLE II: AP. The average precision (AP) score for ScanNet 3D instance segmentation benchmark. SSEN with 10 rotations indicate semantic labels obtained with 10 rotations and then averaged. Even without the 10 rotations, we still achieve far better results.

| Method             | AP  | AP  | AP  | AP  | AP  | AP  | AP  | AP  | AP  | AP  | AP  | AP  | AP  | AP  | AP  | AP  | AP  |
|--------------------|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| SSEN (with 10 rotations) | 0.384 | 0.378 | 0.432 | 0.214 | 0.272 | 0.455 | 0.194 | 0.019 | 0.192 | 0.022 | 0.998 | 0.022 | 0.998 | 0.022 | 0.998 | 0.022 |
| SSEN (without 10 rotation) | 0.348 | 0.775 | 0.432 | 0.214 | 0.272 | 0.455 | 0.194 | 0.019 | 0.192 | 0.022 | 0.998 | 0.022 | 0.998 | 0.022 | 0.998 | 0.022 |
| MTML              | 0.282 | 0.577 | 0.380 | 0.182 | 0.107 | 0.430 | 0.001 | 0.298 | 0.124 | 0.992 | 0.094 | 0.666 | 0.142 | 0.992 | 0.094 | 0.666 |
| MTML (FE)         | 0.171 | 0.235 | 0.154 | 0.075 | 0.036 | 0.475 | 0.001 | 0.144 | 0.121 | 0.932 | 0.094 | 0.666 | 0.142 | 0.932 | 0.094 | 0.666 |
| PanopticFusion    | 0.214 | 0.230 | 0.330 | 0.275 | 0.103 | 0.229 | 0.000 | 0.345 | 0.124 | 0.932 | 0.094 | 0.666 | 0.142 | 0.932 | 0.094 | 0.666 |
| MASC              | 0.234 | 0.403 | 0.249 | 0.113 | 0.142 | 0.412 | 0.000 | 0.345 | 0.124 | 0.932 | 0.094 | 0.666 | 0.142 | 0.932 | 0.094 | 0.666 |
| 3D-SIS            | 0.161 | 0.407 | 0.185 | 0.068 | 0.043 | 0.346 | 0.001 | 0.134 | 0.088 | 0.060 | 0.060 | 0.060 | 0.060 | 0.060 | 0.060 |
| DPC               | 0.209 | 0.405 | 0.267 | 0.230 | 0.106 | 0.285 | 0.053 | 0.278 | 0.088 | 0.130 | 0.034 | 0.097 | 0.229 | 0.181 | 0.273 | 0.269 | 0.695 | 0.124
Fig. 5: **Qualitative comparison with other methods.** We compare SSEN with other BoNet [37] and GSPN [38] on ScanNet [4] validation dataset. Our method is able to clearly distinguish different instances, predicting more precise instance labels than other methods. (Different color indicates different instance, therefore the difference in color between ground truth and prediction is not important.)

Prior performance is especially noticeable when the objects are in close proximity to each other. For example, in the third and last row of the image, BoNet [37] and GSPN [38] fails to identify the aligned chairs as individual objects, marking them as a single instance or either not marking them all. In comparison, our network clearly separates the nearby instances in the embedded space and distinguish them as different instances.

**Visualization of the Spatial-Semantic Embedding.** We claim that our embedding space can capture semantic context
while maintaining the spatial context of the input, and creates
correct clusters for instance segmentation. The effect of
spatial context can be visualized by moving the 3D position
of an instance, as shown in Fig. 6. In this figure, we copy
a single instance of the chair out of the populated chairs
in the scene and move the chair along the empty row. The
chair is highlighted in green in the input scene (first row), and
the corresponding segments in instance segmentation (middle
row) and the embedding space with dimension 3 (last row)
are circled. Firstly, note that the chairs of the same rows
are also aligned in a line in the embedding space with the
same ordering. We can verify this by comparing the color
patterns of prediction and the embedding space along the
line. Also, every row of the 3D scene is mapped parallel to
each other in the embedding space as the input scene. As
the green chair moves between the two rows (image from
the left column to right column), so does the embedding
of the chair. This example gives us high confidence that
our metric space resembles the spatial information of the
input scene. Furthermore, as the chair moves, the whole
embedding space changes pushing nearby instances away
from the moving chair. This implies that the embedding
captures the semantic information of the scene, where the
different instances are further away while the points of the
same instances are mapped together. This demonstrates that
our embedding represents not only the spatial information
but the semantic information as well.

The mapping between the physical 3D space and the
learned metric space can be also understood comparing the
distance to the center of the clusters in embedding space.
Figure 1 visualizes the distances to the cluster means for
individual points, which also partially reflects the confidence
of the clusters. We can see that the 3D connectivity or
other possible challenges can be effectively alleviated in the
learned embedding space with densely clustered points.

V. CONCLUSION

We present a simple, efficient metric learning approach for
3D instance segmentation. Given a large-scale indoor scene,
our deep network maps the voxels of the same instance object
into a condensed cluster of embedded points. The embedded
space encodes both semantic information as well as the
spatial structure of the original input. The suggested pipeline
outperforms all methods in both speed and performance.
In the future, we would like to continue analyzing the
embedded space. Also, we will further explore real-time
instance segmentation by removing the clustering stage, and
training a network in an end-to-end fashion.
