

## METHOD AND DEVICE FOR POINT CLOUD BASED OBJECT RECOGNITION

### Technical Field

[0001] The present disclosure relates to methods and devices for point cloud based object recognition.

### Background

[0002] One of the requirements for an autonomous robot to operate is the recognition of objects in its workspace. This includes detecting objects present in its workspace, identifying the objects (e.g. recognizing an object as a screw) and identifying instances of objects (e.g. being able to distinguish between multiple similar screws). A typical approach is that the robot acquires images of its workspace (e.g. RGB images but other types of sensor data is also possible), generates a point cloud from the image data and performs object recognition using the point cloud.

[0003] A point cloud is a collection of points with spatial coordinates and possibly additional features such as colour or intensity. Visualizing a point cloud in a scene provides intuitive and accurate information of 3D space. Modern technologies such as 3D imaging, photogrammetry, and SLAM (Simultaneous Localisation and Mapping) can produce coloured point clouds. The applications of point cloud cover a large variety of fields, from augmented reality, autonomous navigation, to Scan-to-BIM (Building Information Modelling) in construction.

[0004] For semantic and instance segmentation, deep learning approaches directly process the unordered point set. Other methods include the volumetric approach which requires voxelization of the input data, and multi-view approach.

[0005] Regarding Semantic Segmentation, approaches may be used which do not take into consideration the spatial context around the vicinity. On the other hand, approaches to capture a larger spatial context can be divided into three categories: point-based, graph-based and CNN-based.

[0006] Regarding the point-based approach, several methods use neighbourhood context, Recurrent Neural Network (RNN) or kernel to aggregate local information. For example, point-wise pyramid pooling to capture the spatial context at different scales may be used, and the across-block relationship may be explored with an RNN. Kernels may be used to extract the local features and train the shape context using the self-attention network.

Another approach constructs a graph in each layer dynamically in feature space, allowing point clouds being grouped even over long distances. However, the above-mentioned methods aggregate information on all the input points through each layer. Much information is overlapping, and the network becomes unnecessarily huge.

[0007] Graph-based approaches incorporate a graph convolutional neural network into proposed network structures. For example, the whole scene (e.g. robot workspace) may be partitioned into small patches based on geometric features, and then a graph convolutional neural network is applied to predict the semantic label of each patch. A local neighbourhood point set may also be transformed into the spectral domain, and the structural information is encoded in the graph topology.

[0008] Regarding the CNN-based approach, it should be noted that different from a 2D image, 3D point cloud data does not have a regular grid-like partition scheme, and different design choices can be made for kernel shape and kernel weight. On the one hand, kernel function can be regarded as a weight matrix, and the weight is defined based on features in the neighbourhood. Options are a relation shape convolution, in which the kernel function is mapped with an MLP based on surrounding geometry, adjusting the learning based on feature difference and obtaining the kernel function with MLPs based on the difference between geometry and propagated features.

[0009] On the other hand, the kernel can be modelled with locations, and its location can be fixed in place or trainable. For example, a fixed spherical bin kernel can be used to extract the local features or the 3D kernel can be projected into 2D by projecting the points onto an annular ring which is normal to local geometry. While the point locations of all the above kernels are pre-defined, point convolution may be generalized by modelling the kernel point with trainable locations.

[0010] Approaches to address instance segmentation include usage a similarity matrix and a confidence map, exploring the mutual aid between the semantic-instance tasks and

using semantic-aware instance segmentation and instance-fused semantic segmentation. Further, the joint relationship that is modelled by multi-value conditional random fields may be exploited or a bounding box of each instance may be predicted and subsequently a point mask may be predicted to obtain the segmentation result. Multi-task learning may be used, predicting both instance embedding and point offset in 3D space.

[0011] Still, improved approaches for object recognition on the basis of point clouds are desirable.

### Summary

[0012] According to one embodiment, a method for object recognition is provided comprising obtaining a plurality of input points in three dimensional space wherein each point of the plurality of input points has at least one colour feature and at least one geometric feature, selecting a set of query points from the plurality of input points, determining, for each query point, a set of neighbouring points from the plurality of input points, associating each query point with a respective input feature set and each neighbouring point of each query point with a respective input feature set, determining, by a neural network, for each combination of query point and neighbouring point of the query point an attention value from the at least one colour feature and the at least one geometric feature of the query point and the neighbouring point, determining, for each query point, an output feature set from the input feature sets of the neighbouring points of the query point in accordance with the attention values determined for the neighbouring points and performing object recognition based on the output feature set of the query points.

[0013] According to one embodiment, the query points are selected from the plurality of input points using furthest point sampling.

[0014] According to one embodiment, the neighbouring points of each query point are determined using k-nearest-neighbour.

[0015] According to one embodiment, the neighbouring points are determined for a query point using k-nearest-neighbour with dilation.

[0016] According to one embodiment, the attention values are determined by at least one multi-layer perceptron.

[0017] According to one embodiment, determining the attention value for each combination of query point and neighbouring point of the query point comprises determining a colour-based attention value for the combination from the at least one colour feature of the query point and the neighbouring point and a geometric feature-based attention value for the combination from the at least one geometric feature of the query point and the neighbouring point and determining the attention value based on the colour-based attention value and the geometric feature-based attention value.

[0018] According to one embodiment, the method comprises determining the colour-based attention value by a colour-based attention value determining multi-layer perceptron and determining the geometric feature-based attention value by a geometric feature-based attention value determining multi-layer perceptron.

[0019] According to one embodiment, determining the attention value for each combination of query point and neighbouring point of the query point further comprises determining a position-based attention value for the combination from the three-dimensional position of the query point and the three-dimensional position of the neighbouring point and determining the attention value based on the position-based attention value.

[0020] According to one embodiment, the method comprises determining the position-based attention value by a position-based attention value determining multi-layer perceptron.

[0021] According to one embodiment, determining the attention value for each combination of query point and neighbouring point of the query point comprises determining a input feature-based attention value for the combination from the input feature set of the query point and the input feature set of the neighbouring point and determining the attention value based on the input feature-based attention value.

[0022] According to one embodiment, the method comprises determining the attention value by feeding the the colour-based attention value, the geometric feature-based attention value, the position-based attention value and the input feature-based attention value to a partial attention value combining multi-layer perceptron.

[0023] According to one embodiment, the method comprises determining, for each combination of query point and neighbouring point of the query point, the attention value

from a difference of the at least one colour feature between the query point and the neighbouring point and a difference of the at least one geometric feature between the query point and the neighbouring point.

**[0024]** According to one embodiment, a method for object recognition is provided comprising processing a point cloud in a sequence of neural network layers starting with a first neural network layer and ending with a last neural network layer, comprising, for each neural network layer,

- obtaining a plurality of input points in three dimensional space for the neural network layer wherein each point of the plurality of input points has at least one colour feature and at least one geometric feature;
- selecting a set of query points from the plurality of input points for the neural network layer;
- determining, for each query point, a set of neighbouring points from the plurality of input points;
- associating each query point and each neighbouring point of each query point with an input feature set for the neural network layer;
- determining, for each combination of query point and neighbouring point of the query point an attention value from the at least one colour feature and the at least one geometric feature of the query point and the neighbouring point; and
- determining, for each query point, an output feature set for the neural network layer from the input feature sets of the neighbouring points of the query point for the neural network layer for the neural network layer in accordance with the attention values determined for the neighbouring points.

**[0025]** According to one embodiment, the method comprises performing object recognition based on the output feature set of the query points of the last neural network layer.

**[0026]** According to one embodiment, the sequence of neural network layers implements an encoder and the method comprises performing the object recognition based on the output feature set of the query points of the last neural network layer by a decoder.

[0027] According to one embodiment, the input feature set of each point of the query points and the neighbouring points for the first neural network layer comprises the at least one colour feature and the at least one geometric feature of the point.

[0028] According to one embodiment, for each neural network layer except for the first neural network layer, the input points comprise the query points of the neural network layer preceding the neural network layer in the sequence of neural network layers and the input feature set of each point of the query points and the neighbouring points comprises the output feature set for the query points of the neural network layer preceding the neural network layer in the sequence of neural network layers.

[0029] According to one embodiment, the plurality of input points are contained in a point cloud representing an environment of a robot device and performing object recognition comprises recognizing objects in the environment of the robot device.

[0030] According to one embodiment, determining the output feature set comprises determining a processed feature set for each neighbouring point and determining the output feature set by combining the processed feature set of the neighbouring points.

[0031] According to one embodiment, determining the output feature set comprises max-pooling of the processed feature sets over the neighbouring points.

[0032] This means that for each feature channel, the feature value which is the maximum among the neighbouring points for that channel is selected.

[0033] According to one embodiment, performing object recognition comprises performing semantic segmentation, instance segmentation, or both.

[0034] According to one embodiment, an object recognition device is provided comprising a processor configured to perform one of the methods described above.

[0035] According to one embodiment, a robotic control device is provided comprising an object recognition device as above and configured to control a robot device based on results of the object recognition.

[0036] According to one embodiment, a computer program element is provided including program instructions, which, when executed by one or more processors, cause the one or more processors to perform one of the methods described above.

[0037] According to one embodiment, a computer-readable medium is provided including program instructions, which, when executed by one or more processors, cause the one or more processors to perform one of the methods described above.

[0038] It should be noted that embodiments described in context of one or the methods are analogously valid for the other method and the transport system controller.

### **Brief Description of the Drawings**

[0039] In the drawings, like reference characters generally refer to the same parts throughout the different views. The drawings are not necessarily to scale, emphasis instead generally being placed upon illustrating the principles of the invention. In the following description, various aspects are described with reference to the following drawings, in which:

FIG. 1 shows a robot.

FIG. 2 shows a neural network layer according to an embodiment.

FIG. 3 shows a visualization of positions and features of points of a point cloud.

FIG. 4 shows a neural network according to an embodiment.

FIG. 5 illustrates the effect of vicinity merging.

FIG. 6 shows a visualization of attention.

FIG. 7 shows a flow diagram of a method for object recognition according to an embodiment.

### **Description**

[0040] The following detailed description refers to the accompanying drawings that show, by way of illustration, specific details and aspects of this disclosure in which the invention may be practiced. Other aspects may be utilized and structural, logical, and electrical changes may be made without departing from the scope of the invention. The various aspects of this disclosure are not necessarily mutually exclusive, as some aspects of this disclosure can be combined with one or more other aspects of this disclosure to form new aspects.

[0041] The embodiments described herein may for example be applied for controlling a robot device, e.g. a robot arm or household robot but also autonomous vehicles, access control systems, any kind of industrial machine etc.

[0042] FIG. 1 shows a robot 100.

[0043] The robot 100 includes a robot arm 101, for example an industrial robot arm for handling or assembling a work piece (or one or more other objects). The robot arm 101 includes manipulators 102, 103, 104 and a base (or support) 105 by which the manipulators 102, 103, 104 are supported. The term "manipulator" refers to the movable members of the robot arm 101, the actuation of which enables physical interaction with the environment, e.g. to carry out a task. For control, the robot 100 includes a (robot) controller 106 configured to implement the interaction with the environment according to a control program. The last member 104 (furthest from the support 105) of the manipulators 102, 103, 104 is also referred to as the end-effector 104 and may include one or more tools such as a welding torch, gripping instrument, painting equipment, or the like.

[0044] The other manipulators 102, 103 (closer to the support 105) may form a positioning device such that, together with the end-effector 104, the robot arm 101 with the end-effector 104 at its end is provided. The robot arm 101 is a mechanical arm that can provide similar functions as a human arm (possibly with a tool at its end).

[0045] The robot arm 101 may include joint elements 107, 108, 109 interconnecting the manipulators 102, 103, 104 with each other and with the support 105. A joint element 107, 108, 109 may have one or more joints, each of which may provide rotatable motion (i.e. rotational motion) and/or translatory motion (i.e. displacement) to associated manipulators relative to each other. The movement of the manipulators 102, 103, 104 may be initiated by means of actuators controlled by the controller 106.

[0046] The term "actuator" may be understood as a component adapted to affect a mechanism or process in response to be driven. The actuator can implement instructions issued by the controller 106 (the so-called activation) into mechanical movements. The actuator, e.g. an electromechanical converter, may be configured to convert electrical energy into mechanical energy in response to driving.

[0047] The term "controller" may be understood as any type of logic implementing entity, which may include, for example, a circuit and/or a processor capable of executing software stored in a storage medium, firmware, or a combination thereof, and which can issue instructions, e.g. to an actuator in the present example. The controller may be configured, for example, by program code (e.g., software) to control the operation of a system, a robot in the present example.

[0048] In the present example, the controller 106 includes one or more processors 110 and a memory 111 storing code and data based on which the processor 110 controls the robot arm 101. According to various embodiments, the controller 106 controls the robot arm 101 on the basis of an object recognition neural network 112 whose parameters are stored in the memory 111 and which is executed by the processor 110. It may also be trained by the processor 110 or it may be trained by another device and then stored in memory 111 for execution by the processor 110.

[0049] According to various embodiments, the robot controller 106 acquires image data from one or more cameras 113 of the robot's workspace which may include objects 114.

[0050] The one or more cameras 113 may produce RGB pictures or also depth information and, depending on the application, thermal data etc.

[0051] From the image data (or generally sensor data), the controller 106 generates a point cloud (e.g. by running a corresponding program on the processor 110). It then processes the point cloud for object detection.

[0052] One basic task in the point cloud processing is segmentation that partitions the point cloud into groups, each of which exhibits certain homogeneous characteristics. In particular, semantic segmentation groups the points with similar semantics (e.g. into screws and girders for a construction robot or chair and table for a household robot), and instance segmentation further divides the points into object instances (e.g. distinguishes screws among themselves or chairs among themselves).

[0053] A point cloud is a collection of points with spatial coordinates and possibly additional features such as colour or intensity.

[0054] For semantic segmentation the concept of convolution on an unordered point set is effective. For example, convolutional kernels with attentional weight may be used. However, while a weight matrix only depending on geometry or propagated feature

difference may be sufficient for a semantic task, the relationship between colour and geometry is more important when delineating an instance boundary.

[0055] Intuitively, humans delineate the instance boundary by paying more attention to geometry or colour in different circumstances, e.g. humans distinguish a white board from the wall mainly based on colour of the frame, and two instances of walls based on verticality. Therefore, according to various embodiments, to model the attention mechanism that is adaptive to a neural network (referred to as BEACon network) is used (e.g. as neural network 112).

[0056] In the following, a generalized version of point set convolution is given and it is demonstrated how the convolution performed by the BEACon network fits into those definitions. For each layer of the BEACon neural network, each instance boundary is represented as differences in multiple feature spaces including geometry and colour spaces, and the attentional weight is generated by feeding boundary information to a set of multi-layer perceptron (MLP).

[0057] After the instance embedding is obtained, the Cut-Pursuit algorithm may be used for clustering. Additional, a vicinity merging algorithm may be used, specifically recognizing objects in large indoor spaces.

[0058] Experiments on the S3DIS dataset show a significant improvement on instance tasks in comparison to most recent works. Tests of the BEACon network on the PartNet dataset demonstrate its effectiveness on part instance segmentation.

[0059] In the following, the construct the BEACon network according to various embodiments is described. First, a generalization the idea of point set convolution, which is used as a guideline of designing a layer (referred to as the B-Conv layer) used multiple times in the BEACon network. The network structure and loss function are described further below as well as the vicinity merging algorithm.

[0060] Given a point cloud with point set  $\mathcal{P} \in \mathbb{R}^{N \times 3}$  and corresponding feature set  $\mathcal{F} \in \mathbb{R}^{N \times C}$  the general point convolution of  $\mathcal{F}$  by a kernel  $g$  at a point  $x \in \mathbb{R}^3$  may be defined as:

$$(\mathcal{F} * g)(x) = \sum_{x_i \in \mathcal{N}_x, f_i \in \mathcal{N}_f} g(x_i - x) f_i \quad (1)$$

where  $\mathcal{N}_x$  is a subset of the point set and consists of  $K$  elements which are neighbors to point  $x$  (i.e.  $\mathcal{N}_x$  is the neighbouring point set) with feature set  $\mathcal{N}_f$  defined around the query point  $x$ . However, the kernel function can be generalized to take the difference between features as well. In addition, the input feature for a particular layer can be processed by feature mapping function before the convolution operation, denoted as  $\mathcal{M}(\cdot)$ .

$$(\mathcal{F} * g)(x) = \sum_{x_i \in \mathcal{N}_x, f_i \in \mathcal{N}_f} g(x_i - x, f_i - f) \cdot \mathcal{M}(f_i) \quad (2)$$

It should be noted that in (2) the aggregation function for convolution is summation. This aggregation function can be more general and replaced by other functions such as max pooling. In image processing, the image will have fewer feature elements because of stride operation. In point set convolution, a similar approach is accomplished by sampling query points  $\mathcal{P}_q$  from the point set  $\mathcal{P}$ . Common sampling methods include inverse density sampling, furthest point sampling, and grid down-sampling.

$$\mathcal{P}_q = \mathcal{S}(\mathcal{P}) \quad (3)$$

$$(\mathcal{F} * g)(x) = \mathcal{A}(g(x_i - x, f_i - f) \cdot \mathcal{M}(f_i)) \quad (4)$$

The generalized point set convolution can be represented as in equations (3) and (4),

where  $\mathcal{S}(\cdot)$  is the sampling function,  $\mathcal{A}(\cdot)$  denotes the aggregation function,  $x_i \in \mathcal{N}_x, f_i \in \mathcal{N}_f$  and  $x \in \mathcal{P}_q$ .

[0061] The BEACon network comprises multiple layers, referred to as B-Conv layer.

[0062] FIG. 2 shows a B-Conv layer 200, illustrated in terms of generalized point set convolution.

[0063] Embedding boundary information into an attentional matrix is the core operation of the B-Conv layer 200. This information can guide the BEACon network to learn more discriminative local features in the neighbourhood. In classic image processing, an edge is commonly computed on the basis of gradient of the nearby pixels. Multiple criteria can be used to produce the binary label. However, for point cloud the gradient of either geometry or colour alone cannot guarantee the boundary of the desired instance. Rather, the relative relationship between geometry and colour difference describes the instance boundary and can provide more clues to the attentional matrix.

[0064] With these considerations, the instance boundary is formally defined as differences in four spaces: 3D space, colour space, geometric feature space and propagated feature space.

[0065] This is reflected by the B-Conv layer 200 having a first MLP 201 for deriving attention weight information from difference in 3D position (of a neighbourhood point to a respective query point), a second MLP 202 for deriving attention weight information from difference in colour, a third MLP 203 for deriving attention weight information from difference in geometric features and a fourth MLP 204 for deriving attention weight information from difference in propagated features (input from a preceding B-Conv layer). These MLPs 201 to 204 can be seen as partial attention value determining perceptrons since they generate „partial“ attention values which are used to generate the final attention values of attentional weight 208.

[0066] The difference in 3D space (i.e. 3D position) transforms the neighbourhood area of a query point into a local coordinate system around the query point, while the differences in colour space and geometric feature space provide other similarity measures between neighbouring points (in particular query point and neighbourhood point). This is illustrated in FIG. 3

[0067] FIG. 3 shows a visualization of the query point (outlined white dot at the origin) and the difference (to neighbourhood points) in 3D position  $\Delta XYZ$ , difference in colour  $\Delta RGB$ , and difference in geometric features  $\Delta F_{geo}$  in their corresponding spaces. The scattering dimension is omitted in the plot of geo-feature space. It can be observed that a picture on the wall can only be separated in colour space. To some extent, BEACon can

be seen to learn the “shapes” in all of those spaces and generates the attentional weight by exploring their inter-relationship.

[0068] Since the propagated feature has a better describability of a larger spatial context, the propagated feature space (i.e. the fourth MLP 204) is added if the B-Conv layer 200 is not the input layer of the BEAConv network. Intuitively, more attention should be given to points nearer to the respective query point in 3D space, but the BEAConv network can also adjust its attention based on feature distribution in all the other three spaces.

[0069] Furthest point sampling  $S(\cdot)$  is applied to extract the query points 206 with shape  $N_q \times 3$  from pool points 207 (i.e. the input points to the B-Conv layer 200) with shape  $N \times 3$ . A kNN (k-nearest neighbour) approach is used to search a fixed number of neighbours 207 near the query points 206 with a pre-defined dilation rate  $D$ . The B-Conv layer 200 departs from here to generate the attentional weight 208 and propagated feature 209. To calculate the differences, the query point feature is subtracted from neighbouring features (i.e. features of the neighbours 207) in the four spaces. A kernel function  $g(\cdot)$  embeds the instance boundary in two stages.

[0070] The first stage uses the four MLPs 201, 202, 203, 204 and extracts high level features unique to the four (difference) spaces respectively. After concatenation, the second stage uses a fifth MLP 210 (referred to as „partial attention value combining MLP“) to explore the inter-relationship between those features and generate the attentional weight 208 with dimension  $N_p \times K \times C'$ . The number  $K$  is the number of neighbouring points, i.e. it does not include the query point. In other words, the query point is acting as an anchor to find the neighbouring points and calculate the feature difference when calculating attention but the feature of the query point itself is not used in feature propagation. To generate the propagated feature 209, the gathered neighbourhood (input) features are fed to a sixth MLP 211, as  $\mathcal{M}(\cdot) = MLP(\cdot)$ .

[0071] The attentional weight 208 is multiplied element-wise with the propagated feature 209. This means that the attentional weight value for a neighbour of a respective query point for a channel is multiplied with the propagated feature value of that neighbour of that query point for that channel.

[0072] Although the aggregation function is defined as a summation in convolution operation, it can experimentally be shown that  $A(\cdot)$  as max pooling function can learn more discriminative features in the neighbourhood. Therefore, for each query point max pooling over the neighbours is used as aggregation to generate the layer output 212 according to various embodiments.

[0073] It should be noted that BEACon processes down-sampled query points, and applies the kNN dilation rate to increase the layer receptive field. Further, BEACon learns a weight matrix (i.e. attention weight 208) to scale the neighbouring features. BEACon decouples the difference into separate feature spaces and explicitly models the influence of geometry and colour on attentional weight.

[0074] FIG. 4 shows the BEACon network 400 according to an embodiment, wherein a branch for semantic segmentation (top) and a branch for instance segmentation (bottom) are shown. The number on the encoder layer indicates the size of the respective output matrix, e.g. , 128 points with 256 features after the third layer.

[0075] The query points of (n-1)th B-Conv layer are the pool points for nth B-Conv layer. It should be noted that the query points are not necessarily fewer than pool points, and it can be even more than the pool points.

[0076] The BEACon network thus comprises two parallel networks for semantic and instance segmentation. It comprises B-Conv layers 401, interpolation layers 402, inverse B-Conv layers 403 and fully connected layers 404. The semantic segmentation branch and the instance segmentation branch share the same encoder but have different decoders. At the end of the network 400, the semantic segmentation branch generates the semantic probability and the instance segmentation branch generates the embedding of the input point clouds.

[0077] The initial input feature for a point (i.e. the feature set  $F_{prop}$  for the first B-Conv layer) is composed of XYZ, RGB and geometric features. The geometric features may for example include linearity, planarity, scattering, and verticality and may be generated by corresponding pre-processing. To preserve the finer-scale features, the network 400 comprises skip-links between corresponding layers of the encoder and the decoder.

[0078] The decoder starts with interpolation layers to restore the scale of the original point cloud. kNN is still used to search for the neighbouring points, but in this case, the

number of query points is larger than the number of pool points. The interpolated point feature is a linear combination of nearest points, and the weight is calculated as the inverse of point distance.

[0079] The inverse B-Conv layer is an interpolation layer followed by the B-Conv layer. The skip-linked feature is concatenated with the interpolated feature as input to the B-Conv layer, and a new neighbourhood searching is conducted before the standard operation of the B-Conv layer. To keep the model small and adjust the feature in the neighbourhood at the finest scale, Inverse B-Conv Layer is only applied at the last convolution layer.

[0080] The output layer is defined with a simple classifier in mind, with several fully connected layers and dropout layers. During training, the losses are defined separately for the semantic and instance branch, and their sum is used to update the whole neural network 400.

[0081] The semantic segmentation branch is supervised by the classical cross-entropy loss. The instance segmentation branch, however, does not have a fixed number of labels during the runtime and is adopted with a class-agnostic instance embedding learning. The loss function can be formulated as

$$L_{ins} = L_{var} + L_{dist} + \eta \cdot L_{reg} \quad (5)$$

where  $L_{var}$  aims to pull the instance embedding towards its instance centre,  $L_{dist}$  encourages separation between instance clusters, and  $L_{reg}$  is the regularization term.

Each term can be further defined as follows:

$$L_{var} = \frac{1}{I} \sum_{i=1}^I \frac{1}{N_i} \sum_{j=1}^{N_i} [\|\mu_i - e_i\|_1 - \delta_v]_+^2 \quad (6)$$

$$L_{dist} = \frac{1}{I(I-1)} \sum_{i_A=1, i_B=1}^I \sum_{i_A \neq i_B}^I [2\delta_d - \|\mu_{i_A} - \mu_{i_B}\|_1]_+^2 \quad (7)$$

$$L_{reg} = \frac{1}{I} \sum_{i=1}^I \|\mu_i\|_1 \quad (8)$$

where  $I$  is the number of ground-truth instances,  $N_i$  is the number of points in instance  $i$ ,  $\mu_i$  is the mean embedding of instance  $i$ ,  $\|\cdot\|_1$  is the  $l_1$  distance,  $e_j$  is the instance embedding of an input point,  $\delta_v$  and  $\delta_d$  are margins that define the attractive force and repulsive force, and  $[x]_+ = \max(0, x)$ . During the test time, the Cut-Pursuit algorithm is to cluster in stance embedding, e.g. for the entire robot workspace (e.g. room). The category of the instance is determined by the mode of the semantic label for that instance.

[0082] For large indoor spaces, it is common to separate the space into smaller volumes. However, this introduces problems – an instance may be divided into multiple parts, and because of the separated geometry, the embedding becomes different even for the same instance. For example, the handle of the chair may be separated from the whole chair structure and can be classified as clutter. To navigate through this problem, the predicted semantic label is concatenated at the end of instance embedding before feeding into the Cut-Pursuit algorithm, making the embedding more consistent if they belong to the same category.

[0083] In addition, according to various embodiments, a vicinity merging process specifically for large indoor spaces.

[0084] FIG. 5 illustrates the effect of vicinity merging by showing segmentation before merging 501 and segmentation after merging 502. Generally, the connected instances are merged together if they belong to the same category.

[0085] The vicinity merging algorithm is based on a simple rule – if two instances are from the same semantic category and directly connect, they should be merged into one instance. For other special categories, common knowledge may be used to add more rules to the merging criteria. Planarity, for example, is an additional condition for merging wall instances.

[0086] The BEACon network can for example be trained using the S3DIS dataset. It contains 6 areas and 270 rooms, most of which are office room settings. Totally 13

classes are introduced in this dataset, including structural components (ceiling, floor, wall, beam, column) and in-room objects (door, window, table, chair, sofa, bookcase, board, clutter). Each point has both semantic and instance annotations. To make the network less sensitive to scan noise and suitable for future applications with data scanned with different modality, the room point cloud is for example grid-down-sampled with size 2 cm. Geometric features are calculated based on the 20-nn search in the entire room. The room is then divided into  $1.2\text{m} \times 1.2\text{m}$  blocks with two strategies. For semantic branch each block has an overlap of 0.8m, so each point is predicted three times and the predicted probabilities are averaged. For instance branch the blocks are sampled in a non-overlapping fashion, so the entire room is predicted exactly once for instance embedding. Each block is further divided into batches with maximum points of 4096. After prediction, the per-point labels are back-projected to the full point set for evaluation purpose.

[0087] Another dataset which may be used for training is PartNet which consists of 573,585 part instances over 26,671 3D models covering 24 object categories. Semantic and instance annotations can be prepared for each category. The number of part instance per-object ranges from 2 to 220 with an average of 18, and each object consists of 10000 points.

[0088] Similarly to S3DIS, the geometric feature are calculated based on all the points in one object and the points are randomly sampled into 4 batches with 2500 points. For the S3DIS dataset, each point is represented with a 10-dim feature vector, including 3D coordinates (XYZ), colour (RGB) and geometric features. For example,  $\delta_v = 0.5$ ,  $\delta_d = 1.5$  and  $\eta = 0.001$  (step size) for the loss function. To augment the dataset, a perturbation of  $\pi/32$  on the z-axis and 0.001 scale variance in all directions is for example applied. An Adam optimizer may be used for the training with a base learning rate of 0.001 and a decay rate of 0.8 for every 5000 steps. The minimum learning rate is for example capped at  $10^{-6}$ . The embedding dimension for instance segmentation is for example set to 5, and regularization strength for Cut-Pursuit is set to 3 with a 5-nn graph.

[0089] During training, 2048 points are randomly sampled from each batch and trained for 60 epochs with batch size 4. During testing, all the available points in the block are

used as input. For semantic branch, each point is predicted three times and the predicted probabilities are averaged.

[0090] For the PartNet dataset, the point cloud of the shape is randomly sampled on the CAD model, which causes inaccurate surface colour representation because some models have one outer surface and one inner surface. Thus the colour information is not used, and all the other settings may be kept similar as for S3DIS.

[0091] For evaluation of semantic prediction, the accuracy and IoU (intersection over union) across all the categories are obtained, and mean accuracy (mAcc) and mean IoU are calculated by averaging the per-class accuracy and IoU. In addition, the overall accuracy (oAcc) is also calculated for all the predicted points.

[0092] For instance segmentation, the coverage and weighted coverage are evaluated, along with the precision and recall. Coverage is the average instance-wise IoU of prediction matched with ground-truth. The weighted coverage can be calculated by multiplying the ratio of current ground truth instance points and all the ground truth instances points. The precision and recall are defined with the threshold 0.5, and mean precision (mPrec) and mean recall (mRec) are obtained by averaging the per-category results.

[0093] As mentioned above, for datasets such as S3DIS vicinity merging may be used. For each category, an iteration through all the instances is performed and the ones that are directly connected are merged. This procedure is for example repeated until no instance can be merged anymore. For ceiling and floor, all the instances are merged unselectively. For walls, first, the small instances are filtered out and then RANSAC algorithm is used to fit planes to the instances. The instances will only be merged if they belong to the same plane and directly connected. For chairs, instances that are directly connected or have intersections when projected to a horizontal plane will be merged.

[0094] The evaluation metrics for S3DIS follow the fifth fold validation and 6-fold cross validation. Table 1 shows semantic segmentation results on the S3DIS dataset for BEACon and conventional (recent) algorithms.

| Method   | oAcc         | mAcc         | mIoU         | oAcc <sub>6</sub> | mAcc <sub>6</sub> | mIoU <sub>6</sub> |
|----------|--------------|--------------|--------------|-------------------|-------------------|-------------------|
| ASIS     | 86.90        | 60.90        | 53.40        | 86.20             | 70.10             | 59.30             |
| PointWeb | 86.97        | 66.64        | 60.28        | 87.31             | 76.19             | 66.73             |
| HPEI     | 87.18        | 68.30        | 61.85        | 88.20             | 76.26             | 67.83             |
| GACNet   | 87.79        | -            | 62.85        | -                 | -                 | -                 |
| SSP+SPG  | 87.90        | 68.20        | 61.70        | 87.90             | 78.30             | 68.40             |
| KPConv   | -            | <b>72.80</b> | <b>67.10</b> | -                 | <b>79.10</b>      | <b>70.60</b>      |
| BEACon   | <b>88.71</b> | 68.65        | 62.66        | <b>88.58</b>      | 75.88             | 66.89             |

Table 1

[0095] Although not specifically designed for semantic segmentation, the BEACon network has the competitive performance works. Instance results are shown in Table 2.

| Method   | mCov         | mWCov        | mPrec        | mRec         | mCov <sub>6</sub> | mWCov <sub>6</sub> | mPrec <sub>6</sub> | mRec <sub>6</sub> |
|----------|--------------|--------------|--------------|--------------|-------------------|--------------------|--------------------|-------------------|
| SGPN     | 32.7         | 35.5         | 36.0         | 28.7         | 37.9              | 40.8               | 38.2               | 31.2              |
| ASIS     | 44.6         | 47.8         | 55.3         | 42.4         | 51.2              | 55.1               | 63.6               | 47.5              |
| 3D-BoNet | -            | -            | 57.5         | 40.2         | -                 | -                  | <b>65.6</b>        | 47.6              |
| BEACon   | <b>60.94</b> | <b>62.97</b> | <b>60.06</b> | <b>56.73</b> | <b>64.03</b>      | <b>66.38</b>       | 61.51              | <b>62.38</b>      |

Table 2

[0096] BEACon network outperforms conventional methods by a large margin in all four metrics. Compared with ASIS, it achieves more than 15% improvement on mCov and mWConv, with 4.09% improvement on mean precision and 14.56% on mean recall. Taking a closer look at per category result in Table 3, BEACon is performing better than ASIS on 12 classes out of 13 for weighted coverage.

|      | method | ceiling      | floor        | wall         | beam        | column       | window       | door         | table        | chair        | sofa         | bookcase     | board        | clutter      |
|------|--------|--------------|--------------|--------------|-------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|
| WCov | ASIS   | 80.1         | 76.4         | 66.1         | 53.4        | 9.2          | 58.8         | 49.8         | 50.6         | 59.4         | 9.9          | 32.3         | 38.7         | 42.0         |
|      | BEACon | <b>93.84</b> | <b>98.19</b> | <b>76.77</b> | <b>0.00</b> | <b>28.94</b> | <b>73.82</b> | <b>65.80</b> | <b>59.20</b> | <b>83.11</b> | <b>49.04</b> | <b>62.61</b> | <b>70.38</b> | <b>57.56</b> |

[0097] The results indicate that attentional convolution can substantially benefit the instance segmentation.

[0098] Compared with ground truth, misclassified points may blend in the correct predictions. Due to the fully-convolutional nature of the network, the noise is hard to remove without post-processing. Similar geometry and colour among objects may cause fail cases.

[0099] For instance results, each instance may randomly be assigned with a colour. The colour does not have a meaning but as an indication of different instances. Most of the instances can be correctly recalled. However, one drawback of the vicinity merging algorithm is that it makes some classes indistinguishable between two objects which are directly connected.

[00100] According to one embodiment, the BEACon network has 2.5M parameters, which is 56% more parameters than ASIS (1.6M). However, the Cut-Pursuit algorithm is much faster and processes the whole room point cloud at once. For input with 4096 points in office-39, although the BEACon network inference time is 84ms (54ms for ASIS), the overall time is 200ms, which is 1.2× faster than ASIS (241ms).

[00101] Ablation studies shows that the BEACon network pays attention to the relationship between colour and geometry difference, and extracts a more discriminative feature around the neighbourhood. Table 4 shows results where  $g$  is geometry,  $c$  is colour,  $f$  is geometric feature,  $F$  is propagated feature,  $z$  is the height of the point and  $\parallel$  means concatenation.

| Method                         | oAcc         | mAcc         | mIoU         | mCov         | mWCov        | mPrec        | mRec         |
|--------------------------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|
| Attention Mechanism            |              |              |              |              |              |              |              |
| <i>no - attn</i>               | 81.41        | 53.62        | 45.59        | 42.27        | 44.29        | 43.36        | 40.74        |
| <i>sum - attn</i>              | 87.43        | 66.57        | 60.30        | 52.97        | 55.00        | 59.14        | 49.53        |
| Partial Attention              |              |              |              |              |              |              |              |
| $\Delta c    \Delta F$         | 84.93        | 63.64        | 56.27        | 53.20        | 54.93        | 51.18        | 49.93        |
| $\Delta f    \Delta F$         | 84.92        | 62.87        | 54.93        | 54.02        | 55.88        | 58.04        | 50.11        |
| $\Delta g    \Delta F$         | 88.50        | 68.74        | 62.16        | 60.09        | 62.16        | 55.99        | 55.12        |
| Input Feature                  |              |              |              |              |              |              |              |
| $g    c$                       | 88.39        | 68.10        | 61.90        | 60.42        | 62.41        | 59.33        | 55.27        |
| $g    f$                       | 88.05        | 67.88        | 61.49        | 59.42        | 61.33        | 55.89        | 54.11        |
| $c    f$                       | 87.66        | 65.31        | 59.40        | 57.46        | 59.39        | 57.69        | 54.86        |
| $z    c    f$                  | 88.07        | <b>68.90</b> | 62.53        | 59.82        | 61.75        | 57.97        | 55.74        |
| Cut-Pursuit & Vicinity Merging |              |              |              |              |              |              |              |
| MS+BM                          | -            | -            | -            | 49.92        | 52.16        | 56.37        | 48.82        |
| MS+VM                          | -            | -            | -            | 51.89        | 54.03        | 59.23        | 44.68        |
| BEACon                         | <b>88.71</b> | 68.65        | <b>62.66</b> | <b>60.94</b> | <b>62.97</b> | <b>60.06</b> | <b>56.73</b> |

Table 4: Ablation studies on S3DIS dataset (Area 5)

[00102] To show the effectiveness of the attention mechanism, a specifically designed a baseline network without attention kernel, termed *no - attn* in Table 4 was used where the attention calculation is removed and  $\Delta g$  is concatenated with the input of each layer to provide localized information.

[00103] It can be seen that BEACon shows an average of 13.26% performance gain for semantic and 9.95% gain for instance task over the baseline model.

[00104] FIG. 6 shows a visualization of the attention.

[00105] The attention is calculated as the histogram of neighbourhood index after the max-pooling operation. In other words, a neighbouring point with maximum attention would have most of its features remained after the aggregation function. The result is extracted from layer 3, where each query point has 32 neighbours with a dilation rate  $D = 2$ . Compared to the *no - attn* network, BEACon has a smaller attention spread when the query point is near the edge of the picture and has a larger spread at the centre of the

picture. While BEACon always puts maximum attention near the query point, the *no - attn* network tends to divert the attention randomly. For example, when the query point is on a chair, BEACon puts most of the attention on the structure of the chair, while *no - attn* network spreads its attention to the wall, causing the wrong feature being aggregated down the line.

**[00106]** Further, tests show that max-pooling performs best as the aggregation function.

**[00107]** Experiments on partial attention show the benefit of bringing geometry and colour difference together. Concatenated with  $\Delta F$  to generate the attention weight,  $\Delta c$  gives a slightly better result on semantic task, while  $\Delta f$  performs better on instance task.  $\Delta g \parallel \Delta F$  resembles the attention mechanism commonly used in semantic segmentation. Compared to it, BEACon only has a minor improvement on semantic score, but has a large performance gain on mPrec and mRec. The results indicate that the geometry-colour based attention not necessarily improve the semantic task, but can largely benefit the instance segmentation.

**[00108]** The initial input feature does not have a great impact on network performance, as shown in Table 4 section 3. When geometry is missing as the input feature,  $c \parallel f$  still beats most of the partial attention results. This indicates that instead of features as input, feature difference based attention can better delineate the instance boundary.

**[00109]** One simple strategy to analyze the effect of Cut-Pursuit (CP) is to directly replace it with MeanShift (MS). However, the computation complexity of MeanShift increase quadratically as the number of input points goes up. It is impractical to process the entire room at once using MeanShift. Therefore MeanShift is used only inside each batch. It takes 55 seconds to test and evaluate the entire Area 5, which is five seconds longer than BEACon (CP+VM). MeanShift has also been tested with BlockMerging (BM) strategy. BlockMerging requires the blocks to have overlap. Unlike Cut-Pursuit with vicinity merging (VM), instance embedding has to be predicted three times in this case. The entire evaluation of Area 5 takes 110 seconds.

**[00110]** The advantage of CP lies in its speed and effect. It can also process the entire room at once. One drawback of BM is that it requires an overlapped area between processing block and processed blocks. VM does not have such a limitation.

[00111] Effectiveness of the BEACon network for part instance segmentation can be shown using the four largest categories in the PartNet dataset, following the evaluation protocol in GSPN (Generative Shape Proposal Network) where the 3rd level is used. For a network that simultaneously processes both tasks, BEACon has a close semantic score to the network that specifically designed for semantic segmentation. BEACon's instance segmentation out-performs the best method in PartNet, with a maximum 25.02% improvement on the chair category. Even without colour information, BEACon can distinguish the instance based on small geometric differences.

|      | Method  | Chair        | Lamp         | Storage      | Table        |
|------|---------|--------------|--------------|--------------|--------------|
| mIoU | PartNet | <b>43.9</b>  | <b>25.3</b>  | <b>49.4</b>  | <b>33.9</b>  |
|      | BEACon  | 41.97        | 25.04        | 48.15        | 32.69        |
| mAP  | SGPN    | 19.4         | 14.6         | 14.4         | 21.5         |
|      | GSPN    | 26.8         | 21.9         | 18.3         | 26.7         |
|      | PartNet | 29.0         | 18.7         | 27.5         | 23.9         |
|      | BEACon  | <b>54.02</b> | <b>36.37</b> | <b>47.55</b> | <b>38.32</b> |

Table 5: Segmentation results on PartNet dataset

[00112] In summary, according to various embodiments, a method is provided as illustrated in FIG. 7.

[00113] FIG. 7 shows a flow diagram 700 of a method for object recognition according to an embodiment.

[00114] In 701, a plurality of input points in three dimensional space is obtained wherein each point of the plurality of input points has at least one colour feature and at least one geometric feature.

[00115] In 702, a set of query points is selected from the plurality of input points.

[00116] In 703, for each query point, a set of neighbouring points from the plurality of input points is determined.

**[00117]** In 704, each query point is associated with a respective input feature set and each neighbouring point of each query point with is associated with a respective input feature set;

**[00118]** In 705, for each combination of query point and neighbouring point of the query point an attention value is determined by a neural network from the at least one colour feature and the at least one geometric feature of the query point and the neighbouring point;

**[00119]** In 706, for each query point, an output feature set is determined by combining the input feature sets of the neighbouring points of the query point and the feature set of the query point in accordance with the attention values determined for the neighbouring points.

**[00120]** In 707, object recognition is performed based on the output feature set of the query points.

**[00121]** According to various embodiments, in other words, a neural network is provided which incorporates boundary embedded attention mechanism for instance segmentation and explicitly models the influence of both geometry and colour changes on attentional weight. Experimental results prove its benefit than incorporating geometry alone, especially for instance segmentation.

**[00122]** A geometric feature may be understood as a feature of an object constructed by a set of geometric elements like points, lines, curves or surfaces. It can be a corner feature, an edge feature, a blob, a ridge, a set of salient points, image texture and so on, which can be detected by feature detection methods. According to various embodiments, the at least one geometric feature includes one or more values indicating linearity, planarity, scattering, and verticality.

**[00123]** The neural network (BEACon) according to various embodiments can be seen to be motivated by how humans perceive geometry and colour to recognize objects and that the relationship between geometry and colour plays a more important role when delineating an instance boundary.

**[00124]** At the core of BEACon, attentional weights are introduced in the convolution layer to adjust the neighbouring features, with the weight being adapted to the relationship between geometry and colour changes. This means that instance

segmentation is improved by designing the attentional weights with the embedded boundary information. As a result, BEACon makes use of both geometry and colour information, takes instance boundary as an important feature, and thus learns a more discriminative feature representation in the neighbourhood.

**[00125]** The method of FIG. 7 may be performed by a object recognition device, e.g. implemented by a robotic control device.

**[00126]** The components of the object recognition device or robotic control device may be implemented by one or more circuits. In an embodiment, a "circuit" may be understood as any kind of a logic implementing entity, which may be special purpose circuitry or a processor executing software stored in a memory, firmware, or any combination thereof. Thus, in an embodiment, a "circuit" may be a hard-wired logic circuit or a programmable logic circuit such as a programmable processor, e.g. a microprocessor (e.g. a Complex Instruction Set Computer (CISC) processor or a Reduced Instruction Set Computer (RISC) processor). A "circuit" may also be a processor executing software, e.g. any kind of computer program, e.g. a computer program using a virtual machine code. Any other kind of implementation of the respective functions which will be described in more detail below may also be understood as a "circuit" in accordance with an alternative embodiment.

**[00127]** According to various embodiments, a object instance segmentation method for a scene defined by a point cloud is provided which is characterized by the following operations:

- i) acquiring the input point cloud comprising the following features:
  - (a) 3D spatial coordinates;
  - (b) colour values; and
  - (c) geometric features;
- ii) generating a plurality of localized centroids within the point cloud scene;
- iii) associating each localized centroid point with one or more neighbouring points;
- iv) determining the feature difference between each of the centroid and the corresponding neighbouring points;
- v) assigning distributed attentional weights based on the determined differences; and
- vi) determining object instances in the scene.

**[00128]** While specific aspects have been described, it should be understood by those skilled in the art that various changes in form and detail may be made therein without departing from the spirit and scope of the aspects of this disclosure as defined by the appended claims. The scope is thus indicated by the appended claims and all changes which come within the meaning and range of equivalency of the claims are therefore intended to be embraced.