A Fuzzy Spatial Relationship Graph for Point Clouds Using Bounding Boxes

Andrew R. Buck*, Derek T. Anderson*, James M. Keller*, Robert H. Luke III†, and Grant Scott*
*Electrical Engineering and Computer Science (EECS) Department, University of Missouri, Columbia, MO, USA
†US Army DEVCOM C5ISR Center, Fort Belvoir, VA, USA
Email: {buckar, andersondt, kellerj, scottgs}@missouri.edu, robert.h.luke2.civ@mail.mil

***Abstract*—Three dimensional point cloud data sets are easy to acquire and manipulate, but are often too large to process directly for embedded real-time applications. The spatial information in a point cloud can be represented in a variety of reduced forms, such as voxel grids, Gaussian mixture models, or spatial semantic structures. In this article, we show how a segmented point cloud can be represented as a spatial relationship graph using bounding boxes and triangular fuzzy numbers. This model is a lightweight encoding of the relative distance and direction between objects, and can be used to describe and query for particular spatial configurations using linguistic terms in a multi-criteria framework. We show how this approach can be applied on a hand-segmented subset of the NPM3D data set with several illustrative examples. The work herein has useful applications in many applied domains, such as human-robot interaction with unmanned aerial systems.**

I. Introduction

A three dimensional (3D) point cloud is a common way to represent physically sensed objects in an environment. These data sets are often generated by a time-of-flight (ToF), structured light, structure from motion (SfM), multi-view stereo (MVS), or stereo vision system and they can grow to be quite massive. For many applications, including mobile robotics, raw point clouds are often too large to store and process directly in real-time. In these scenarios, some form of compression or abstraction is required. One popular approach is to convert a point cloud into a voxel space, where individual voxels are 3D objects (typically cubes). An example of a slightly higher-level abstraction is a 3D occupancy grid map via a mixture of Gaussians [1]. While these approaches reduce data, they are low-level representations.

Another route involves representing scenes by a segmented point cloud, where each point is assigned a class label and an instance ID. Herein, we assume that such a data set can be acquired, either through manual annotation or automatic segmentation methods, e.g., PointConv [2], KPConv [3], etc. Once objects have been identified, an agent can perform tasks like reasoning about the spatial configuration of objects in the scene. This may be important to help determine interesting things, perhaps to decide where to go next, what object to interrogate, or how to explore an environment.

Herein, we propose a general framework to compute fuzzy spatial relationships between 3D point cloud objects using bounding boxes and triangular fuzzy numbers (TFNs). As an example, we demonstrate our approach on a portion of the NPM3D benchmark suite [4], which contains ground truth segmentations (Fig. 1). For reference, the considered point cloud has about 10 million points with approximately 200 labeled objects spanning just over one city block. The segmented point cloud is reduced to a collection of annotated bounding boxes and object centroids, from which we compute a graph of fuzzy spatial relationships. This in turn can be used to quickly and efficiently identify objects based on spatial queries. For instance, we can find a "person standing near a tree" or a "row of cars." For egocentric applications, we may wish to identify our next destination as an object "not too far away" and "near the edge of the explored map." To accomplish this, we define measures of distance and direction between objects and evaluate the degree to which candidate objects satisfy the criteria. This framework can be used as part of a larger system to guide agent actions or better understand a scene.

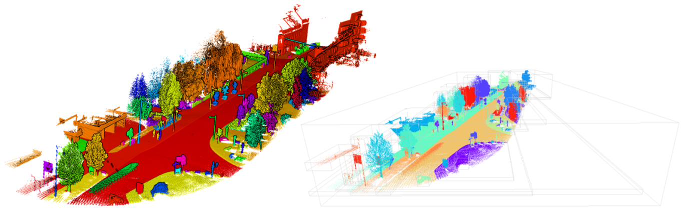


Fig. 1. A segmented point cloud from the NPM3D benchmark suite. Objects are reduced to their bounding boxes and centroids, from which we construct a fuzzy spatial relationship graph to efficiently understand the scene.

The concepts expressed above are driven by a real-world need to support human-robot teaming in the context of augmented reality (AR) for shared spaces, where robots in this context are unmanned aerial systems (UASs) and users are wearing AR headsets. One of our research needs is to build lightweight 3D representations of environments in real-time. The spatial relationship graph (SRG) proposed herein is one such structure that can be used to develop domain specific robot behaviors, e.g., multi-criteria decision making involving complex and dynamic combinations of exploration, mapping, detection and tracking, etc. Furthermore, our linguistic SRG not only captures the inherent underlying uncertainty of spatial relations in a scene, it also comes with the added benefit of assisting human-robot communication. Aspects of our SRG can be graphically displayed in a users AR headset or spatial queries like those discussed above can be used to let humans and robots talk to each other. In prior work, we demonstrated the utility of the latter in the context of spatial language driven navigation for ground robotics [5, 6].

This current article makes the following contributions. First, we outline an efficient and effective way to use triangular fuzzy sets to model 3D spatial relationships in a spatial relationship graph (SRG). To the best of our knowledge, this is new, as prior work has focused on black box neural solutions in 2D imagery [7], 2D histogram of forces (HOF) [8, 9], or 3D HOFs [10] that do not lend themselves to real-time applications. Second, we propose a set of directional, distance, object, and combinations therein, anchored linguistic SRG queries to support UAS navigation and human-robot teaming.

II. Background

A. Point Cloud Segmentation

The reader can refer to recent literature such as [11] for SfM and MVS, [12] for passive ranging, or any number of other articles on SLAM, LiDAR, ToF, structured light, or stereo vision. These are the primary sensors and sources for generating 3D point clouds. In recent years, commercial products at low cost have been introduced, e.g., Intel's RealSense RGBD (where D is depth), which is based on active sensing and ToF. Furthermore, the reader can refer to work such as PointConv [2], KPConv [3], or other semantic segmentation algorithms and neural networks to produce arbitrary segments with unknown class labels or objects with labels. Herein, we do not focus on the data stream generation nor its segmentation. The current article builds on these works and we focus on translating these low to mid level object and environment representations into high-level structures that can be acted on.

B. Fuzzy Spatial Relationships

As in Section II-A, we quickly summarize related spatial relationship research. As noted in the Introduction section, deep neural networks have been proposed to learn spatial relations in 2D imagery [7]. While intriguing, a shortcoming of an approach like this is a lack of rich linguistic support, e.g., object A is surrounded by B, object A is above and to the right of B, etc. Furthermore, the system has to be trained, current data sets do not possess this type of labeling, and labeling the imagery is a demanding task, something not likely to scale. On the other hand, Matsakis et al. have produced very rich and deep work on spatial relations via histograms of forces (HoF) and fuzzy vocabularies to support them [8, 9]. Herein, we define a SRG to be a graph where nodes are entities (e.g., objects in a scene) and edges are spatial relations, e.g., HoFs. One advantage of the HoF is its rigorous definition, theorems, and years of analysis. In [10], Matsakis et al. discuss extensions of 2D HoF to 3D vector objects.

A natural question is, why not use the HoFs? First, to the best of our knowledge, no efficient 3D HoF computational algorithm has been proposed. Keep in mind, efficient herein is with reference to large 3D point data sets and real-time algorithmic performance. That is, we need spatial relations that can be calculated in a fraction of a second. This is only one piece in a larger puzzle of robot autonomy and human-robot teaming in a shared AR space. Next, there is error in 3D mapping algorithms and objects in the visible spectrum result in hulls. That is, additional processing, e.g., calculating an umbra, is required to determine filled 3D objects. It is not clear what the result of a 3D HoF would be on a partially observed and error prone object. Herein, we instead focus on a light weight structure that can ideally absorb such errors and be of high level benefit. As we show in the results section, our TFN approach is perhaps already a good enough descriptor to power the downstream tasks outlined above (robotic navigation, exploration, human-robot teaming, etc.). In future work we will work to develop an analytical procedure or more extensive experiments, e.g., involving evaluation of robotic behavior, to further quantify and investigate these interesting questions.

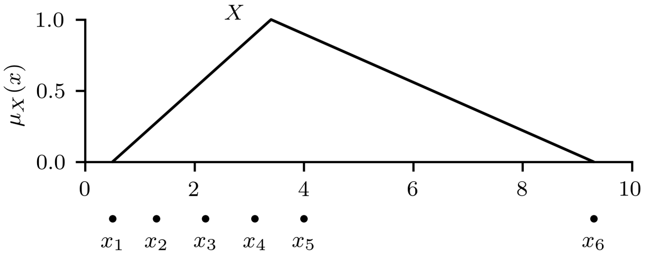


Fig. 2. A triangular fuzzy number X can be defined over a set of numbers $x_1, ..., x_n$ to represent the minimum, mean, and maximum values of the set.

C. Fuzzy Numbers

Fuzzy numbers provide a way to represent imprecision in the specification of a real-valued number [13]. The membership function $\mu_A(x)$ of a fuzzy number A maps the value x into the range $[0, 1]$, where $\mu_A(x)$ represents the degree to which x belongs to A. In this work, we consider only triangular fuzzy numbers (TFNs), each represented as a triplet Tri(a, b, c), where $a \leq b \leq c$ and $a, b, c \in \mathbb{R}$. Here, the interval $[a, c]$ represents the support where $\mu_A(x) > 0$ and $\{b\}$ is the singleton core set where $\mu_A(x) = 1$.

There are many possible ways to define a TFN from a set of real-valued numbers $x_1, ..., x_n$, but we focus here on the following intuitive approach. Let $X = \text{Tri}(a, b, c)$ be a TFN, where $a = \min\{x_1, ..., x_n\}$, $b = \frac{1}{n}\sum_{i=1}^{n} x_i$, and $c = \max\{x_1, ..., x_n\}$. This captures the minimum, maximum, and average values of the set (Fig. 2). An alternative approach might use the median and some form of outlier detection to increase robustness.

1) Fuzzy Arithmetic: Arithmetic operations on TFNs are defined here as an extension of interval arithmetic, although other operators could be chosen. For two TFNs $A = \text{Tri}(a_1, a_2, a_3)$ and $B = \text{Tri}(b_1, b_2, b_3)$, the result of a function $f(A, B)$ is a new TFN $C = \text{Tri}(c_1, c_2, c_3)$. Here, c_1 and c_3 are the minimum and maximum possible values respectively that could result from $f(a,b)$, where $a_1 \leq a \leq a_3$ and $b_1 \leq b \leq b_3$. Since the middle element is chosen to represent the mean value, c_2 is defined as $f(a_2, b_2)$.

Using this notation, we define the following operations used throughout this work.

$$A + B = \text{Tri}(a_1 + b_1, a_2 + b_2, a_3 + b_3) \tag{1}$$

$$A - B = \text{Tri}(a_1 - b_3, a_2 - b_2, a_3 - b_1) \tag{2}$$

$$\lambda A = \text{Tri}\left(\min\{\lambda a_1, \lambda a_3\}, \lambda a_2, \max\{\lambda a_1, \lambda a_3\}\right) \tag{3}$$

$$\begin{aligned} A^2 &= \text{Tri}\big(a_{\text{min}}, a_2^2, \max\{a_1^2, a_3^2\}\big)\,, \\ a_{\text{min}} &= \begin{cases} \min\{0, a_1^2, a_3^2\}, & \text{if } a_1 \leq 0 \leq a_3 \\ \min\{a_1^2, a_3^2\}, & \text{otherwise} \end{cases} \end{aligned} \tag{4}$$

$$\sqrt{A} = \text{Tri}(\sqrt{a_1}, \sqrt{a_2}, \sqrt{a_3}), \qquad 0 \leq a_1 \leq a_2 \leq a_3 \tag{5}$$

Note that $\lambda \in \mathbb{R}$ is a scalar multiplier, and when squaring a fuzzy number that spans both positive and negative values, the minimum value will be set to zero.

2) Fuzzy Similarity: Two TFNs A and B (defined as above) can be compared to determine the degree to which they are equivalent. Note that for this application we are not looking for the amount of overlap between the sets, but rather a best-case sense of how well an element from A might be represented by B and vice versa. Of the many possible similarity operators, an appropriate choice here to measure the similarity between two TFNs is

$$S(A, B) = \max_{x \in \mathbb{R}} \left\{ \min\left(\mu_A(x), \mu_B(x)\right) \right\}. \tag{6}$$

For TFNs, this can be computed quickly by finding the maximum of four intersecting line segments as shown in Fig. 3. Consider two lines

$$\begin{aligned} x - (a_h - a_l)y - a_l = 0 \\ x - (b_h - b_l)y - b_l = 0 \end{aligned}$$

where $\mu_A(a_l) = 0$, $\mu_A(a_h) = 1$, $\mu_B(b_l) = 0$, and $\mu_B(b_h) = 1$. Solving for y reveals the level where the lines intersect,

$$y = \frac{b_l - a_l}{(a_h - a_l) - (b_h - b_l)}. \tag{7}$$

If the denominator here is 0, the two lines are parallel. If $a_l = b_l$ and $a_h = b_h$, the two lines are also coincident. Only when $0 \leq y \leq 1$ do the lines intersect as part of the membership function. Let us define a function

$$h(a_l, a_h, b_l, b_h) = \begin{cases} 1, & \text{if the lines are coincident} \\ 0, & \text{if the lines are otherwise parallel} \\ y, & \text{if } 0 \leq y \leq 1 \\ 0, & \text{otherwise.} \end{cases} \tag{8}$$

The similarity of A and B can now be defined as

$$S'(A, B) = \max_{\substack{a_l \in \{a_1, a_3\} \\ b_l \in \{b_1, b_3\}}} h(a_l, a_2, b_l, b_2). \tag{9}$$

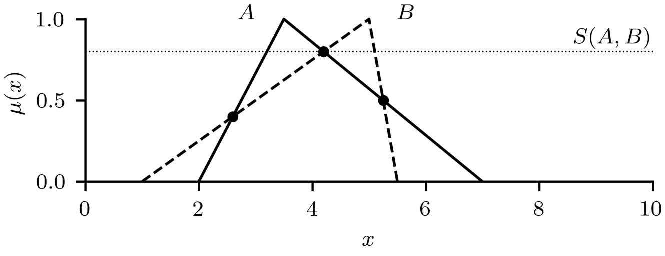


Fig. 3. The similarity of two TFNs A and B is defined as the maximum of the intersecting points.

3) Defuzzification: There are times, typically at the end of a calculation, when a fuzzy number needs to be reduced to a single crisp value. This may be done to provide a reference point upon which a decision can be made, or to facilitate working with scalar numbers. For a TFN $X = \text{Tri}(a, b, c)$, any value $x \in [a, c]$ could be chosen as the crisp result of defuzzification. We can parameterize this decision with a variable $\xi \in [0, 1]$ that interpolates the result between a and c. If b is defined to represent the average value, then a natural interpolation approach would be

$$\Gamma(X|\xi) = \begin{cases} a + 2\xi(b - a), & \xi \leq 0.5 \\ b + 2(\xi - 0.5)(c - b), & \xi > 0.5. \end{cases} \tag{10}$$

The parameter ξ can be viewed as an optimism or pessimism term, where small values shift $\Gamma(X|\xi)$ toward a and large values shift $\Gamma(X|\xi)$ toward c. When $\xi = 0.5$, the defuzzified value is equal to b. This provides a straightforward method to interpolate between the TFN control points and select among the minimum, maximum, or average values.

III. Method

A. Bounding Boxes

Consider a collection of points in 3D space that represent a segmented object. That is, all points in this set have the same label and instance ID. The axis-aligned bounding box of these points spans the minimum and maximum values of any point in this set along each axis. Additionally, the centroid of this object is the mean value of the points in the set and can be thought of as the object's center of mass. Note that while the centroid will always be contained within the bounding box, it may fall outside of the actual object boundary if the object is not convex. Fig. 4 shows two point cloud objects and their corresponding bounding boxes and centroids.

We denote the position of an object A with three TFNs,

$$\begin{aligned} A_x &= \text{Tri}(x_{\text{min}}, x_{\text{mean}}, x_{\text{max}}) \\ A_y &= \text{Tri}(y_{\text{min}}, y_{\text{mean}}, y_{\text{max}}) \\ A_z &= \text{Tri}(z_{\text{min}}, z_{\text{mean}}, z_{\text{max}}), \end{aligned}$$

which represent the extents of the axis-aligned bounding box of A, and the centroid of A. Likewise, we can define the position of an object B with TFNs B_x, B_y, and B_z. The

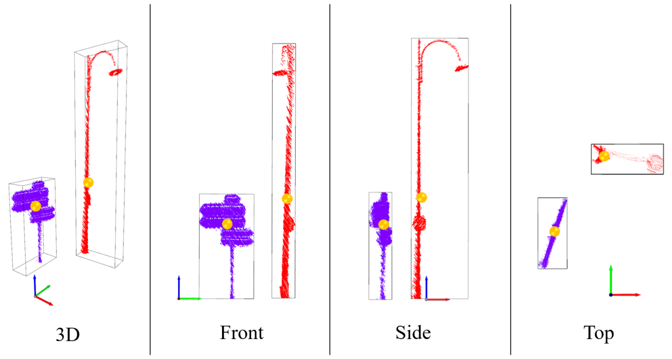


Fig. 4. Two point cloud objects, A (purple) and B (red), represented by their axis-aligned bounding boxes and centroids.

relative position of an object B with respect to an object A can be computed as $B - A$. Taken along each axis, three distance values can be computed using Eq. 2 as

$$D_x = B_x - A_x \tag{11}$$

$$D_y = B_y - A_y \tag{12}$$

$$D_z = B_z - A_z. \tag{13}$$

These differences represent the minimum, mean, and maximum distance between the two objects in each direction. The overall distance from A to B can then be computed as

$$D = \sqrt{D_x^2 + D_y^2 + D_z^2}. \tag{14}$$

Equation 14 makes use of Eq. 1, 4, and 5, and represents the overall average distance between the two objects, as well as the minimum and maximum possible distances. Because the distances are computed between bounding boxes and not the object points themselves, the computation is very fast, but also captures extreme values for the minimum and maximum. The centroid value is therefore critical in representing the underlying distribution, while the endpoints describe the possible extent. Fig. 5 shows the fuzzy spatial relationship computed for the example in Fig. 4.

B. Spatial Relationship Graph

A point cloud environment can have many individual segmented objects, spread out over large distances. Often, we are concerned with only the local neighborhood of a particular object and would like to ignore any relationships to objects that are far away. A spatial relationship graph (SRG) can be defined over the objects to show this local topology. Using the distance measure defined above, we define a neighborhood graph that contains each segmented object as a node and includes an edge between each pair of objects that are closer than some threshold distance d. Since we consider the graph to be crisp, the distances between objects are defuzzified with a parameter ξ using Eq. 10. When ξ is high, only object pairs that are completely within the threshold distance are included as edges in the graph (similar to complete linkage clustering). When ξ is low, the closest distance between object pairs is considered (similar to single linkage clustering). Fig. 6 shows an example SRG computed for different values of ξ with a fixed distance threshold.

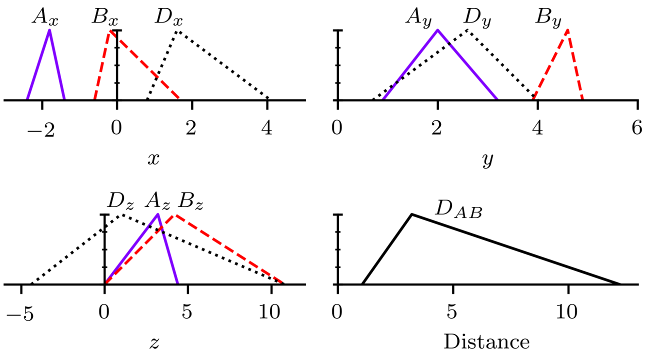


Fig. 5. The fuzzy spatial relationship between objects A and B in Fig. 4 is computed as the difference between the TFNs representing the objects along each axis. The overall distance is computed as the Euclidean norm of these directional fuzzy numbers.

The SRG provides a way to identify objects that have relevant spatial relationships to each other. A common type of spatial query is to determine an object (or set of objects) that is (are) a particular distance away from, and in a certain direction of, some reference object R. These queries are explored further in the following sections. The SRG can be used to restrict the search to neighboring objects of R, denoted as $N(R)$ Changing the distance threshold d and defuzzification parameter ξ can change the average number of neighbors in the graph, and in the extreme case (when d is large) this can result in a fully-connected graph. The distance pruning rule can be adapted for specific object types, e.g. to connect an egocentric actor in the scene to distant objects while keeping other object types more locally connected. This can result in a graph that more naturally expresses the relationships between objects in the scene, restricting connections to only object pairs that have some inherent relationship.

C. Distance Queries

The fuzzy neighborhood graph provides a way to evaluate the distance between objects in an imprecise way, such as with natural language. A linguistic distance term can be modeled as a TFN to represent a query such as "nearby" or "not too far away."[1] The similarity of this query to the computed fuzzy distance between two objects gives the degree to which the spatial relationship between these objects matches the query.

The distance between the axis-aligned bounding boxes of two objects is computed as a TFN by Eq. 14. Consider a query distance $Q = \text{Tri}(q_1, q_2, q_3)$ and the distance between a pair of objects A and B as $D_{AB} = \text{Tri}(d_1, d_2, d_3)$. The degree

[1] We only consider triangular fuzzy numbers in this work, although the ideas could be easily extended to other representations, such as trapezoids.

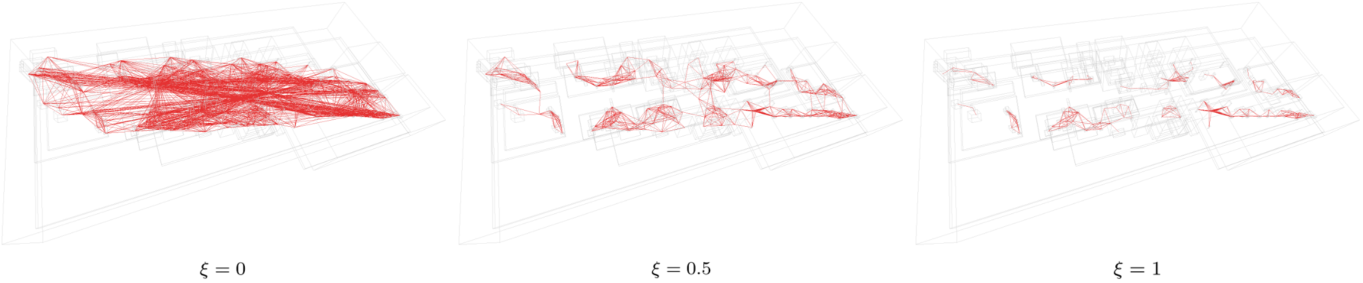


Fig. 6. The spatial relationship graph computed for a set of objects with a fixed distance threshold of 10 meters and different values of ξ.

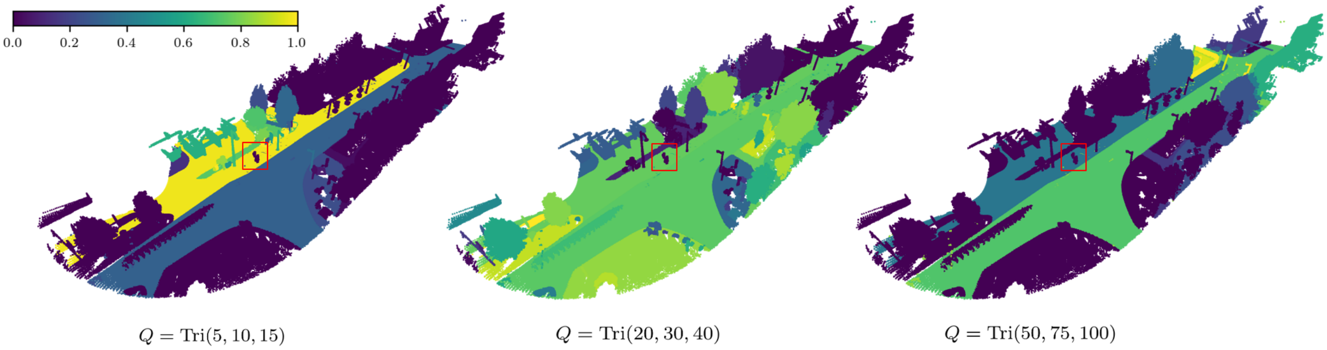


Fig. 7. An example scene from the NPM3D data set showing heat maps for different distance queries from a person in the scene, outlined with a red box. The queries could be interpreted linguistically as “nearby,” “somewhat distant,” and “far off.” Note that the imprecision in these terms is reflected in how objects are evaluated.

to which A and B are distance Q apart from each other would be given by Eq. 9 as

$$s_{\text{dist}} = S'(Q, D_{AB}). \tag{15}$$

Here, s_{dist} is defined to be in the range $[0, 1]$. This feature could be used along with other features to filter and select objects in the fuzzy neighborhood graph. For instance, we could find all objects in $N(R)$ that are some query distance Q away from a reference object R.

In Fig. 7, we show an example scene from the NPM3D data set centered on a person (identified with a red box). The person could be considered an egocentric actor in the scene, from which we want to know the distances to all other objects. Four different distance queries are shown with a heat map indicating the degree of match between each object and the specified query distance. These queries could be interpreted as linguistic terms such as “nearby,” “somewhat distant,” or “far off.” The imprecise nature of the bounding box representation allows queries to span both wide and narrow ranges, but only entire objects are considered. This means that if an object spans the entire scene—such as the roadway, which is not divided into separate objects—it may always be evaluated as being near to other objects in the scene. This could be useful for semantic reasoning (the minimum distance to the road is always small, but the maximum can be quite large), or with some additional processing, large objects could be further segmented into a hierarchical representation with a built-in size limit.

D. Directional Queries

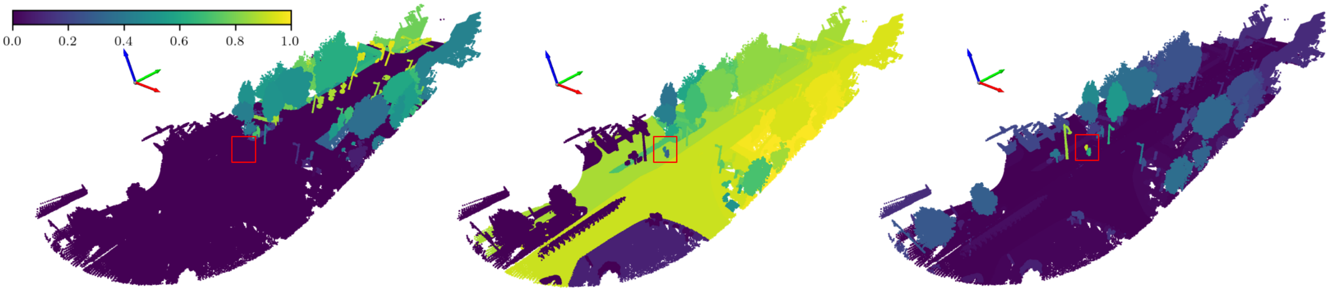
The fuzzy relative position between two objects A and B encodes both the distance and direction between them in the three TFNs, D_x, D_y, and D_z (Eq. 11–Eq. 13). This information can be used to evaluate queries such as the degree of support for the statement “B is in direction $\hat{\mathbf{u}}$ from A,” where $\hat{\mathbf{u}}$ is a unit vector pointing in the direction of interest. Consider the difference values

$$\begin{aligned} D_x &= \text{Tri}(d_{x1}, d_{x2}, d_{x3}) \\ D_y &= \text{Tri}(d_{y1}, d_{y2}, d_{y3}) \\ D_z &= \text{Tri}(d_{z1}, d_{z2}, d_{z3}). \end{aligned}$$

A vector $\mathbf{v}_{AB} = [d_x, d_y, d_z]$ represents the position of a point in B relative to a point in A if $d_{x1} \le d_x \le d_{x3}$, $d_{y1} \le d_y \le d_{y3}$, and $d_{z1} \le d_z \le d_{z3}$. We can determine the degree to which B is in direction $\hat{\mathbf{u}}$ from A by comparing $\hat{\mathbf{u}}$ with a normalized version of $\mathbf{v}_{AB}$.

The maximum length of $\mathbf{v}_{AB}$ is computed by measuring the distance from the origin to each of the corners of the box defined by D_x, D_y, and D_z. If

$$v_{\max} = \max_{\substack{x \in \{d_{x1}, d_{x3}\} \\ y \in \{d_{y1}, d_{y3}\} \\ z \in \{d_{z1}, d_{z3}\}}} \sqrt{x^2 + y^2 + z^2}, \tag{16}$$



$\xi = 0$
$\hat{\mathbf{u}} = [0, 1, 0]$

$\xi = 1$
$\hat{\mathbf{u}} = [0.707, 0.707, 0]$

$\xi = 1$
$\hat{\mathbf{u}} = [0, 0, 1]$

Fig. 8. An example scene from the NPM3D data set showing heat maps for different directional queries from a person in the scene, outlined with a red box. The queries show concepts such as "strictly to the north," "generally to the northeast," and "generally above" (assuming an ENU reference frame).

then the normalized difference values are defined as

$$\hat{D}_x = \text{Tri}\left(\frac{d_{x1}}{v_{\text{max}}}, \frac{d_{x2}}{v_{\text{max}}}, \frac{d_{x3}}{v_{\text{max}}}\right) \tag{17}$$

$$\hat{D}_y = \text{Tri}\left(\frac{d_{y1}}{v_{\text{max}}}, \frac{d_{y2}}{v_{\text{max}}}, \frac{d_{y3}}{v_{\text{max}}}\right) \tag{18}$$

$$\hat{D}_z = \text{Tri}\left(\frac{d_{z1}}{v_{\text{max}}}, \frac{d_{z2}}{v_{\text{max}}}, \frac{d_{z3}}{v_{\text{max}}}\right). \tag{19}$$

The fuzzy dot product between the normalized difference vector $\hat{\mathbf{D}} = [\hat{D}_x, \hat{D}_y, \hat{D}_z]$ and a query vector $\hat{\mathbf{u}}$ is defined as

$$S_{\text{dir}} = \hat{\mathbf{D}} \cdot \hat{\mathbf{u}} = \hat{D}_x \hat{u}_x + \hat{D}_y \hat{u}_y + \hat{D}_z \hat{u}_z, \tag{20}$$

where $\hat{u}_x$, $\hat{u}_y$, and $\hat{u}_z$ are the crisp scalar components of $\hat{\mathbf{u}}$. Recall that the multiplication of a TFN with a scalar may result in swapping the minimum and maximum values if the scalar is negative (Eq. 3). S_{dir} is a TFN bounded in the range $[-1, 1]$ that represents how much B is in direction $\hat{\mathbf{u}}$ from A. The value can be reduced to a crisp scalar as

$$s_{\text{dir}} = \max\left\{0, \Gamma(S_{\text{dir}}|\xi)\right\}. \tag{21}$$

Here, s_{dir} is defined to be in the range $[0, 1]$ using ξ as a defuzzification parameter with Eq. 10. Small values of ξ imply that B must be nearly entirely in direction $\hat{\mathbf{u}}$ from A to score highly, whereas large values of ξ allow for only part of B to be in the specified direction. Limiting s_{dir} to the unit interval allows directional similarity to be evaluated along with distance similarity in a general multi-criteria framework.

Fig. 8 shows several directional queries on the example scene from the NPM3D data set. Here, we show concepts such as "strictly to the north," "generally to the northeast," and "generally above." As with the distance queries, the imprecision in the linguistic terms is reflected in how objects are evaluated.

E. A Multi-Criteria Framework

The distance and direction between objects in the fuzzy neighborhood graph provide two features that can be used to filter and select objects that match certain criteria. In general, we assume that each object in the scene has a vector of features or attributes that describe it. These may be intrinsic to the object itself, such as a class label or world-space position, or the features may be computed relative to another object, such as the spatial relationship to a reference object as described in the previous sections. We require each feature to map to a normalized scalar value s_i in the range $[0, 1]$ so that the features can be compared on the same scale. Here, the features represent the degrees to which an object satisfies a particular set of criteria. Given a normalized feature vector $\mathbf{s} = [s_1, ..., s_k]$ for each object in the scene, the objects can be compared in k-dimensional objective space. Choosing an object that best satisfies all of the (perhaps conflicting) criteria involves understanding the trade-offs between criteria, and perhaps computing the Pareto optimal set.

For this work, we chose to scalarize each feature vector $\mathbf{s}$ into a single value that can be ranked using a scalarization function $g_\theta(\mathbf{s})$, where θ represents the parameterization of the function. While many options exist for choosing this function, two simple methods are the average and minimum operators. Assuming all features have equal weight, the average operator is defined as

$$g_{\text{avg}}(\mathbf{s}) = \frac{1}{k}\sum_{i=1}^{k} s_i, \tag{22}$$

and the minimum operator is

$$g_{\text{min}}(\mathbf{s}) = \min_i s_i. \tag{23}$$

The average operator treats all features equally, whereas the minimum operator considers only the least-satisfied criteria, ensuring that objects that score highly have no significant unsatisfied criteria. Extending this idea, one could implement weighted preferences for each feature, and perform a generalized ordered weighted average to better capture the decision-maker's intentions. Once all feature vectors have been scalarized, they can be ranked and the corresponding objects visualized as a heat map. The object with the largest scalarized value would be chosen as the option that best satisfies the stated criteria.

IV. Examples

The spatial relationship graph described in the previous section can be applied to real-world problems in a variety of ways. The methods are generic and extendable so they can be adapted for specific context domains. In this section, we describe in greater detail how these ideas can be used for reasoning about the spatial configuration of objects in a scene. The following two examples illustrate potential applications of the SRG using the NPM3D data set, but are not the only ways in which this approach can be used.

A. Example 1: Multi-Criteria Selection

This example demonstrates using distance, direction, and object type to select a specific object in a scene. Consider Fig. 9, where a pedestrian in the NPM3D data wants to refer to a lamp post that is "nearby" and "to the front-left." This may be part of a larger natural language application that involves interaction between the human and a robot. For the purposes of this example, we assume that we can model the linguistic distance term "nearby" as the TFN $\mathrm{Tri}(0, 5, 10)$ and the direction "front-left" as the vector $\hat{\mathbf{u}} = [0.707, 0.707, 0]$.

To begin, we use a pre-computed SRG over the entire scene with a distance threshold of 35 meters and $\xi = 0.5$. This is a relatively compact data structure that encodes the spatial relationships between any pair of objects determined to be within 35 meters of each other. The node representing the pedestrian has many edges, but only those leading to lamp post objects need to be evaluated for this query. Immediately, we are left with just five objects to consider.

For each of these five lamp post objects, we compute the similarity of the distance from the pedestrian to the query distance of $\mathrm{Tri}(0, 5, 10)$. Fig. 9a shows that only the two closest objects have a non-zero distance score.

We then compute the directional similarity of each lamp post object to the query vector $\hat{\mathbf{u}} = [0.707, 0.707, 0]$ with $\xi = 1$. Fig. 9b shows that four of the objects have non-zero directional similarity scores. Interestingly, objects that are farther away have higher scores than nearby objects. This can be attributed to the uncertainty that arises from assessing the primary direction between two nearby objects with spatial extent. As the distance between the objects increases, they appear more like points and there is less uncertainty in the direction, leading to greater similarity to the query vector.

Finally, we combine the distance and direction scores using the minimum operator. The results are shown in Fig. 9c, where only one lamp post object has a non-zero score. This would be the object selected by the agent for this example query.

B. Example 2: Choosing a Exploration Target

In this example, we imagine that the pedestrian in the previous example represents a robotic actor that seeks to continue exploring using its current model of the environment. This could be a UAS that is tasked with building a 3D model of the scene. We would like to select an object of interest in some under-explored area of the map to study in more detail. Presumably the agent can navigate to this location using its own independent planning algorithm. Our criteria are that the new target object is not too far away, in front of the agent, and near the edge of the explored map.

The first two criteria are similar to the previous example. In this case, we have chosen to only consider objects within 30 meters ($\xi = 0.5$) of the agent. Figs 10a and 10b show the evaluation scores for the query distance $\mathrm{Tri}(10, 20, 30)$ and the direction $\hat{\mathbf{u}} = [1, 0, 0]$ with $\xi = 1$. To select objects near the edge of the map, we count the number of edge connections for each node in the SRG with a threshold of 10 meters and $\xi = 0.25$. Objects with fewer connections are likely to be near the edge of the map, whereas objects with many connections tend to be surrounded by other objects. The inverse of this count, normalized over the whole graph gives a frontier score for each object as an additional criteria, shown in Fig. 10c. Objects with greater frontier scores appear near the edge of the map. Combining all three of these criteria with the minimum operator gives the final assessment for each object, shown in Fig. 10d. The highest scoring objects in this example are across the street from the agent, near the edge of the explored area, suggesting that one of these might be an appropriate target to investigate next.

V. Conclusion

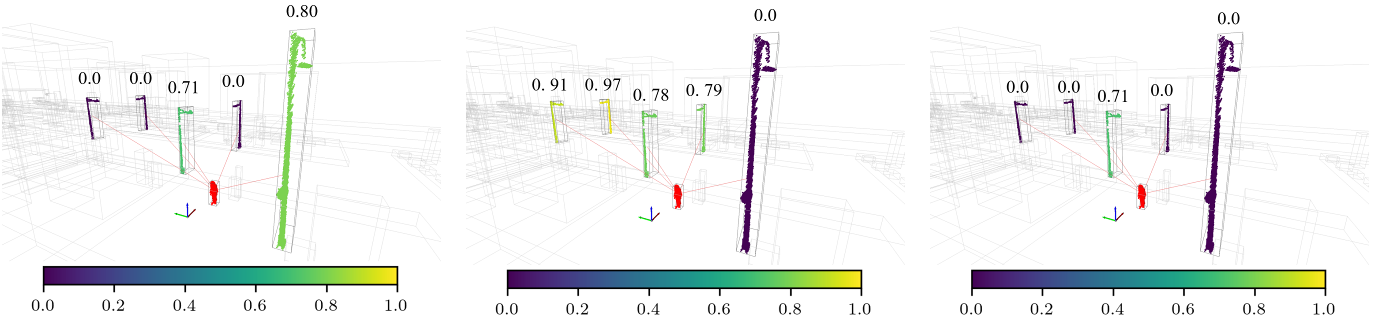
The spatial relationship graph proposed in this article can be used to reduce a segmented point cloud into a lightweight form with many potential applications. We demonstrated how the graph could be queried to find objects that satisfy certain distance and directional constraints in a multi-criteria framework. This approach is suitable for real-time scenarios that require low computational overhead and can tolerate a high degree of imprecision. High levels of uncertainty are bound to be present in real-world settings, where an object may not be completely observed. The bounding box and centroid method presented here captures both the average spatial relationships between objects, as well as the maximum and minimum extents.

The distance and direction relationships between bounding boxes as defined here use an axis-aligned representation, which is most appropriate for global queries. Objects that do not align well with the world axes may suffer an increase in imprecision from this choice of coordinate frame. An oriented bounding box representation may be able to provide increased accuracy, but with other trade-offs, such as increased storage requirements and a more complicated spatial relationship computation. We have limited our discussion here to the axis-aligned approach for simplicity, but extensions and limitations of this method are subjects for future work.

References

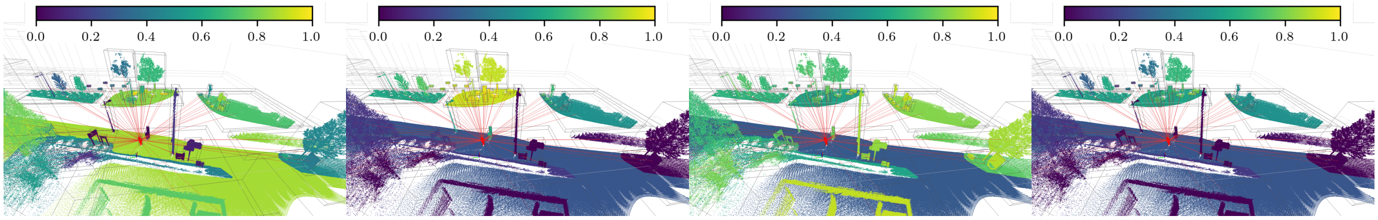
[1] A. Dhawale, X. Yang, and N. Michael, "Reactive collision avoidance using real-time local gaussian mixture model maps," in *Proceedings of (IROS) IEEE/RSJ International Conference on Intelligent Robots and Systems*, October 2018, pp. 3545 – 3550.

[2] W. Wu, Z. Qi, and L. Fuxin, "Pointconv: Deep convolutional networks on 3D point clouds," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.



(a) (b) (c)

Fig. 9. Example showing a combined distance and directional query to identify a lamp post that is "near" and "to the front-left" of a pedestrian (shown in red). (a) Distance query for the query distance $\mathrm{Tri}(0, 5, 10)$. (b) Directional query for the vector $\hat{\mathbf{u}} = [0.707, 0.707, 0]$ and $\xi = 1$. (c) Combined query using the minimum operator. Zoom in to see full detail.



(a) (b) (c) (d)

Fig. 10. Example showing how an egocentric agent might select an object of interest to move towards and inspect more closely. Only objects with a max distance of 30 meters ($\xi = 0.5$) from the agent (represented by a pedestrian in the scene) are considered as neighbors. (a) Distance query for the query distance $\mathrm{Tri}(10, 20, 30)$. (b) Directional query for the vector $\hat{\mathbf{u}} = [1, 0, 0]$ and $\xi = 1$. (c) Frontier score (objects with fewer neighbors score higher). (d) Combined query using the minimum operator. Zoom in to see full detail.

[3] H. Thomas, C. R. Qi, J.-E. Deschaud, B. Marcotegui, F. Goulette, and L. J. Guibas, "KPConv: Flexible and deformable convolution for point clouds," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

[4] X. Roynard, J.-E. Deschaud, and F. Goulette, "Paris-Lille-3D: A large and high-quality ground-truth urban point cloud dataset for automatic segmentation and classification," *The International Journal of Robotics Research*, vol. 37, no. 6, pp. 545–557, 2018.

[5] G. Chronis and M. Skubic, "Sketch-based navigation for mobile robots," in *The 12th IEEE International Conference on Fuzzy Systems, 2003. FUZZ '03.*, vol. 1, 2003, pp. 284–289 vol.1.

[6] M. Skubic, P. Matsakis, B. Forrester, and G. Chronis, "Extracting navigation states from a hand-drawn map," in *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No.01CH37164)*, vol. 1, 2001, pp. 259–264 vol.1.

[7] M. Haldekar, A. Ganesan, and T. Oates, "Identifying spatial relations in images using convolutional neural networks," 2017.

[8] P. Matsakis, J. M. Keller, L. Wendling, J. Marjamaa, and O. Sjahputera, "Linguistic description of relative positions in images," *IEEE Trans. Syst. Man Cybern. Part B*, vol. 31, no. 4, pp. 573–588, 2001. [Online]. Available: https://doi.org/10.1109/3477.938261

[9] P. Matsakis, J. M. Keller, O. Sjahputera, and J. Marjamaa, "The use of force histograms for affine-invariant relative position description," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, no. 1, pp. 1–18, 2004.

[10] J. Reed., M. Naeem., and P. Matsakis., "A first algorithm to calculate force histograms in the case of 3D vector objects," in *Proceedings of the 3rd International Conference on Pattern Recognition Applications and Methods - ICPRAM,*, INSTICC. SciTePress, 2014, pp. 104–112.

[11] J. L. Schonberger and J.-M. Frahm, "Structure-from-motion revisited," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[12] C. Godard, O. Mac Aodha, M. Firman, and G. J. Brostow, "Digging into self-supervised monocular depth estimation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

[13] G. J. Klir and B. Yuan, *Fuzzy Sets and Fuzzy Logic: Theory and Applications.* Upper Saddle River, NJ: Prentice Hall, 1995.