Exploiting Depth Camera for 3D Spatial Relationship Interpretation

Jun Ye
University of Central Florida
4000 Central Florida Blvd
Orlando, Florida 32826
jye@cs.ucf.edu

Kien A. Hua
University of Central Florida
4000 Central Florida Blvd
Orlando, Florida 32826
kienhua@cs.ucf.edu

ABSTRACT

Interpretation of spatial relations between objects is essential to many applications such as robotics, video surveillance, spatial reasoning, and scene understanding. Current models for spatial logic are two-dimensional. With the advance in new sensing technology, inexpensive depth sensors become widely available and 3D scene reconstruction can be applied in various application scenarios. In this paper, we propose a 3D spatial logic and algorithms for interpretation of spatial relationships among objects in 3D space. More specifically, these techniques are developed for LVDBMS (Live Video DataBase Management System), a generic platform for live video computing. We extend the original directional relationships into 3D directional relationships, and introduce a simple yet effective way to build 3D object models based on depth sensors. A highly accurate and efficient algorithm is also proposed to compute the spatial relationships between two objects by sampling the entire space from the reference object. Experimental results based on a real indoor scene and an RGB-D dataset are given to demonstrate the effectiveness of our techniques.

Categories and Subject Descriptors

I.4.8 [**Computing Methodologies**]: Image Processing and Computer Vision – Scene Analysis

General Terms

Design,Algorithms,Security

Keywords

Spatial relation interpretation, Directional relationships, depth sensor, 3D video surveillance

1. INTRODUCTION

A spatial relation specifies how an object is located in space in relation to the reference object. Efficient computation of such relations between objects is essential in various applications. In robotics, it helps robots identify shapes and positions, and recognize the target [1, 2]. It can also be employed in robot route planning and navigation by exploiting the position of objects in the scene [3]. In video surveillance, it can support spatial data queries and trigger alerts if certain events happen [4], for example, two vehicles collide on the highway. In artificial intelligence, spatial properties and relations can help improve spatial reasoning [5]. In scene understanding, spatial relationships are employed to detect the 3D spatial layout of scenes [6, 7].

Spatial relationships have a large number of instances. They can be roughly divided into two categories, topological relationships and metric ones [8]. The latter can be further split into distance relationships and directional relationships. Topological relationships include *coincide, intersect, touch externally, touch internally, contains, inside, disjoint*, etc. A complete set of topological relationships were proposed in [9]. Distance relationships include *at, nearby, in the vicinity, far away*, etc. Directional relationships can support *left, right, in front of, at the back of*, or more specifically, the eight directional operators (*north, south, west, east, northwest, northeast, southwest and southeast*). Many authors have stressed on topological and distance relationships [10], nevertheless, directional relationships are also of great significance in scenarios such as video surveillance, robotics and scene understanding. For example, in video surveillance, the directional relationship "*north of*" can provide a more specific predicate than distance relationship "*near*" by pointing out a specific direction thus can support more accurate spatial queries.

In this paper, we only focus on directional relationships between two different objects; and we assume one object does not contain or contact the other. We investigate techniques for the interpretation of 3D spatial relationships for the LVDBMS (*Live Video DataBase Management System*) [4, 11]. The LVDBMS is a general-purpose framework for managing and processing live motion imagery data for surveillance and analytical applications. This system allows automatic monitoring and management of a network of live cameras from a single location. The user is able to specify a monitoring task by formulating a query to describe a spatiotemporal event. The query is in the form of combinations of some logical, spatial, and temporal operators. When the specified event occurs, an action associated with the query is triggered. In other words, the LVDBMS treats a camera as a special class of storage, and processes the queries against the live video feed as a new category of databases. This general-purpose platform allows rapid development of

MMSys'13, February 26-March 1, 2013, Oslo, Norway.

live video applications much like database applications are generally implemented atop a database management system today. A critical functionality of the LVDBMS is the interpretation of spatial relationships among the objects. The current system performs this task based on 2D spatial logic. This simplified approach has some limitation. As an example, when someone appears spatially overlapping with a chair from the view point of a camera, it does not necessarily mean that this person is sitting at the chair. With a depth sensor such as the Kinect sensor [12], 3D logic can be exploited to resolve this ambiguity and more accurately interpret the spatial relationship between this person and the chair. Kinect sensor is capable of providing high quality synchronized RGB and depth streams. A point cloud data structure can be further generated and used for scene reconstruction. With its affordable price and relatively high accuracy, it is now feasible for consumers to deploy their systems and reconstruct 3D scene that were only available by using expensive laser equipment in the past.

The challenge is that we cannot simply reuse the 2D-based algorithms applied in [4, 11] to compute the 3D spatial logic. New algorithms must be designed to take advantage of the point cloud and effectively model the scene. In addition, since our spatial logic is used for live video computing, it is necessary to design algorithms that can run in real time.

In this paper, we extend the original directional relationships into 3D space. We add two major directions into the original direction set and define a set of new directional relationships that are suitable for 3D scene. We introduce a simple yet effective way to extract objects from the RGB and the depth streams from the Kinect sensor and build 3D models. We then propose a highly accurate and efficient algorithm to compute the spatial relationships between objects with complex structures.

The contributions of this paper are as follows:

1. we define a new set of directional spatial operators that can be effectively applied in 3D space;

2. we introduce an efficient 3D object modeling method using Kinect sensor; and

3. we propose an accurate sampling-based algorithm to compute the 3D directional relationships of objects with complex structure.

The remainder of this paper is organized as follows. Related work in spatial relationships is surveyed in Section 2. We describe the proposed set of 3D directional operators in Section 3. In Section 4, we introduce the 3D object modeling algorithms. The algorithm for computing spatial operators is presented in Section 5. Demonstrations and experiments are presented and analyzed in Section 6. Finally, we conclude the paper in Section 7.

2. RELATED WORK

The nature of spatial relations among objects requires them to be described in an approximate (fuzzy) framework [13]. This is quite intuitive and common to human understanding when we interpret directions. Most of the existing methods adopt the fuzzy framework and interpret spatial relationships by a number or a range indicating the degree of applicability; and the spatial relationship of two complex objects can be interpreted by a combination of primitive spatial operators. A good survey of different methods for computing spatial relationships is presented in [10].

A large number of methods in literature are angle-based. In [14], the angle between the connection of two points and the x-axis is measured and a cosine function is applied to compute the direction "*to the right*" of one point with respect to the other point. By changing the x-axis to other directions, any 2D directional relationships can be computed similarly. Another way to represent an arbitrary direction is to use a combination of basic directions [10]. For example, "*oblique right*" is defined by "*above and right of*" or "*below and right of*". [14] proposes a simple solution to evaluate the spatial relationship of two objects using the centroid method. Objects are represented by their centroid and the angle-based methods are employed to compute the directional relationships. [15] computes the angle between any pair of points in the two objects, and generates a histogram of angles and a weighted histogram of angles. A fuzzy set is defined to represent the compatibility between each histogram and the spatial relations. [14] also computes the angle of all points between two objects, the difference is that they do not incorporate histograms; instead, they aggregate the result of every pair of points by the weighted mean. Those methods based on all points of objects can achieve higher accuracy at the cost of more computational overhead.

Angle-based methods cannot achieve good performance in the scenario of complex objects. Machine learning-based methods are also proposed to address the complex spatial relationships. In [16], neural networks are incorporated as classifiers to learn different spatial relationships between a few basic shapes. Since each type of shape requires an independent classifier, such a method is not scalable for many applications. [3] trains an SVM (Support Vector Machine) to detect the contact points of two objects. Those points are further clustered into groups to establish the Contact Points Network and serves as a skeletonized description of the objects. This skeleton graph is finally used to identify the spatial relationships of the objects. The skeleton graph has limited capability in representing complex spatial relationships; and this approach can only support two simple spatial relationships which are "*on*" and "*adjacent*".

A morphological approach [17] has been proposed to assess the spatial relationships of multiple objects with respect to the same reference simultaneously. This method contains two steps. First, a fuzzy landscape is computed around the reference object; and then the target object is compared against the fuzzy landscape by a fuzzy pattern matching approach. The result is a range of minimum and maximum likelihood of a certain directional relationship. Since the entire space (landscape) around the reference object is computed, this method can address the spatial relationships between complex objects. However, it needs one computation for each direction of interests, which inevitably increases the computational overhead.

3. DIRECTIONAL SPATIAL RELATIONSHIPS IN 3D SPACE

In this section, we briefly review the directional relationships in 2D space and discuss their limitations in 3D space. We later introduce how we extend them into the 3D space and give a complete set 3D spatial relationships.

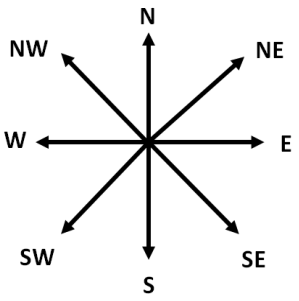


Figure 1: Directional relationships in 2D space.

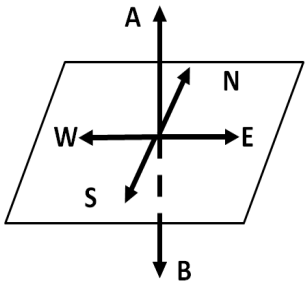


Figure 2: Six major directions in 3D space. (A represents the direction "above", B represents the direction "below")

A 2D space has different primitive directional relationships such as "*left*", "*right*", "*front*" and "*back*". The same relationships can also be interpreted by the geographic terms such as "*east*", "*west*", "*north*" and "*south*". The only difference is, the former are relative relations while the latter are absolute or global relations. In this paper, we adopt the global term as our directional relationship systems. Figure 1 shows the eight primitive directions in 2D space.

We note that, these eight directions are defined with the assumption that, all objects are roughly located on the same plane. Let's consider the following situation. Two objects both locate at the NE (*northeast*) direction to the reference object, but one of them is above the plane, the other is below the plane. In such scenario, NE is too vague to distinguish the difference between the directions of these two objects. However, such difference is significant in 3D space and may lead to different results in the application. Therefore, traditional direction relationships are not capable of providing accurate information in 3D space. To solve the issue, an intuitive way is to add two major directions "*above*" and "*below*" into consideration. Figure 2 shows the six major directions in 3D space.

We note that Figure 2 is actually the Cartesian coordinate system. We can generate the 3D directional relationships by combining any three orthogonal major directions in Figure 2. This is exactly the same as how NE, NW, SE and SW are defined in 2D space. Similarly, a complete set of 3D directional relationships consisting of 26 primitive directions are obtained. Figure 3 shows how this direction set is fit to a 3×3 cube.

As shown in Figure 3, the original eight 2D directions are preserved in the middle level, two directions A and B are added, which means "*above*" and "*below*", respectively. Another 16 new primitive directions are generated by concatenating "A" or "B" with the eight original 2D directions. For example, ANW means "*northwest*" in the "*above*". In such way, a 3D space is divided into 26 subspaces and each one is capable of representing a primitive 3D directional spatial relationship. We note that, the traditional 2D directions are a subset of the new 3D directional spatial relationship set.

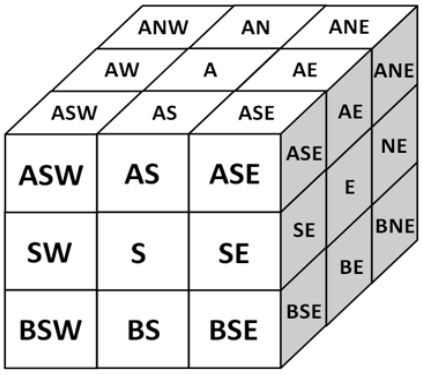


Figure 3: 3D direction set in a 3×3 cube.

4. 3D OBJECT MODELING

In this section, we introduce the 3D object modeling method. We use the OpenNI SDK [18] to obtain the calibrated and synchronized RGB stream and depth stream. Two sample images from the streams are shown in Figure 4. Since the depth camera has a smaller field of view than the RGB camera, OpenNI SDK zooms out the original depth image and fills the blank region with black pixels (invalid pixels only for padding) in order to get a match with the RGB image. In such way, each valid pixel in the depth image (Figure 4(a)) has a corresponding pixel in the RGB image(Figure 4(b)).

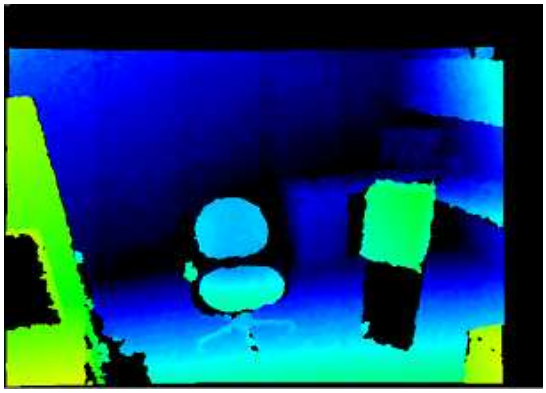
A six-tuple point cloud data structure $<r, g, b, i, j, d>$ can be obtained for each matched pixel between the depth image and the RGB image, where i and j are the index of the pixel in the image. d is the raw reading of pixel (i, j) from the depth image, it can be further converted to the real depth value z which is the distance from the surface of the object to the depth sensor. OpenNI SDK provides the conversion. Using the calibration parameters of both cameras, each pixel (i, j) in the RGB image and depth image can be converted to a 3D coordinate $<x, y, z>$ in the camera coordinate system with the camera as the origin. This can be computed by (1) and (2).

$$x = \frac{(i - u_0) \times z}{a_x}, \tag{1}$$

$$y = \frac{(j - v_0) \times z}{a_y} \tag{2}$$

where (u_0,v_0) represent the principal point, which would be ideally in the center of the image. a_x and a_y represent focal length in terms of pixels in the x-axis and the y-axis directions. These parameters can be found in the intrinsic matrix of the camera after calibration.

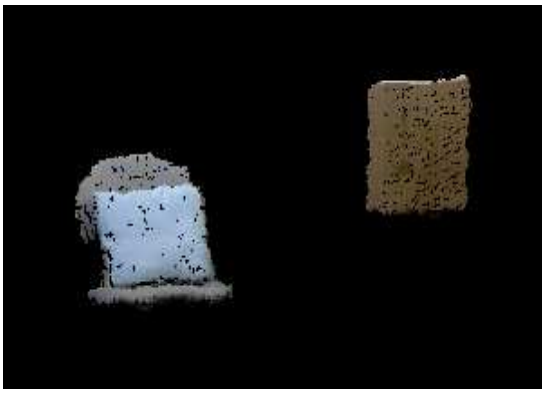
Next, objects are extracted from the scene by image segmentation techniques. Since the object segmentation from stereo images is not the contribution of this paper, we simply implement the Meanshift segmentation algorithm [19] on



(a) Depth image

(b) RGB image

Figure 4: Depth image and RGB image of the same frame.



(a) point cloud of the objects

(b) Original image

Figure 5: point cloud of objects of interests.

both the RGB image and the depth image. We firstly subtract the floor and wall from the scene using segmentation on only the RGB image. Then objects are segmented from the remaining content of the scene using only the depth image. Since we are not focusing on automatic object recognition, we manually select the segments of the object of interest and extract the object by combining all the neighboring segments. One of the objects serves as the reference, the other serves as the target. The point clouds of the two objects are then extracted from the scene as shown in Figure 5. Please note that RGB image is only used for segmenting back ground pixels (wall, floor), they do not contribute to the following modeling procedure. We put color in Figure 5 only for illustration purpose.

We use a single Kinect sensor in our algorithm and mount it to a fixed position about 1.8 meters above the floor. Since only one sensor is employed, only half of the scene is sensed by the device. Information on the backside of objects is totally missing. For example, the points of the back of the chair and the back of the box are unknown to the Kinect sensor. Therefore, the point clouds are incomplete and cannot generate 3D models that can represent the true spatial relationships between objects. For example, incomplete point cloud can make an object thinner than its true thickness and thus can generate inaccurate spatial relationships. To address this issue, we fill up the point cloud of each object by applying heuristics similar to Manhattan world suggestion [20] which is commonly assumed in 3D scene understanding. We assume that the hidden points on the back of an object have the same depth value of the corresponding point on the top or bottom contour of the object. By this assumption, hidden points on the back, side and bottom of the object are added to the point cloud and make it complete. Before filling up the point cloud, we need to perform a rotation transformation to the point cloud so that the original floor plane in the camera coordinate system is parallel to the floor plane in the world coordinate system. This is done by rotating the scene according to the extrinsic parameters of the camera (camera pose). We can see in Figure 5(b) that the original scene has some skew due to the camera pose, after rotation, the object model is aligned according to the floor plane (Figure 5(a)). Algorithm 1 shows the detailed algorithm to fill up a complete point cloud model.

Algorithm 1 can fill up the hidden point on the back, side and bottom of the object and generate a complete 3D model of the object. Figure 6 illustrates an example of the completed object model.

We can see in Figure 6 that the bottom of the chair is filled up by blue pixels, the back of the chair is filled up by green pixels and the side of the chair is filled up by red pixels. We note that, when aligning the 3D model, we only rotate the scene around x-axis and z-axis to make the floor plane in the camera system parallel with the floor plane in the world system, we did not rotate the scene around y-axis. We intentionally leave the skew in the y-axis untouched because it represents the direction the camera is facing and spatial relationships are computed based on it.

Algorithm 1 Algorithm of filling up the point cloud.

1: Compute the contour of the object in the depth image,
2: For each pair of points on the contour sharing the same X coordinate (top contour point and bottom contour point), insert a point at the same X coordinate from the top contour to the bottom contour. All inserted points have the same depth value of the top or bottom contour point whichever has a larger depth value,
3: For each point on the top and bottom contour, insert a point at the same X and Y coordinates with the depth value going further along the Z axis until reaching the correspond back point inserted in step 2. The step size of the depth value is selected based on experiment,
4: For each pair of points on the contour sharing the same Y coordinates (left contour point and right contour point), do the same as step 3.

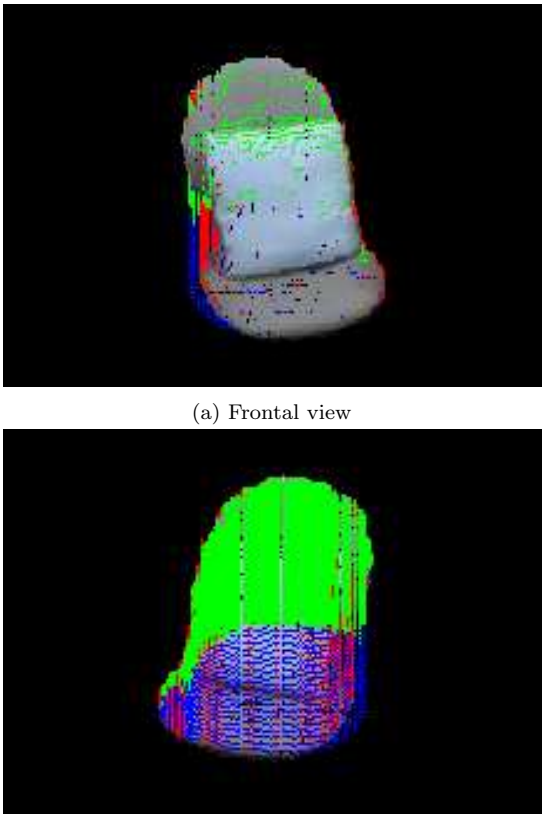
5. ALGORITHM FOR COMPUTING DIRECTIONAL RELATIONSHIPS

In this section, we describe our algorithm for computing directional relationships in detail. Most conventional methods are not capable of detecting the directional relationships between objects with complex structures. The challenge lies in how to maintain a global interpretation of the directional relationship while still considering the impact of each substructure of both objects. Normally, any single primitive direction relationship is too vague to interpret the global relationship in 3D space. The global relationship should be in the form of combinations of several primitive relationships.

The idea of our algorithm comes from another technique named "ray tracing". This technique is commonly applied in computer graphics for scene rendering [21]. In ray tracing, a number of rays are generated from the camera and go into the scene. If the ray intersects with the object in the scene, a corresponding texture on the surface of the object will be displayed on the screen when rendering. It can be noted that ray tracing is a sampling technique. Similarly, our technique applies the same idea to achieve a global representation of spatial relationships. The basic idea of our algorithm is: we sample the entire space wrapping the reference object by generating a number of rays from the reference object. Each ray represents a specific 3D primitive direction. If the ray hits any part of the target object, the target object is partially in the direction represented by the ray. Results of each ray are concatenated to form a feature vector and a Gaussian filter is employed to generate the final relationship.

5.1 Generating Rays into the Sphere

We uniformly generate 840 rays into the sphere wrapping the reference object. The origin of the ray can be any point in the point cloud of the reference object. Each ray has a direction of $<\theta, \phi>$, where $\theta \in [0, \pi], \phi \in [0, 2\pi)$. We quantize the range of θ into 20 intervals ($\theta_0 = 0, \theta_1 = \frac{\pi}{20}, \cdots, \theta_{20} = \pi$) and the range of ϕ into 39 intervals ($\phi_0 = 0, \phi_1 = \frac{\pi}{20}, \cdots, \phi_{39} = \frac{39\pi}{20}$). We note that, this quantization strategy leads to increased ray density at poles of the sphere, we will further reduce the number of rays at poles to make the density equal to other regions. We note that geodesic sphere can generate an equal division of a sphere, we will address this issue in our future work. Figure 7 illustrates how a ray is generated.



(a) Frontal view

(b) Back view

Figure 6: Frontal view (left) and back view (right) of a complete object model.

In Figure 7, θ is the angle between the ray and the positive direction of y-axis, ϕ is the angle between the projection of the ray on x-z plane and the positive direction of x-axis in a counter-clockwise manner.

5.2 Ray and the Target Object Intersection

In this subsection, we explain how to determine if the ray hits the object. In our algorithm, this is achieved by rotation and projection. In the rotation process, we first translate the entire scene to the origin of the ray and rotate the scene according to the ray's direction. More specifically, the scene is rotated around x-axis and y-axis with $\frac{\pi}{2} - \theta$ and $\phi - \frac{\pi}{2}$, respectively. We later project the target object and the ray to the x-y plane. The projection of the ray on x-y plane is then a single point. If the ray hits the object, the projection of the ray should be inside the projection of the target object. Finally, each ray will generate a result of "1" or "0", with "1" indicating the ray hits the object and "0" otherwise. We sequentially concatenate the result of all 840 rays and obtain a 840-dimension feature vector.

To reduce the computational overhead, we compute the bounding box of the target object and implement ray-bounding

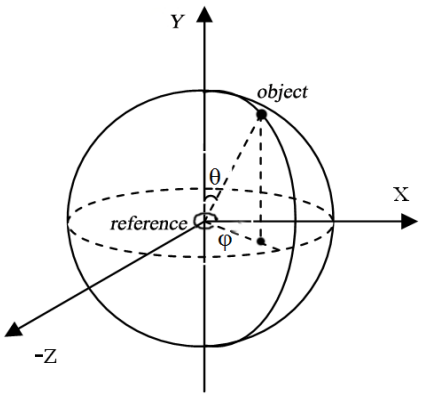


Figure 7: Generating rays from the reference object.

Table 1: Mapping from θ to Primitive Directions

θ_i	**Primitive direction**
$\theta_0, \theta_1, \theta_2$	A
$\theta_3, \theta_4, \theta_5, \theta_6, \theta_7$	AE ANE AN ANW AW ASN AS ASE
$\theta_8, \theta_9, \theta_{10}, \theta_{11}, \theta_{12}$	E NE N NW W SW S SE
$\theta_{13}, \theta_{14}, \theta_{15}, \theta_{16}, \theta_{17}$	BE BNE BN BNW BW BSW BS BSE
$\theta_{18}, \theta_{19}, \theta_{20}$	B

box intersection to speedup the above process. Bounding box is computed as follows. The min point (P_{min}) and max point (P_{max}) of the point cloud are found to establish a bounding box of an object [21]. (3) and (4) show how P_{min} and P_{max} are computed from the object model.

$$P_{min} = (x_{min}, y_{min}, z_{min}), \tag{3}$$

$$P_{max} = (x_{max}, y_{max}, z_{max}), \tag{4}$$

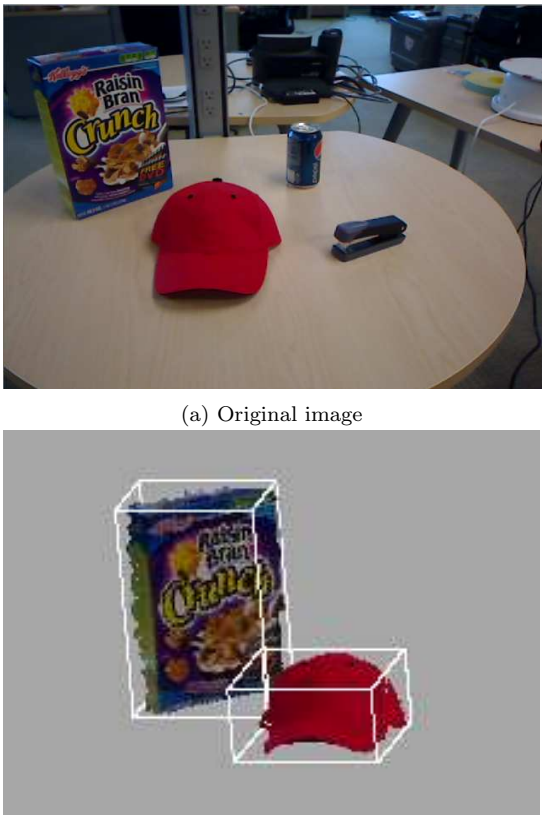
where x_{min} and x_{max} are the min and max x coordinate of the 3D object in the aligned point cloud, respectively. So are y_{min}, y_{max} and z_{min}, z_{max}. For each ray, if it does not intersect with the bounding box, we do not perform the process of ray and target object intersection. This can significantly reduce the computation. Figure 8 shows an example of the bounding box.

It is possible that a tiny object may fail to be hit by any rays. Our solution for this scenario is to generate a ray directly toward the centroid of the tiny object and compute $<\theta, \phi>$. We then find the nearest neighboring ray and assign its intersection result to be true. We note that in this case, our solution reduces to the centroid method [14].

5.3 Generating the Global Directional Relationship with Gaussian Filter

We note that the number of rays is much larger than the number of primitive directions as defined in Section 3, therefore a mapping process is applied to map the 840-dimension feature vector into 26 primitive directions. Table 1 and Table 2 show how this mapping is performed.

It can be found in Table 1 and Table 2 that, every primitive direction has exactly 25 rays, with both θ and ϕ having 5 consecutive values. The only exceptions are primitive direction A and B which are at the poles. These two directions



(a) Original image

(b) Bounding box

Figure 8: Bounding box of the reconstructed object.

have 120 rays each. To make them uniform as the others, we further down sample them to have only 25 rays.

For every primitive direction, we mask the corresponding 25 rays by a 5×5 operator which preserves the original layout of the ray in the sphere. Table 3 gives an example of the 25 rays in primitive direction NE.

From Table 3, we note that, all 25 rays contribute to NE, nevertheless, some rays are more significant than the others. For example ray $<\theta_{10}, \phi_5>$ is right in the center of the NE direction, it coincides with the NE direction in the 2D space. In order to demonstrate the significance of some predominant rays, we filter the 5×5 operator by 2D normalized

Table 2: Mapping From ϕ to Primitive Directions

ϕ_i	**Primitive direction**
$\phi_{38}, \phi_{39}, \phi_0, \phi_1, \phi_2$	AE E BE
$\phi_3, \phi_4, \phi_5, \phi_6, \phi_7$	ANE NE BNE
$\phi_8, \phi_9, \phi_{10}, \phi_{11}, \phi_{12}$	AN N BN
$\phi_{13}, \phi_{14}, \phi_{15}, \phi_{16}, \phi_{17}$	AWN WN BWN
$\phi_{18}, \phi_{19}, \phi_{20}, \phi_{21}, \phi_{22}$	AW W BW
$\phi_{23}, \phi_{24}, \phi_{25}, \phi_{26}, \phi_{27}$	ASW SW BSW
$\phi_{28}, \phi_{29}, \phi_{30}, \phi_{31}, \phi_{32}$	AS S BS
$\phi_{33}, \phi_{34}, \phi_{35}, \phi_{36}, \phi_{37}$	ASE SE BSE

Table 3: 25 Rays Contribute to the Primitive Direction NE

$<\theta_8,\theta_3>$	$<\theta_8,\theta_4>$	$<\theta_8,\theta_5>$	$<\theta_8,\theta_6>$	$<\theta_8,\theta_7>$
$<\theta_9,\theta_3>$	$<\theta_9,\theta_4>$	$<\theta_9,\theta_5>$	$<\theta_9,\theta_6>$	$<\theta_9,\theta_7>$
$<\theta_{10},\theta_3>$	$<\theta_{10},\theta_4>$	$<\theta_{10},\theta_5>$	$<\theta_{10},\theta_6>$	$<\theta_{10},\theta_7>$
$<\theta_{11},\theta_3>$	$<\theta_{11},\theta_4>$	$<\theta_{11},\theta_5>$	$<\theta_{11},\theta_6>$	$<\theta_{11},\theta_7>$
$<\theta_{12},\theta_3>$	$<\theta_{12},\theta_4>$	$<\theta_{12},\theta_5>$	$<\theta_{12},\theta_6>$	$<\theta_{12},\theta_7>$

Gaussian filter given by (5),

$$g(x,y) = \frac{1}{A} \cdot \frac{1}{2\pi\sigma^2} \cdot e^{-\frac{x_+^2 y^2}{2\sigma^2}}, \tag{5}$$

$$A = \sum_{x=-2}^{2} \sum_{y=-2}^{2} g(x,y), \tag{6}$$

where σ=1 , A is the summation of $g(x,y)$. The final result is the summation of all 25 rays given in (7),

$$Result = \sum_{x=-2}^{2} \sum_{y=-2}^{2} f(x,y)g(x,y), \tag{7}$$

$$f(x,y) = \begin{cases} 1, & \textit{ray hits the object;} \\ 0, & \textit{otherwise.} \end{cases}, \tag{8}$$

where, $f(x,y)$ represents whether a ray hits the object or not. When all 25 rays hit the object, The result is normalized to 1. The value of the result indicates how predominant the part of the object lies on this direction. The greater the result is, the more significant rays hit the object and therefore, it is more confident to say the part of the object lie in this primitive direction.

The global result can be represented in the combination of several primitive directions. For example, NE=0.8 and ANE=0.2, which means the target is mainly in the northeast direction but also with a little bit on the above of the reference object.

5.4 Centroid-based Method and Landmark-based Method

In this subsection, we introduce two strategies to represent the reference object, a centroid-based method and a landmark-based method. Similar to [14], the centroid-based method represents the reference object by its centroid. The motivation behind this representation is that as the ray goes far away from the reference point into the space, the structure of the reference object reduces to a single point in the 3D space. Therefore in the centroid-based method, all the rays are originated from the centroid of the reference object, it is highly computational efficient and can be very useful when the reference object is significantly smaller than the target object. The disadvantage is, when reference object is with complex shape or with a larger size than the target object, single centroid can be inaccurate to represent the structural information of the reference object and thus compromises the performance.

The landmark-based method uses a set of landmark points of the reference object model to represent the reference object. For each landmark point, a ray is generated from it and the centroid-based method is applied to it. Landmark points must be evenly selected from the reference object, and

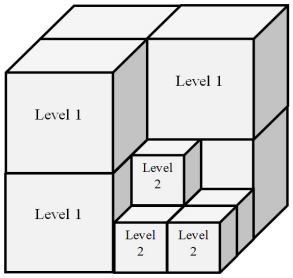


Figure 9: Two-level octree decomposition.

Figure 10: Reference object (cereal box) is represented by 33 landmark points.

every landmark point has an equal contribution to the entire spatial relationship. In this paper, we applied the octree coding [22] to find the landmark points. An octree is a tree data structure in which each internal node has exactly eight children. Octrees are most often used to partition a three dimensional space by recursively subdividing it into eight octants. We use an octree to subdivide the bounding box of the reference object. Since each level of decomposition will lead to eight octants, n levels of decomposition will create 8^n subcubicles. We only apply two levels of octree decomposition considering the tradeoff between performance and computational intensity. An example of the two-level octree decomposition is shown in Figure 9.

After two levels of octree decomposition, the space wrapped by the bounding box of the reference object is evenly divided into 64 sub-cubicles. In each sub-cubicle, the centroid of all points inside is employed as its landmark point. We note that, not all sub-cubicles are filled with at least one point of the reference object, there may be empty sub-cubicles. Therefore, the number of landmark points is not fixed. But the maximum number of landmark points is 64. An example of a 33-landmark reference object representation is shown in Figure 10.

Compared to the centroid-based method, the landmark-based method can better represent the reference object in the face of complex object structure by taking each substructure of the reference object into consideration, nevertheless, the computation is more intensive. In the experiment section, both methods are tested and their performance is compared against each other.

6. EXPERIMENTS

In order to intensively evaluate the performance of our algorithm, we conduct experiments on both a real indoor environment and a public RGB-D dataset [23].

6.1 Demonstration in An Indoor Environment

In this section, we demonstrate the performance of our algorithm in a real indoor scene. We mount a single Kinect sensor to a tripod 1.8 meters above the floor. The sensor is facing down toward the scene with an angle between z-axis and the floor plane to be 22.5^o. This is to simulate cameras in the LVDBMS which are always mounted at a high spot. We assume the camera is fixed. The Kinect sensor has a practical sensing range of $1.2 \sim 3.5$ meters, therefore we can only utilize a small part of the scene. Nevertheless, this is only the limitation of the hardware device not the limitation of our algorithm. Considering the computational efficiency, we applied only the centroid-based method in our demonstration experiment.

Figure 11 shows four scenarios we used to demonstrate the effectiveness of our algorithm. In the above scene, we assume the white board is on the north. In Figure 11(a) and Figure 11(b), we evaluate the directional relationships of the box and the chair, in Figure 11(c) and Figure 11(d), we evaluate the directional relationship between two boxes. Table 4 summarized the experimental results of the above scenes. In Table 4, only the primitive directions with values greater than zero are listed.

In Figure 11(a), the chair is roughly in the northwest direction to the box and also below the box. Experimental result has correctly represented this observation with BNW and NW all greater than zero. If we swap the reference object and the target object, the corresponding result is also correct. For example, in Figure 11(a), BNW becomes ASE and NW becomes SE if the reference and target are switched. We note that the output values between the corresponding pair of relationships are not identical, in Figure 11(c), $AN = 0.1684$ while $BS = 0.5530$. This is because the large box and small box have different size. Since the small box takes less space than the large one, the small box may be hit by fewer rays than the large box does. That's why the BS has a larger value than AN in Figure 11(c). This can also explain why the spatial relationships are not symmetrical in Figure 11(a) where BW cannot find a match up when the target and reference objects are switched. However, the predominant relationships, NW and BNW, are perfectly matched with SE and ASE in Figure 11(a). The above experiments have demonstrated the effectiveness of our algorithm in a real indoor scene.

Table 4: Experimental Results of A Real Scene

Scene	Reference object	Target object	Spatial relationship
(a)	box	chair	$BNW = 0.0183, NW = 0.0439, BW = 0.00367$
(a)	chair	box	$ASE = 0.0183, SE = 0.0915$
(b)	box	chair	$BSE = 0.01464, SE = 0.2380$
(b)	chair	box	$ANW = 0.01464, NW = 0.1684$
(c)	large box	small box	$AN = 0.1684$
(c)	small box	large box	$BS = 0.5530$
(d)	large box	small box	$A = 0.328$
(d)	small box	large box	$B = 0.2739$

6.2 Experiment on RGB-D Dataset

To extensively test the performance of our algorithm, we also conduct experiments on a public RGB-D dataset [23]. This large RGB and depth image dataset contains 300 common household objects, such as bowls, cups, dishes, computers, etc. It is recorded using a Kinect camera that captures synchronized and aligned 640×480 RGB and depth images. The camera is slowly moving in the scene so that it can provide images from different views. The dataset is captured in eight different household scenes, for example, a desk, a kitchen table, a meeting room table, etc. Figure 12 gives some image examples of the dataset.

The reason we use this dataset is as follows. Firstly, this dataset contains many indoor environments where multiple objects appear in the same scene, it is potentially convenient for us to investigate the spatial relationships between different objects in one image. Secondly, most objects in this dataset are captured in $1 \sim 2.5$ meters distance away from the camera. In this range, Kinect can provide good accuracy because according to [24], Kinect's random error of depth measurements increases quadratically with increasing distance from the sensor. Thirdly, the RGB and depth images are already calibrated in this dataset, we can perform our algorithm directly on the images.

Since spatial relationships between objects in consecutive images are very similar, we select 43 representative image frames from 8 different scenes as the pool of our image set. Among the 43 images we obtain 214 pairs of objects and investigate their spatial relationships, for example, in Figure 12(b) the white hat and the green bowl are a pair of objects and their spatial relationship is, the white hat is to the *northeast* of the green bowl.

6.2.1 Estimating Camera Extrinsic Parameters from Images

The original purpose of the RGB-D dataset is 3D object recognition, it does not provide any camera calibration parameters. This brings about a challenge to our experiment because those parameters are crucial for 3D model reconstruction and alignment. In our experiment, we address this issue in the following way. For the camera intrinsic parameters (*e.g.* focal length, principal point, etc), we use the same parameters that are obtained when calibrating our own Kinect camera. We assume the camera intrinsic parameters do not vary significantly between the same products. For the extrinsic parameters camera pose), we estimate them directly from the image. Extrinsic parameters can be estimated by computing the normal of the floor plane in the camera coordinate frame, which contains all the information needed to compensate the skew of the camera orientation. Since images in the database are captured in a household scene, there is always a table or a desk in the scene (see Figure 12). By manually selecting three points on the table plane, a pair of non-parallel line segments can be obtained. The *cross product* of these two line segments gives us the initial guess of the normal of the table plane, which can be further used to estimated the table plane by RANSAC plane fitting [25] .The parameters (rotation matrix) needed to align the 3D object model can be computed by (9) and (10),

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

Figure 11: Examples of spatial relationships in a lab scene.

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

Figure 12: RGB-D dataset [23] examples.

$$\Delta\theta = \arccos\left(\frac{(1-\Delta x^2)+\Delta y^2-\Delta z^2}{2\sqrt{1-\Delta x^2}\Delta y}\right), \tag{9}$$

$$\Delta\phi = \arctan\left(\frac{\Delta x}{\Delta y}\right), \tag{10}$$

where Δx, Δy and Δz are the x, y and z component of the normalized normal vector, respectively. Then the table plane can be aligned parallel to the floor plane of the world coordinate system by rotating around x-axis by $\Delta\theta$ and rotating around z-axis by $\Delta\phi$.

6.2.2 Labeling the Ground Truth

Another challenge is how to obtain the ground truth (GT). All existing RGB-D datasets are not designed for detecting the spatial relationships between objects, therefore there is no GT of spatial relationships available. As a result, we label the GT by ourselves. For each pair of objects, we manually assign all the possible spatial relationships as candidate relationships according to human observation. Since spatial relationship is a fuzzy concept, it is extremely difficult to find a complete GT between two objects. However, the predominant spatial relationship should not differ greatly from human observation. For example, in Figure 12(b) the white hat is to the *northeast* of the green bowl, *northeast* is the predominant relationship. Therefore we select NE as the major relationship because it best suits human observation. We note that in the face of complex objects or objects of different sizes, a single major relationship may be insufficient. For example, in Figure 12(d), the size of the cereal box is much larger than the size of the soda can. Given the soda can as reference object, the spatial relationship between them are ANW, NW, AN and N, we see that no single relation can completely represent the spatial relationship between them. Therefore we label all four relationships as candidate relationships and select ANW as the major relationship. In our experiments, both major relationships and candidate relationships are employed as GT for performance evaluation.

Table 5: Experimental Results on RGB-D Dataset by Major Spatial Relationship (centroid-based)

Total pairs	**Corrected detected pairs by major relationship**	**Major relationship precision**
214	187	87.4%

6.2.3 Experimental Results of Centroid-based Method

In this set of experiment, centroid-based method is evaluated. Experimental results based on major relationship are showed in Table 5. The relationship with the largest output value among all candidate relationships is selected as the major relationship. Precision is adopted as the performance metric by comparing the detected result against GT using only the major relationship. We use the precision of the major relationship to measure how well the result coincides with human observation. From the result in Table 5, we can see that 87.4% of the object pairs are correctly determined by their major relationships.

To evaluate the capability of our algorithm in detecting a complete set of spatial relationships between two objects, we also use candidate relationships in the performance measurement. In this experiment, every relationship with a output value greater than zero is preserved as a candidate relationship. We note that each candidate relationship may have different output values, and their contributions to the entire relationship set are different. However, it is extremely difficult to assign the significance when labeling the GT, therefore we do not distinguish the significance of candidate relationships, they all have the same contribution to the final result. We adopt *recall* and *precision* as the performance metrics. Their definitions are as given by (11) and (12),

Table 6: Experimental Results on RGB-D Dataset by All Candidate Relationships (centroid-based)

Total candidate relationships in GT	Correctly detected	Total detected	Recall	Precision
384	279	315	72.8%	88.6%

Table 7: Experimental Results on RGB-D Dataset by Major Spatial Relationship (landmark-based)

Total pairs	Corrected detected pairs by major relationship	Major relationship precision
214	191	89.3%

$$\text{recall} = \frac{\text{num of correctly detected relationships}}{\text{total num of relationships in the GT}}, \quad (11)$$

$$\text{precision} = \frac{\text{num of correctly detected relationships}}{\text{num of detected relationships}} \quad (12)$$

Experimental results based on candidate relationships are summarized in Table 6. We manually labeled 384 candidate relationships as GT, 279 of them are correctly detected by our algorithm, meanwhile there are also 36 false alarms. Only 72.8% of the total candidate spatial relationships are discovered based on the centroid-based method. There are two main reasons for this. On one hand, labeling the GT from 2D images is extremely challenging, especially when objects in the 2D image are presented with skew and rotation. Therefore, the GT regarding the candidate relationships is not very precise itself. On the other, the RGB-D dataset does not provide the camera calibration information, we have to use the default intrinsic parameters and estimate the extrinsic parameters (camera pose) by ourselves, this can also bring interference into the results.

6.2.4 Experimental Results of Landmark-based Method

In this set of experiment, landmark-based method is evaluated. All experimental settings and evaluation criteria are exactly the same as in the previous sub-section.

Experimental results based on major relationship are summarized in Table 7. We can see that landmark-based method can achieve a higher precision than the centroid-based method regarding only the major spatial relationship. The result coincides with our expectation since landmark points are more effective to represent the shape of the reference object than the single centroid. We note that objects in the RGB-D dataset are mostly regular household items, they have regular shapes and similar sizes, such as cups, bowls, torches, staples, cereal boxes, etc. Therefore, the performance of landmark-based method does not vary significantly to the centroid-based method in this case. We expect the landmark-based method can demonstrate more advantages with objects of more complex shape and different sizes.

Experimental results based on candidate relationships are summarized in Table 8. Again, we have 384 candidate relationships in the GT, 317 of them are correctly detected with 59 false alarms. Since the landmark-based method stacks over the results of each landmark point, any point can have an interference to the final results. To make the results stable we further remove those candidate relationships with an accumulated value less than 0.01.

Table 8: Experimental Results on RGB-D Dataset by All Candidate Relationships (landmark-based)

Total candidate relationships in GT	Correctly detected	Total detected	Recall	Precision
384	317	376	82.5%	84.3%

Table 9: Comparison on the Speed of Both Methods

centroid-based	landmark-based	average landmark points
0.0418s	1.37s	32.7

Table 8 shows the landmark-based method can discover more candidate relationships than the centroid-based method at the expense of a higher false-alarm rate. Since the reference object is decomposed into different parts, by combining the local results of each landmark point, global spatial information is reinforced. When comparing the results of Table 8 and Table 6, we can see that landmark-based method can significantly improve the *recall* (from 72.8% to 82.5%) without sacrificing too much *precision* (drop from 88.6% to 84.3%) which demonstrates its advantage over the centroid-based method.

The landmark-based method can outperform the centroid-based method on metrics of *recall* and *precision*, however, the computation is inevitably more intensive than the centroid-based method. Table 9 summarizes the average time needed to detect the spatial relationship between a pair of objects in the RGB-D dataset by both methods. The experiments are conducted on a system of Intel i7 quad cores CPU and 24GB RAM, no multi-thread programming is used. The elapsed time only includes the 3D object modeling and the spatial relationship detection, it does not include the time for image segmentation. We note that centroid-based method takes only 41.8 ms to detect the spatial relationships, which is capable for real time application. Whereas the Landmark-based method takes an average of 1.37 seconds to do the same thing, almost 32 times slower than the centroid method. The average number of landmark points generated for reference object is also 32.7, which coincides with the ratio of the elapsed time between these two methods. Although the landmark-based method is theoretically slower, parallelism can be exploited to increase its speed and we leave it to the future work.

7. CONCLUSIONS

Accurate interpretation of spatial relationships between objects is essential in many applications such as robotics, video surveillance, and scene understanding. With the popularity of affordable depth sensors, 3D scene reconstruction can be applied in various applications to better interpret spatial relationships among objects. In this paper, we present a new 3D spatial relationship interpretation technique for LVDBMS, a generic platform for live video computing. We introduce a simple yet effective way to build 3D models using a Kinect sensor. A highly accurate algorithm is proposed to

compute the spatial relations between two objects by sampling the entire space from the reference object. Experimental results on both a real scene and an RGB-D database have demonstrated the effectiveness of our techniques.

For future work, we would like to address the issue of building complete 3D models by employing multiple depth sensors. We note that using a single fixed sensor inevitably loses half of the information in the scene. Although we fill in the model based on heuristics, the estimation may still bring errors to the model and affect the performance of the following procedures. This limitation can be addressed by fusing the information from multiple depth sensors or exploiting the mobility of single depth sensor to generate a complete 3D object model. We would also like to leverage parallel computing techniques to improve the speed of the landmark-based method.

8. ACKNOWLEDGMENTS

This material is based upon work supported by the National Science Foundation under Grant Number CNS 0917082. Any opinions, findings, and conclusions or recommendations expressed in this materials are those of the authors and do not necessarily reflect the views of the National Science Foundation.

9. REFERENCES

[1] H. S. Koppula, A. A, T. Joachims, and A. Saxena. Labeling 3d scenes for personal assistant robots. In *arXiv:1106.5551v1*.

[2] A. Kasper, R. Jakel, and R. Dillmann. Using spatial relations of objects in real world scenes for scene structuring and scene understanding. In *Proceedings of The 15th International Conference on Advanced Robotics (ICAR)*, pages 20–23, June 2011.

[3] B. Rosman and S. Ramamoorthy. Learning spatial relationships between objects. *International Journal of Rotbotics Research*, 30(1):1328–1342, 2011.

[4] R. Peng, A. J. Aved, and K. A. Hua. Real-time query processing on live videos in networks of distributed cameras. *International Journal of Interdisciplinary Telecommunications and Networking*, 2(1), 2010.

[5] J. Liu. A method of spatial reasoning based on qualitative trigonometry. *Artificial Intelligence*, 98(1):137–168, January 1998.

[6] V. Hedau, D. Hoiem, and D. Forsyth. Recovering the spatial layout of cluttered rooms. In *Proceedings of IEEE The 12th International Conference on Computer Vision (ICCV)*, pages 1849–1856. IEEE, Sept 2009.

[7] A. Gupta, S. Satkin, A. A. Efros, and M. Hebert. From 3d scene geometry to human workspace. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1961–1968. IEEE, June 2011.

[8] B. J. Kuipers and T. S. Levitt. Navigation and mapping in large-scale space. *AI magazine*, 9(2), 1988.

[9] M. Egenhofer and R. Franzosa. Point-set spatial relations. *International Journal of Geographical Information Systems*, 5(2):161–174, 1991.

[10] I. Bloch. Fuzzy spatial relationships for image processing and interpretation: a review. *Image and Vision Computing*, 23(2):89–110, 2005.

[11] A. J. Aved and Kien A. Hua. A general framework for managing and processing live video data with privacy proctection. *Multimedia Systems*, 18(2):123–143, Feb 2012.

[12] Microsoft. Kinect sensor website. *http://www.xbox.com/en-US/kinect/*.

[13] J. Freeman. The modeling of spatial relations. *Computer Graphics and Image Processing*, 4:156–171, 1975.

[14] J. M. Keller and X. Wang. Comparison of spatial relation definitions in computer vision. In *Proceedings of The Third International Symposium on Uncertainty Modeling and Analysis and Annual Conference of the North American Fuzzy Information Processing Society*, pages 679–684, Sept 1995.

[15] K. Miyajima and A. Ralescu. Spatial organization in 2d segmented images: representation and recognition of primitive spatial relations. *Fuzzy Sets and Systems*, 65(2-3):225–236, August 1994.

[16] J. M. Keller and X. Wang. Learning spatial relationships in computer vision. In *Proceedings of The Fifth IEEE International Conference on Fuzzy Systems*, pages 118–124, Sept 1996.

[17] I. Bloch. Fuzzy relative position between objects in image processing: a morphological approach. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(7):657–664, July 1999.

[18] Rgbd-demo. *http://labs.manctl.com/rgbdemo/*.

[19] D. Comaniciu and P. Meer. Mean shift: a robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(5):603–619, May 2002.

[20] J. M. Coughlan and A. L. Yuille. Manhattan world: compass direction from a single image by bayesian inference. In *Proceedings of The 7th IEEE International Conference on Computer Vision (ICCV)*, pages 941–947, 1999.

[21] A. S. Glassner. *An Introduction to Ray Tracing*. Academic Press, USA, 1991.

[22] Donald Meagher. Geometric modeling using octree encoding. *Computer Graphics and Image Processing*, 19(2):129–147, June 1982.

[23] K. Lai, L. Bo, X. Ren, and D. Fox. A large-scale hierarchical multi-view rgbd object dataset. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, May 2011.

[24] K. Khoshelham. Accuracy analysis of kinect depth data. In *Proceedings of International Society for Photogrammetry and Remote Sensing (ISPRS)*, August 2011.

[25] M. A. Fischler and R. C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.