Querying 3D Data by Adjacency Graphs

Nils Bore, Patric Jensfelt and John Folkesson

Centre for Autonomous Systems at KTH Royal Institute of Technology

Abstract. The need for robots to search the 3D data they have saved is becoming more apparent. We present an approach for finding structures in 3D models such as those built by robots of their environment. The method extracts geometric primitives from point cloud data. An attributed graph over these primitives forms our representation of the surface structures. Recurring substructures are found with frequent graph mining techniques. We investigate if a model invariant to changes in size and reflection using only the geometric information of and between primitives can be discriminative enough for practical use. Experiments confirm that it can be used to support queries of 3D models.

1 INTRODUCTION

Rapid advances in computing and 3D sensing have led to larger and larger 3D data sets being collected by robots and stored for future reference. With the advent of digital cameras and the Internet, a similar situation arose for 2D images, spurring the development of ways to analyze and mine the large amounts of data; these needs now arise for 3D data.

The ability to represent a robot's working environment with simple structures of composite geometric primitives enables both compact representations and the possibility for the robot to reason about its surroundings at a more abstract level. For example, at a high level a bookshelf consists of two vertical sides and horizontal shelves. Most indoor environments consist of combinations of simple substructures repeated throughout the space. Take an office space as an example. It is typically made up of tables, chairs, bookshelves, doorways, pillars, etc. which could be further broken down to simpler parts, e.g. corners or edges.

We would like our robot to be able to look back over its stored data to find specific structures. This would be helpful in a semantic mapping context; perhaps instructed to put a label 'doorway' on all structures that 'look like' some example. It can also be used in an unsupervised transfer learning context, e.g. the robot learns to associate a certain human behavior near a sink in a kitchen. It then finds a similar structure in another room and infers a similar human behavior as a prior. The capability needed is one of being able to query 3D data with representative examples of a structure.

Our approach is based on the idea of having a qualitative representation that can be queried for parts that might be similar. We focus on finding general structures by looking at the surface topology of an indoor environment. We believe that identification of frequent substructures could be an important part

of a robot's understanding of space. The structures could potentially be used as building blocks for a robotic map representation, enabling efficient representation of 3D data gathered by modern robots.

We build on the work in [1] and adapt a popular adjacency graph model to represent configurations of geometric primitives. To find the frequent substructures we look for frequent subgraphs using the *gSpan* algorithm [2].

We contribute a new way of defining discrete pairwise relations in the adjacency graph and propose to have full connectivity locally. This enables us to achieve greater consistency between matched structures. In addition, we extend the approach by learning a graph to search for from a set of example pointclouds.

2 RELATED WORK

The use of frequent patterns for image detection and classification has been studied within the computer vision community. In [3], Nowozin *et al.* demonstrate good classification results with a method based on a combination of graph mining and boosting. The authors suggest that a representation of spatial relations between features is powerful compared to bag-of-words representations, and note that it has the important advantage of easier human interpretation. Jiang & Coenan [4], like [5], propose to use frequent patterns across a set of images as features for classification. As in this paper, both approaches utilize some variant of the popular *gSpan* graph mining algorithm [2]. Within a robotics context, Aydemir *et al.* [6] use *gSpan* to predict what may lie beyond the explored part of the environment.

Many recent papers both in 2D and 3D contexts use over-segmentation to partition a scene into areas that are to be labeled. Those often employ graphical models over adjacent areas to infer semantic labels, primarily by using some kind of probabilistic inference over the graph. An early example of this kind of inference on a stitched point cloud map was presented by Anand *et al.* [7]. As is natural in a 3D context, they use e.g. local shape features for the patches and geometric relations such as co-planarity as pairwise features. Silberman *et al.* [8] focus primarily on inferring geometrical structure in the form of support relations. They demonstrate that segmenting the scene simultaneously with inferring scene topology improves segmentation quality.

Another approach within the scene analysis context that is more similar to ours is the work by Nüchter *et al.* [9]. Their method segments a scene into planes and form discrete pairwise angle features over the segments. Using pre-compiled knowledge of typical angle and co-planarity constraints between plane classes, the system labels each plane according to e.g. floor, ceiling or doorway. Their algorithm achieves this by finding a global labeling that satisfies the local inter-planar constraints.

Farid & Sammut [10] use a similar model for supervised classification of compounds of planes. To achieve this they use a classification scheme based on *inductive linear programming*. Given a set of object groups that are to be classified, a set of *Prolog* clauses are learned for each object such that at least

one returns true upon being shown an object example but none returns true when shown a negative.

In robotics, several papers have dealt with the problem of finding furniture-sized objects from 3D data without supervision. Common to all such methods is that they look for recurring objects. Shin *et al.* [11] use the relation of gradually discovered shape parts in addition to features to gain more information about potential objects. The authors propose a variant of the branch-and-bound joint compatibility test to find multiple object instances. In [12] the authors find repetitive objects in precise indoor LIDAR data. Using a segmentation of point clouds into locally planar patches, the authors group combinations of patches into spatially consistent objects. They use shape descriptors of the patches together with geometric consistency within the objects. To limit the possible number of necessary combinations, several pruning steps based on patch size and individual patch is similarity is required.

The idea to model perception of 3D objects through their decomposition into primitive parts was introduced by Biederman [13]. Adjacency graphs over planes have been used for 3D roof detection from aerial LIDAR data, see e.g. [14]. In [15], Schnabel *et al.* present a representation of adjacency graphs over primitives that is similar to ours. The authors demonstrate a system that allows a user to look for a structure by specifying a query graph that can then be found within large scale environments. Our model differs in how we define discrete pairwise relations and have full connectivity locally. This enables us to search the graph for repeated structure and achieve greater consistency between matched structures. In addition, we extend their approach by learning a graph to search for from a set of example point clouds.

Our work differs from unsupervised object detection approaches like [12] in that, instead of looking for repeating structures, we look for functional parts by finding the *most frequently* repeating structures globally. We also consider more of the environment, including building structure. This is enabled by frequent subgraph mining techniques, which, to the best of our knowledge, is applied here for the first time to extract patterns in 3D point cloud data. A trade-off when using these techniques is that we have to derive precise discrete attributes.

3 METHOD

A popular approach to model semantic properties of a space has been to study graphs constructed over over-segmented scenes [7, 8]. Our approach is to similarly construct an adjacency graph over the scene but to instead identify topological structures within that graph. However, to do so, we need a graph that for one type of 3D structure consistently returns the same segmentation and graph structure. This means that over-segmentation is not an option. Instead we need to make the assumption that the surfaces that we study are unambiguous. Therefore, similar to [9, 10], [15], we make the assumption that interesting parts can be represented by geometric primitives such as planes or cylinders. This makes sense at a larger scale where much of the environment is made up

of constellations of such shapes. It further enables us to define clear pairwise relations through the relative angles and the primitive types provide us with node labels. The algorithm works with discrete properties, an inherent trait of this kind of graph mining.

We assume that we have an algorithm for segmenting a point cloud into planes, cylinders and spheres. First, some general graph concepts are introduced.

3.1 Preliminaries

A *labeled graph* is defined as a tuple $G = (V, E, \alpha)$ of nodes V and edges $E \subseteq V \times V$ together with a function $\alpha : V \cup E \to \mathbb{L}$ that maps nodes and edges to discrete labels. The *order* of a graph is $|V|$, the number of nodes. Two graphs $G_1 = (V_1, E_1, \alpha_1)$ and $G_2 = (V_2, E_2, \alpha_2)$ are said to be *isomorphic* if there exists a bijective function $f : V_1 \to V_2$ such that

- $\alpha_1(v_1) = \alpha_2(f(v_1)), \forall v_1 \in V_1$
- $\forall e_1 = (v_1, v_1') \in E_1 \; \exists e_2 = (f(v_1), f(v_1')) \in E_2$
  s.t. $\alpha_1(e_1) = \alpha_2(e_2)$ and conversely,
- $\forall e_2 = (v_2, v_2') \in E_2 \; \exists e_1 = \left(f^{-1}(v_2), f^{-1}(v_2')\right) \in E_1$
  s.t. $\alpha_2(e_2) = \alpha_1(e_1)$.

This simply means that there is a mapping f that associates every node in G_1 with a node in G_2 in such a way that all the labels and edges are maintained. A graph G is called a *subgraph* of $\hat{G} = (\hat{V}, \hat{E}, \alpha)$ if there exists some subset $(V \subseteq \hat{V}, E \subseteq \hat{E}, \alpha)$ isomorphic to G.

A collection of graphs $D = \{G_1, \ldots, G_n\}$ is said to form a *graph dataset*. Further we define $D_G = \{G_i \in D; \text{G subgraph of } G_i\}$. The *support* of G in D is then the number of times G appears as a subgraph in D, namely $|D_G|$.

3.2 Graph Construction

In our graph, the nodes $v \in V$ correspond to primitives. Each pair of primitives in a scene are connected through an edge, with one exception discussed later. Edges $e = (v_1, v_2) \in E$ describe the spatial relation between two primitives, as described by the *distance* and *angle* labels, $\alpha(e) = (l_d, l_a)$. The distance label l_d can assume two values, *close* and *distant*. A *close* edge connects two primitives (v_1, v_2) if any two points of the surfaces are closer than 0.01m (0.25m when looking at large structure data), otherwise the edge is labeled *distant*.

To assign each edge an angle label l_a, we first define the meaning of an angle γ between two primitives. Generally, the idea is to define it as the angle between the rotational symmetry axes n_1 and n_2 of the two primitives, i.e. $\gamma = \arccos(|n_1 \cdot n_2|)$. Of course, in the case of the sphere, that is ambiguous and any pair involving one is defined to have angle zero. Planes, however, have a notion of direction since they are rotationally symmetric around the surface normal. If the normals n_1 and n_2 are taken to be unit length and on the visible sides of the planes, the angle between two *distant* planes is $\gamma = \arccos(n_1 \cdot n_2)$.

Another exception is *close* planes, where we define the angle based on the angle of intersection. An inwards edge (e.g. wall facing the floor) will have angle 90° whereas an outwards edge (e.g. corner of a building) will have angle 270°.

The angles in our data are mostly parallel or orthogonal, with few exceptions. This justifies a discretization of the angles. To find the angle label l_a of an edge, we discretize the angle of its connecting primitives around multiples of 90°. In order not to include shapes not conforming to this model, all primitive pairs with relative angle further away than $\sim 11°$ from this are discarded in the following analysis. Additionally, we introduce an extra label for distant co-planar planes, enabling us to represent e.g. walls interrupted by cabinets or doors.

3.3 Subgraph extraction

Given a collection of point clouds from different scenes, a graph of primitives is extracted for each scene. The graphs together form a graph dataset D. We want to study which substructures are the most frequent for different substructure complexities. Within our framework, this translates to finding the subgraphs of order n with the highest support in the graph dataset. We use the *gSpan* algorithm for this purpose. The algorithm expands each graph to a unique *Depth First Code* (DFC). It then does a depth-first search of the graphs to effectively find the most frequent subgraphs in a graph dataset. The algorithm has found extensive use in e.g. molecule mining for finding common molecule substructures [16]. We use a *gSpan* implementation by Kudo *et al.* [17]. To make sure that the internal relations between the primitives in all scenes corresponding to a certain subgraph are consistent, we require that the frequent graphs be complete. We therefore limit the *gSpan* algorithm to look only for subgraphs $G = (V, E)$ with $|E| = \frac{n(n-1)}{2}$. Further, for the subgraphs to represent something connected in the scene, most of the primitives need to belong to the same surface structure. A number of *close* edges greater than or equal to a constant n_{adj} is therefore also required. If nothing else is stated, at least half of the edges have to be *close*. One could require that the subgraph be connected by *close* edges but as we will see this was not necessary on our data. It can easily be added if needed.

3.4 Study of Isomorphic Graphs

We are investigating to what extent we can use pure surface topology to characterize the typical structures. Within one group of isomorphic subgraphs we can therefore have nodes corresponding to primitives of different sizes. However, in the following analysis, it will prove useful to be able to remove instances with large size deviations. To do this, we construct from each instance of a subgraph in a scene, a vector u_i where each element represents a measure of the size of one primitive. For example, in Section 5.2 we use the areas of the extracted planes. Thus, a subgraph found in m scene instances will have vectors $U = \{u_1, \ldots, u_m\}$ describing the different sizes. To separate the subgraphs based on size also, one could imagine doing clustering over this vector space. For this paper, we are only

interested in removing matched graph instances with sizes dissimilar from the provided examples. Based on the nearest neighbor distance between an instance and the example set size vectors, we remove far-away matches.

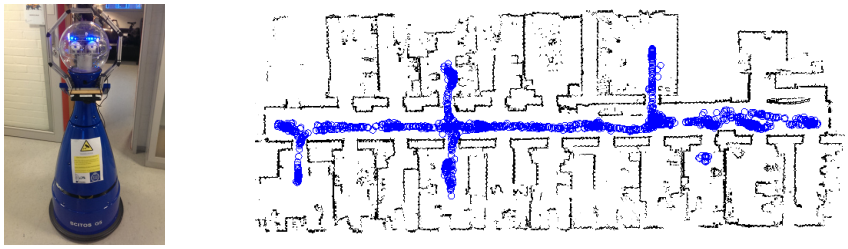

Fig. 1: The Scitos G5 robot, during the capturing of the data set with the snapshot positions overlaid on the floor map. The camera is looking down at 43°.

4 EXPERIMENTAL SETUP

4.1 Primitive Extraction

One major challenge with using geometric primitives is that they can be costly to extract, especially in noisy sensor data. We use a RANSAC algorithm [18] since it is known to be robust to noise in the form of outliers. The basic algorithm in the context of shape recognition works by sampling a number of points, called a *minimal set*, from which a shape hypothesis can be formed. Several hypotheses are formed by sampling minimal sets of points repeatedly. The algorithm returns the shape hypothesis that is supported by the most inlier points. An *inlier* to a shape is defined as a point whose minimal distance to the shape surface is less than some threshold λ.

However, using this algorithm to extract several shapes from one point cloud can be unnecessarily costly since the minimal sets are sampled across the entire cloud, with no prior on size or locality. We therefore use a RANSAC modification which was introduced by Schnabel *et al.* [1]. Their algorithm makes use of the observation that points in a smaller neighborhood are more likely to belong to the same surface. The result of the method is a segmentation of a point cloud into primitives, with some points remaining.

4.2 Environment & Setup

We conduct our experiments using a Scitos G5 platform with an Asus Xtion depth-sensing camera mounted in front. We did two experiments, one in which the robot drives around autonomously and captures individual RGB-D images and another in which many point clouds were combined into a single 3D map.

In the first experiment we want to avoid having many images from nearly the same camera pose so we only save images from distinct view points. A new image is captured only when the robot has turned more than the field of view or traveled more than a certain distance. Granted, this does not mean that the same structure is not observed several times during a run but the intention is to make the distribution of the scans roughly uniform across the floor. The robot performed two runs over approximately three hours each, together making up a dataset of 1846 frames, see Figure 1. Along the way, it went in to three offices and a kitchen. In this first experiment we extract planes, cylinders and spheres.

To construct the 3D map for the second experiment, we drove the robot around the office and collected local 3D sweeps using a camera pan-tilt unit (PTU) mounted on the head. These were then assembled into a big map using the transform from the PTU and stock laser localization [19]. To form a graph and search this very large point cloud was computationally infeasible. We therefore build graphs and search inside a window of a fixed size. The window is then slid to a partially overlapping position and the search repeated until the entire map is covered. Since planes are dominating at this more coarse level of resolution, we limit ourselves to plane primitives. Also, as the robot always knows the position of the floor, so it is given its own label with edge definitions equivalent to other planes.

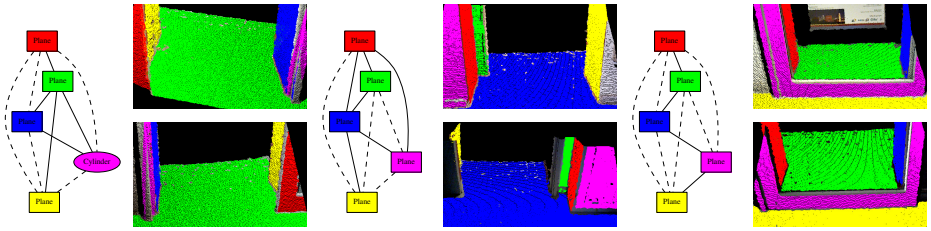

Fig. 2: Graphs representing the doors and the classified scenes. The color of primitives in the scenes and the corresponding nodes of the graph match. The *far* edges are dotted, while the *close* edges are solid. From left to right: Open from outside, open from inside and closed from outside. The bottom middle frame has similar geometry to an open door from the inside and is falsely classified as such.

5 RESULTS

5.1 Experiment 1

In the first experiment we search a set of individual RGB-D snapshots. We show the robot doors in three different configurations: open from outside, open from inside, and closed from outside. Our robot was positioned to take twelve snapshots of different doors in each configuration. The framework then extracts the most

frequent subgraph from each door configuration, defining a "template" graph for each type. The doors are re-identified by finding instances of these graphs within the set of 1846 scene graphs. To make the subgraphs more discriminative, we choose a graph order of $n = 5$ with $n_{\text{adj}} = 4$.

The results can be seen in Figure 2. Apart from the floor, all graphs have the door frames and parts of the door in common. Among all frames, the robot successfully found five instances of open doors from the outside, seven open doors from the inside and 33 closed doors from the outside. For each of the open door categories, one false positive was found, containing some primitives not part of the doors. Among the 33 extracted closed doors, all were found to be correctly classified. The most common subgraph for each configuration together with some instances of the graph are presented in Figure 2. This is an encouraging result and confirms that the representation can be descriptive enough to find instances of one particular structure.

As might be expected, the sphere primitives found little use in this data set as they were detected in only a few places. The cylinders were detected more often, most consistently in specific structures like the trash cans. While simple primitives other than planes were not as common in this particular environment, we note that they allow the method to represent some more of these special cases.

Many doors were not detected, probably partly because the entire structure was not in the field of view of one of our snapshots. This is a limitation of our method when applied to single snapshots as we did here. If we look at the example of the closed doors, the method is sensitive to the degree of occlusion between the door and the floor since that can determine if the corresponding edge will be labeled as *close* or *distant*. The following section explores one approach to overcome these problems.

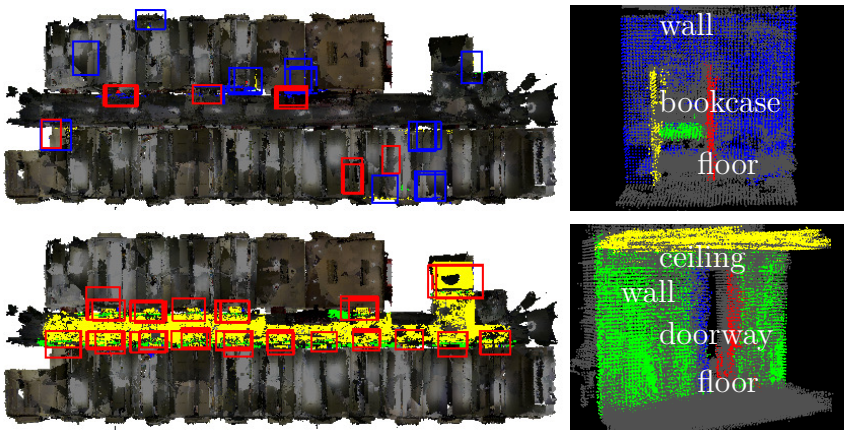

Fig. 3: Top row: Result of a bookshelves query. The blue rectangles are graph template matches removed by size constraints. Bottom row: Here we show the most frequent structures found using a $3m \times 3m$ window size; mostly doorways.

5.2 Experiment 2

In the second experiment we search a very large point cloud map. A user selects windows by clicking points in a display of the point cloud map. In this experiment we select four areas with bookcases. The average width of these windows determine the scale at which we look for similar structures. From the examples, the most frequent subgraph of order 5 is extracted, giving us a template graph representing the bookcase. The top row of Figure 3 summarizes the result. On the left are all the locations of returns from the query. On the right is one instance. In this case the query returns many false candidates. We can then use other information, size in this case, to further remove candidates. From the example graph instances, we create a vector set U over plane areas, constructed as in Section 3.4. By removing found instances with size vector further than a certain nearest-neighbor distance from U we can remove all instances not representing a bookcase. These are shown as blue rectangles while the red ones are all true bookcases. We also looked at the most frequent structures over the sliding windows. The result for a $3m \times 3m$ sliding is presented in the bottom row of Figure 3. The most frequent graph represents most of the doors found in the environment. As should be expected, when the user queries for doors by marking four doorways, the result is mostly the same. A four times larger window finds structures matching walls and ceiling of a room.

The results show that the problems observed in the door identification are fixed by considering a full view of the scenes instead of partial views; this time 14 out of 18 doors in the office floor are detected without any false positives.

6 CONCLUSION AND FUTURE WORK

We proposed a method to construct adjacency graphs over geometric primitives from point cloud data. We demonstrated the ability to re-identify structures. The approach was also verified to be able to search for structure in a large scale 3D map. Our results indicated that we can combine topology with other cues, here size but one could extrapolate to for example color, for reliable classification. An advantage of the graph mining in 3D versus the case with traditional images is that we can define clear, descriptive relations between local features.

We see this as a first step towards building a general 3D point cloud query framework. This would extend the search criteria to include such things as the separation of the point cloud by planes which could lead to concepts such as "enclosed by". One could for example then look for structures "inside" the rooms that we found here.

7 Acknowledgments

The work presented in this papers has been funded by the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement No 600623 ("STRANDS"), VR project "XPLOIT", and the Swedish Foundation for Strategic Research (SSF) through its Centre for Autonomous Systems.

References

1. R. Schnabel, R. Wahl, and R. Klein, "Efficient ransac for point-cloud shape detection," *Computer Graphics Forum*, vol. 26, pp. 214–226, June 2007.
2. X. Yan and J. Han, "gspan: Graph-based substructure pattern mining," in *Proceedings of the 2002 IEEE International Conference on Data Mining*, ICDM '02, (Washington, DC, USA), pp. 721–729, IEEE Computer Society, 2002.
3. S. Nowozin and et al., "Weighted substructure mining for image analysis," in *Conference on Computer Vision and Pattern Recognition*, pp. 1–8, IEEE, 2007.
4. M. C. Jiang and F. Coenen, "Graph-based image classification by weighting scheme," in *Applications and Innovations in Intelligent Systems XVI*, pp. 63–76, Springer, 2009.
5. H. Cheng, X. Yan, J. Han, and C.-W. Hsu, "Discriminative frequent pattern analysis for effective classification," in *Data Engineering, 2007. ICDE 2007. IEEE 23rd International Conference on*, pp. 716–725, IEEE, 2007.
6. A. Aydemir, P. Jensfelt, and J. Folkesson, "What can we learn from 38,000 rooms? reasoning about unexplored space in indoor environments," in *Intelligent Robots and Systems, IEEE/RSJ International Conference on*, pp. 4675–4682, 2012.
7. A. Anand and et al., "Contextually guided semantic labeling and search for three-dimensional point clouds," *The International Journal of Robotics Research*, vol. 32, no. 1, pp. 19–34, 2013.
8. N. Silberman and et al., "Indoor segmentation and support inference from rgbd images," in *Computer Vision–ECCV 2012*, pp. 746–760, Springer, 2012.
9. A. Nüchter and J. Hertzberg, "Towards semantic maps for mobile robots," *Robotics and Autonomous Systems*, vol. 56, no. 11, pp. 915–926, 2008.
10. R. Farid and C. Sammut, "A relational approach to plane-based object categorization," in *Robotics Science and Systems Workshop on RGB-D Cameras*, 2012.
11. J. Shin, R. Triebel, and R. Siegwart, "Unsupervised discovery of repetitive objects," in *IEEE International Conference on Robotics and Automation, ICRA 2010, Anchorage, Alaska, USA, 3-7 May 2010*, pp. 5041–5046, IEEE, 2010.
12. O. Mattausch, D. Panozzo, C. Mura, O. Sorkine-Hornung, and R. Pajarola, "Object detection and classification from large-scale cluttered indoor scans," in *Computer Graphics Forum*, vol. 33, pp. 11–21, Wiley Online Library, 2014.
13. I. Biederman, "Recognition-by-components: a theory of human image understanding.," *Psychological review*, vol. 94, no. 2, p. 115, 1987.
14. V. Verma, R. Kumar, and S. Hsu, "3d building detection and modeling from aerial lidar data," in *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, vol. 2, pp. 2213–2220, IEEE, 2006.
15. R. Schnabel, R. Wessel, R. Wahl, and R. Klein, "Shape recognition in 3d point-clouds," in *Proc. Conf. in Central Europe on Computer Graphics, Visualization and Computer Vision*, vol. 2, Citeseer, 2008.
16. K. Jahn and S. Kramer, "Optimizing gspan for molecular datasets," in *Proceedings of the Third International Workshop on Mining Graphs, Trees and Sequences (MGTS-2005)*, pp. 77–89, 2005.
17. T. Kudo and et al., "An application of boosting to graph classification," in *Advances in neural information processing systems*, pp. 729–736, 2004.
18. 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*, vol. 24, no. 6, pp. 381–395, 1981.
19. D. Fox, W. Burgard, F. Dellaert, and S. Thrun, "Monte carlo localization: Efficient position estimation for mobile robots," *AAAI/IAAI*, vol. 1999, pp. 343–349, 1999.