Region Identification Methods for Efficient and Automated Motion Planning

Jory Denny jdenny@cse.tamu.edu
Anshul Agrawal anshula@cse.tamu.edu
Evan Greco egreco@cse.tamu.edu
Lydia Tapia ltapia@cse.tamu.edu
Nancy M. Amato amato@cse.tamu.edu

Technical Report TR10-002
Parasol Lab.
Department of Computer Science and Engineering
Texas A&M University

September 30, 2010

Abstract

Realistic environments for motion planning problems typically consist of multiple areas, e.g., in a house there are rooms, hallways, and doorways. Decomposition of the environment into smaller areas, or regions, has been shown to impact the quality of planning for adaptive methods. Also other complex problems, such as planning for groups of agents with environmental preferences, can be aided with automated region identification. However, despite the advantages of regions, automatic region identification is not always easy, inexpensive, or consistent. In this paper we explore and compare three methods of automatic region identification and quantify the effects of the resulting regions on planning. The three strategies used for region identification are the statistical methods: k-means clustering, PG-means clustering, and Hierarchical clustering. First, we explore the differences in the regions produced by these three methods in a variety of rigid body and articulated linkage environments. Then, we quantitatively compare the usefulness of the region methods in automated motion planning through the application of the regions within the Unsupervised Adaptive Strategy for Motion Planning (UAS). Our results indicate both regions that seem intuitive and non-intuitive to the problems solve motion planning problems with increased efficiency and automation. That is, we found that planning generally benefitted from using regions, and the benefit was not tied to the naturalness of the region.

1 Introduction

Environments in motion planning are rarely simple. Most realistic and challenging environments consist of multiple subproblems, such as a collection of cluttered, free, and narrow spaces. For example, planning in a house would require planning for rooms, hallways, and doorways. The automatic identification of these regions has been shown to help facilitate motion planning [4, 15, 20, 26] by helping map appropriate planners to resulting regions. Also, complex planning problems, such as planning the natural behaviors for a group of agents [3], could be directed through the use of regions that identify natural barriers in the environment. These regions can bind agents to a particular region or can create complex interactions between the agents. Also, regions can be used to parallelize the framework by providing a straight-forward decomposition of the problem.

Despite the fact that regions have been used in many motion planning solutions including cell-decomposition [18, 25], water-shedding [4], workspace decomposition [15], and feature-senstive motion planning [20, 26], the identification is not always easy, inexpensive, or consistent. Factors such as the relationship between the planning space, C-space, and the problem workspace and the environment characteristics can impact the quality of the resulting regions.

In this paper, we explore three methods for automatic region identification: k-means clustering [13], PG-means clustering [7], and Hierarchical clustering [12]. K-means clustering groups points (in the case of motion planning, robot configurations) based upon proximity and tries to create equal sized clusters. PG-means clustering groups based upon a dimensionality reduction of the space and computes clusters in the reduced space. Another method, Hierarchical clustering, groups each point with its nearest node and creates a tree like representation of the data showing the regions at each level of merging. These methods do not always create regions that might be intuitive to the human eye. However, we demonstrate that these regions, though not intuitive to the workspace representation, improve the efficiency of automated motion planning solution.

The three region identification methods are evaluated by two criteria. First, we compare the regions created by each method by contrasting the resulting regions with the natural regions in the environment. Second, we measure the impact of the regions on motion planning. This is quantitatively measured through the application of the Unsupervised Adaptive Strategy (UAS) for motion planning [26]. In this domain, homogeneous regions reflect more efficient planning. Results are shown on a variety of environments with complex regions.

2 Preliminaries

A robot is a movable object whose position and orientation can be described by n parameters, or degrees of freedom (DOFs), each corresponding to an object component (e.g., object positions, object orientations, link angles, link displacements). Hence, a robot's placement, or configuration, can be uniquely described by a point $(x_1, x_2, ..., x_n)$ in an n dimensional space (x_i being the ith DOF). This space, consisting of all possible robot configurations (feasible or not) is called *configuration space* (C-space) [19]. The subset of all feasible configurations is the *free C-space* (C-free), while the union of the unfeasible configurations is the *blocked C-space* (C-obstacles). Thus, the MP problem becomes that of finding a continuous trajectory for a point in C-free connecting the start and the goal configurations. In general, it is intractable to compute explicit C-obstacle boundaries, but we can often determine whether a configuration is feasible or not quite efficiently, e.g., by performing a collision detection (CD) test in the *workspace*, the robot's natural space.

Randomized motion planners explore C-space and produce a data structure containing feasible configurations and some information about the connectivity of C-free. One such planner, the *Probabilistic Roadmap Method* (PRM) [14], builds a roadmap (graph) of the free C-space. The first phase in this process, *node generation*, is where collision-free configurations are sampled and added as nodes to the roadmap. In the second phase, *node connection*, neighboring nodes are selected by a *distance metric* as potential candidates for connection. Then, simple *local planners* attempt connections between the selected nodes, represented as roadmap edges.

Although the initial PRMs were successful in solving many problems previously thought unsolvable, they were not successful in problems where the solution path must pass through a narrow passage in the C-space. In order to address this deficit, many PRM variants have been introduced. For example, OBPRM [1] generates samples near C-obstacle surfaces by first generating a random sample and searching along a random direction until the sample's collision state changes. Another variant, Gaussian PRM [6], generates pairs of samples that are a distance d apart, where d has a Gaussian distribution, until one sample is collision-free and the other is not, and retains the free sample as a roadmap node. Many other heuristics have been proposed [5, 16, 23]. The performance of these methods is dependent on the problem instance, e.g., OBPRM performs better in cluttered areas and uniform random sampling is quite efficient in open areas.

3 Related Work

3.1 Roadmap-based Region Identification Methods

In this section, we survey methods that have been proposed to identify regions in an environment. Each of these methods extracts information from a small PRM roadmap in order to gain knowledge about the characteristics of the environment. As the partitions are made, they can be evaluated for homogeneity and split if homogeneity requirements are not met.

Gap Partitioning. Gap Partitioning [22] uses the knowledge of the free sampled nodes in order to place partitions that will serve as the boundaries of regions. Specifically, it targets large gaps between the free nodes. This method is deterministic but bases its partitions on the obstacles, either in C-space or in the workspace. By placing the partitions in areas of large gaps it would separate obstacle-space from free-space. Since the partitions are based upon obstacles, gap partitioning favors environments with clear separation between obstacle-space and free-space.

Random Partitioning. Random Partitioning [20, 22] is a elementary subdividing strategy by which a random point is used to split the data based upon one random DOF. By using random partitions, it can be expected that many partitions may be required to identify homogeneous regions. Also, random partitioning does not guarantee an optimal number of regions. Since it is not deterministic and does not use the features of the environment, it also does not guarantee the quality of partitions.

Entropy Based Partitioning. Entropy Based Partitioning [22] uses the information theory concept of entropy to partition an environment. By measuring the entropy, the amount of disorder, that a partition introduces into a set of features, it can pick partitions that maximize the loss of entropy. The end result will be two or more partitions of lowest entropy based on a given feature, e.g., the placement of collision and collision-free nodes, from the environment.

Clustering. Clustering Techniques have been widely studied. In clustering, the goal is to find clusters with similar values from a given data set. By finding these clusters, they should accurately represent homogeneous regions in the data. K-means clustering [13] is a popular clustering technique, which finds centroids and assigns the data points to their closest centroid. Then the algorithm averages the nodes to find a new centroid and repeats the assigning process. When no change has occurred in the reassignment phase, then the final assignment of points to centroids represents the clusters. In the case of the basic algorithm, the user must specify the number of clusters for k-means, although there have been methods proposed to determine the optimal number of clusters based upon the variance of the data.

3.2 Planning Methods with Regions

This section provides an introduction to many of the adaptive planning methods that use region identification to solve the motion planning problem.

Feature Sensitive Motion Planning Framework. This approach was introduced as a method that used machine learning to characterize and partition a planning problem [20]. In this approach, the planning space is recursively subdivided until a machine learning method is able to classify a subdivision as appropriate for a planner from a given library. This topology mapping may be defined in either workspace or C-space. The strength of this method lies in its ability to identify a model of a problem's topology which allows identification of homogeneous regions

appropriate for certain planners. However, other than recursive subdivision calls, it is not able to adapt planner applications over time. Another drawback of this approach is that it requires a mapping of samplers to regions, typically generated by machine learning techniques that require an "expert" to label hundreds of examples of training data. Such a mapping must be repeated as new planners are developed.

Unsupervised Adaptive Strategy. In the Unsupervised Adaptive Strategy (UAS) [26] for solving motion planning problems, a learning approach, based upon unsupervised learning, is used to adapt motion planning to specific regions with as little intervention from the user as possible. This method is an extension of Feature Sensitive Motion Planning. It identifies regions, then automatically applies planners to the region through the application of Hybrid PRM. This method showed an improvement in quality of roadmaps [26], with little overhead, over Hybrid PRM [11] and Feature Sensitive Motion Planning [20].

First, UAS learns about the problem's environment by creating a small training roadmap R_{train}. The vertices, V, and edges, E, of this roadmap define a set of features F that characterize the problem. Next, F is used within an automated method to define a set of regions R that capture the approximate topology of the original environment. Finally, the motion planning method Hybrid PRMis applied in each of the regions probabilistically based on a measure of difficulty of each regions, e.g., a region's approximate visibility. Therefore, the Hybrid applied in each region can learn which planners are best to apply in each region.

RESAMPL [24]. This method uses local region information (e.g., entropy of neighboring samples) to make decisions about how and where to sample, which samples to connect together, and to find paths through the environment. This use of spatial information about the planning space enables RESAMPL to increase sampling in regions identified as "narrow" and decrease sampling in regions identified as "free".

Workspace Adaptation Methods. Many methods have been proposed to explore the impact of adaptation in response to the features of the planning workspace. A recent adaptation of the Hybrid PRM method [15] uses workspace information, extracted from a cell decomposition, to define locations where samplers should be applied. Another workspace-based approach applies the watershed method (previously applied in image processing) to identify narrow passageways in the workspace [4]. After such features are identified, the planning can be adapted based on the characterization of the region. While these approaches rely heavily on spatial characteristics in order to decide where to apply a planner, they do not consider the change in the topology that is discovered as a space is explored. Their performance also degrades in more complex problems (different C-space types) where difficult (e.g., narrow) regions of C-space can no longer be identified from difficult regions of the workspace (such as with articulated linkages and other constrained robots).

3.3 Metrics for Planner Performance

Adaptive motion planning strategies require metrics to evaluate the performance of planning approaches. While many common metrics have been used for evaluation (e.g., solution of a query, time, CD counts), it is still often difficult to get a clear measure of planner performance. Methods that require a discretization of space have been proposed [8]. In the problems studied here, we applied a group of metrics that have been explored on non-discrete spaces in order to classify the contribution of samples produced by planners [21].

If two configurations, q_1 and q_2, can be connected by a sequence of valid motions, they are considered *visible* to each other. For example, the straight-line local planner will decide that q is visible to q' if the straight segment from q to q' is composed of only valid configurations. In [21], a *visibility ratio* is assigned to each configuration to approximate the visibility of a single configuration to its neighbors. This ratio is defined in terms of the number of successful connections over the number of connection attempts involving that configuration. In this paper, we approximate the *visibility ratio* by using a small roadmap. When this roadmap is constructed, we can inexpensively tally up the number of attempted and successful connections and get a good approximation of the topology of the space. In this paper, we refer to this feature of a node as its visibility.

4 Region Identification

In this section, we describe the algorithms for the three statistical region identification methods, k-means, PG-Means, and Hierarchical clustering. We also explain how determining the optimal number of clusters is provided by PG-Means and can be added to fully automate k-means and hierarchical clustering.

4.1 k-means Clustering

The basic algorithm of k-means takes an integer value k and features of the environment as the data set, such as x position, visibility, or rotation, as input. Then it starts by randomly selecting k centroids within the bounding range of each of the features. After the initial centroids are created, each node is assigned to its closest centroid.

After this assignment of nodes has occurred, the averages of the clusters are determined. These values become the new centroids. The reassigning phase continues until either n iterations have passed or no reassignment occurs in an iteration. The algorithm gives k clusters. Each cluster is then converted to a region, and k regions are created from this method. The advantage of this method is speed, but the user must provide the number of clusters desired.

The optimal number of regions can be automatically determined through analysis of the Elbow Criterion [10,17] of k-means. In the elbow criterion, a plot of the average variance of the data vs. the varying k value is found. Then the second derivative of this plot is found. The optimal number of clusters is the k which holds the absolute maximum value on this second derivative plot.

4.2 Projected Gaussian-means Clustering (PG-means)

In the initial step, PG-means [7] generates a model of k clusters. The method used to learn the new model is Expectation-Maximization (EM); however, any other method, e.g., K-Means, can be used to obtain a new model. EM is a technique where the likelihood of generating an expected model of clusters is maximized based on logarithmic statistics. Once the best model of k clusters is obtained, the model and data set are projected to a 1-dimensional arbitrary axis to expedite the process of testing. After the projection phrase, the model is tested for its fitness with the data set. The Kolmogorov-Smirnov Test (KS-Test) is used to compare the model to the data set since it theoretically determines whether two given data sets differ significantly. At this time, the model may be accepted; if it is rejected, a new model is created using EM based on the previous learned model, and the PG-means process is repeated until a model is accepted. The resulting information that the PG-means clustering provides is the number, location, and orientation of the final cluster model. Again these clusters are collected as regions to be used by our motion planning application.

4.3 Hierarchical Clustering

Hierarchical clustering [12] recursively groups the given set of nodes based on certain proximity distance metrics. Binary agglomerative Hierarchical clustering performs a bottom-up recursive clustering of the two closest nodes at each step.

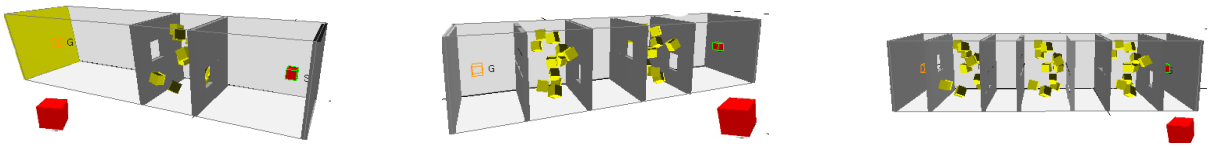
There are three major conceptual steps involved in the clustering process. First, a *distance array* is created as a list of all distances between each of the nodes in the input data set based on a certain distance metric in the configuration space. Second, a *Linkage Matrix* which is a connection analysis tool used to represent the Hierarchical clustering tree is created from the distance array. Finally, we apply the elbow criterion [10, 17] on the variance vs. number of clusters curve (as described above with k-means). This criterion returns the optimal number of clusters k for the given environment, and we slice the Hierarchical clustering tree at a height where there are only k leaf nodes. All the sampled nodes that lie in the sub-tree of these leaf-nodes fall under the same cluster. The final clusters that are collected are the regions for our application.

5 Experiments

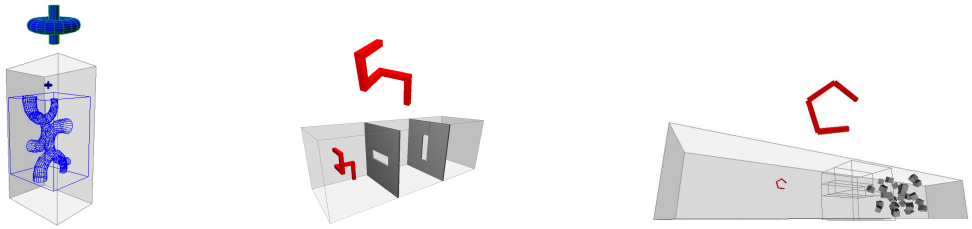
In this section, the three statistical region identification methods are evaluated. First, we compare the regions created by the different methods with each other. Second, we compare motion planning performance using the regions identified by the methods in the Unsupervised Adaptive Strategy for motion planning.

5.1 Experimental Setup

For these experiments, we use the Unsupervised Adaptive Strategy, which uses a learning technique to select different PRM-based motion planners to incrementally build a roadmap in its regions. We implemented all planners using the C++ motion planning library developed by the Parasol Lab at Texas A&M University. RAPID [9] is used for collision detection computations. In these PRM-based planners, after nodes are sampled, connections are attempted between each node and its k-nearest neighbors according to some distance metric; here we use $k = 20$, C-space Euclidean distance, and a rotate at s planner [2] with $s = 0.5$. The Unsupervised Adaptive Strategy for Motion Planning is implemented as described in [26]. We used basic PRM, OBPRM and Gaussian PRM in order to create the initial training roadmap. After this, we use region identification to incrementally build a complete roadmap. For a given region, we use Hybrid PRM sampling in order to focus the building of the roadmap in these regions. Hybrid adapts itself to its environment, so for high visibility regions it learns to use uniform random sampling in order to augment the already highly visible regions, and it learns to use either Gaussian PRM or OBPRM for low visibility regions.



(a) Walls One (b) Walls Two (c) Walls Three



(d) Maze (e) Hook (f) Regions

Figure 1: Rigid body (a, b, c, d, e) and articulated linkage (f) environments studied. The robot must travel from one end to the opposite end in all environments. Robot enlarged and shown on the side.

We explore different rigid body and articulated linkage environments of varying topology, see Figure 1. In these environments, the witness queries have been designed to force the robot to traverse the entire problem space. This ensures that they capture the problem complexity. Each environment has regions of high and low visibility. It will be difficult for each region identifier to identify regions and successfully allow the UAS to solve the query.

- **Walls (One, Two, Three)**: The Walls environments were designed to be of increasing complexity. In all three environments the robot is a cube that must travel from one end of the environment to the other. First, Walls One consists of a cluttered region bounded by walls with two free space regions on the ends. Next, Walls Two adds two more free and cluttered regions separated by walls. Finally, Walls Three adds two more free and cluttered regions.

- **Maze**: This environment consists of two open areas connected by a maze of narrow tunnels. The tunnels vary from smaller than the robot (impassable) to just slightly wider than the robot. The robot is in the shape of a spinning top, and it is often very difficult for a planner to find feasible motions in the maze. This environment has three regions visually intuitive regions in the workspace, two free and one constrained.

- **Hook**: The Hook environment has two walls with slits between them. The robot, a hook, must rotate and translate between the two slits to move from one side of the environment to the other. This environment can be separated into three or five regions that are visually intuitively, two free and one constrained or three free and two constrained.

- **Regions**: The Regions environment consists of a long narrow tunnel followed by a cluttered region with free regions on either side. The robot, a four link articulated linkage, must elongate itself to pass through the tunnel and then change to a more compact form to navigate the cluttered region. This environment has three or four visually identifiable regions in the workspace, two free and one constrained or two free and two constrained.

To have a variety of sampling techniques, we have chosen sampling techniques known to work well in a variety of environments. The samplers selected are: uniform random sampling [14], Gaussian sampling [6], and OBPRM [1]. For Gaussian sampling, we use two values of the Gaussian distance d: the robot's minimum diameter, r_d, and $2r_d$.

We use the following metrics to evaluate performance of the methods: (i) ability to solve a user-defined witness query, (ii) number of collision detection calls as a measure of time, (iii) number of nodes generated to solve the query, and (iv) the ratio of CD calls to nodes as a measure of efficient planning.

The region identification methods were set up with their own parameters. k-means, Hierarchical, and PG-means all used a set of features extracted from the PRM roadmap, the training roadmap. There are many roadmap-based features that have been previously explored for region identification [20]. In the results shown here, clustering is based on a set of features that are independent of robot type: visibility, x-position, y-position, and z-position. These features are extracted from the nodes or the set of configurations in the training roadmap. Then, we define each region as the bounding box of each node set. Due to the use of positional values as features,

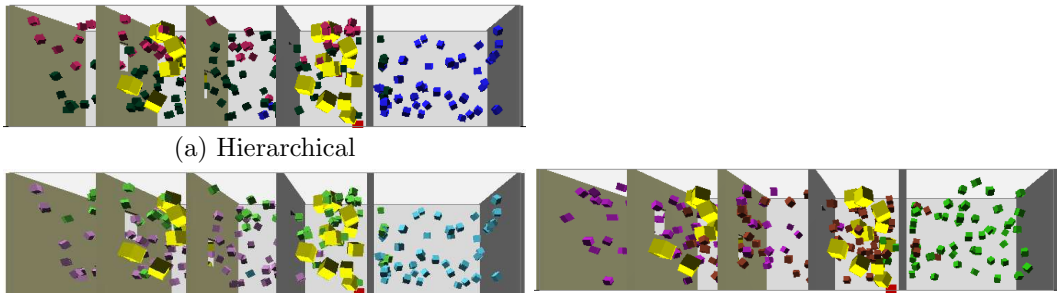
clusters may result in overlapping regions. The k value used in k-means clustering was determined by the elbow criterion [10, 17] discussed in 4.1, which is why the environments use a value of $k = 3$. Also, PG-means uses the following conformational parameters: $\alpha = 0.001$ and number of projections equal to 12. The number of regions can range depending on the accepted projection model. Hierarchical clustering was set up with the distance matrix as Euclidean distance, the linkage matrix as Wards distance [12]. Again the elbow criterion was used to determine the stopping criterion as discussed in 4.3.

In order to get a good approximation of the problem but not contribute much to the cost of problem solution, we constructed a small training roadmap R_{train}. For this study all six environments were constructed with training roadmaps of 200 nodes. In previous work, this number was found to capture problem spaces well while keeping the cost of training low [26]. The training set was created from a mix of samplers including OBPRM, Basic PRM and Gauss PRM samplers (in the proportion 50% OBPRM, 40% PRM, and 20% Gauss PRM). The k closest connections for each node were based on a k value of $k = 5$ and were done with the Rotate-at-S local planner with $s = 0.5$. On each training set we ran the region identifier and collected our data from these regions (Table 1). We collected average total nodes, average total CD calls, the ratio between nodes and CD calls, and the number of regions created by the method. We compare each region identifier on each training set with the same random seed being used. These data sets are averaged over 10 random seeds. If the method failed to solve the query on a certain seed, the number was averaged on successful runs.

5.2 Region Study

In this section, we explore the differences in the regions created by the three methods. Through examples of typical regions and statistics about the resulting regions, we explore the quality of the regions for each problem. Because of the similarity of the three Walls environments, we show one typical example of them with Walls Two. We eliminate the Maze environment from this discussion because all the methods create very similar regions both in quantity and location. These methods performed region identification on a training roadmap and a set of features as described above.

The first example we study is of Walls Two. The typical outputs of the regions are shown in Figure 2. Each of the regions are represented by the robot shown at half-size and are colored by region membership. In general, none of these methods are able to identify regions that might be considered visually intuitive according to the problem's workspace. Both hierarchical clustering and k-means clustering group the nodes in very similar fashions with regions are of similar size. In these two examples we can see two regions on the left side of the image (top and bottom) then a single region encompassing the free region on the right. We notice that two of three regions do not encompass strictly homogeneous regions. However, PG-means clustering is slightly different. In this example it creates three regions which overlap by a large amount. The first region extends from the first free space on the left to the middle free space. The second region encompasses both cluttered areas. The final region covers from the rightmost cluttered and free space region. These regions are not visually intuitive. But, as we will show in 5.3, these regions help the solution to the motion planning problem.



(a) Hierarchical

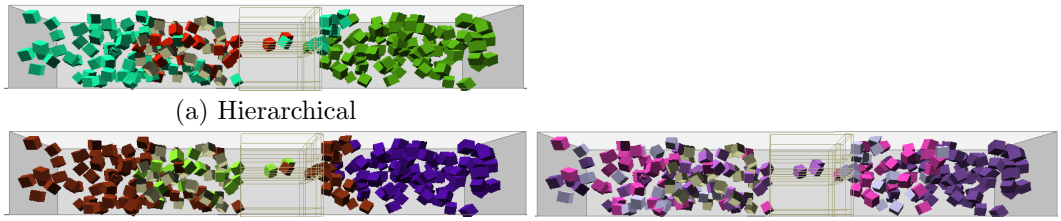
(b) k-means

(c) PG-means

Figure 2: Typical regions created from the three methods for Walls Two environment. Each region is shown by colored nodes, which represent the nodes of the training set that belong to the region. Robot is shown as a small box. All methods produced three regions.

The next environment we will examine is Regions. These are shown in Figure 3. In this environment, we can again see that the regions created from hierarchical clustering and k-means clustering are very similar. We see that one region is created which encompasses the free space, cluttered space, and the narrow passage. The next region is enclosed within the cluttered region and narrow passage of the space. Lastly, there is a final region which covers the free space on the right. While these regions may intuitively seem like they are spread across the

environment, there are specific regions that are focused on the difficult parts of the problem: the narrow passage and the cluttered area. PG-means clustering however shows regions which are highly overlapping. It created four regions, but these regions do not seem visually intuitive to the problem's workspace. The first two regions overlap and enclose the free space on the left and the narrow passage. There is one small region within the clutter and narrow passage. Finally the last region is the free space on the left. These regions have a high degree of overlap but we will see in 5.3 that this will not hurt the solution strategy.

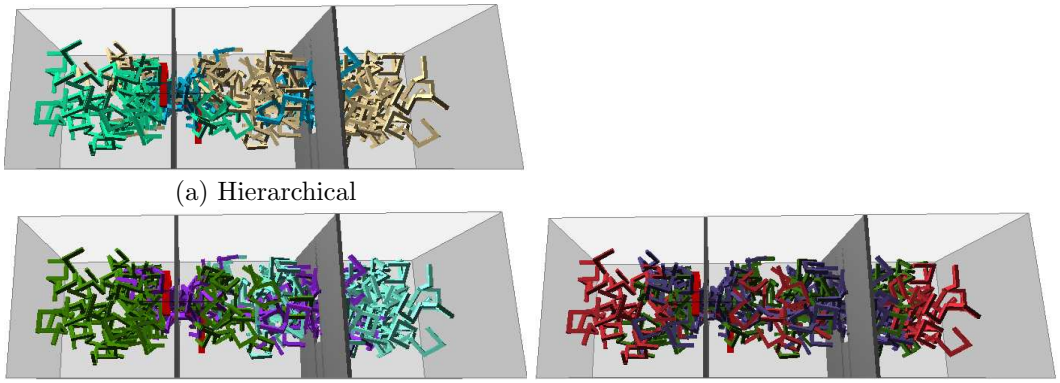


(a) Hierarchical

(b) k-means (c) PG-means

Figure 3: Typical regions for Regions environment. Each region is shown by colored nodes, which represent the nodes of the training set which belong to the region. Robot is shown as a box. Both Hierarchical and k-means produced 3 regions, whereas PG-means produced 4.

In the final environment, Hook, we see similar effects, except this environment contains narrow passages which are difficult to detect. The typical regions produced are shown in Figure 4. Again, hierarchical and k-means create similar regions. The inner most region encompasses both narrow passages and the other two regions split the rest of the space in half. PG-means has a series of regions encompassing one another. So the smallest region encompasses the two narrow passages. The next two regions encompass a slightly larger portion of the workspace. The overlap of the regions in all the methods will actually cause a problem for the Hybrid PRM samplers for these regions. The regions consist of a large amount of free space.



(a) Hierarchical

(b) k-means (c) PG-means

Figure 4: Typical regions for Hook environment. Each region is shown by colored nodes, which represent the nodes of the training set which belong to the region. All methods produced three regions.

5.3 Performance in Motion Planning Solution

In this section we are evaluating if the region identification methods demonstrate differences in motion planning problem solution. We do this by taking the three automated methods and use them to identify the regions, R, in UAS. Then, we compare these results against each other and the standard, non-region methods, of Hybrid PRM and PRM. Both articulated linkages and rigid bodies are studied in a variety of environments. In the previous section, it is clear that the three methods produce vastly different regions. However, it is unclear if these regions, which intuitively do not seem to relate to the workspace, will improve or hinder automated motion planning.

The results are shown in Table 1. Recall that in these experiments, the measure of difficulty that automatically assigns node distribution for UAS was based upon the visibility ratio of each region from R_{train}. For example, the visibility ratio is directly proportional to the probability of sampling in a region. This results in regions of low visibility receiving a high number of samples, and high visibility often receiving a small number of nodes.

The three Walls environments were created to show results on a set of problems of increasing complexity. As shown in Table 1, Walls Three takes more nodes and CD calls to solve for all methods than does Walls One. Also, in these environments we can see that regions produced by all methods are highly beneficial by reducing the number of CD calls. For example, in Walls One they are reduced to about 10% of that of Hybrid PRM and PRM. In Walls Two, we see the three UAS methods requiring around 15% of the CD calls that Hybrid PRM and PRM. In Walls Three it is about 33% of the other two methods. Notice that these reductions are independent of the method used to identify the regions. As discussed above, in Section 5.2, the methods usually found 3 regions for all three environments.

For the Maze environment we can see again that with all three region identification methods UAS makes fewer CD calls than Hybrid PRM. PRM was unable to solve this environment with a limit of 5000 nodes. In the Maze, small differences in the regions created can make big jumps in efficiency of solving the planning problem. For example, hierarchical clustering has a 20% decrease in the ratio of nodes to CD calls when compared to k-means clustering, and a 12% decrease compared to PG-means. However, despite the variation in the performance different regions, they provided a 60% decrease on the average number of of CD calls that Hybrid PRM had.

In the Regions environment we see a difference in the average CD calls taken for UAS with the region identification methods. We see that hierarchical clustering can solve the query in fewer CD calls than the other methods, and k-means can solve the query faster than PG-means. This difference provides insight that different region identifiers might perform better in certain cases and no one of these methods is best in our application. In this environment we see the obvious benefit of regions, because Hybrid PRM is unable to solve the query. In these cases we see Hybrid PRM converging on PRM because of the large open spaces. We can also see that PRM alone can only solve the query a few times.

In the Hook environment, where we see a slightly bigger difference in regions created (see Section 5.2), very badly formed regions seem to have an effect on the ability of the problem to be solved by UAS (as shown by queries solved being around 50%). The regions formed do not appropriately represent the narrow passages of the environment but instead encompass a large portion of the free space. When this happens the instances of Hybrid PRM for each region converge on PRM. This makes sampling very efficient but the problem exists of it taking many nodes to find a solution, if it finds one at all. In the cases where this problem solved it still solved with less CD calls and lower Node to CD Call ratio than Hybrid PRM.

From these environments we can see a few things. First, regions help the solution of the motion planning problem no matter if they reflect the homogeneity of the workspace (e.g., encompassing only free space or narrow passage). There are few cases when improperly placed regions can hurt problem solution. Second, we see a significant efficiency increase in sampling individual nodes with the use of regions as demonstrated by lower CD calls and lower Ratio of CD calls to Nodes.

6 Conclusion

This work demonstrates how different methods for automatic region identification compare within a motion planning framework. First, we compared the regions that each identifier makes to show how the methods differ. This allowed us to see differences between the regions and the nonhomogeneity of the regions as produced by the various methods. Second, we compared the performance of the methods in a motion planning framework, UAS. In UAS, regions of vary homogeneity still improved the solution over Hybrid PRMand PRM. Thus, regions will produce more efficient motion planning solutions. We see that in some cases regions can decrease the ability to solve the queries.

There are many motion planning applications where automatic region identification may prove useful. In the future we plan to extend our application of these methods to these areas including: complex planning problems, such as planning the natural behaviors for a group of agents and decomposition for parallel solution. Along with this exploration we want to take an in-depth look at the benefit of homogeneous vs. nonhomogeneous region identification methods in UAS.

Achknowledgements

Tapia supported in part by the National Science Foundation under Grant # 0937060 to the Computing Research Association for the CIFellows Project. We would like to acknowledge Thasia Madison, who contributed to the early part of this work while participating in the CRA's CRA-W and CDC's DREU program during summer 2009.

References

[1] N. M. Amato, O. B. Bayazit, L. K. Dale, C. V. Jones, and D. Vallejo. OBPRM: An obstacle-based PRM for 3D workspaces. In *Robotics: The Algorithmic Perspective*, pages 155–168, Natick, MA, 1998. A.K. Peters. Proc. Third Workshop on Algorithmic Foundations of Robotics (WAFR), Houston, TX, 1998.

[2] N. M. Amato, O. B. Bayazit, L. K. Dale, C. V. Jones, and D. Vallejo. Choosing good distance metrics and local planners for probabilistic roadmap methods. *IEEE Trans. Robot. Automat.*, 16(4):442–447, August 2000.

[3] O. B. Bayazit, J.-M. Lien, and N. M. Amato. Roadmap-based flocking for complex environments. In *Proc. Pacific Graphics*, pages 104–113, Oct 2002.

[4] J. Berg and M. Overmars. Using workspace information as a guid to non-uniform sampling in probabilistic roadmap planners. *Int. J. Robot. Res.*, 24(12):1055–1072, 2005.

[5] R. Bohlin and L. E. Kavraki. Path planning using Lazy PRM. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 521–528, 2000.

[6] V. Boor, M. H. Overmars, and A. F. van der Stappen. The Gaussian sampling strategy for probabilistic roadmap planners. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, volume 2, pages 1018–1023, May 1999.

[7] Y. Feng and G. Hamerly. Pg-means: Learning the number of clusters in data. In *Neural Information Processing Systems*, pages 393–400, 2006.

[8] R. Geraerts and M. H. Overmars. Reachablility-based analysis for probabilistic roadmap planners. *Robotics and Autonomous Systems*, 55:824–836, 2007.

[9] S. Gottschalk, M. C. Lin, and D. Manocha. OBB-tree: A hierarchical structure for rapid interference detection. *Comput. Graph.*, 30:171–180, 1996. Proc. SIGGRAPH '96.

[10] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, 2001.

[11] D. Hsu, G. Sánchez-Ante, and Z. Sun. Hybrid PRM sampling with a cost-sensitive adaptive strategy. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 3885–3891, 2005.

[12] A. K. Jain, M. N. Murty, and P.J. Flynn. Data clustering: A review. *ACM Computing Surveys*, 31:264–323, 1999.

[13] T. Kanungo, D. M. Mount, N. Netanyahu, C. Piatko, R. Silverman, and A. Y. Wu. An efficient k-means clustering algorithm: Analysis and implementation. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 24(2002):881–892, 2002.

[14] L. E. Kavraki, P. Švestka, J. C. Latombe, and M. H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Trans. Robot. Automat.*, 12(4):566–580, August 1996.

[15] Hanna Kurniawati and David Hsu. Workspace-based connectivity oracle - an adaptive sampling strategy for prm planning. In *Algorithmic Foundation of Robotics VII*, pages 35–51. Springer, Berlin/Heidelberg, 2008. book contains the proceedings of the International Workshop on the Algorithmic Foundations of Robotics (WAFR), New York City, 2006.

[16] S. M. LaValle and J. J. Kuffner. Randomized kinodynamic planning. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 473–479, 1999.

[17] L. Lieu and N. Saito. Automated discrimination of shapes in high dimensions. In *Proc. Society of Photo-Optical Instrumentation Engineers (SPIE)*, volume 6701, pages 67011V–1–67011V–12, 2007.

[18] T. Lozano-Pérez. Automatic planning of manipulator transfer movements. *IEEE Trans. Syst. Man Cybern.*, SMC-11(10):681–698, 1981.

[19] T. Lozano-Pérez and M. A. Wesley. An algorithm for planning collision-free paths among polyhedral obstacles. *Communications of the ACM*, 22(10):560–570, October 1979.

[20] Marco Morales, Lydia Tapia, Roger Pearce, Samuel Rodriguez, and Nancy M. Amato. A machine learning approach for feature-sensitive motion planning. In *Algorithmic Foundations of Robotics VI*, pages 361–376. Springer, Berlin/Heidelberg, 2005. book contains the proceedings of the International Workshop on the Algorithmic Foundations of Robotics (WAFR), Utrecht/Zeist, The Netherlands, 2004.

[21] Marco A. Morales A., Roger Pearce, and Nancy M. Amato. Metrics for analyzing the evolution of C-Space models. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 1268–1273, May 2006.

[22] Marco A. Morales A., Lydia Tapia, Roger Pearce, Samuel Rodriguez, and Nancy M. Amato. C-space subdivision and integration in feature-sensitive motion planning. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 3114–3119, April 2005.

[23] C. L. Nielsen and L. E. Kavraki. A two level fuzzy PRM for manipulation planning. Technical Report TR2000-365, Computer Science, Rice University, Houston, TX, 2000.

[24] Samuel Rodriguez, Shawna Thomas, Roger Pearce, and Nancy M. Amato. (RESAMPL): A region-sensitive adaptive motion planner. In *Algorithmic Foundation of Robotics VII*, pages 285–300. Springer, Berlin/Heidelberg, 2008. book contains the proceedings of the International Workshop on the Algorithmic Foundations of Robotics (WAFR), New York City, 2006.

[25] J. T. Schwartz and M. Sharir. On the "piano movers" problem I: The case of a two-dimensional rigid polygonal body moving amidst polygonal barriers. *Commun. Pure Appl. Math.*, 36:345–398, 1983.

[26] Lydia Tapia, Shawna Thomas, Bryan Boyd, and Nancy M. Amato. An unsupervised adaptive strategy for constructing probabilistic roadmaps. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 4037–4044, May 2009.

Walls One Environment					
Method	**Hierarch.**	**k-means**	**PG-means**	**Hybrid**	**PRM**
Avg. Nodes	**348**	374	370	1141	760
Avg. CD Calls	**75820**	81939	96333	1038714	1288808
Ratio of CD calls to Nodes	**217.9**	**219.1**	260.3	910.4	1695.8
Avg. # of Regions	3.2	3.0	2.6	N/A	N/A
Queries Solved (%)	100	100	100	100	100
Walls Two Environment					
Method	**Hierarch.**	**k-means**	**PG-means**	**Hybrid**	**PRM**
Avg. Nodes	**700**	810	835	1333	1025
Avg. CD Calls	**131892**	152487	151382	878136	974519
Ratio of CD calls to Nodes	**188.4**	**188.3**	**181.3**	658.8	950.8
Avg. # of Regions	3.2	3.0	3.3	N/A	N?A
Queries Solved (%)	100	100	100	100	100
Walls Three Environment					
Method	**Hierarch.**	**k-means**	**PG-means**	**Hybrid**	**PRM**
Avg. Nodes	1421	**1375**	1528	1789	1640
Avg. CD Calls	**160028**	**157986**	175120	575599	470146
Ratio of CD calls to Nodes	**112.6**	**114.9**	**114.6**	321.7	286.7
Avg. # of Regions	3.2	3.0	2.7	N/A	N/A
Queries Solved (%)	100	100	100	100	100
Maze Environment					
Method	**Hierarch.**	**k-means**	**PG-means**	**Hybrid**	**PRM**
Avg. Nodes	3402	**3057**	3230	3189	N/A
Avg. CD Calls	**1015115**	1139366	1092975	2572757	N/A
Ratio of CD calls to Nodes	**298.4**	372.7	338.4	806.8	N/A
Avg. # of Regions	3.2	3.0	1.7	N/A	N/A
Queries Solved (%)	60	60	70	100	0
Regions Environment					
Method	**Hierarch.**	**k-means**	**PG-means**	**Hybrid**	**PRM**
Avg. Nodes	**856**	988	1165	N/A	3640
Avg. CD Calls	**166469**	194597	214227	N/A	5938581
Ratio of CD calls to Nodes	194.5	197.0	**183.9**	N/A	1631.5
Avg. # of Regions	3.0	3.0	3.8	N/A	N/A
Queries Solved (%)	100	100	100	0	6
Hook Environment					
Method	**Hierarch.**	**k-means**	**PG-means**	**Hybrid**	**PRM**
Avg. Nodes	3090	4081	4030	**1789**	N/A
Avg. CD Calls	137654	**105229**	**99933**	268319	N/A
Ratio of CD calls to Nodes	44.5	**25.8**	**24.8**	149.9	N/A
Avg. # of Regions	3.3	3.0	1.2	N/A	N/A
Queries Solved (%)	60	40	50	100	0

Table 1: Performance results for all environments. Lowest values are shown in bold. Values are average over the sucessful runs. Problems that were unsolvable by a certain planner are indicated by 0% queries solved. Methods were given a maximum of 5000 nodes to solve the query.