Hierarchical Map Building and Planning based on Graph Partitioning *

Zoran Zivkovic and Bram Bakker and Ben Kröse
Intelligent Systems Laboratory Amsterdam
University of Amsterdam
Kruislaan 403, 1098 SJ Amsterdam, The Netherlands
{*zivkovic,bram,krose*}*@science.uva.nl*

***Abstract*— Mobile robot localization and navigation requires a map - the robot's internal representation of the environment. A common problem is that path planning becomes very inefficient for large maps. In this paper we address the problem of segmenting a base-level map in order to construct a higher-level representation of the space which can be used for more efficient planning. We represent the base-level map as a graph for both geometric and appearance based space representations. Then we use a graph partitioning method to cluster nodes of the base-level map and in this way construct a high-level map, which is also a graph. We apply a hierarchical path planning method for stochastic tasks based on Markov Decision Processes (MDPs) and investigate the effect of choosing different numbers of clusters.**

***Index Terms*— mobile robots, hierarchical map building, topological map, path planning**

I. Introduction

Mobile robot localization and navigation requires an internal representation of the environment. Several researchers (e.g. [12], [18]) have proposed a hierarchy of maps to represent large environments at different resolutions, or levels of abstraction, simultaneously.

Typically two levels of abstraction are used: a base-level map and a higher-level "topological" map. The higher-level, abstract map may be used to represent larger areas of a building, for instance as a graph connecting rooms and corridors, without representing the exact spatial relationship of individual locations within rooms and corridors. Such a high-level map is used to construct abstract plans to navigate from one room to another, without having to worry about exact spatial details within the rooms. The base-level map can be used for precise navigation from one room to the next and to target locations within an individual room, without having to worry about other rooms. In this paper we address the problem of segmenting the base-level map so as to arrive at the higher level map.

*The work described in this paper was conducted within the EU FP6-002020 COGNIRON ("The Cognitive Companion") project.

There are various ways of segmenting base-level maps. In human augmented mapping, a human supervisor indicates which places are to serve as nodes in the graph [1]. Geometrical methods such as generalized Voronoi graphs [6] are used to segment the geometric space representations. It is also possible to use sensory data directly for the creation of a higher level map. In [16], [10], [21] a set of images of the robot's environment is grouped based on the presence of a number of automatically extracted landmarks.

In this paper we describe an alternative algorithm for segmenting the base-level map similar to [21]. We start from a graph representation of the base-level map. The graph consists of nodes that represent admissible space locations and the links between nodes represent the admissible transitions between the locations. In [21] images are used to construct the graph of an appearance based representation. We extend [21] by showing how some common space representations used in robotics are readily represented in this way.

Segmentation of the base level map then becomes a problem of graph partitioning. We use the "normalized graph cut" criterion [9] and its efficient approximate solution [9], [15] to perform the segmentation. We apply a hierarchical path planning algorithm for stochastic tasks based on Markov Decision Processes (MDPs) [2] and investigate the effect of choosing different numbers of clusters.

The paper is organized as follows. In Section 2 we describe simple methods for generating base-level graphs for different space representations. Section 3 describes the graph-theoretic segmentation method for extracting a higher level conceptual map. The algorithm from [2] for hierarchical path planning is presented briefly in Section 3. Our experimental results in Section 4 show the results of segmenting different real space representations into varying numbers of segments, and demonstrate the efficiency of the hierarchical path planning.

II. Base-level graph

Throughout the paper we will denote by $(\mathcal{S}^0, W^0)$ the graph that describes the base-level map. Here $\mathcal{S}^0$ is the set of

graph nodes. In our examples the graph nodes will correspond to the admissible areas of space. Let q^0 be the number of nodes. The $q^0 \times q^0$ matrix W^0 is called the "similarity matrix". For each pair of nodes i,$j \epsilon [1, ..., q^0]$ the value of the element W^0_{ij} from the symmetric matrix W^0 defines the similarity of the nodes. In our examples $W^0_{ij} = 1$ typically denotes that the transition is possible between the nodes and $W^0_{ij} = 0$ that it is not possible.

There are two general approaches used to construct the base-level map in robotics: geometric and appearance based. In both cases it is straightforward to represent the base-level map as a graph.

A. Geometric base-level graph

In the geometric approach the map is represented as a 2D geometric model of the workspace of the robot, indicating admissible and non-admissible areas [18]. The admissible and non-admissible areas are often represented as a grid, yielding an "occupancy grid" representation. A simple graph representation for a grid-based model is commonly obtained in the following way [18]. Grid cells that correspond to the admissible areas are considered as the nodes of the base-level graph. Transitions between neighboring admissible cells are the edges that connect the nodes of the graph. This is also a natural and straightforward representation to be used for path planning, where the graph nodes are regarded as the possible states of the robot and the transitions as possible actions given a current state (see Section IV).

In Figure 1a we present an occupancy grid representation of an office environment obtained from real data [18]. The black pixels of the presented image correspond to non-occupied cells. Each black pixel becomes a node of the graph $(\mathcal{S}^0, W^0)$. This gives $q^0 = 11773$ graph nodes (states) for this map. For each pair of nodes i,j that correspond to neighboring cells the value of the element W^0_{ij} from W^0 is set to the inverse of the distance between the cells. For example, a black pixel form Figure 1a surrounded by only black pixels represents a graph node connected to its 8 neighboring nodes. For the cell neighbors in the horizontal and vertical direction $W^0_{ij} = 1$, and in the diagonal direction $W^0_{ij} = 1/\sqrt{(2)}$. All other elements of W^0 are set to zero. Note that there are many ways to extend this representation. For example the occupancy grid representations are often not binary but contain some measure of uncertainty about the cells and this information could be included in the graph. However, this is beyond the scope of this paper.

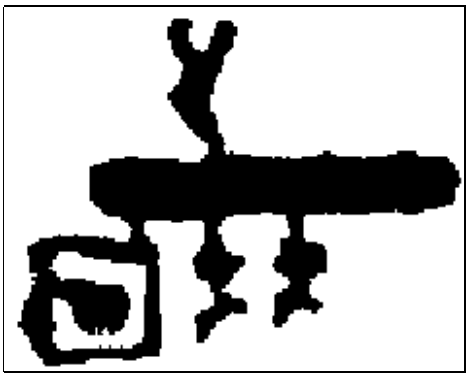
B. Appearance based base-level graph

A second approach used to construct base-level maps is appearance-based representations. These are representations where the environment is not modelled geometrically, but as an 'appearance map' that consists of a collection of sensor readings obtained at various poses [14], [8], [11]. Here the various poses can be regarded as the nodes of the low-level graph and the admissible transitions as the graph edges. Again this is a natural representation for path planning using an appearance based representation.

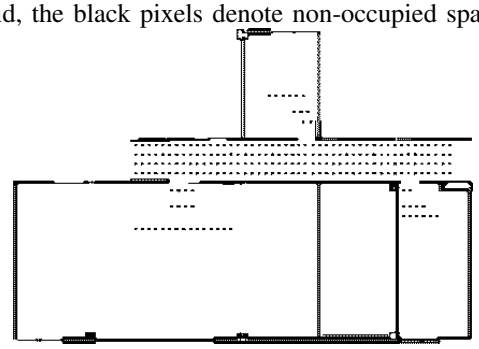
As an example we take the map from [21] where the appearance based representation consists of a set of omnidirectional images taken at different locations. As an example in Figure 1b we illustrate a set of $q^0 = 234$ locations where the images were taken in an office environment. Note that the ground-truth locations are used only for presenting the results, they are not used by our methods. As the result from q^0 images we get a graph $(\mathcal{S}^0, W^0)$ with q^0 nodes. For each pair of nodes i,j the value of the element W^0_{ij} defines the similarity of the nodes to be equal to 1 if and only if it is possible to perform 3D reconstruction of the local space from the two images corresponding to the nodes. Otherwise there is no link between the nodes and $W^0_{ij} = 0$. For localization and navigation the robot can use the same 3D reconstruction algorithm as the one used to define the edges of the graph. If there is a non-zero edge in the graph this also means that if the robot is at one of the connected nodes (corresponding to one image), it can determine the relative location of the other node (corresponding to the other image). If there are no obstacles in between, the robot can directly navigate from one node to the other (as, e.g., in [17]). If there are obstacles, one could rely, for example, on an additional reactive algorithm for obstacle avoidance using range sensors. As in [21], we use the standard 3D reconstruction 8-point algorithm [7] and the Scale Invariant Feature Transform (SIFT) features [13] as the automatically detected landmarks in the images.

In this way, a graph is constructed from the data set. One such graph is presented in Figure 3. This graph contains, in a natural way, information about how the space in an indoor environment is separated by walls and other barriers. Images from a convex space, for example a room, will have many connections between them, and just a few connections to images from another convex space, for example a corridor, that is connected with the room via a narrow passage, for example a door.

As with geometric representations, there are various ways to define the similarity metric for W^0. The simple metric we use is directly related to the robot navigation task. Additional information can be associated with the edges of the graph, e.g., Euclidean distance between the nodes if metric positions of the images are known or reconstructed, some measure of quality and robustness of the 3D reconstruction given a pair of images, etc.



a) a geometric space representation - a binary occupancy grid, the black pixels denote non-occupied space.



b) an illustration of an appearance based map - an omnidirectional image was taken at each location denoted by a dot.

Fig. 1. Example base level 2D maps. Bird's eye view of two office environments. For the appearance based map we show the ground floor plan to give some idea about the environment layout.

III. Constructing Higher Level Topological Maps Using Graph Cuts

The central idea behind our method to construct the higher level map is to cut the graph $(\mathcal{S}^0, W^0)$ representing the lower level map (described above) into q^1 separate subgraphs $\{(\mathcal{S}^0_1, W^0_1)..., (\mathcal{S}^0_{q^1}, W^0_{q^1})\}$. Each of the clusters becomes a higher level node (state) in the higher level graph $(\mathcal{S}^1, W^1)$.

A. Normalized graph cut

We will start by introducing some graph-theoretic terms. The *degree* of the i-th node of a graph $(\mathcal{S}, W)$ is defined as the sum of all the edges that start from that node: $d_i = \sum_j W_{ij}$. For nodes $\mathcal{S}_j$ (where $\mathcal{S}_j$ is a subset of $\mathcal{S}$), *volume* is defined as $vol(\mathcal{S}_j) = \sum_i d_i$. $vol(\mathcal{S}_j)$ describes the "strength" of the interconnections within the subset $\mathcal{S}_j$. A subgraph $(\mathcal{S}_j, W_j)$ can be "cut out" from the graph $(\mathcal{S}, W)$ by cutting a number of edges. The sum of the values of the edges that are cut is called a graph cut:

$$cut(\mathcal{S}_j, \mathcal{S}\backslash\mathcal{S}_j) = \sum_{i\epsilon\mathcal{S}_j, j\epsilon\mathcal{S}\backslash\mathcal{S}_j} W_{ij} \quad (1)$$

where $\mathcal{S}\backslash\mathcal{S}_j$ denotes the set of all nodes except the ones from $\mathcal{S}_j$. One may cut the base level graph into q_1 clusters by minimizing the number of cut edges:

$$\sum_j^{q^1} cut(\mathcal{S}_j, \mathcal{S}\backslash\mathcal{S}_j). \quad (2)$$

This would mean that the graph is cut at the weakly connected places, which in our case would usually correspond to natural segmentation at doors between the rooms or other narrow passages. However, such segmentation criteria often leads to undesirable results. For example, if there is an isolated node connected to the rest of the graph by only one link, then (2) will be in favor of cutting only this link. To avoid such artifacts we use a *normalized* version:

$$\sum_j^{q^1} \frac{cut(\mathcal{S}_j, \mathcal{S}\backslash\mathcal{S}_j)}{vol(\mathcal{S}_j)}. \quad (3)$$

Minimizing this criterion means cutting a minimal number of connections between the subsets but also choosing larger subsets with strong connections within the subsets. This criterion naturally groups together convex areas, like a room, and makes cuts between areas that are weakly connected.

B. Approximate solution - spectral clustering

For completeness of the text we briefly sketch a well-behaved spectral clustering algorithm from [15] that leads to a good approximate solution of our criteria:

1) Define D to be a diagonal matrix of node degrees $D_{ii} = d_i$ and construct the normalized similarity matrix $L = D^{-1/2}W^0D^{-1/2}$.
2) Find $x_1, ..., x_{q^1}$ the q^1 largest eigenvectors of L and form the matrix $X = [x_1, ..., x_{q^1}] \in \mathcal{R}^{q^0 \times q^1}$.
3) Renormalize rows of X to have unit length $X_{ij} \leftarrow X_{ij}/(\sum_j X^2_{ij})^{1/2}$.
4) Treat each row of X as a point in $\mathcal{R}^{q^1}$ and cluster using for example the k-means algorithm.
5) The i-th node from $\mathcal{S}^0$ is assigned to cluster j if and only if the row i of the matrix X was assigned to the cluster j.

Instead of the k-means step in [19] a more principled but more complex approach is used, following [20] where a good initial start for the k-means clustering is proposed. We tested the mentioned algorithms, and in practice, for our type of problems, they lead to similar solutions.

C. Quality of the segmentation

The quality of the segmentation can be assessed by comparing to the ideal case where the clusters are highly separated. In such a case the normalized similarity matrix L becomes block diagonal. If there are q_1 clusters present then there will be q_1 blocks and the first q_1 eigen values will be equal to 1. In the non-ideal case eigenvalues will be less then one and decreasing. If q_1 is the "optimal" number of clusters then we expect a sudden drop for the eigenvalue $q_1 + 1$. This is a common heuristic [15] to decide the quality of a segmentation for a given q_1 but in practice highly unreliable and lacks theoretical justification [20].

Another, more robust heuristic [20] is based on the following observation. In the ideal case the eigenvectors will correspond to the matrix blocks having all other values corresponding to other matrix blocks equal to zero. This means that for every row of the matrix X (described above) that contains the eigenvectors there will be at most one non zero entry. So we could define for example the following quality measure:

$$Quality = \sum_{1}^{q^0} \sum_{1}^{q^1} \frac{X_{ij}^2}{max_j X_{ij}^2} - 1. \quad (4)$$

Note that in the ideal case the eigenvectors correspond to a repeated eigenvalue and it could be that the eigenvectors we get as a solution span the same subspace but are rotated by an unknown rotation R. Therefore in order to use the quality measure (4) we need to find this rotation and transform X to its canonical form $\hat{X} \leftarrow RX$. See [20] for details.

IV. Hierarchical path planning for stochastic tasks

For path planning, robot maps are commonly formalized as Markov Decision Processes (MDPs), such that the planning task becomes a dynamic programming problem [4], [5]. This formalization is appropriate for such robot planning tasks because it is efficient, because it can take into account noise in the execution of actions and uncertain state transitions, and because the resulting policies are optimal in the sense that they lead to lowest expected cost (e.g. distance travelled). Furthermore, it allows for straightforward inclusion of cost factors other than distance travelled, such as energy consumption and obstacle avoidance.

A. MDPs

An MDP $\mathcal{M}$ is a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R} \rangle$. $\mathcal{S}$ is a finite set of states s, some of which may be terminal states. The states are the nodes of the graph $(\mathcal{S}, W)$. The $\mathcal{A}$ is a finite set of actions a, whose availability may depend on the state. $\mathcal{T} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$ defines the state transition function that describes the probability $p(s'|s, a)$ that the system will move from state s to s' after performing the action $a \in \mathcal{A}$. The actions for state s in our case are the possible transitions for this graph node defined by the graph $(\mathcal{S}, W)$. In the simplest case we set the probability of each transition to 1 or 0. $\mathcal{R} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow R$ defines the expected immediate real-valued reward $r(s, a, s')$ when action a is taken in state s and the transition to s' is made. For path planning $r(s, a, s')$ can simply be negative value of the distance travelled between s and s', defined by the values from the graph similarity matrix W.

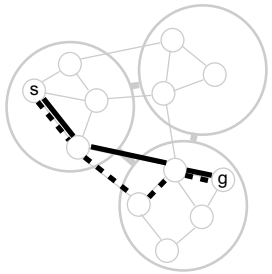


Fig. 2. Illustration of a case when the hierarchical planning method may lead to longer paths from a start state s to a target state g. The base-level graph nodes are grouped into three higher level nodes indicated by the larger circles. The solid line is the optimal path. The dashed line is the path computed by the hierarchical method, which optimally goes to the bottom high-level state, and within the bottom high-level state optimally goes to the target state.

The objective of planning in MDPs is to determine a policy $\pi : S \rightarrow A$ which maximizes the total (future, cumulative, possibly discounted) reward, or minimizes the total cost. Planning is normally done by estimating a value function, which represents expected total reward or cost obtainable from each state, through dynamic programming methods [4].

B. Hierarchical dynamic programming

The standard MDP framework can be extended by adding hierarchical structure [2]. We define two types of MDPs making up a complete hierarchical system. They are both derived from a given standard, flat MDP. The first type, $\mathcal{M}^n$, a tuple $\langle \mathcal{S}^n, \mathcal{A}^n, \mathcal{T}^n, \mathcal{R}^n \rangle$, represents the given MDP at a particular level of abstraction. n indexes the level in the hierarchy, N is the number of levels in the hierarchy (in our case $N = 2$). The base-level MDP is $\mathcal{M}^0$ corresponding to the base level graph $(\mathcal{S}^0, W^0)$. $\mathcal{M}^n$ for $n \geq 1$ is constructed from $\mathcal{M}^{n-1}$ by segmenting the states in $\mathcal{S}^{n-1}$. In our case, this is performed by segmenting the base-level graph using the method described above. The state transition function $\mathcal{T}^n$ and reward function $\mathcal{R}^n$ are estimated by averaging over the corresponding lower level state transitions and rewards (see [2] for details). The second type of MDPs making up the complete hierarchical system is defined only for $n \geq 1$ and is denoted by $\mathcal{M}^{n-1}_{s^n_k, s^n_m}$. $\mathcal{M}^{n-1}_{s^n_k, s^n_m}$ is an MDP that represents

the *lower* level $(n-1)$ task of navigating from *higher* level (n) state s_k^n to state s_m^n. Its states, state transition function, and reward function are straightforward subsets of the lower-level MDP $\mathcal{M}^{n-1}$.

This hierarchy of MDPs allows us to efficiently compute a value function/policy for the entire state space to every possible target state of the original, flat MDP $\mathcal{M}^0$. For a specific low-level target state, the higher level target states in $\mathcal{M}^n$ for all $0 < n < N$ are determined in which this low-level target state lies. At their own levels, they become terminal states. Next, the path planning task is performed from the highest level down, using dynamic programming at each level. State transitions from s_k^n to s_m^n dictated by a value function at level n are modelled by the appropriate $\mathcal{M}^{n-1}_{s_k^n, s_m^n}$ and subsequently planned, again using dynamic programming. In this way, the complete planning task to a low-level target state is solved recursively. Algorithm 1 provides pseudocode for the complete hierarchical planning method.

[18] proposes a planning method that is similar to this hierarchical dynamic programming method. However, that method was specifically designed for metric-topological maps and deterministic tasks, and it assumes that topological state transitions always have the same cost. It was also designed to do most of the planning off-line. In contrast, this method was designed for more diverse, deterministic and stochastic hierarchical maps with an arbitrary number of levels in the hierarchy, and it computes higher level state transition probabilities and costs in a more principled way. Furthermore, it was designed for online use, with plans being generated only when needed.

In contrast to standard, "flat" dynamic programming using only the base-level MDP, the hierarchical algorithm can, when paths must be planned to multiple target states, in many cases reuse value functions computed for earlier target states. A second advantage over standard flat dynamic programming is that the state spaces for individual value functions at all levels are reduced, leading to fewer operations per sweep through the state set and faster convergence. A disadvantage is that in some cases the system may converge to somewhat longer paths to target locations than standard flat dynamic programming. This situation can arise because low-level value functions which are optimal with respect to reaching the next high-level state from the current high-level state are not always optimal with respect to reaching the final target state (see Figure 2). However, the idea of using the normalized graph cut algorithm is that this problem is minimized because that algorithm segments the base-level map at narrow passages. This means that hierarchical planning cannot do much worse than flat planning, because both must pass through this narrow passage.

$s_g^0 \leftarrow$ new target state for $\mathcal{M}^0$
for all $0 < n < N$ **do**
  $s_g^n \leftarrow$ determine target state for $\mathcal{M}^n$
$V^{N-1} \leftarrow$ *Solve*($\mathcal{M}^{N-1}$, $N-1$).

Function *Solve*($\mathcal{M}$, n):
while $\delta > \Delta$ (a tiny threshold) **do**
  for all $s \in \mathcal{S}$ **do**
    $V_{new} \leftarrow \max_a \sum_{s'} p(s'|s,a)[r(s,a,s') + V(s')\ ]$
    if $|V_{new} - V(s)| > \delta$ **then**
      $\delta \leftarrow |V_{new} - V(s)|$
    $V(s) \leftarrow V_{new}$
if $n > 0$ **then**
  for all $s \in \mathcal{S}$ **do**
    if $s = s_g^n$ **then**
      if $V^{n-1}_{s_g^n, s_g^{n-1}}$ does not exist **then**
        Construct $\mathcal{M}^{n-1}_{s_g^n, s_g^{n-1}}$ from $\mathcal{M}^{n-1}$
        $V^{n-1}_{s_g^n, s_g^{n-1}} \leftarrow$ *Solve*($\mathcal{M}^{n-1}_{s_g^n, s_g^{n-1}}$, $n-1$)
    else
      $s^* \leftarrow \arg\max_a \sum_{s'} p(s'|s,a)[r(s,a,s') + V(s')]$
      if V^{n-1}_{s,s^*} does not exist **then**
        Construct $\mathcal{M}^{n-1}_{s,s^*}$ from $\mathcal{M}^{n-1}$
        $V^{n-1}_{s,s^*} \leftarrow$ *Solve*($\mathcal{M}^{n-1}_{s,s^*}$, $n-1$)
Return V

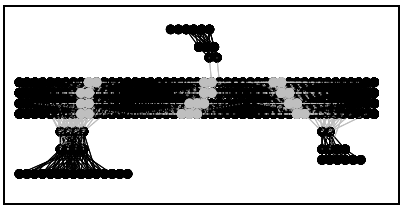
Algorithm 1: Pseudocode of hierarchical dynamic programming algorithm.

V. EXPERIMENTS

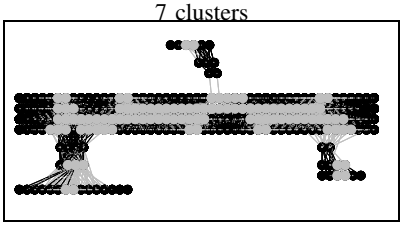
The experiments described here were designed to investigate the validity of the method to extract the higher level map from the base-level map and to investigate the effect of choosing different number of clusters on the path planning.

A. Appearance based map

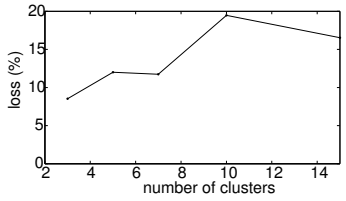
We applied the graph cut clustering on the appearance based map from Section 2 with 2, 5, 7, 10, and 15 clusters. The graph and the clustering for 7 and 15 clusters are shown in Figure 3. For the various numbers of clusters we use the hierarchical value iteration method described above to compute policies to 1000 randomly selected states. The results are compared to flat value iteration performed on the flat, base-level MDP $\mathcal{M}^0$. In Figure 3 we summarize the results. The number of value updates until convergence and computation time is significantly lower for the hierarchical method than for the flat method (as expected; see the explanation above). However, the average maximum likelihood path length for the hierarchical method is 10% to 20% higher than the flat method's (optimal) value. This average loss with respect to



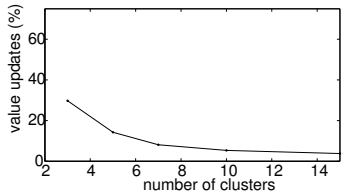
7 clusters



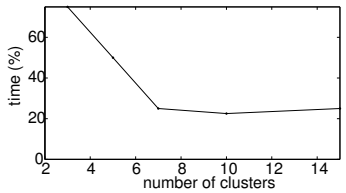
15 clusters



average loss in % with respect to the optimal path



average number of value function updates in % with respect to the flat planning



average computation time in % with respect to the flat planning

Fig. 3. Examples of segmentation and results of planning for various number of clusters for the appearance based graph. In the segmented graphs the cut links are shown in gray. Some nodes along the cuts are also shown in gray for better visibility.

the optimal path depends both on the number of clusters and the structure of the base-level graph. The standard quality measure (4) indicates 7 as the "best" number of clusters which leads to a reasonable compromise.

B. Occupancy grid map

The graph cut clustering method is applied to the occupancy grid from Section 2 with 5, 8, 10, 15, 19, 20, 30, and 50 clusters. The results comparing the hierarchical planning method to the flat planning method are summarized in Figure 4. The gains in computation time for this larger map are huge while the average path length gets only slightly worse than the optimal one computed using the flat planning. The graph cut algorithm leads to good segmentations and hierarchically computed paths close to optimal for the whole range of number of clusters. A number of clusters of 20 or 30 seems to be a particularly good choice. The quality measure (4) indicates 10 as the number of clusters which leads to a "natural" segmentation (see Figure 4). This number of clusters leads again to a reasonable compromise for planning.

Voronoi graph based segmentation is a standard method for segmenting occupancy grids. For our graph the Voronoi segmentation leads to 19 clusters. Figure 4 shows the segmentation for the Voronoi graph as well as the segmentation for our method with the same number of clusters. Hierarchical planning results indicate that our method leads to better segmentation for planning (see Figure 4). The average loss with respect to the optimal path is almost 3 times less and the computation time is more than 2 times less for the same number of clusters.

VI. Conclusions and Future Work

We presented an algorithm for automatically generating hierarchical maps. Base-level maps are represented as graphs, and higher-level maps are derived from base-level maps by graph partitioning, yielding another, smaller graph. Experiments on real data show that meaningful higher level-maps can be obtained and that planning can be made highly efficient. The hierarchical maps lead to average paths close to optimal for different number of clusters.

Different criteria lead to a different "optimal" number of clusters. These criteria include intuitiveness of the resulting higher level map, sparseness of the higher level map, optimality of the paths planned using the higher level map, and computational cost of planning. The quality of clustering criteria described in Section 3 yield intuitive, "natural" segmentations. To minimize computation costs, segmentations that have another number of clusters and that seem less natural may have to be chosen.

The optimality of paths compared to flat planning depends on structure of the graphs and the number of clusters in a complex way. The "natural" segmentation indicated by the quality of clustering criteria seems to usually give a reasonable choice with respect to optimality of the paths.

Compared to the standard Voronoi-based approach, our method for map segmentation is more general since it can be applied also to maps such as appearance-based maps. Our experiments indicate also that our segmentation method leads to hierarchical planning with less computation time while

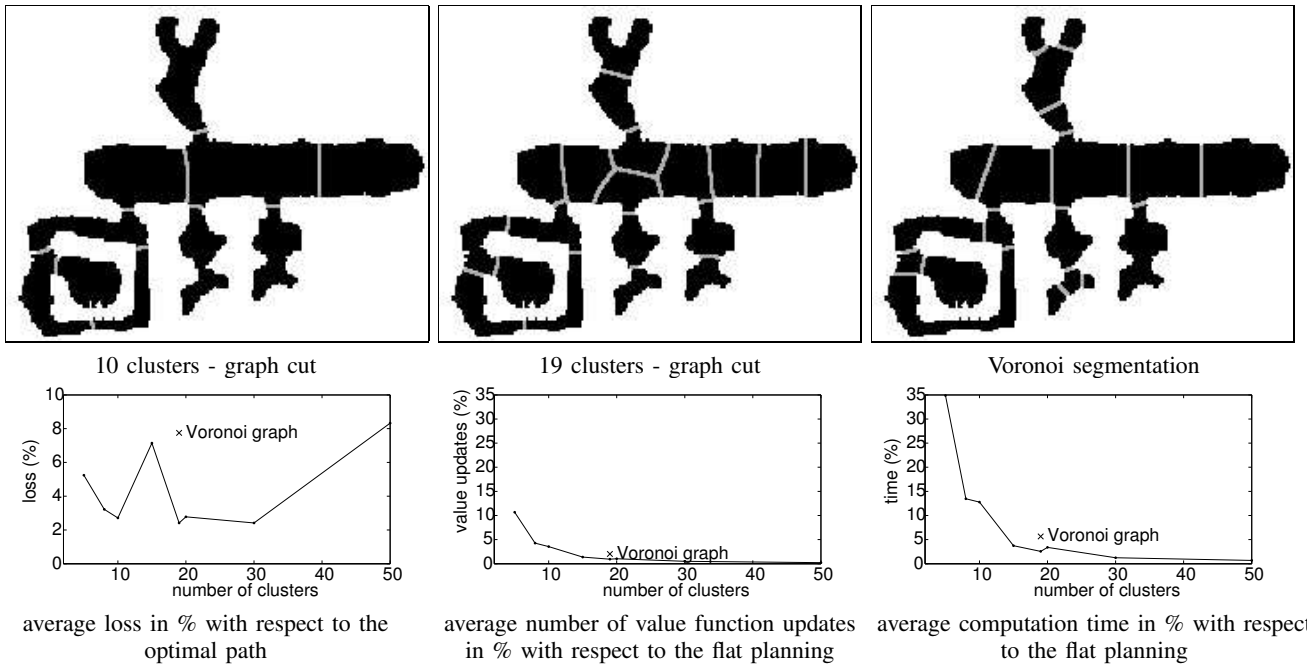


Fig. 4. Examples of segmentation and results of planning for various number of clusters for the occupancy grid. Above we present segmentation into 10 clusters chosen as the "best" by the quality measure from Section 3. We present the standard Voronoi segmentation and graph cut segmentation for the same number of 19 clusters. The results for the hierarchical path planning using the Voronoi-based segmentation are indicated by the cross below.

the average loss in optimality of the paths is better as well. Furthermore, the new method gives also more design choices. For example the similarity matrix that describes the base-level graph can be defined in various ways, different types of information can be included as discussed in Section 2, and different numbers of clusters can be chosen. Finally, extending the new representation to semi-supervised segmentation [3] and online, real-time segmentation may also be possible and we will focus on this in future work.

REFERENCES

[1] P. Althaus, H. Ishiguro, T. Kanda, T. Miyashita, and H. I. Christensen. Navigation for human-robot interaction tasks. In *In Proc. IEEE Int. Conf. on Robotics and Automation*, 2004.

[2] B. Bakker, Z. Zivkovic, and B. Kröse. Hierarchical dynamic programming for robot path planning. In *In Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2005.

[3] M. Belkin and P. Niyogi. Semi-supervised learning on riemannian manifolds. *Machine Learning, Special Issue on Clustering*, 56:209–239, 2004.

[4] D. P. Bertsekas. *Dynamic programming and optimal control*. Athena Scientific, Belmont, MA, 1995.

[5] J. M. Buhmann, W. Burgard, A. B. Cremers, D. Fox, T. Hofmann, F. E. Schneider, J. Strikos, and S. Thrun. The mobile robot RHINO. *AI Magazine*, 16(2):31–38, 1995.

[6] H. Choset and K. Nagatani. Topological simultaneous localisation and mapping: Towards exact localisation without explicit localisation. *IEEE Transactions on Robotics and Automation*, 17(2):125–137, 2001.

[7] R. Hartley and A. Zisserman. *Multiple view geometry in computer vision, second edition*. Cambridge University Press, 2003.

[8] S. D. Jones and J. L. Crowley. Appearance based processes for visual navigation. *In Proc. IEEE Int. Conf. on Intell. Robots and Syst.*, 1997.

[9] J.Shi and J.Malik. Normalized cuts and image segmentation. *IEEE Trans. Pattern Anlysis and Machine Intelligence*, 22(8):888–904, 2000.

[10] J. Kosecka and F. Li. Vision based markov localization. *In Proc. IEEE Robotics and Automation Conference*, 2004.

[11] B.J.A. Krose, N. Vlassis, R. Bunschoten, and Y. Motomura. A probabilistic model for appearance-based robot localization. *Image and Vision Computing*, 6(19):381–391, 2001.

[12] B. J. Kuipers. Representing knowledge of large-scale space. Technical Report TR-418, MIT Artificial Intelligence Laboratory, July 1977.

[13] D.G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. Journal of Computer Vision*, 2(60):91–110, 2004.

[14] S. Nayar, S. Nene, and H. Murase. Subspace methods for robot vision. *CUCS-06-95, Technical Report, Department of Computer Science, Columbia University*, 1995.

[15] A. Ng, M. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. *In Proc. Advances in Neural Information Processing Systems 14*, 2001.

[16] P.Lamon, A.Tapus, E.Glauser, N.Tomatis, and R.Siegwart. Environmental modeling with fingerprint sequences for topological global localization. *In Proc. IEEE/RSJ Int. Conf. on Intell. Robots and Systems*, 2003.

[17] I.Shimshoni R.Basri, E.Rivlin. Visual homing: Surfing on the epipoles. *Int. Journal of Computer Vision*, 33(2):117–137, 1999.

[18] S. Thrun. Learning metric-topological maps for indoor mobile robot navigation. *Artificial Intelligence*, 99(1):21–71, 1998.

[19] S. X. Yu and J. Shi. Multiclass spectral clustering. *In Proc. Int. Conf. on Computer Vision*, pages 11–17, 2003.

[20] L. Zelnik-Manor and P. Perona. Self-tuning spectral clustering. *In Proc. Advances in Neural Information Processing Systems*, 2004.

[21] Z. Zivkovic, B. Bakker, and B. Kröse. Hierarchical map building using visual landmarks and geometric constraints. In *In Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2005.