Online Semantic Exploration of Indoor Maps

Ziyuan Liu, Dong Chen and Georg von Wichert

Abstract—In this paper we propose a method to extract an abstracted floor plan from typical grid maps using Bayesian reasoning. The result of this procedure is a probabilistic generative model of the environment defined over abstract concepts. It is well suited for higher-level reasoning and communication purposes. We demonstrate the effectiveness of the approach through real-world experiments.

I. INTRODUCTION AND RELATED WORK

Most of today’s mapping approaches aim to construct a globally consistent, metric map of the robot’s operating environments. See Fig. 4 a) for a typical result. Such maps enable the robot to localize itself with respect to the environment. Based on this capability, the robot can also plan a path and navigate towards a goal, that will be specified by its metric position in the global map reference frame. However, the robots do not understand their environment in terms of typical semantic concepts like rooms, corridors or even functionally enriched concepts like kitchen or living room. Furthermore, the robots do not understand relations like adjacency, connectivity via doors, or properties like rectangularity that – if known to be relevant to the given environment – could help to build the maps in the first place.

Assigning semantics to spatial maps in robotics has not been looked at as intensely as metric or topological mapping. Still, several important contributions to the field have already been made. They can be clustered into two major groups. The first group consists of methods based on place labeling, some notable examples are [19], [15], [4], [5], [12]. These methods assign semantic labels to places or regions of the accessible work space of the robot. They are very much in the tradition of [16].

A second group is formed by approaches assigning semantic labels to parts or objects of the perceived structure of the environment, like traversable terrain, trees or similar structures in outdoor environments or walls, ceilings, and doors in indoor settings [10], [3], [13], [17], [18], [9], [14].

II. PROBLEM DESCRIPTION

Different from those methods mentioned above, we aim to construct a probabilistic generative model of the world around the robot, that is essentially based on abstract semantic concepts but at the same time allows to predict the continuous percepts that the robot obtains via its noisy sensors. This abstract model has a form similar to a scene graph, a structure which is widely used in computer graphics. The graph (see Fig. 4 c) in our case consists of rooms and doorways connecting the rooms and can be visualized as a classical floor plan (see Fig. 4 b).

The scene graph and the semantically annotated world state are represented as a vector of parameters $W$ observed through the occupancy map $M$. In Bayesian framework we can use a maximum posterior approach to infer the most probable state $W^* \in \Omega$ from the space of possible worlds $\Omega$ given the map $M$.

$$W^* = \arg\max_{W \in \Omega} p(W|M),$$

where

$$p(W|M) \propto p(M|W)p(W).$$

Here $p(W|M)$ is the posterior distribution of $W$ given a map $M$, and $p(W)$ is the prior specifying, which worlds $W$ are possible at all. $p(M|W)$ is the likelihood function describing how probable the observed map $M$ is, given the different possible worlds represented by a parameter vector $W$. The actual semantic model is represented in the structure of the parameter vector $W$, while semantically relevant constraints go into the prior $p(W)$.

III. A GENERATIVE MODEL FOR OCCUPANCY GRIDS

The prior $p(W)$ in (2) expresses a set of assumptions concerning the structured world based on context knowledge as follows:

1) a room has four walls and possesses a rectangular shape.
2) a room has at least one door, and a door is placed on a wall.
3) each cell in the map should only belong to one room.
4) walls of an indoor environment have two main orientations (see Fig. 4 b)).

The prior $p(W)$ penalizes worlds that are not fully compliant with the above assumptions:

$$p(W) = \alpha_1 \times \alpha_2 \times \alpha_3 \times \alpha_4,$$

where $\alpha_1$, $\alpha_2$, $\alpha_3$ and $\alpha_4$ are the corresponding penalization terms for the point 1), 2), 3) and 4) of the prior information respectively. $\alpha_1$, $\alpha_2$, and $\alpha_4$ are set to $\psi_1$, $\psi_2$, $\psi_4 \in (0, 1)$ if there is a conflict with the corresponding constraint, and $\alpha_3$ is
model parameters should be learned from training data, so that they pied, unexplored and free pixels, respectively. Ideally, these unknown pixels and free space pixels in the world $S$ accordingly:

$$
\alpha_3 = \prod_{c(x,y) \in M} \psi_3^{\gamma(c(x,y))},
$$

$$
\gamma(c(x,y)) = \begin{cases} 
\sigma(c(x,y)) - 1, & \sigma(c(x,y)) > 1, \\
0, & \text{otherwise},
\end{cases}
$$

where $\psi_3$ satisfies $\psi_3 \in (0,1)$. $c(x,y)$ denotes one grid cell in the map $M$. $\sigma(c(x,y))$ indicates the number of rooms, to which $c(x,y)$ belongs, $\alpha_3$ is a cell-wise penalization of the overlap between different rooms.

Furthermore, for our generative model, we need to specify the likelihood function $p(M|W)$. Since $M$ is represented by an occupancy grid with statistically independent grid cells $c \in M$, we only need to come up with a model $p(c|W)$ for all cells at their locations $(x, y)$ in the map $M$:

$$
p(M|W) = \prod_{(x,y) \in M} p(c(x,y)|W). \tag{5}
$$

For our model $p(c(x,y)|W)$, we first discretize the cell state $M(x,y)$ by classifying the intensity values into three classes $C_M(x,y)$ according to:

$$
C_M(x,y) = \begin{cases} 
2, & 0 \leq M(x,y) \leq h_o, \\
1, & h_o < M(x,y) \leq h_u, \\
0, & h_u < M(x,y) \leq h_f,
\end{cases} \tag{6}
$$

where $h_o$, $h_u$ and $h_f$ are the intensity thresholds for occupied, unexplored and free pixels, respectively. Ideally, these parameters should be learned from training data, so that they do not largely depend on the input data. Based on our world model $W$ we can also predict expected cell states $C_W(x,y)$ accordingly:

$$
C_W(x,y) = \begin{cases} 
2, & (x,y) \in S_w, \\
1, & (x,y) \in S_u, \\
0, & (x,y) \in S_f,
\end{cases} \tag{7}
$$

where $S_w$, $S_u$ and $S_f$ are the set of all the wall pixels, unknown pixels and free space pixels in the world $W$ respectively. $p(c(x,y)|W)$ can then be represented in the form of a lookup-table.

In principle the likelihood $p(c(x,y)|W)$ plays the role of a sensor model. In our case it captures the quality of the original mapping algorithm producing the grid map (including the sensor models for the sensors used during the SLAM process), and could be learned from labeled training data. However, for the experiments described in section V we used the manually selected values given in Table I.

| $C_W(x,y)$ | 0   | 1   | 2   |
|------------|-----|-----|-----|
| 0 (wall)   | 0.5 | 0.1 | 0.1 |
| 1 (unknown)| 0.3 | 0.8 | 0.1 |
| 2 (free space) | 0.2 | 0.1 | 0.8 |

**Table I**

The lookup table for $p(c(x,y)|W)$.

IV. SEARCHING THE SOLUTION SPACE

For solving (1) we need to efficiently search the large and complexity structured solution space $\Omega$. Here we adopt the approach of [20], who propose a data driven Markov chain Monte Carlo (MCMC) technique for this purpose. The basic idea is to construct a Markov Chain that generates samples $W_i$ from the solution space $\Omega$ according to the distribution $p(W|M)$ after some initial burn-in time. One popular approach to construct such a Markov chain is the Metropolis-Hastings (MH) algorithm [11], [7]. In MCMC techniques the Markov chain is constructed by sequentially executing state transitions (in our case from a given world state $W$ to another state $W'$) according to a transition distribution $\Phi(W'|W)$ of the sub-kernels. An example of $\Phi(W'|W)$ is given in Table II. In order for the chain to converge to a given distribution, it has to be reversible and ergodic [1]. The MH algorithm achieves this by generating new samples in three steps. First a transition is proposed according to $\Phi(W'|W)$, subsequently a new sample $W'$ is generated by a proposal distribution $Q(W'|W)$, and then it
is accepted with the probability $\lambda$.

$$\lambda(W, W') = \min\left(1, \frac{p(W'|M)Q(W|W')}{p(W|M)Q(W'|W)}\right)$$  \hfill (8)

The resulting Markov chain can be shown to converge to $p(M|W)$. However, the selection of the proposal distribution is crucial for the convergence rate. Here, we follow the approach of [20] to propose state transitions for the Markov chain using discriminative methods for the bottom-up detection of relevant environmental features (e.g., walls, doorways) and construct the proposals based on these detection results. An overview of our online semantic exploration algorithm is described in Algorithm 1.

**Algorithm 1** Online Semantic Exploration

**Require:** input map $M$

while certain accuracy condition not satisfied do

if input map updated then

generate the classified map $C_M(x, y)$;

generate new room candidates;
end if

select one sub-kernel according to the transition probabilities $\Phi(W'|W)$;

calculate the acceptance probability $\lambda(W, W')$ of the selected sub-kernel according to the MH algorithm [3];

draw a random float number $\theta$, $\theta \in U(0, 1)$;

if $\theta < \lambda(W, W')$ then

accept;
else

reject;
end if

end while

**A. MCMC Kernels**

In order to realize the Markov chain in form of the Metropolis-Hastings algorithm, we arrange the kernels that alter the structure of the world as reversible pairs:

- Kernel pair 1: ADD or REMOVE one room.
- Kernel pair 2: SPLIT one room or MERGE two rooms.
- Kernel pair 3: SHRINK or DILATE one room.
- Kernel pair 4: ALLOCATE or DELETE one door

Fig. 2 shows an example of the four reversible MCMC kernel pairs. The world $W$ can transit to $W'$, $W''$, $W'''$ and $W''''$ by applying the sub-kernel REMOVE, MERGE, SHRINK and DELETE, respectively. By contrast, the world $W'$, $W''$, $W'''$ and $W''''$ can also transit back to $W$ using corresponding reverse sub-kernels. In the following, we discuss the kernels in detail.

1) **ADD**: The sub-kernel ADD tries to put one new room into the world. Once the input map is updated, several room candidates are generated around the current robot position, then one room is selected from the generated rooms in a resampling style [8]. Each of the generated room candidates are weighted according to how well their walls match the observations provided by the occupancy grid map. The weight of a room $\omega_r$ is defined as the lowest wall weight $\omega_{w_j}$ among its four walls, where $j, j \in \{r_1, r_2, r_3, r_4\}$, indexes the wall, with $r_i, i \in \{1, 2, 3, 4\}$, indicating the $i$th wall of room $r$.

$$\omega_r = \min_{j \in \{r_1, r_2, r_3, r_4\}} \omega_{w_j}.$$  \hfill (9)

The wall weight $\omega_{w_j}$ is calculated as:

$$\omega_{w_j} = \frac{n(w_j)}{l(w_j)},$$  \hfill (10)

where $l(w_j)$ indicates the length of wall $w_j$ and can be computed from the coordinates of its two end points $(x_{w_{j,1}}, y_{w_{j,1}})$, $(x_{w_{j,2}}, y_{w_{j,2}})$:

$$l(w_j) = \sqrt{(x_{w_{j,1}} - x_{w_{j,2}})^2 + (y_{w_{j,1}} - y_{w_{j,2}})^2}.$$  \hfill (11)

The term $n(w_j)$ counts the number of wall pixels that match with the map:

$$n(w_j) = \sum_{(x, y) \in w_j} t(x, y),$$  \hfill (12)

where

$$t(x, y) = \begin{cases} 1, & C_M(x, y) = 0, \\ 0, & \text{otherwise}. \end{cases}.$$  \hfill (13)

The normalized weights $\omega'_r$ is calculated as:

$$\omega'_r = \sum_{r \in B} \omega_r,$$  \hfill (14)

where $B$ indicates the set of all the room candidates generated using the robot position. Then, we calculate the cumulative weights $A_r$ for room $r$:

$$A_r = \sum_{i=1}^{r} \omega_i.$$  \hfill (15)

Finally, we can draw a room candidate $n$ out of $B$, by generating a random number $k, k \in [0, 1]$,

$$n = \min\{i | k \leq A_i\}.$$  \hfill (16)

There are many ways to generate room candidates using robot position. Fig. 3 depicts one of the simplest ways of generating room candidates, which generates just one room of certain minimum size using robot position as the center of the generated room.
2) SPLIT: The sub-kernel SPLIT tries to decompose one member room into two rooms. To do this, a member room \( r \) is drawn randomly from the current world \( W \) according to a uniform distribution. Subsequently, we use Hough line detection in the room \( r \) to find all line segments which are potential splitting possibilities. Let \( E_r \) denote the set of the detected line segments within room \( r \). Each detected line segment \( e, e \in E_r \) is weighted, using its length \( l(e) \):

\[
\omega_e = l(e),
\]

where the length \( l(e) \) is similarly calculated as done in (17). Then we normalize the weights and build the cumulative distribution of \( E_r \). Furthermore, we draw one line segment out of \( E_r \), as done in (14) to (16).

The selected line segment is then extended to intersect with the walls of the room, so that two rooms are generated as the result of SPLIT.

3) MERGE: The sub-kernel MERGE tries to combine two members rooms of the current world \( W \). The first room \( r \) is drawn from the set of all member rooms of world \( W \) according to a uniform distribution. Additionally, a second room \( s \) needs to be selected from the rest of the member rooms. For sampling \( s \), we define a new weight \( a_r(s) \), which is the reciprocal of the distance \( d(c_r, c_s) \) between the center point \( c_r \) of room \( r \) and the center point \( c_s \) of room \( s \):

\[
d(c_r, c_s) = \sqrt{(c_r.x - c_s.x)^2 + (c_r.y - c_s.y)^2},
\]

where \( (c_r.x, c_r.y) \) and \( (c_s.x, c_s.y) \) are the image coordinates of the two center points. The weight \( a_r(s) \) is calculated as follows:

\[
a_r(s) = \frac{1}{d(c_r, c_s)},
\]

Applying the same sampling technique as that in (14) to (16), we can obtain the second room \( s \). Then we try to combine them into one room. The underlying idea for using \( a_r(s) \) in the sampling is that the closer two rooms are, the more likely they can be combined.

4) SHRINK and DILATE: The kernel pair SHRINK/DILATE tries to move a wall \( w_j \) of a member room \( r \) of the current world \( W \) along one of the main orientations. Here, \( j, j \in \{ r_1, r_2, r_3, r_4 \} \), indexes the wall, with \( r_i, i \in \{ 1, 2, 3, 4 \} \), indicating the \( i \)th wall of room \( r \). For selecting the room \( r \) from the set of all member rooms, we define a new weight \( b_r \):

\[
b_r = \begin{cases} \frac{1}{\omega_w}, & \omega_w \leq h_b \\ \frac{1}{\omega_w}, & \text{otherwise} \end{cases},
\]

where \( \omega_w \) is the room weight defined in (9), \( h_b \) is a predefined threshold for the weight. Using \( b_r \), a room is drawn according to (14) to (16).

Once the room is selected, one wall \( w_j \) needs to be drawn from its four walls. Following the same idea, we define a new weight \( v_{w_j} \) for sampling the wall:

\[
v_{w_j} = \begin{cases} \frac{1}{\omega_{w_j}}, & \omega_{w_j} \leq h_v \\ \frac{1}{\omega_{w_j}}, & \text{otherwise} \end{cases},
\]

where \( \omega_{w_j} \) is the wall weight defined in (10), and \( h_v \) is a predefined threshold. \( h_b \) and \( h_v \) are just the upper bound for the corresponding weight, thus they do not depend on the input data. Again, the wall is drawn according to \( v_{w_j} \), as done in (14) to (16). After the wall is selected, we propose to shift it along one of the main orientations using a zero-mean Gaussian distribution. In principle, the algebraic sign decides whether a SHRINK or a DILATE is proposed, e.g. if a positive sign proposes a SHRINK, then a negative sign will propose a DILATE. In general, SHRINK and DILATE sub-kernel have both 50% chance to be proposed.

5) ALLOCATE: This sub-kernel tries to assign one door to two member rooms so as to explore connectivity information of the semantic world. A door detector which is based on blob detection [2] proposes door candidates for the sub-kernel ALLOCATE. We draw one door candidate from the set of all candidates according to their weights. Here, the weight \( \omega_g \) of a door \( g \) is similar to the weight of walls \( \omega_{w_j} \) that is defined in (10):

\[
\omega_g = \frac{n'(g)}{l(g)},
\]

where \( l(g) \) is calculated the same as in (11), and \( n'(g) \) is computed as follows:

\[
n'(g) = \sum_{(x,y) \in g} t'(x,y),
\]

where

\[
t'(x,y) = \begin{cases} 1, & C_M(x,y) = 2 \\ 0, & \text{otherwise} \end{cases}
\]

Using the weight \( \omega_g \), one door candidate is drawn from the set of all detected candidates, as done in (14) to (16).

Then, the MH algorithm decides whether this door will be accepted. Here we do not detail on how to detect the door candidates.

6) REMOVE and DELETE: The sub-kernel REMOVE and DELETE have similar functionality, which is to cancel one existing member room and one of the assigned doors respectively. There are no special discriminative methods used for these two sub-kernels. They just draw one member from the corresponding set (existing rooms or assigned doors) and propose to cancel this member, then the MH algorithm decides whether this proposal is accepted. Following the idea
that the worse a member matches the map, the more likely it should be canceled, we use the weight $b_r$ defined in (20) for room sampling. Similarly, we define a new weight $z_g$ for door sampling:
\[
z_g = \begin{cases} 
\frac{1}{\omega_g}, & \frac{1}{\omega_g} \leq h_g \\
h_g, & \text{otherwise,}
\end{cases}
\]
(25)
where $\omega_g$ is the door weight defined in (22), and $h_g$ is a predefined threshold.

**B. Proposal Probability $Q(W'|W)$ and $Q(W|W')$**

The proposal probability $Q(W'|W)$ describes how probable the world $W$ can transit to the world $W'$, and by contrast, $Q(W|W')$ is the probability for transiting back to the world $W$, given the world $W'$. Intuitively, $Q(W'|W)$ is the product of the normalized weight of the selected elements (room candidate, splitting line, wall etc.) in the corresponding MC sub-kernel defined in the previous section. For instance, in the ADD or REMOVE sub-kernel, $Q(W'|W)$ is equal to the corresponding normalized weight of the selected room candidate or that of the selected member room.

Compared with $Q(W'|W)$, the calculation of $Q(W|W')$ is less intuitive, because the back transition is virtual and must be defined. In the example of ADD, $Q(W'|W)$ should perform the same function as the sub-kernel REMOVE, namely, the world $W'$ transits back to the world $W$ by canceling the room that is added in the transition from $W$ to $W'$, thus $Q(W'|W)$ of ADD should be the normalized weight of the added room in the sub-kernel REMOVE. $Q(W|W')$ for other sub-kernels can be similarly defined.

**V. EXPERIMENTAL RESULTS**

In experiments, we use the gmapping package in ros.org [6] to provide input maps for our algorithm. The selection probabilities of the MC sub-kernels are listed in Table II. The values in Table II are just an example. There are many possibilities on how to choose them. The simplest one is to equally set all the probabilities. In this sense, these values are independent of the input data.

![Fig. 4](image)

**Table II**

| Sub-kernel | Probability |
|------------|-------------|
| ADD        | 0.2         |
| REMOVE     | 0.05        |
| SPLIT      | 0.125       |
| MERGE      | 0.125       |
| SHRINK     | 0.2         |
| DILATE     | 0.2         |
| ALLOCATE   | 0.05        |
| DELETE     | 0.05        |

**Fig. 4.** Final result of the semantic exploration. a) The last input map $M$ obtained from gmapping. b) The classified map $C_M(x, y)$ with three intensity values (black=wall, grey=unexplored, white=free). c) The analyzed world $W$ (green=wall, black=unknown, blue=free, orange=door). d) A direct comparison between the analyzed world and the input map: walls (blue) and doors (orange) of the world $W$ drawn into the map. Here the violet point shows the robot position.

through many successful DILATE. In part 6, 10) and 14) another three rooms are generated using the robot position and join the world by successful ADD. In part 10) two rooms are joined together by a successful MERGE. In part 16) one new room is generated independently on the robot position in the way that its neighbor room became first bigger through DILATE and was then decomposed into two rooms through a successful SPLIT.

Final result of our semantic exploration algorithm for another indoor environment is shown in Fig. 6. Since there is no furniture in this environment, it totally complies with our semantic model, thus, the result is better than that in Fig. 4.

**VI. CONCLUSIONS AND FUTURE WORK**

This paper proposes a new approach for automatically extracting semantic information from preprocessed sensor data. We propose to do this by means of a probabilistic generative model and MCMC-based reasoning techniques. We construct an abstracted semantic and top-down representation of the domain under the consideration: a classical indoor environment consisting of several rooms, that are connected by doorways. We use Bayesian reasoning to build this semantic map, so that it is aligned with the preprocessed sensor observations, that a robot made during an environment exploration and mapping stage. This introduces a bottom-up path into the approach and employs data driven discriminative environment feature detectors to analyze the continuous
While we currently generate representations that more or less resemble a classical floor plan (including semantics however), the extension of our work to more functionally enhanced representations (e.g. differentiating several room types based on the context, adding other types of concepts like general objects or furniture) is obvious and will be pursued in the future. It is also straightforward to extend the concept towards 3D environment representations.

ACKNOWLEDGMENTS
This work is founded by the Institute for Advanced Study at the Technische Universität München.

REFERENCES
[1] C. M. Bishop. Pattern recognition and machine learning. Springer, 2007.
[2] Fu Chang, Chun jen Chen, and Chi jen Lu. A linear-time componentlabeling algorithm using contour tracing technique. Computer Vision and Image Understanding, 93:206–220, 2004.
[3] B. Douillard, D. Fox, and F. Ramos. Laser and vision based outdoor object mapping. In Proc. of Robotics: Science and Systems (RSS), pages 9–16, 2008.
[4] S. Friedman, H. Pasula, and D. Fox. Voronoi random fields: Extracting the topological structure of indoor environments via place labeling. In Proc. of the International Joint Conference on Artificial Intelligence (IJCAI), 2007.
[5] N. Goerke and S. Braun. Building semantic annotated maps by mobile robots. In Proceedings of the Conference Towards Autonomous Robotic Systems, Londonderry, UK, 2009.
[6] G. Grisetti, C. Stachniss, and W. Burgard. Improved techniques for grid mapping with rao-blackwellized particle filters. IEEE Transactions on Robotics, 23(1):34–46, 2007.
[7] W.K. Hastings. Monte carlo sampling methods using markov chains and their applications. Biometrika, 57:97–109, 1970.
[8] G. Kitagawa. Monte carlo filter and smoother for non-gaussian nonlinear state space models. Computational and Graphical Statistics, 5(1):1–25, 1996.
[9] A.K. Krishnan and K.M. Krishna. A visual exploration algorithm using semantic cues that constructs image based hybrid maps. In IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 1316 –1321, Oct. 2010.
[10] B. Limketkai, L. Liao, and D. Fox. Relational object maps for mobile robots. In International Joint Conference on Artificial Intelligence, volume 19, page 1471, 2005.
[11] N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller, and E. Teller. Equation of state calculations by fast computing machines. Journal of Chemical Physics, 21:1087–1092, 1953.
[12] O.M. Mozos and W. Burgard. Supervised learning of topological maps using semantic information extracted from range data. In IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 2772–2777. IEEE, 2006.
[13] A. Nüchter and J. Hertzberg. Towards semantic maps for mobile robots. Robotics and Autonomous Systems, 56(11):915–926, 2008.
[14] M. Persson, T. Duckett, C. Valgren, and A. Lilienthal. Probabilistic semantic mapping with a virtual sensor for building/nature detection. In International Symposium on Computational Intelligence in Robotics and Automation, pages 236 –242, June 2007.
[15] A. Pronobis, P. Jensfelt, K. Sjöö, H. Zender, G. M. Kruijff, O. M. Mozos, and W. Burgard. Semantic modelling of space. In Cognitive Systems, volume 8 of Cognitive Systems Monographs, pages 165–221. Springer Berlin Heidelberg, 2010.
[16] S. Thrun and A. Bäckken. Integrating grid-based and topological maps for mobile robot navigation. In Proceedings of the AAAI Thirteenth National Conference on Artificial Intelligence, Portland, Oregon, 1996.
[17] Jingchen Tong, Dong Chen, Yan Zhuang, and Wei Wang. Mobile robot indoor semantic mapping using 3d laser scanning and monocular vision. In 8th World Congress on Intelligent Control and Automation (WCICA), pages 1212 –1217, July 2010.
[18] Tingqi Wang and Qijun Chen. Object semantic map representation for indoor mobile robots. In International Conference on System Science and Engineering (ICSSE), pages 309 –313, June 2011.
[19] D.F. Wolf and G.S. Sukhatme. Semantic mapping using mobile robots. IEEE Transactions on Robotics, 24(2):245–258, 2008.
[20] S.C. Zhu, R. Zhang, and Z. Tu. Integrating top-down/bottom-up for object recognition by data driven markov chain monte carlo. In IEEE Computer Vision and Pattern Recognition (CVPR), 2000.