Autonomous Object Discovery on a Mobile Platform

Alex Grubb
Advisor: Dr. Paul Rybski

CMU SCS Senior Honors Thesis
Spring 2007

Abstract

This paper studies the problem of a mobile agent autonomously discovering objects within an unknown environment. We demonstrate an unsupervised clustering algorithm capable of grouping environmental features into distinct groups, and show how this clustering algorithm can be used to segment the world into seperate objects. The problem of learning the structure of the environment is also addressed, and a spring-inspired model for solving the simultaneous localization and mapping (SLAM) problem is used to map the environment.

1 Introduction

In this paper we consider the problem of autonomous object discovery on a mobile robot, in the specific context of discovering the locations and rough object models of landmark objects within an environment, using an unsupervised clustering algorithm for grouping features present in the environment. At the same time, we seek to learn the structure of the environment and use the learned environmental structure to build models of each landmark object discovered, and guide discovery of further objects.

The landmark objects we try to discover are large, immobile objects within an environment, and should ideally be objects that are easily human-recognizable, such as furniture or decorative elements. These objects can then be used to localize the robot, and the learned models can be used for object recognition in other tasks.

Learning about these objects is critical for human-robot interaction tasks, and for mobile robots operating in typical human environments, as these are typical objects that such robots will have to deal with. By autonomously discovering objects within the environment, we aim to remove the tedious step of hand-training an object recognition system to detect objects specific to a new and unkown environment, and allow the robot to adapt to new environments quickly and with minimal outside interference, making these mobile robots accessible to non-technical users.

Our approach to solving this problem is to combine existing efforts in the areas of simultaneous localization and mapping, along with efforts in video data mining and object discovery in video data. Specifically, we aim to use existing mapping techniques to learn the structure of the environment, and then segment the features found into semantically meaningful groups. This involves pulling in knowledge from existing object discovery work, and modifying to both deal with the uncertainty issues inherent in mobile robotics

and take advantage of the added information about position, camera calibration, and the information provided by mapping.

1.1 Simultaneous Localization and Mapping

Mapping and localization is one of the key research topics within the field of mobile robotics, and many advances have been made in this area in the past few years. Traditionally, most SLAM research has been done using some kind of range sensor, such as a laser range finder [2], but more recently there have been a large number of advances for solving the SLAM problem with respect to visual data and other sensors.

While some visual mapping research can be fit into a more traditional SLAM framework, due uses of things like stereo camera systems to approximate depth [10], in our specific context we are concerned more with some of the other approaches, which deal with bearing-only data [3], while still producing the same structural map of the environment. There are also a number of other flexible SLAM frameworks, such as the one we have adapted for this research, which uses a spring and mass model inspired by the maximum likelihood estimate for the optimal map [9].

1.2 Object Discovery and Feature Segmentation

The main component of our research involves the grouping of environmental features in to distinct object groups. A number of approaches have been proposed to perform similar tasks on video data, particularly using a variety of spatio-temporal techniques [7] to group features into meaningful groups. A different for improving object models and discovering the full extent of objects given a minimal number of source images [6].

Other techniques for segmentation have also been tried, including using pairwise co-occurence of features in segmented image regions [15] and other segmentation methods utilizing or based off of the Mean Shift algorithm [1]. Other approaches also utilize different underlying segmentations to group image features [8].

Another line of object discovery research involves clustering configurations of features based on spatial and temporal overlap [14] [11] [12]. The advantage present in these methods is that they are more robust to feature detection and matching errors, due to the large number of features considered in every configuration.

1.3 Existing System

For our research, we used the CORAL Lab's CMAssist mobile robots, which were designed as platforms for human-robot interaction studies and as competitors in the Robocup@Home event at Robocup 2007. The focus of the event and the robots' design is to allow autonomous operation in a household or office setting, mostly performing assistant type tasks where knowledge of the environment and key objects is a critical factor.

An object recognition system was already in place on the robots, using local image features with PCA-SIFT [4] descriptors to recognize key pieces of furniture within the environment, such as couches or a telelvision, and roughly determine object location and orientation. The object models are based off of single labeled images provided for the robot beforehand, with a Hough transformation used to determine the object's location [5].

The vision system being used is a 4-way camera called a CAMEO, which consists of four separate cameras at 90 degrees to one another. We are not using any kind of stereo vision or other range finding system, so all data being used is bearing-only data about feature locations, and position and odometry information for the robot.

2 Mapping

When simultaneously localizing and mapping, we wish to find optimal positions for both the recorded robot positions, and the positions of all features detected in the environment. This can be expressed as a maximum-likelihood estimation problem of finding the most likely map M given a set of sensor readings S, or finding the map M which will maximize:

$$P(M|S) = \prod_{s \in S} P(M|s) \tag{1}$$

We can obtain a form of this problem that is very similar to the potential energy equation for a spring:

$$\frac{k}{2}(e - \hat{e})^2 \tag{2}$$

by taking the $-log$ of the maximum likelihood equation (essentially changing the problem to one of minimizing the energy):

$$-log(P(M|S)) = \prod_{s \in S} -log(P(M|s)) \tag{3}$$

and then using the assumption that $P(M|s)$ is gaussian, which will give the following result when taking the $-log$ in the one dimensional case:

$$\frac{1}{2}(s-\mu)^T \sigma (s-\mu) \tag{4}$$

The similarity between these two equations is the basis of a spring and mass model [9] for the map. Recorded robot positions and feature positions are modeled using masses, with each mass being interconnected using a linear spring for determining distance and two torsional springs, to determine the angle of each mass relative to the linear spring connecting them. Once an initial spring and mass model has been created, it can be relaxed using a number of numerical simulation methods, and the optimal map can then be derived from the final positions of the masses in the model.

To model the bearing-only feature readings present in our sensor, we need to let the linear spring connecting each mass representing a feature to each mass representing a pose stretch infinitely, allowing for any possible range. This can easily be modeled by setting the linear spring constant s_k to 0. The only constraints affecting placement of the features then are the bearing constraints enforced by each torsional spring, allowing the relaxation step to effectively find the optimal feature location, given only bearing information for that feature.

2.1 Initial Model Generation

To generate the initial mass and spring model, we first generate the model data for all recorded poses of the robot, using the odometry data. For each mass M_i, we first record the starting position as the position of pose p_i given by the odometry data. Then, we calculate the resting values of the springs, consisting of one linear and two torsional springs, between every set of consecutive poses. The linear spring between the two poses is calculated to have rest length ℓ_i equal to the distance between p_i and p_{i+1}:

$$\ell_i = \sqrt{(p_{i,x} - p_{(i+1),x})^2 + (p_{i,y} - p_{(i+1),y})^2} \tag{5}$$

The rest angles for each torisional spring θ_i and ϕ_{i+1} are then calcuated to have rest angles equal to the distance between the recorded pose and the connecting linear spring:

$$\theta_i = p_{i,\theta} - \text{atan2}(p_{i+1,x} - p_{i,x}, p_{i+1,y} - p_{i,y}) \tag{6}$$

$$\phi_{i+1} = p_{i+1,\theta} - \text{atan2}(p_{i,x} - p_{i+1,x}, p_{i,y} - p_{i+1,y}) \tag{7}$$

Next, initial feature positions are calculated by first tracking each feature over a series of frames, using the configuration tracking algorithm outlined in the next section. If a feature is tracked over a sufficient number of frames, simple line intersection is used in combination with the bearing and robot position corresponding to each frame, to find a rough guess as to the position of the feature.

As stated above, an infinitely malleable linear spring is used to connect the feature f_j to each position p_i that the feature was detected at. The rest angle θ_i, j is then set to be the bearing the feature was detected at for pose p_i. No torsional springs are used on the mass representing the feature, as we also lack information about which direction the feature is oriented, and so the orientation is unconstrained.

Due to the inherent uncertainty in tracking features, false positives will occasionally be added to the spring and mass model as tracked features, leading to the application of incorrect forces in the model. This can cause position estimates to be relaxed to erroneus positions, and the relaxation step to blow up computationally due to the unsatisfiable constraints that the false matches impose. One easy way to counter this problem is to linearly scale the torsional spring constant s_k of all the non-odometry related torsional springs based on the total number of feature detections being added into the map. This will prevent false positives from having too much of an impact on the relaxation step, and will prevent the computation from getting out of hand by limiting the total energy present in the part of the model representing features.

3 Unsupervised Clustering

To actually perform the object discovery, we use a greedy, unsupervised clustering algorithm similar to the one developed by Sivic and Zisserman [14] for video data-mining. Using configurations of the nearest neighbors of each feature, we first track these configurations across every frame, building

clusters of configurations corresponding to a tracked feature. These initial clusters are then greedily merged into larger clusters, based on their overlap.

For the clustering, it is both efficient and convenient to first vector quantize the PCA-SIFT descriptor of all detected features using K-means clustering, with every cluster being treated as it's own *v*isual word [13]. For the rest of the clustering, two features will be considered to be the same feature if they are both the same visual word, or both nearest to the same cluster center. By vector quantizing all the features, determining if two features match is now a very efficient and simple process, as we can just look up what visual word each feature belongs to.

For each vector quantized feature, we also generate a configuration consisting of the N spatially nearest neighbors of that feature in the image. These configurations are only defined by the features present in them, and orientation or layout information for configuration is not stored and is not used for any of the following algorithm.

We say that two neighborhoods match with M matching neighbors if M of the nearest neighbors in one configuration are present in the other configuration. Again, this is based strictly on membership, and layout of the neighbors is not taken into account. This allows the configurations to be matched robustly across many viewpoints, and allows for errors due to missed feature detections to be recovered from easily.

The final pre-processing step of the algorithm is to remove outlying configurations which either occur too frequently or to seldom. This is done by matching each configuration to every other configuration, and counting the number of matches with $K = 3$ neighbors in common. The top and bottom %5 are then treated as outliers and pruned. The center feature is also required to match for two configurations to be a match at this stage.

3.1 Tracking Configurations

Once we have computed the visual words present in all captured frames and computed the nearest neighbor configurations for each of them, we then track these configurations across every frame, building initial clusters of configurations, using the scores from the pre-processing stage to figure out the order to cluster in.

For this stage, we require that M nearest neighbors match, but the center feature is not required to match for two configurations to match. This provides some measure of robustness against missed detections.

The greedy clustering algorithm is as follows:

- Select the configuration with the highest remaining score from the first stage that hasn't yet been grouped into a cluster.
- Find the best matching configuration for the selected one in each frame , and add them to the new cluster if M or more of the neighbors match.
- Remove all configurations in the new cluster and set them aside.
- Repeat

This produces initial clusters with at most one configuration from a given cluster in each frame. These initial clusters can also be thought of as one configuration tracked over time, with each match representing a detection in a given frame.

3.2 Clustering

Following the initial tracking, we then grow the clusters using a spatiotemporal clustering algorithm. Again, we use the number of matches found in the last stage as a method of determining which clusters should be grown first. Here, we consider two clusters, or tracked configurations to match if they overlap in p precent of all frames that either cluster was present in.

The cluster growing algorithm is as follows:

- Select the initial cluster with the largest size.
- For every other unmerged cluster, do the following:
  - Count the number of frames where the configuration for the initial cluster in that frame matches the configuration for the other cluster in that frame, with O nearest neighbors in common.
- If the percent of frames where both clusters match is greater than p percent of the total frames either cluster is detected in,merge the two clusters.
- Repeat

4 Discussion

Figure 1 shows the results for mapping the first few tracked features for a small traversal. One interesting thing to note is the relative accuracy of the initial positions for each feature, compared to their final resting positions,

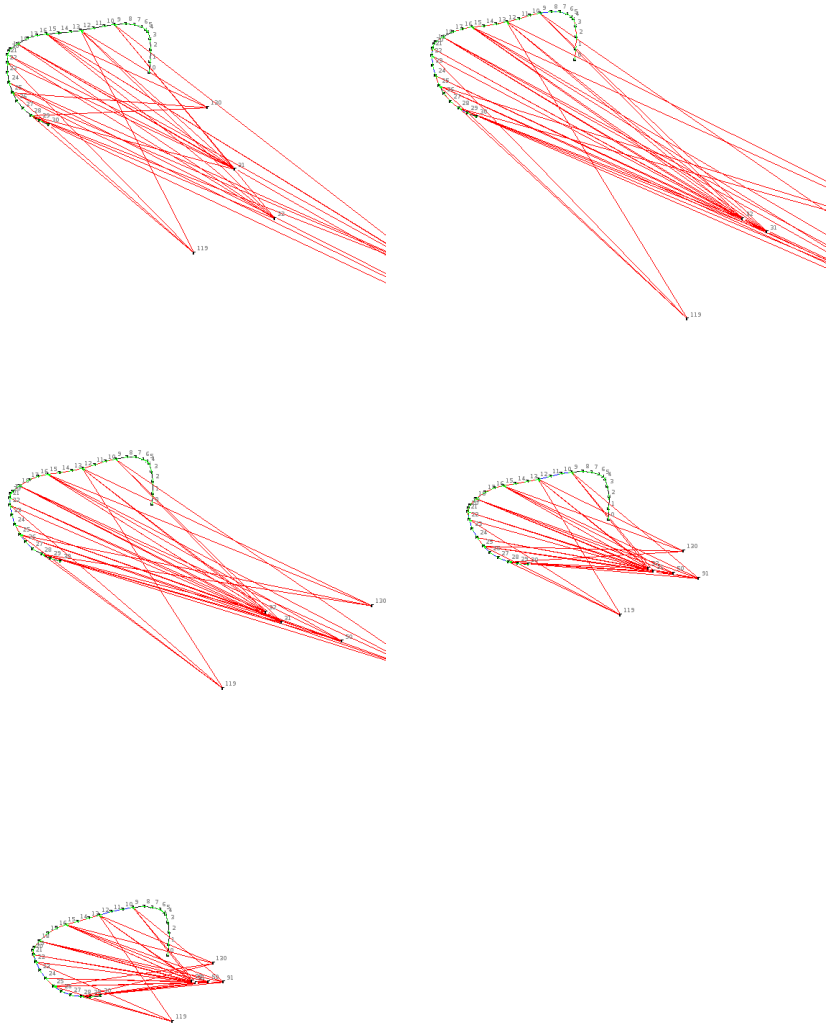

Figure 1: Mapping results for a small number of features.

althought the spring simulation causes the feature locations to move quite far away from their final positions, and then slowly relaxes them back to the correct configuration.

One important aspect of the mapping that is not visualized here is the false positive case, where a number of features are matched but their bearing data can not correspond to a valid position. When large number of features are included in the map, this problem becomes apparent as a small number of incorrect matches are inevitably included in the map. The forces introduced by these incorrect matches can be enough to warp the map and result in very poor solutions for the map.

These results indicate that a good avenue for further research is in the improvement of the way the spring and mass model represents features, so that the map can handle false positives for feature matches with minimal effect on the accuracy of the resulting map. One potential way to deal with this problem would be to model each feature detection as its own mass, and place linear springs with 0 rest length between each detection that is considered a match. By tweaking the parameters it could be possible to model the potential for false positives in the feature tracking.

Another way this model for mapping could be further advanced would be to improve the feature tracking and matching itself. Currently, our system does not take advantage of the added information we gain from working on a mobile robot. By using odometry data being recorded by the robot and optical flow modeling, the window where a configuration could appear in the next frame could be limited, and the tracking algorithm made much more robust.

For clustering, we first show results for one set of parameters ($N = 20, M = 6, O = 6$) in Figure 2. While the first and fourth clusters are well localized to small, reasonably sized areas given the objects we set out to discover. Additionally, the seem to focus on the couch and bookcase areas, seperating out the interesting objects from the rest of the room. The other two clusters however spill out over the rest of the environment, becoming too large to be useful.

Another feature of note is that while the clusters tend to extend throughout the environment they are for the most part non-overlapping, and have segmented the environment spatially fairly well. This is critical for segmenting out regions containing objects of interest, so that we obtain a good idea of the location of a given object, and also end up with a complete set of features that make up that object without many holes in the coverage.

Finally, we examine the benefit of varying the parameters used in the unsupervised clustering algorithm. In Figure 3 we can see the effects that

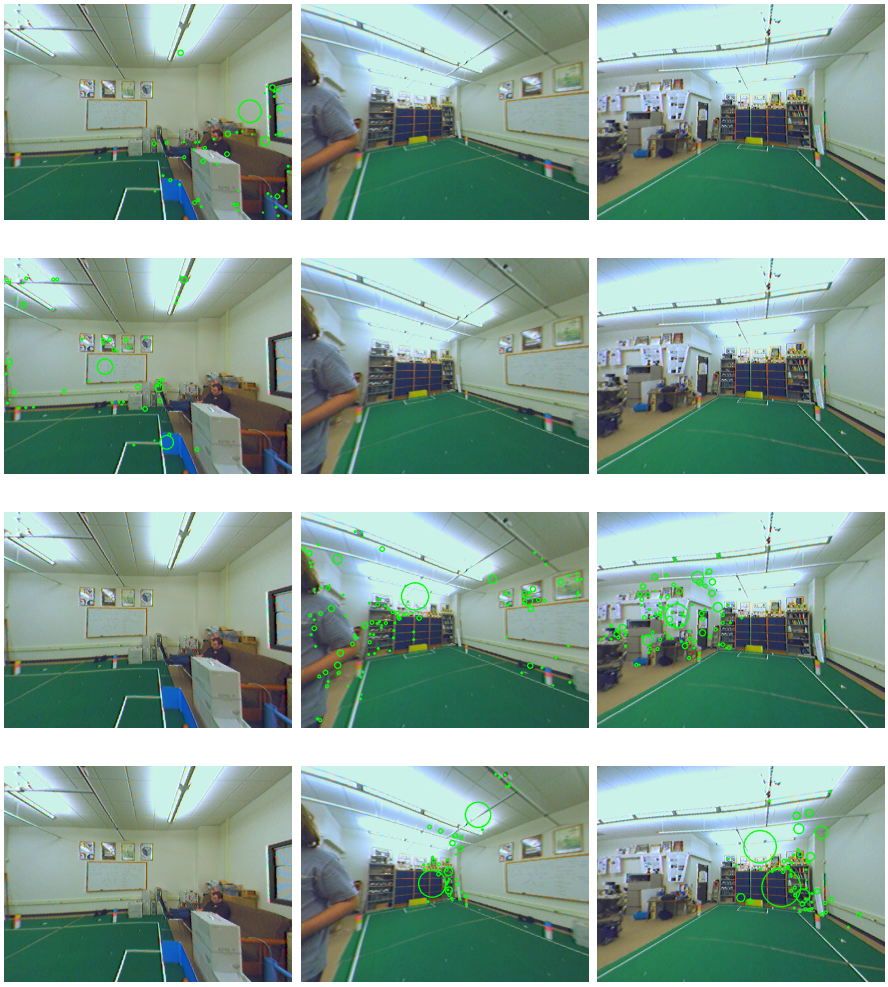

Figure 2: Four largest clusters for clustering with $N = 20, M = 6, O = 6$.

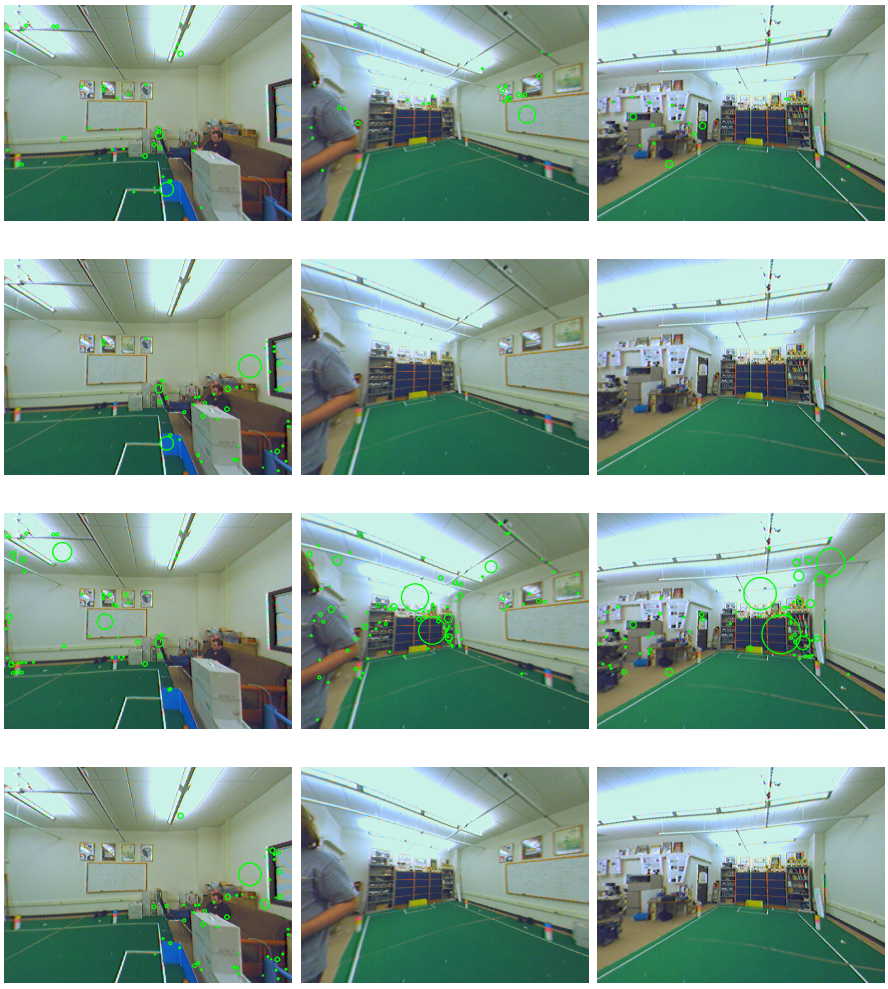

Figure 3: Largest clusters for clustering with $N = 20$ and $(M = 3, O = 3)$, $(M = 3, O = 6)$, $(M = 6, O = 3)$, and $(M = 6, O = 6)$, respectively.

the different parameters have on the largest cluster returned. In terms of the first cluster, it is clear that the O parameter is critical for keeping the cluster from growing too large (unsurprisingly, as this is the parameter which determines minimum overlap for merging clusters).

Additionally, the value of M seems to have little effect on the cluster generated, as we can see in the $M = 3, O = 6$ and $M = 6, O = 6$ clusters strong similarity (but there does seem to be some effect when M and O are both varied, as in the first set of parameters). These results do show that overall varying the O parameter does show some promise as a potential way for improving the segmentation.

The potential for varying the parameters on the clustering algorithm opens up many options for further work. Using multiple segmentations or multiple results from the same algorithm with different parameters is a well established method in the object discovery field [15] [8], and could be applied to our system in a similar method.

One method in particular that we believe shows a lot of promise is that of iteratively refining the clusters returned by re-running the algorithm with varying the parameters, in the hopes of improving the clusters generated. This would require having some measure of the quality of the clusters returned. We propose that one way to measure this quality could be the physical extent of cluster, which can be measured using the map we've built up.

Once we have a measure of the rough size we want for a given cluster, we can re-run the algorithm on that cluster, and as we saw above, vary O accordingly, in an attempt to either increase or decrease the size of the cluster. This will allow clusters that are roughly the size of objects that we are concerned about to be formed. Many other possibilities for tweaking the parameters in the algorithm and for iteratively refining the clusters generated exist, and leave open many areas for exploration.

5 Conclusion

In this paper, we have shown a method of clustering the features in an unknown environment and have made first steps toward discovering objects on a mobile platform. In addition, we have demonstrated a method for using an existing SLAM solution in a novel way, when dealing with bearing-only data from visual sensors. Finally, we've shown that the method of unsupervised clustering inspired by Sivic and Zisserman [14] shows promise for use on a mobile platform, and also seems to show that an iterative,

multiple segmentation approach to object discovery may have some merit.

References

[1] D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. *IEEE PAMI*, 24:603–619, 2002.

[2] A. I. Eliazar and R. Parr. Dp-slam 2.0. *ICRA*, 2:1314–1320, 2004.

[3] P. Jensfelt, D. Kragic, J. Folkesson, and M. Björkman. A framework for vision based bearing only 3D SLAM. *ICRA*, pages 1944–1950, 2006.

[4] Y. Ke and R. Sukthankar. Pca-sift: A more distinctive representation for local image descriptors. *CVPR*, 02:506–513, 2004.

[5] D. Lowe. Object recognition from local scale-invariant features. *ICCV*, 2(2):1150–1157.

[6] D. Lowe. Local feature view clustering for 3d object recognition. *CVPR*, 1:682–688, 2001.

[7] R. Megret and D. DeMenthon. A survey of spatio-temporal grouping techniques. Technical Report LAMP-TR-094/CS-TR-4403, University of Maryland, College Park, 1994.

[8] B. C. Russell, A. A. Efros, J. Sivic, W. T. Freeman, and A. Zisserman. Using multiple segmentations to discover objects and their extent in image collections. *CVPR*, 2006.

[9] P. Rybski. *Building Topological Maps using Minimalistic Sensor Models*. PhD thesis, University of Minnesota, 2003.

[10] S. Se, D. G. Lowe, and J. J. Little. Vision-based global localization and mapping for mobile robots. *IEEE Robotics*, 21:364–375, 2005.

[11] J. Sivic, B. C. Russell, A. A. Efros, A. Zisserman, and W. T. Freeman. Discovering objects and their localization in images. *ICCV*, 1:370–377, 2005.

[12] J. Sivic, F. Schaffalitzky, and A. Zisserman. Object level grouping for video shots. *IJCV*, 67(2):189–210, 2006.

[13] J. Sivic and A. Zisserman. Video Google: A text retrieval approach to object matching in videos. *ICCV*, 2:1470–1477, 2003.

[14] J. Sivic and A. Zisserman. Video data mining using configurations of viewpoint invariant regions. *CVPR*, 1:488–495, 2004.

[15] T. Stepleton and T. S. Lee. Using co-occurrence and segmentation to learn feature-based object models from video. *WACV/MOTION'05*, 1:129–134, 2005.