A Unified Framework for Planning and Execution-Monitoring of Mobile Robots

M. Gianni, P. Papadakis, F. Pirri
ALCOR, Cognitive Robotics Lab
DIS, University of Rome "La Sapienza", Italy

M. Liu, F. Pomerleau, F. Colas
Autonomous Systems Lab,
ETH Zurich, Switzerland

K. Zimmermann, T. Svoboda, T. Petricek
Center for Machine Perception
Czech Technical University, Prague

G.J.M Kruijff, H. Khambhaita, H. Zender
German Research Center for Artificial Intelligence (DFKI GmbH)
Saarbrücken, Germany

Abstract

We present an original integration of high level planning and execution with incoming perceptual information from vision, SLAM, topological map segmentation and dialogue. The task of the robot system, implementing the integrated model, is to explore unknown areas and report detected objects to an operator, by speaking loudly. The knowledge base of the planner maintains a graph-based representation of the metric map that is dynamically constructed via an unsupervised topological segmentation method, and augmented with information about the type and position of detected objects, within the map, such as cars or containers. According to this knowledge the cognitive robot can infer strategies in so generating parametric plans that are instantiated from the perceptual processes. Finally, a model-based approach for the execution and control of the robot system is proposed to monitor, concurrently, the low level status of the system and the execution of the activities, in order to achieve the goal, instructed by the operator.

Introduction

In real-world applications where environmental constraints are minimal, highly efficient multi-modal perception is a prerequisite for action planning and execution (Muscettola et al. 2002). The problem to be addressed translates into building meaningful, higher-level representations of the real-world from incoming raw data that are acquired from cameras and laser scanners and formulating these representations into a domain where reasoning and goal generation takes place (Murphy 2004; Maxwell et al. 2004; Carbone et al. 2008).

A standard set of perception capabilities that need to be embodied into a mobile robot regards simultaneous localization and mapping (SLAM) (Durrant-Whyte and Bailey 2006), and object detection and localization within the map (Murphy et al. 2006), (Gould et al. 2008), (Kjellström, Romero, and Kragic 2011). Furthermore, human-guided operation of the robot can be performed in a natural way using dialogue, wherein the robot receives instructions by the human operator using speech recognition that translates natural language into predefined tasks.

In this paper, we present a mobile robot system that integrates into the Robot Operating System (ROS) (Quigley et al. 2009), multi-modal perception from vision and mapping with action planning. With respect to vision, object detection is performed using several learnable detectors that are rapidly updated while 3D localization is estimated via several object detections that are used by a greedy algorithm based on RANSAC (Hurych, Zimmermann, and Svoboda 2011). This information is used to augment a graph-based representation of the metrical map that is dynamically constructed during exploration using an unsupervised topological segmentation method (Liu, Colas, and Siegwart 2011). The result of this process is a set of nodes that are annotated with properties related to the detected objects wherein connections between the nodes determine the traversability between the corresponding areas. By formulating the perceived knowledge about the environment into a suitable logic representation that is maintained into the knowledge base (Pirri and Reiter 2000; Pirri 2011), the logic-based planner can build a set of tasks, whose goal is communicated by the operator, through dialogue. The planner both verifies the consistency of their executability and monitors its execution reporting possible failures.

Topological Segmentation of Metric Map

The perception of the topology of the environment through mapping is initially represented within a metric layer and in the following as a higher-level topological layer, used for action planning.

Using the sensors of the robot, we first build a metric map using standard SLAM algorithm.[1] From wheel odometry and a 180° 2D laser scanner, we obtain an occupancy grid using Rao-Blackwellized particle filtering (Grisetti, Stachniss, and Burgard 2007).

Based on this occupancy grid, we perform a topological segmentation of the free space in the metric layer. Instead of working at the discretization level of the occupancy grid, we down-sample it to zones of size $1m^2$ that will be referred to as "nodes" in the rest of this section.

We use spectral clustering (Brunskill, Kollar, and Roy 2007) as the segmentation method of the metrical map. The

[1] We use the GMapping software in ROS: http://www.ros.org/wiki/gmapping.

general algorithm of spectral clustering requires the neighborhood graph together with the corresponding *adjacency matrix* W with $n \times n$ elements $W(i,j) = \omega_{ij}$, where n is the number of nodes in the graph. Among the different approaches that have been considered in the definition of the weights ω_{ij} between nodes, in our work, we define it as $\omega_{ij} = e^{-\frac{l(i,j)}{\sigma^2}}$ where $l(i,j)$ is the distance between the centers of nodes i and j. Following the notation of our previous work in (Liu, Colas, and Siegwart 2011), the algorithm of spectral clustering is shown in Alg. 1

Algorithm 1: Spectral Clustering on Topological Segmentation

Input: $W \rightarrow$ Adjacency matrix, where $W(i,j)$ indicates the weight between two nodes i and j;
$k \rightarrow$ The number of clusters-areas;
$T = \{T_1, T_2, \ldots T_k\} \simeq \{1, 2, \ldots, k\}$;
Output: *The list of indices corresponding to the nodes.*
Calculate the normalized graph Laplacian using $L_{sym} := I - D^{-1/2} W D^{-1/2}$ or $L_{rw} := I - D^{-1} W$, where $D = \text{diag}\{d_1, \ldots, d_n\}$ and $d_i = \sum_{j=1}^{n} \omega_{ij}$;
Calculate the k smallest eigenvectors $u_1, \ldots, u_k$ of L (either L_{sym} or L_{rw}), form the matrix $U = [u_1 \ldots u_k] \in \mathbb{R}^{n \times k}$;
Set $\hat{U}$ to be U with rows normalized to the unit L_2 norm;
Use k-means clustering on the rows of $\hat{U}$;
Assign label T_i to cluster j if and only if row j of $\hat{U}$ is assigned to cluster j;

In order to enable the robot to execute plans that remain consistent during the discovery of new, previously unexplored areas, we need to ensure the consistency of the topological map segmentation in time. In other words, a new segmentation of the map should build upon the last instance of segmented map instead of employing a segmentation of the complete map. To enable the spectral clustering method to work incrementally, we set up a simple mechanism as shown in Alg. 2 for each iteration.

Algorithm 2: Incremental Segmentation

Input: $M_t \rightarrow$ Decomposition results of the occupancy grid map;
$Obs_t \rightarrow$ Threshold determining sufficient new observations;
Output: *The updated list of topological regions* T.
Retrieve the newly updated free nodes from M_t, with a total surface area $Area_t$;
if $Area_t < Obs_t$ **then**
  return ;
else
  new_list_of_regions = Do_SpectralClustering(M_t);
  Clean_up();
  Register(T, new_list_of_regions);
end if

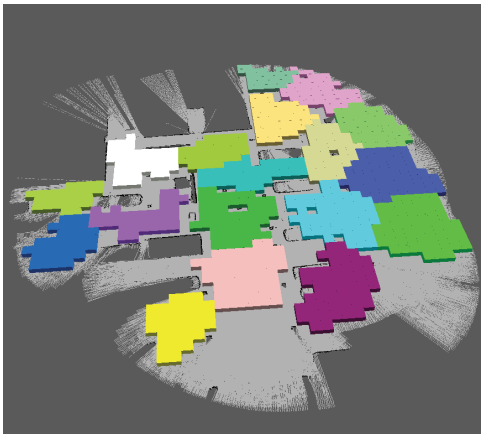

Figure 1: Topological decomposition of a metrical map using the proposed methodology.

In Figure 1 we show an example of topologically segmented metrical map based on the described methodology.

In order to use the topological decomposition of the metrical map for plan generation, we extract the centroid of each area and use it to denote the position of a node within the graph-based representation of the topological map while the edges between the nodes are determined according to the proximity of one node with respect to another based on a distance threshold. The topological segmentation of the metrical map is a continuous process running in parallel with object detection and localization that in turn, augments the domain knowledge stored in the topological graph with incoming information from vision.

Object Detection and Annotation of Topological Graph

Visual perception of the environment by the mobile robot is implemented by object detection and localization using an omnidirectional camera. The information about the detected objects is then introduced into the graph-based representation of the topologically segmented map in the form of graph nodes around the object that are attached onto the graph and by storing information related to the detected object.

Online learnable object detector We use an online learnable object detector (Hurych, Zimmermann, and Svoboda 2011) that is based on an efficient combination of (Lepetit, Lagger, and Fua 2005) and (Kalal, Matas, and Mikolajczyk 2010). There are two key ideas we are using: Kalal et al. (Kalal, Matas, and Mikolajczyk 2010) showed that a tracker allows for efficient boosting of a detector and Lepetit et al. (Lepetit, Lagger, and Fua 2005) showed that multiple detectors using the same set of features can run almost at the same rate as a single detector. In contrast to (Kalal, Matas, and Mikolajczyk 2010), we use several rapidly updated detectors instead of a tracker that use the same features but are updated with different parameters, yielding similar boosting ability as a tracker, while running in real-time.

Category car detector The above described online learnable detector essentially detects *instances* of a certain object. What is crucial is the visual similarity of the instance to the object that has been selected for learning. The individuality of the learnable detector makes general car detection hard. Car surface is usually highly specular while changing illumination influences its appearance significantly. We applied Adaboost detector from OpenCV library (Bradski and Kaehler 2008) for detection of rear car part. For learning we partly used available datasets and partly assembled our own.

3D localization of detected objects Object detectors localize objects of interest in captured images. This information needs to be transformed into the corresponding position in the 3D world so that the robot can infer strategies in approaching or avoiding objects depending on the task.

For object localization in 3D, a new ROS component has been developed. This component depends on the robot pose estimated from odometry and 2D laser scan data, on static transforms which relate the omnidirectional (Ladybug3) camera to the robot base and on internal camera calibration. These transforms are used to resolve directions in which the objects were detected within a particular reference frame. From several detections of a particular object class, a number of objects and their poses are estimated using a greedy algorithm based on RANSAC. In this way, the system tries to interpret all these detections following the minimum description length principle. An example of car detection and localization within the map is shown in Figure 2 while the localization algorithm is given in Alg. 3.

After the 3D localization of an object we associate a set of nodes around the object that correspond to the areas that the robot can reach (left, right, back, front) in order to approach the object. In this way, the topological graph representation of the metric map is enriched with perceptual information coming from vision giving an augmented graph-based representation that the planner is using to generate complex plans.

Figure 2: Example of car detection in one of the LB3 cameras and car models placed in 3D map.

Algorithm 3: 3D localization of objects

Input: $D \rightarrow$ Set of unprocessed object detections consisting of position and direction in common reference frame;
$D_{min} \rightarrow$ minimum number of consistent detections for object to be localized; $i_{max} \rightarrow$ maximum number of iterations;
Output: *Set of detected objects O with resolved position and set of consistent detections.*

```
Initialize set of objects O ← {};
while |D| ≥ D_min do
    Initialize new object (p*, D*) with empty consensus set D* ← {};
    for i ← 1 to i_max do
        Compute position p from two detections D_0 randomly chosen from D;
        Find detections D_1 from D consistent with p;
        Initialize refinement counter j ← 0;
        while |D_{j+1}| > |D_j| do
            Set j ← j + 1;
            Refine position p using all detections D_j;
            Find detections D_{j+1} from D consistent with p;
        end while
        if |D_j| > |D*| then
            Update new object p* ← p, D* ← D_j;
        end if
    end for
    if |D*| ≥ D_min then
        Accept new object O ← O ∪ {(p*, D*)};
        Remove processed detections D ← D \ D*;
    else
        return O;
    end if
end while
```

Dialogue

In this paper, we consider settings in which a human operator is at a remote location, away from the disaster area into which the robot is deployed. The human and the robot interact through an Operator Control Unit (OCU). The OCU is illustrated in Figure 3.

The OCU provides the operator with a visualized map, camera stream, as well as plan and dialogue histories. The OCU facilitates multi-modal interaction: The operator and the robot can use spoken dialogue to interact, and the operator can use pointing gestures (mouse) to select waypoints or landmarks for the robot to go to. A gesture can but need not be accompanied by an utterance. Selecting a waypoint, or saying ”Go here [select waypoint]” have in principle the same interpretation, namely that the operator intends the robot to go to the selected location.

The operator and the robot are working together on a task – namely to jointly explore an environment. This makes the interaction inherently task-driven. The idea of interpreting communicative acts in terms of their underlying intention (operator goal) therefore plays a fundamental role in the OCU design. It enables us to connect interaction to action in

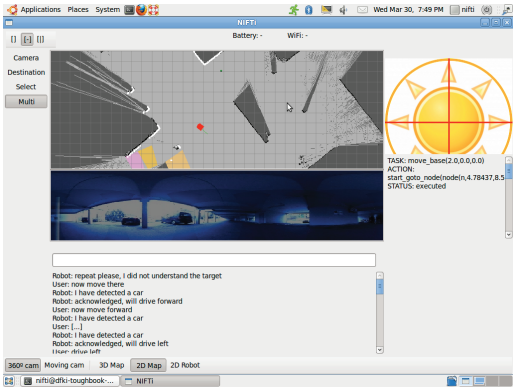

Figure 3: Operator Control Unit, with visualization of map, camera stream, and plan- and dialogue histories

the world, in a way as proposed by the GUI design guidelines of (Goodrich and Olsen 2003), and models of situated dialogue processing like in (Kruijff, Janíček, and Lison 2010). The interpreted intention is grounded in the "world" by resolving the references in the utterances to aspects in the robot's situation awareness that is maintained and updated into the knowledge base of the planner. These references can be referring expressions like "the car" as in "go to the car," but can also be deictic references such as "here" or selected waypoints or detected objects. The resulting representation provides a smooth bridge to planning by stating what (abstract) goal the robot is to achieve, and relative to what locations.

The Logic-based Robot Control

The robot control is endowed with a declarative temporal model of the controllable activities and a planning engine. The structure of the architecture is shown in Figure 4.

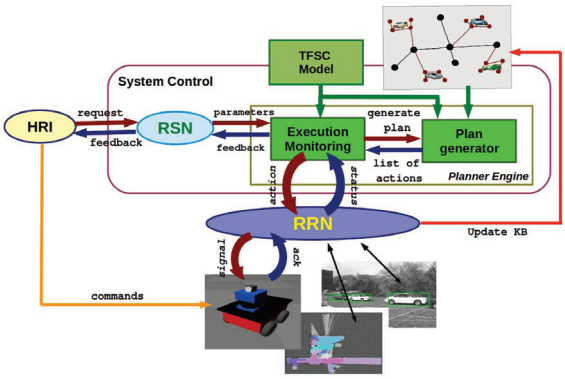

Figure 4: The robot control architecture.

The declarative temporal model is specified in the Temporal Flexible Situation Calculus (TFSC) (Finzi and Pirri 2005) and explicitly represents the main components and processes of the robot system, the cause-effect relationships as well as the temporal constraints among the processes. The TFSC extends the language of a basic theory of actions in the Situation Calculus (Pirri and Reiter 1999; Reiter 2001), combining temporal constraint reasoning and reasoning about actions. It intermediates between Situation Calculus formulae and temporal constraints networks (Dechter, Meiri, and Pearl 1991).

Within this framework, the system is modeled as a set of components specifying their activities over timelines. In the implementation that is presented in this paper, we make use of three components specified as *slam*, *navigation* and *vision* that trigger a set of processes, according to their role. In detail, the *slam* component performs a $toposeg$ process to segment the metric map of the environment, the *navigation* component executes a $goto\_node(nodeId)$ process to reach a target node $nodeId$ and the *vision* component performs a $detect(object)$ process that detects objects in the acquired video stream and localizes them within the map.

These processes are explicitly represented through fluents and instantaneous starting and ending actions which are defined in terms of preconditions and effects. For example, the $toposeg$ process is modeled by the fluent $process(slam, toposeg, s)$ and both the actions $start\_toposeg(t)$ and $end\_toposeg(t)$. The effects are defined by the following successor state axiom:

$$\begin{aligned} process(slam, toposeg, \quad & do(a,s)) \equiv \\ & \exists t\ a = start\_toposeg(t) \vee \\ & process(slam, toposeg, s) \wedge \\ & \neg \exists t' (a = end\_toposeg(t')) \end{aligned}$$

where the action preconditions are:

$$\begin{aligned} Poss(start\_toposeg(t), s) \quad & \equiv idle(slam, s) \wedge \\ & time(s) \leq t \\ Poss(end\_toposeg(t), s) \quad & \equiv process(slam, toposeg, s) \wedge \\ & time(s) \leq t \end{aligned}$$

and the hard time constraints among activities are managed by the TFSC model using Allen-like temporal relations (Allen 1983).

The planning engine is composed of two main logical modules: the plan generator and the execution monitoring. The plan generator relies on a library of Prolog scripts designating the set of tasks which the mobile robot can perform, according to the specified processes, their temporal constraints (compatibilities), and preconditions. For example:

- **Go here**: navigate to a position within the metric map.
- **Move left, right, forward**: execute simple motion commands.
- **Visit graph**: visit all the nodes of the graph-based representation of the metrical map.
- **Visit node**: visit a specific node of the graph-based representation of the metrical map, following the optimal path within the graph according to a selected criterion (e.g. the distance between the nodes).
- **Approach object**: visit a node (left, right, front or back) around a detected object.

These tasks are based on the perceived information of the environment using *vision* and *slam*, which is represented in the knowledge of the planning engine in the form of a graph (see Fig. 5). The nodes of the graph correspond to topologically segmented regions or approachable areas around a detected object and edges determine the traversability between the regions.

The execution-monitoring is a continuous process which ensures that the set of action sequences, generated by the plan generator according to the TFSC model and the current state of the domain knowledge, are consistently executed. Concurrently, at regular time intervals, the execution-monitoring reads the system state and monitors the execution of the activities, in order to detect system malfunctioning that may result in action failures (De Giacomo, Reiter, and Soutchanski 1998; Pettersson 2005).

Both the TFSC model and the planning engine are implemented in ECLIPSE Prolog (Apt and Wallace 2006) which optimally combines the power of a constraint solver (for the time and compatibility constraints) with inference in order to generate the set of action sequences, and also enable the continuous update due to incoming new knowledge (using finite progression).

The planner embedded into the ROS architecture The declarative temporal model and the planning engine are fully embedded into ROS. The underlying implementation involves the following ROS nodes:

1. A ROS *Service node* (RSN);
2. A ROS *Robot node* (RRN).

The RSN allows for RPC request-reply interactions with the user interfaces while the RRN takes care of the communication between the perceptual and physical robot components and the planning engine.

The RSN manages the communication with the human interfaces and embeds into the ROS language the logical part of the control system. It enables the human operator to interact with the control system during the computational cycle. In fact, the operator can interact with the system by posting goals, and he/she can also interrupt the execution of a plan, generated by the planning engine, or directly control the perceptual and physical robot components.

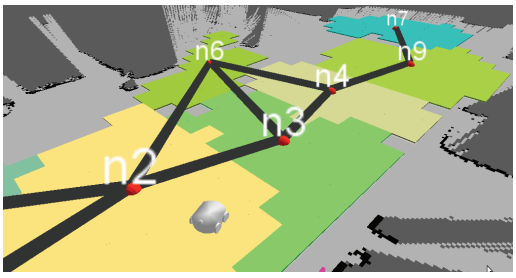

Figure 5: An instance of topological map segmentation together with the corresponding node centers and connections within the nodes that is used by the robot to generate plans.

On the other hand the RRN receives the information from the different perceptual modalities (mapping and vision) in order to build the domain knowledge of the robot. The RRN is also responsible for sending task activation signals to the robot components in order to perform the sequence of actions generated by the planner, according to the operator requests.

More specifically, the RRN takes as input the segmented metric map and creates a logical graph-based representation of the explored area. Nodes and edges of the graph are specified by predicates together with additional information about the edges (capacity, distance) and the nodes (surface area, convexity). According to this representation the plan generator is able to infer shortest paths along the graph, inducing from this a plan, whenever a node-to-node navigation task is requested by the operator.

Similarly, the visual perception provides the RRN with information about detected objects. Whenever an object is detected and localized within the metric map, a new set of nodes is added to the topological graph, circumscribing the object, and the corresponding properties are specified in the action theory. The orientation of the object can be used to determine the correspondence of these nodes with respect to the areas at the front, back, left and right of the object. This allows the plan generator to make a plan, to suitably approach the detected object, and also it enables the model-based planner to reason about the augmented logical structure.

The RSN receives a task request from the user, creates a logical representation of the request and sends the corresponding Prolog term to the execution-monitoring. The execution-monitoring asks the planner to generate an executable set of action sequences, according to the current state of the knowledge base. Once the plan is generated, the execution-monitoring sends these actions for execution to the RRN. Upon receipt of an action, the RRN takes the control managing the physical execution of the action. When an action is performed, the execution-monitoring retrieves the state of the robot, whence it verifies whether the action was successful or not. In the former case, the execution-monitoring sends the next action to the RRN, otherwise it aborts the execution of the remaining plan and notifies the failure to the RSN. In turn, the RSN returns the result of the execution of the action to the interface in so yielding the control to the operator.

Conclusion

In this paper, we have presented a mobile robot system that employs high-level control in order to operate in a real-world setting where the main task is human-assisted exploration of an environment. In the presented system, we have integrated multi-modal perception from vision and mapping with a model-based executive control. We have also showed how the system allows the interaction between the human operator and the robot platform via the dialogue-based communication. In this framework, action planning is performed using a high-level representation of the environment that is obtained through topological segmentation of the metric map and object detection and 3D localization in the map.

This representation has the form of a graph where all the information related to the spatial characteristics of the environment is stored into properties that are annotated to the nodes and the edges of the graph that is used by the planner to generate task-dependent plans. The control system monitors the execution of the action sequences and communicates the status through the dialogue.

Acknowledgement

This paper describes research done under the EU-FP7 ICT 247870 NIFTI project.

References

Allen, J. F. 1983. Maintaining knowledge about temporal intervals. *Commun. ACM* 26(11):832–843.

Apt, K. R., and Wallace, M. 2006. *Constraint Logic Programming using Eclipse*. Cambridge University Press.

Bradski, G., and Kaehler, A. 2008. *Learning OpenCV: Computer Vision with the OpenCV Library*. O'Reilly.

Brunskill, E.; Kollar, T.; and Roy, N. 2007. Topological mapping using spectral clustering and classification. In *IEEE International Conference on Intelligent Robots and Systems*, 3491–3496.

Carbone, A.; Finzi, A.; Orlandini, A.; and Pirri, F. 2008. Model-based control architecture for attentive robots in rescue scenarios. *Autonomous Robots* 24:87–120. 10.1007/s10514-007-9055-6.

De Giacomo, G.; Reiter, R.; and Soutchanski, M. 1998. Execution monitoring of high-level robot programs. In *In Proceedings of the 6th International Conference on Principles of Knowledge Representation and Reasoning*.

Dechter, R.; Meiri, I.; and Pearl, J. 1991. Temporal constraint networks. *Artif. Intell.* 49(1-3):61–95.

Durrant-Whyte, H., and Bailey, T. 2006. Simultaneous localization and mapping: part i. *Robotics Automation Magazine, IEEE* 13(2):99–110.

Finzi, A., and Pirri, F. 2005. Representing flexible temporal behaviours in the situation calculus. In *In Proceedings of IJCAI*.

Goodrich, M. A., and Olsen, D. R. 2003. Seven principles of efficient interaction. In *Proceedings of IEEE International Conference on Systems, Man, and Cybernetics*, 3943–3948.

Gould, S.; Baumstarck, P.; Quigley, M.; Ng, A.; and Koller, D. 2008. Integrating visual and range data for robotic object detection. In *ECCV Workshop on Multi-camera and Multi-modal Sensor Fusion Algorithms and Applications - M2SFA2 2008*.

Grisetti, G.; Stachniss, C.; and Burgard, W. 2007. Improved techniques for grid mapping with Rao-Blackwellized particle filters. *IEEE Transactions on Robotics* 23(1):34–46.

Hurych, D.; Zimmermann, K.; and Svoboda, T. 2011. Fast learnable object tracking and detection in high-resolution omnidirectional images. In *International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications*.

Kalal, Z.; Matas, J.; and Mikolajczyk, K. 2010. P-N Learning: Bootstrapping Binary Classifiers by Structural Constraints. *Conference on Computer Vision and Pattern Recognition*.

Kjellström, H.; Romero, J.; and Kragic, D. 2011. Visual object-action recognition: Inferring object affordances from human demonstration. *Computer Vision and Image Understanding* 115(1):81–90.

Kruijff, G.; Janíček, M.; and Lison, P. 2010. Continual processing of situated dialogue in human-robot collaborative activities. In *Proceedings of the 19th IEEE International Symposium in Robot and Human Interactive Communication*. IEEE.

Lepetit, V.; Lagger, P.; and Fua, P. 2005. Randomized trees for real-time keypoint recognition. In *Computer Vision and Pattern Recognition*, 775–781.

Liu, M.; Colas, F.; and Siegwart, R. 2011. Regional topological segmentation based on Mutual Information Graphs. In *Proc. of the IEEE International Conference on Robotics and Automation (ICRA 2011)*.

Maxwell, B.; Smart, W.; Jacoff, A.; Casper, J.; Weiss, B.; Scholtz, J.; Yanco, H.; Micire, M.; Stroupe, A.; Stormont, D.; and Lauwers, T. 2004. 2003 aaai robot competition and exhibition. *AI Magazine* 25(2):68–80.

Murphy, K.; Torralba, A.; Eaton, D.; and Freeman, W. 2006. Object detection and localization using local and global features. In *Toward Category-Level Object Recognition*, volume 4170 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg. 382–400.

Murphy, R. 2004. Human-robot interaction in rescue robotics. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on* 34(2):138–153.

Muscettola, N.; Dorais, G. A.; Fry, C.; Levinson, R.; and Plaunt, C. 2002. Idea: Planning at the core of autonomous reactive agents.

Pettersson, O. 2005. Execution monitoring in robotics: A survey. *Robotics and Autonomous Systems* 53(2):73–88.

Pirri, F., and Reiter, R. 1999. Some contributions to the metatheory of the situation calculus. *J. ACM* 46:325–361.

Pirri, F., and Reiter, R. 2000. *Planning with natural actions in the situation calculus*. Kluwer Academic Publishers. 213–231.

Pirri, F. 2011. The well-designed logical robot: Learning and experience from observations to the situation calculus. *Artificial Intelligence* 175(1):378–415. John McCarthy's Legacy.

Quigley, M.; Conley, K.; Gerkey, B.; Faust, J.; Foote, T.; Leibs, J.; Wheeler, R.; and Ng, A. 2009. Ros: an open-source robot operating system. In *ICRA Workshop on Open Source Software*.

Reiter, R. 2001. *Knowledge in action : logical foundations for specifying and implementing dynamical systems*. MIT Press.