Graph Grammars for Active Perception

Luis J. Manso, Pablo Bustos, Pilar Bachiller and Marco A. Gutierrez
University of Extremadura,
Cáceres, Extremadura.
lmanso@unex.es

***Abstract*—The complexity of the applications in which robots are being used does not stop growing. Different solutions such as sophisticated control architectures have been proposed in order to deal with complexity in robot control. These solutions make robotic systems more robust, scalable and easier to distribute, understand and monitor. However, it is still not clear how to cope with the complexity of the interaction dynamics that underlie the perception of the environment. With this issue in mind this paper presents the concept of *cognitive graph grammar* and two algorithms that make use of it. Cognitive graph grammars are a grammar-based theoretical framework designed to support cognitive perception and, especially, the active nature of perception. They provide a means to describe how graph-based models can be generated and the behaviors to execute depending on the perceptual context. This is done in such a way that the information provided using this formalism can be used for different perceptive purposes at the same time, such as to link action and perception or to diminish perceptive errors. The paper also describes an experiment in which a cognitive graph grammar is used in an autonomous robot in order to efficiently model an environment made of rectangular rooms with obstacles.**

I. Introduction

It would be desirable to have robots able to interact with non-trivial entities (e.g., compound objects, people). In order to perform these tasks robots have to perceive and model their environment to some extent. Unlike in the earliest experiments of robotics, the floor is not necessarily restricted to textureless shadow-free surfaces anymore, and objects are not necessarily simple perfectly shaped boxes. Nevertheless, the environments in which robots operate are not random. In order to build actually intelligent robots, a priori knowledge about the environment must be properly used.

The complexity that roboticists have to face when developing autonomous robotic systems has been successfully handled from different points of view. Control architectures such as those in [1], [2] suggest how to organize control and information flows. Technologies such as [3], [4] handle implementation issues from a software engineering point of view using component-oriented programming. However, none of these approaches can be directly used as a tool to support perception or to ease the binding of perception and action. The control logic of active perception algorithms still tends to be formed by hard-coded if-then-else constructs that map robot proprioception and its world model to specific perceptual states and actions, which is error-prone. Moreover, they rarely take into account context information, which is useful to produce robust and coherent environment interpretations. Thus, despite the use of the previously mentioned technologies makes robotic systems better designed and easier to manage, the complexity of the control logic associated with the perception of the environment is hardly reduced. This paper presents *cognitive graph grammars*, a theoretical framework that helps roboticists building context-aware active perception systems.

When a robot builds symbolic models of its surroundings it generally does so by recording the perceived environment elements and their relationships. Since this can be seen as the generation of a graph where nodes represent the modeled symbols and edges represent the relationships between them, it is interesting to formally describe how the robot might do that. Precisely, graph grammars describe the rules governing the formation of graphs with a specific structure. Graph grammars generalize the concept of string grammars so that productions can also be applied to graphs. In fact, strings can be seen as undirected graphs such that all nodes –characters–, except those at sentence endings, have an edge linking them to each of their adjacent characters. Thus, graph grammars extend string grammars in order to support input data with arbitrarily complex connection patterns.

This paper introduces the concept of cognitive graph grammar (CGG), a graph grammar-based formalism designed to support symbolic active perception. They provide a means to give raise to different active-perceptual mechanisms by describing how graph-based models can be generated. Building on this formalism, additional algorithms are provided so that descriptions based on CGG can be used for different purposes, such as linking action and perception or improving perception robustness. The paper also provides a proof-of-concept experiment in which a CGG is used to model an environment made of rectangular rooms with obstacles.

The remaining of the paper is organized as follows. Section II reviews previous work on graph grammars and active perception. The core of the paper is found in section III. It provides an introduction to the most widely used graph grammar formalisms, describes the limitations that make necessary a new one, and details the concept of cognitive graph grammars both from a formal and practical point of view. It also describes the different benefits of using CGGs and the perceptual phenomena that can arise when using them. Section IV describes an example of a CGG used in order to perform topological mapping along with the experimental results obtained using it. Finally, section V presents the conclusions and future work.

II. Previous Work

Besides active perception, grammars have been used in robotics and artificial vision for a wide range of applications. An algorithm for graph verification (i.e., given a graph and a graph grammar, check if the graph can be generated using the grammar) is proposed in [6]. In this work, only the vertices of the graphs can be labeled. Graph grammars are used in [7] to achieve self-configuring adaptable software architectures. In this work graphs are of fixed order. A similar approach for coordinating multi-robot systems where robots are represented by graph vertices is proposed in [8]. The graph, which is shared by all the robots of the system, is also of fixed order. Coordination is achieved by modifying the linking pattern and the label (i.e., role) of the robots.

A series of works by the same group is presented in [9], [10] and [11]. In [9] and [11], an attributed graph grammar is designed in order to parse rectangle layouts from images of man-made scenes. The authors consider rectangles as terminal symbols and layouts as production rules. Bottom-up and top-down mechanisms are used in order to improve rectangle detection and parsing. Since different possible models can explain input images, the algorithm chooses the one maximizing the posterior probability or minimizing a descriptor length. A similar approach is used for segmenting and recognizing generic scenes in [10]. A similar approach to the one described in [11] is used in [12] in order to represent and recognize objects. In this case, both the set of primitives and production rules are wider, but the foundations are the same. These approaches have two main differences from what is proposed in this paper: a) they are based on string grammars, reducing what can be solved using their approach; b) they use static input data, which is a very hard restriction for robotics (action is not taken into account).

Graphs are used for task planning in [13]. It also covers how plans can be dynamically modified as sub-tasks are accomplished or conditions change. Grammar rules are proposed in order to modify the current plan.

Spatial Random Tree Grammars are proposed in [14] for image parsing. They are context-free grammars with at most two children in which rules are labeled with information for determining the spatial distribution of their nodes (i.e., vertically or horizontally distributed). While in string grammars this is not necessary (i.e., productions are always horizontally distributed), it guarantees the unambiguous interpretation of parse trees from graph grammars. In this work, a probability distribution is also associated to production rules so the probability of a specific parse can be estimated.

III. Cognitive Graph Grammars

Unlike graph grammars, string grammars do not generally provide enough expressive power to be used in order to build environmental representations. On the other hand, string grammars unambiguously describe how the production rules can be applied. This is because in these grammars two symbols are connected if and only if they are in contiguous positions within the sentence. When a string symbol pattern p_1 is replaced by p_2 it is automatically linked to –and only to– the pattern on the left and right sides of p_1, in the same order. Depending on the formalism used, this assumption does not hold for graph grammars since the connection pattern is not fixed or restricted to a specific order. Additionally, none of the previous graph grammar formalisms were conceived with robots or active perception in mind. Different graph grammar formalisms have been proposed aiming to remove any ambiguity in the connection patterns of the nodes involved [5]. However, they do so by assuming a specific behavior: some of them assume that dangling edges should be automatically removed; others assume that dangling edges prevent productions from being applied. In order to avoid ambiguity without limiting what can be expressed by graph grammars they must also allow to specify negative subpatterns (i.e., parts of the patterns that should not exist in order to apply the rule). Since no formalism provides mechanisms to remove these restrictions, it was found necessary to create a more flexible formalism.

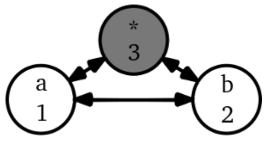


Fig. 1. Example of how the 'avoid' decoration and the wildcard symbol can be used to dismiss any pattern not specifically considered on the right-hand side of grammar rules.

A. CGG formalism

The cognitive graph grammar formalism follows the same principles as single-pushout [5] but, in order to overcome the previously mentioned limitations three modifications are proposed: **a)** the elements of the left hand side of the rules can be decorated with an *avoid* attribute in order to dismiss those matches containing elements decorated in such way (expressed by filling them with gray color); **b)** productions may be accompanied with first-order logic sentences specifying *conditions* and the *operations* performed by the rule if the desired behavior differs from the one of single-pushout; **c)** CGGs have the '$*$' NT symbol, which matches any other symbol. Figure 1 and table I(e) are examples of the use of the avoid attribute. From a formal point of view, cognitive graph grammars are defined as a seven tuple $(N, T, P, B, A_V, A_E, PB)$ such that:

- N and T are the mutually exclusive non-terminal and terminal alphabets, respectively. In particular, N must contain the start and wildcard symbols (S and $*$).
- P is the set of production rules.
- B is the set of possible perceptive behaviors.
- A_V and A_E are the attribute alphabet for vertices and edges, respectively.
- PB is a function mapping P to B.

This formalism allows to unambiguously specify a graph grammar without assuming any specific behavior, as well as to relate behaviors to grammar rules.

B. CGGs properties

As seen in section II, grammars can be used as a tool to support perception. This section describes how to achieve different grammar-based perception-oriented techniques using CGGs. To the knowledge of the authors, all published grammar-based perception techniques can be classified in one of the following types: a) bottom-up parsing, b) model verification, c) context-aware perception restrictions, and d) covert perception. In addition to these, CGGs can also be used to associate perception and action. The remaining of the section describes how to achieve each of these phenomena using CGGs.

Bottom-up parsing is the most common application of grammars: given a sample from the input space, it is parsed in order to recognize its structure. For some applications it might only be necessary to perform **model verification**. It can be seen as a bottom-up parsing where the parse result is ignored, only providing whether or not there was any result.

It is also desirable that robot perception would dynamically adapt to the scenario in which robots are located and their conditions. Here we distinguish between a) passively adapting to the environment by restricting what might be perceived, **context-aware restrictions**; and b) high perceptual layers actively providing top-down information to the bottom perceptual layers in order to influence its output, **covert perception** [16].

Additionally, we propose using CGGs to select the appropriate perception strategy or behavior according to the context. The subset of rules that can be potentially triggered next can be computed by analyzing the grammar rules and the current model. Thus, by associating behaviors to rules, the compatible behavior set is computed as the set of behaviors associated to the rules that can be potentially triggered. It is worth noting that all previous work regarding the use of grammars for perception were applied to static images. The remaining of the section elaborates how these perceptual techniques can be implemented using CGGs.

1) Bottom-up parsing: Regular parsing algorithms are designed to work with static and complete input data. Robot perception generally entails movements that allow robots to sense different parts of the environment. Since these movements change the input data, the standard approaches can not be used to parse it. The process of bottom-up parsing in CGGs is performed by running the rules that are compliant with the current model as long as the terminal symbols they introduce are actually being perceived. When multiple possible rules can be triggered at the same time, the approach presented in [14] can be used in order to provide the most probable parse.

2) Context-aware restrictions on perception: A classic example of this kind of phenomena in humans can be found in [15]: the same visual input can be perceived as different objects depending on the context. Graph grammars express how graphs (symbolic models in our case) can be built. By doing so they also describe how they can not be built, thus providing the power to support context-aware perception. By restricting which world elements can be perceived at each moment according to the limitations of the grammar, the number of false positives (i.e., misrecognized world elements) can be reduced. In order to illustrate this, a simple two-rule grammar example is provided in equation 1. The formalism is not used in this case because this specific property can be provided by all kind of grammars, not just CGGs.

$$\begin{aligned} S &\Longrightarrow arm \cdot A \\ A &\Longrightarrow forearm \end{aligned} \tag{1}$$

If a robot using this grammar is certain that it perceived the arm of a person (so its current model is $'arm \cdot A'$) it can unquestionably discard any other arms. A pseudo-code implementation is shown in algorithm 1. For every potentially applicable rule, it computes the set of non-terminal symbols appearing only on the RHS (not on the LHS) and returns the union of those sets. This set contains the world elements that can be perceived given the grammar and the current model.

Algorithm 1 Contex-aware restrictions algorithm:

```
Require: h: Input graph
Require: G: CGG such that $G = (N, T, P, B, A_V, A_E, PB)$
 1: $U \leftarrow \emptyset$
 2: forall $p = (lhs, rhs) \in P$ do
 3:     if $applicable(p, h)$ then
 4:         forall $s \in (rhs.V - lhs.V)$ do
 5:             if $terminalSymbol(s)$ then
 6:                 $U \leftarrow U \cup s$
 7:             end if
 8:         end forall
 9:     end if
10: end forall
11: return $U$
```

3) Covert perception: The information provided by grammars can also be used to influence bottom-up perception, not just to filter its output. Thus, grammars are also a valid framework to enable covert perception [16]. A priori knowledge of the world can be used to influence bottom-up perception by enforcing or inhibiting the detection of specific parts of the environment. Thus, grammars can not only reduce false positives in object detection but also reduce false negatives. The grammar rule described in table I(b), is a good example of this. It is further explained in section IV. When parts of the object to detect are occluded, bottom-up object detectors tend to decrease their effectiveness dramatically. The partial pattern detected can be used to compute the probability of a false negative. This can be expressed using the Bayes theorem as in equation 2. In the equation, F stands for the event "a not detected entity should be forced into the model", and C stands for the event of the robot having a specific context (potentially partial input).

$$p(F|C) = \frac{p(C|F)p(F)}{P(C)} \tag{2}$$

This is one of the most interesting applications of graph grammars. Examples of this type of technique can be found in [9], [10], [11] or [12].

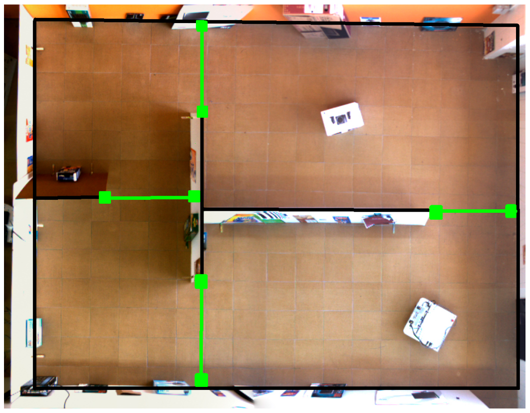


Fig. 2. Overhead view of the environment used for the experiments. It is composed of four rectangular rooms connected in a loop. The robot is the squared object on the right-bottom part of the image. Objects are placed in the environment in order to prove the robustness of the approach.

4) Action selection: Generally, given the grammar and the current graph, only a subset of rules can be potentially applied. Since CGG rules have associated behaviors, computing the subset of potentially applicable rules is equivalent to compute the candidate set of perceptual actions. It can be used to enable robots to decide what to do next. Depending on the grammar and the context, the subset can be formed by several or a single action. Algorithm 2 details how to compute the action set.

Algorithm 2 Action selection algorithm:

```
Require: h: Input graph
Require: G: CGG such that $G = (N, T, P, B, A_V, A_E, PB)$
1: $U \leftarrow \emptyset$
2: forall $p \in P$ do
3:     if $applicable(p, h)$ then
4:         $U \leftarrow U \cup PB(P)$
5:     end if
6: end forall
7: return $U$
```

The novelty of the underlying idea of CGGs in this respect is not the well-known idea of associating behaviors to robot states, but to do it by defining the perceptive state of robots as their set of potentially applicable rules. Moreover, algorithm 2 can be easily extended to only take into account those rules which are of interest depending on the robot goal.

IV. Experiment

In order to illustrate the usage of CGGs, this section provides a real example of a grammar used in a robot that models its environment. In particular, the objective of this grammar is to enable a robot to model a simple world made of rectangular connected rooms in which obstacles can be found. Section IV-B describes the benefits obtained from the use of CGGs. Section IV-C provides experimental results obtained using this grammar.

A. *Grammar definition*

1) Grammar alphabets: The first step is to define the entities that the world model will be composed of. This is an arbitrary decision: it is up to the roboticist to decide which symbols to use. It will depend on the environment or object to model, and the desired level of detail. For this experiment it was decided to use symbols for rooms, doors and obstacles.

r	Used for rooms.
d	Used for doors. Doors will link two different rooms.
o	For obstacles. Obstacles will be located within rooms.

It is also necessary to define the attributes that the symbols will have. In particular, rooms, doors and obstacles have their position (x and y) and their dimensions ($width$ and $length$). Moreover, rooms also have an *active* attribute that is true for the room in which the robot is located and false otherwise. As can be seen in table I, attributes can be used in order to enable or disable the application of the different rules. The resulting alphabets are shown in equation number 3.

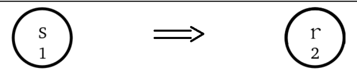
$$\begin{aligned} N &= \{S, *\} \\ T &= \{r, d, o\} \\ A_V &= \{width, length, x, y, active\} \\ A_E &= \emptyset \end{aligned} \tag{3}$$

2) Grammar rules: Once the entities to model are known, the next step is to write the rules that will guide the perception process. In most cases this is a cyclic task (i.e., the symbols may depend on the rules and vice versa), and sometimes it will be necessary to introduce new non-terminal symbols.

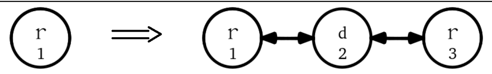
Rule 0 specifies that the start symbol can be transformed into a room. This is the only rule with the start symbol (S) in its left hand side, so the start symbol *can only* be transformed into a room. This implies that, when the perception process starts, the first task robots have to perform is to model the room in which they are located. Rule 1 describes how the discovery of new rooms transforms the model. The rule creates a new room symbol and makes it adjacent to the room in which the robot is located. It is triggered when the robot perceives or goes through a door when there is only one room in the model or when there are no close rooms to perform loop-closing with. Rule number 2 is similar to the rule number 1, but applicable to objects instead of rooms. It links obstacles to rooms, so it is used when the robot perceives new obstacles. Rule 3 is used for loop closing, when the robot realizes that two rooms previously modeled as disconnected are actually adjacent. In this case, a new door is included in the model without including a new room. Rule 4 is also used for loop closing. It describes how the event of finding out that two rooms that were thought to be different are actually the same would affect the graph. This happens when the robot closes a loop without recognizing that the new room it entered is already known. Both rooms collapse and all the ingoing and outgoing links are redirected from r_1 and r_2 to the new room r_3. The formal descriptions of the rules are shown in table I. The resulting P and B sets (containing rules and behaviors)

TABLE I
EXAMPLE RULE 0

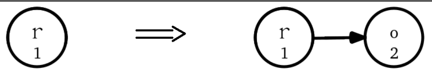
(a) Rule 0

	
Conditions:	
Operations:	$[r_2.active \leftarrow true]$
Behavior:	"explore room".

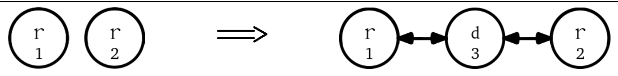
(b) Rule 1

	
Conditions:	$[r_1.active = true]$
Operations:	$[r_1.active \leftarrow false]$; $[r_3.active \leftarrow true]$
Behavior:	"explore room".

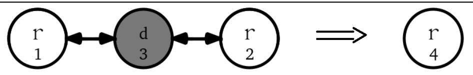
(c) Rule 2

	
Conditions:	$[r_1.active = true]$
Operations:	
Behavior:	"find new room".

(d) Rule3

	
Conditions:	$[r_1.active = true]$
Operations:	
Behavior:	"find new room".

(e) Rule4

	
Conditions:	$[r_1.active = true]$
Operations:	$[r_1.active \leftarrow false]$; $[r_4.active \leftarrow true]$; $[\forall a \in (A_{r_1} \cup A_{r_2})\ A_{r_4} \leftarrow A_{r_4} \cup a]$
Behavior:	"explore room".

are shown in equations 4 and 5. The PB mapping associating a behavior to each of the rules, is shown in equation 6.

$$P = \{Rule0, Rule1, Rule2, Rule3, Rule4\} \tag{4}$$

$$B = \{"explore\ room", "find\ new\ room"\} \tag{5}$$

$$PB = \begin{cases} Rule0 & \Longrightarrow & "explore\ room" \\ Rule1 & \Longrightarrow & "explore\ room" \\ Rule2 & \Longrightarrow & "find\ new\ room" \\ Rule3 & \Longrightarrow & "find\ new\ room" \\ Rule4 & \Longrightarrow & "explore\ room" \end{cases} \tag{6}$$

B. Implications of the use of CGGs

In order to implement the described grammar in a robot it is necessary to have a detector for each of the elements the model will be composed of (e.g., rooms, obstacles). The role of the bottom-up part is to detect atomic world elements and provide them to higher perceptive levels. Top-down might influence or force the detection of parts of the environment. In the case of the previously described grammar, this happens when the robot finds a door on a single room model or there are no rooms to close the loop with: since, in those cases, rule 1 is the only compatible rule introducing a door in the model, the only interpretation is that at the other side of the door there is a new room not previously seen. The grammar-based restrictions imposed by the available productions make the robot ignore impossible or useless signals. For example, the grammar disables the robot to close the loop with an adjacent room in the way that rule 4 does. Another example is that, when the system is started, obstacles, doors and other adjacent rooms are automatically ignored until the robot models the room in which it is located. This situation would be harder to specify and understand using regular programming languages, and would increase the probability of errors.

CGGs provide a means to select the most appropriate robot behavior. In the situation of the last example, the world model would be composed of a single node with the start symbol (S) on it. In this scenario, the only applicable rule is the one that substitutes S with a room node (rule number 0). Thus, the only acceptable behavior would be "explore room". Once the room is detected, the robot will not be able to trigger rule 0 anymore. CGGs enable robots to compute the set of potential actions compliant with the model and the rules. However, since the set might contain several choices, CGGs can not always decide what the robot should do. It is possible that the robot selects a non-achievable behavior, for example, if it tries to find a new room when there are no more rooms. Moreover, in more complex scenarios there may be different potentially applicable rules (potentially interesting actions). These two situations make necessary an additional planning or homeostatic algorithm in order to reach a final decision.

C. Experimental Results

The grammar described in section IV-A has been implemented in an autonomous robot. Figure 2 shows an overhead view of the environment. The experiment starts with a world model containing only the start symbol. According to the behavior selector, the robot adopts the "explore room" behavior, expecting to run rule 0. Figure 3 shows the metric reconstruction of the map after the behavior models the room. Once the robot accomplishes the task, the robot could potentially apply rules 1 and 2. Thus, it interleaves their associated behaviors in order to discover the next room. By repeating this process (computing the candidate behaviors and interleaving them) the robot gets to the point in which all rooms and obstacles are modeled. This is shown in figure 4. Figure 5 shows the final graph model. As it can be depicted

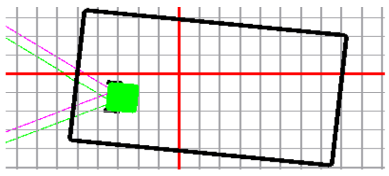


Fig. 3. World model after the robot entered and modeled the initial room.

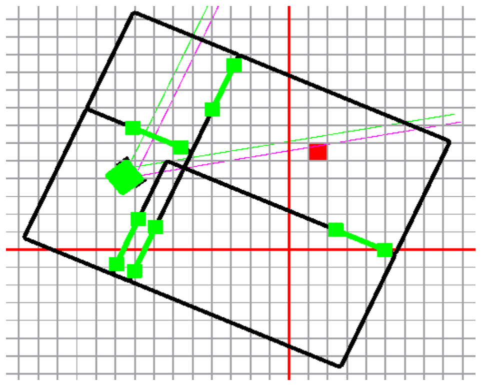


Fig. 4. Metric reconstruction from the topological model after the robot visited all the rooms.

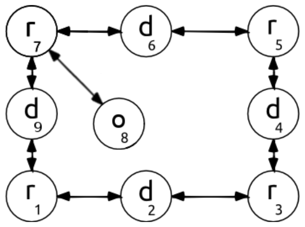


Fig. 5. Structure of the graph after the robot visited all the rooms.

in figure 4, the metric reconstruction shows overlapping errors caused by odometry and sensor uncertainty. These errors are canceled by performing a stochastic gradient descent search in a parameter space defined by the size of the rooms and the positions of the doors (see [17]). The minimization is weighted by the uncertainty of the models, as in standard non-linear graph optimization. This improves the overall result and allows the robot to perform robust loop-closings. The final metric reconstruction is shown in figure 6.

V. Conclusions

This paper presented the concept of cognitive graph grammars, the first grammar-based approach designed for active perception. CGGs provide a means for linking perception and action in order to gather the necessary information robots might need in a robust and context-aware way. At the same time they have interesting top-down properties and perceptive restrictions that help avoiding perception errors (both false negatives and positives). Section IV-C presented a proof of concept experiment, proving CGGs to be a useful and promising approach. We are currently experimenting with more complex grammars and studying methods to automatically infer formal knowledge from grammars. Efforts towards creating a tool in order to automatically translate CGG specifications into regular programming languages using a Domain Specific Language for CGGs are also being made.

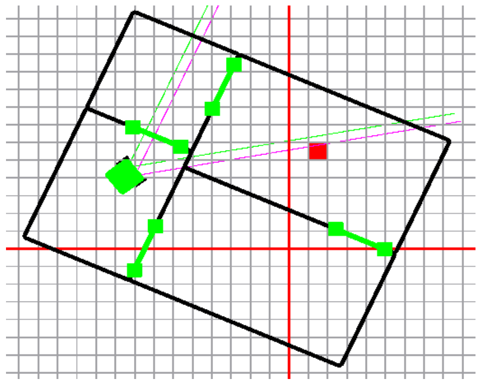


Fig. 6. Final metric reconstruction, after the optimization process.

Acknowledgment

This work has been supported by project IPT430000-2010-2 funded by Spanish Ministry of Science and Innovation, and by project IB10062 of the Extremaduran Government.

References

[1] E. Gat, "On three-layer architectures", in *Artificial Intelligence and Mobile Robots*, p. 195–210, 1998.

[2] M.N. Nicolescu and M.J. Matarić, "A hierarchical architecture for behavior-based robots", in *Proc. of Int. Joint Conf. on Autonomous Agents and Multi-Agent Systems*, pp. 227–233, 2002.

[3] L.J. Manso, P. Bachiller, P. Bustos, P. Nunez, R. Cintas, and L. Calderita, "RoboComp: a tool-based robotics framework", in *Proc. of Conf. on Simulation, Modeling and Programming for Autonomous Robots (SIMPAR)*, pp. 251–262, 2010.

[4] M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Ng, "ROS: an open-source robot operating system", in *Proc. of ICRA Workshop on Open Source Software*, 2009.

[5] R. Heckel, "Graph transformation in a nutshell", *Electronic Notes in Theoretical Computer Science*, vol. 148, no. 1, pp. 187–198, 2006.

[6] J. Bauer and R. Wilhelm, "Abstract Interpretation of Graph Grammars", in *Simulation and Verification of Dynamic Systems*, 2006.

[7] I. Bousassida, C. Chassor, and M. Jmaiel, "Graph grammar-based transformation for context-aware architectures supporting group communication", *Nouvelles Technologies de l'Information*, vol. L, no. 19, pp. 29–42, 2010.

[8] B. Smith, A. Howard, J.M. McNew, J. Wang, and M. Egerstedt, "Multi-robot deployment and coordination with embedded graph grammars", *Journal of Autonomous Robots*, vol. 26, no. 1, pp. 77–98, 2009.

[9] S.C. Zhu, R. Zhang, and Z. Tu, "Integrating bottom-up/top-down for object recognition by data driven markov chain monte carlo", in *Proc. of Conf. in Computer Vision and Pattern Recognition*, vol. 1, pp. 738–745, 2000.

[10] Z. Tu, X. Chen, A.L. Yuille, and S.C. Zhu, "Image parsing: Unifying segmentation, detection and recognition", *Journal of Computer Vision*, vol. 63, no. 2, pp. 113–140, 2005.

[11] F. Han and S.C. Zhu, "Bottom-up/top-down image parsing with attribute grammar", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 1, pp. 59-73, 2009.

[12] L. Lin, T. Wu, J. Porway, and Z. Xu, "A stochastic graph grammar for compositional object representation and recognition", *Journal of Pattern Recognition*, vol. 42, no. 7, pp. 1297–1307, 2009.

[13] J.M. Hasemann, "A robot control architecture based on graph grammars and fuzzy logic", in *Proc. of Conf. on Intelligent Robots and Systems*, vol. 3, pp. 2123–2130, 1994.

[14] J.M. Siskind, J.J. Sherman, I. Pollak, M.P. Harper, and C.A. Bouman, "Spatial random tree grammars for modeling hierarchal structure in images with regions of arbitrary shape", *Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 9, pp. 1504–1519, 2007.

[15] A. Torralba, K.P. Murphy, and W.T. Freeman, "Using the forest to see the trees: exploiting context for visual object detection and localization", *Communications of the ACM*, vol. 53, no. 3, pp. 107–114, 2010.

[16] H. Svensson, A. Morse, and T. Ziemke, "Neural pathways of embodied simulation", in *Anticipatory Behavior in Adaptive Learning Systems*, pp. 95–114, 2009.

[17] P. Bachiller, P. Bustos, and L.J. Manso, "Attentional Behaviors for Environment Modeling by a Mobile Robot", in *Stereo Vision*, pp. 17–40, InTech, 2011.