2017 IEEE International Conference on
Autonomous Robot Systems and Competitions (ICARSC)
April 26-28, Coimbra, Portugal

A Case Study on Automatic Parameter Optimization of a Mobile Robot Localization Algorithm

Oscar Lima, Rodrigo Ventura
Institute for Systems and Robotics
Instituto Superior Técnico
Av. Rovisco Pais 1, 1049-001 Lisbon, Portugal
Email: {olima, rodrigo.ventura}@isr.ist.utl.pt

***Abstract*—Algorithms in robotics typically tend to expose several parameters for the user to configure. This allows both re-usability and fine tuning of the algorithm to a particular setup, but at the expense of significant effort in tuning, since this task is typically done manually. However, recent results in parameter optimization have shown to be quite successful, namely in automatic configuration of boolean satisfiability problem solvers, contributing to a significant increase in scalability. In this paper we address the applicability of these methods to the area of mobile robot localization. In particular, we applied a sequential model-based optimization method to the automatic parameter tuning of the well-known Adaptive Monte Carlo Localization algorithm. Our results show a statistically significant improvement over the default algorithm values. We also contribute with an open source experimental setup, based on the popular Robot Operating System ROS, which can be easily adapted to other algorithms.**

***Index Terms*—Model based optimization robotics, black box optimization, algorithm configuration, automatic algorithm optimization, multi objective parameter optimization, automatic parameter tuning.**

I. Introduction

As algorithms for real robots become more complex, the amount of parameters available that tune their performance tend to increase. Since these parameters are typically hand tuned, this increase turns manually tuning a very tedious task. This difficulty becomes larger whenever the algorithm is being reused by someone other than the original authors. In particular, code reuse, which is an widespread practice observed for instance in the robotics community, may be hindered by these difficulties.

However, in other domains, such as algorithms for boolean satisfiability (SAT), this profusion of parameters have been dealt with significant success by using automatic configuration systems. These methods aim to find the optimal set of parameters that maximize the performance of an algorithm (or a set of algorithms, in case of portfolio approaches) over a given set of instances. The success of this methods have allowed very large problems to become tractable in reasonable time [1].

The goal of this paper is to study the applicability of these methods to the domain of mobile robotics. In particular, we contribute with a case study on mobile robot localization, choosing the well known Adaptive Monte Carlo Localization (AMCL) algorithm [2], which implementation is freely available here[1] thanks to the Robot Operating System (ROS) community. Encouraging code re-usability, their implementation exposes 47 parameters to configure and although some of these can be experimentally estimated, their interaction and effects given the stochastic nature of MCL methods are hard to predict.

To perform the automatic parameter tuning of AMCL, we have chosen a sequential model-based optimization algorithm and in particular its implementation in the open sourced SMAC software[2]. This package is one of the state-of-the-art global optimization methods used by the SAT community.

This paper is organized as follows: in section II we briefly explore the relevant related work to parameter optimization algorithms, then in III we describe the system architecture that we have developed for our experiments, in IV we describe the specifics of how the experimental data was collected and in V, VI, VII we present obtained results, discussion and conclusions.

II. Related Work

In the Automated Algorithm Configuration problem (AAC), given a parameterized algorithm $A(\theta)$, a set P of instances and a error metric $E(A(\theta), P)$, the problem is described as finding a vector θ of parameters for A so that they minimize E [1].

In this constrained optimization problem, the global minimum is searched, having as main challenge to overcome getting stuck in local minimum. Usually the cost function is unknown, but can be discretely sampled. For our case study the algorithm is stochastic and it has an expensive (time consuming) cost function calculation. Therefore the importance of having global optimization methods with heuristics that can train the parameters in a reasonable amount of time.

In literature we find several automated optimization algorithms, such as F-Race [3], ParamILS [1], SMAC [4] and genetic algorithm based in [5]. A survey paper for this last mentioned approach can be found in [6].

A. ParamILS: Iterated Local Search

ParamILS [1] is a parameter optimization algorithm based on local search. Similar to simulated annealing, ParamILS switches between local descend and jumps, in an attempt to

[1] https://github.com/ros-planning/navigation/tree/indigo-devel/amcl
[2] http://www.cs.ubc.ca/labs/beta/Projects/SMAC/

978-1-5090-6234-8/17/$31.00 ©2017 IEEE

find the best global configuration. It has been successfully applied to different domains such as Propositional Satisfiability SAT, Mixed Integer Programming CPLEX, AI Planning [7], Protein folding [8], etc. Unlike our case study, where the goal is to minimize the error of the cost function, for SAT the target is to reduce the CPU time of highly parameterized algorithms such as CPLEX (159 parameters). Their results show speed-ups ranging from 2 to 23 over the default set.

B. Sequential Model-Based Optimization for General Algorithm Configuration (SMAC)

SMAC [4], the improved version of ParamILS (from the same authors), is a tool for optimizing algorithm parameters based on predictive models (to select promising parameter configurations). The cost function model is estimated based on random forests [9], a well-known machine learning regression method based on domain dependent features. For new domains with no defined features, SMAC can be applied with a set of empty or domain independent features, for instance the default's target algorithm parameter set.

SMAC supports both numerical and categorical parameters along with multiple algorithm instances. The authors claim that SMAC is a general configuration tool, non domain specific and therefore can be used for many different algorithms.

The current implementation of SMAC does not support parallelization and runs are evaluated separately. In SMAC the algorithm configuration is treated as a black-box optimization problem.

C. Parameter optimization applied to Localization

In [10] we find a similar work to the one presented in this article: an automatic parameter optimization algorithm applied to Monte Carlo Localization (MCL). In this paper from 2010 the authors automatically tune all relevant parameters of the filter by using the Particle Swarm Optimization (PSO) algorithm [11]. This well known method is useful for multi-objective optimization of non-linear functions. The algorithm configuration process is performed offline by using recorded log data. To rate each parameter set, a benchmark function is used to estimate the mean error based on euclidean distance between the estimated and the reference position, which comes from a camera image based system named SSL-Vision [12]. The approach is evaluated on a humanoid soccer player robot, in the scope of the RoboCup competition. The results are produced faster than with evolutionary algorithm approaches. According to the authors the system was able to find parameter sets that led to a more precise position estimate than the manually tuned set.

1) Differences with our approach: Unlike this work [10], which automatically configures a single algorithm, we claim a general approach. Our framework can be used to offline configure other (ROS based) mobile robot algorithms. To use our framework the user would need to develop a score function and record their own data.

In our work we provide with a better statistical result analysis and cross-validation by testing the trained parameters in a separate data set.

III. System Architecture

At the core of the optimization system we use SMAC[4], which is a Java based general tuning software. A client-server ROS wrapper was implemented to interact via linux shell with the third party tuning software.

SMAC requires a scoring function to evaluate the quality of each run. For AMCL the selected metric measures the average distance between the robot ground truth pose and the best AMCL particle.

The complete cycle is shown in Algorithm 1. The first step is to load the algorithm configurator *C* (SMAC for our case) selected parameters into the server and to select the instance i that will be used. Then we run algorithm *A* and initial conditions are provided. For AMCL this step provides with a pose estimate based on motion capture system data. Next step is to playback the previously recorded robot sensor data, being in our case odometry and laser scanner readings of a robot moving around the environment. Afterwards the quality score function *E* is started and the system waits until the playback is finished. Finally the scoring function result is collected together with the CPU time and the algorithm is terminated. SMAC will continue the loop until the maximum amount of runs is reached, selecting in each step a new promising algorithm configuration. At the end the current best configuration obtained is returned.

Algorithm 1 Automatic parameter optimization pseudocode

```
1:  procedure ALGORITHM CONFIGURATION
2:      best configuration ← initial configuration
3:      i ← 0.
4:      do
5:          C selects promising A config. and instance i
6:          Run A
7:          A ← initial conditions
8:          Playback P(i) sensor data set
9:          Wait until sensor data playback is finished
10:         Terminate A
11:         C ← Compute E, CPU time.
12:         i ← i + 1.
13:     while i < max. allowed runs
14:     return best configuration
```

For deterministic algorithms SMAC always selects a different configuration for the next run (Algorithm 1, line 5). For AMCL we use the non-deterministic option to perform several runs with the same parameters, considering the median quality score to select the next configuration.

In Fig. 1 we present a component diagram of our work showing the specifics of how the developed components interact with each other.

To encourage re-usability, the architecture was design in a modular way. To plug-in a new algorithm a score function must be developed along with configuration files.

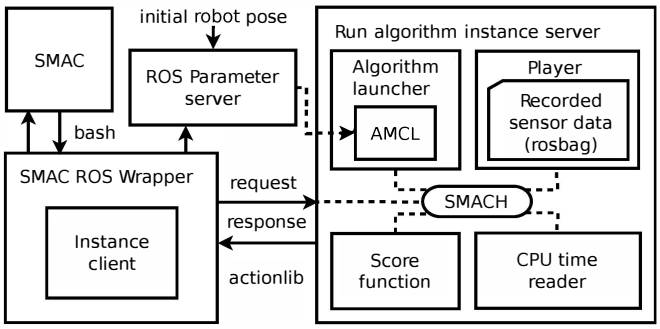


Fig. 1: Automatic algorithm optimization component diagram.

IV. Experiment Setup

The experimental data was collected by teleoperating the MOnarCH robot (Fig. 2a) in the ISRoboNet@Home Testbed[3] (Fig. 2b).

(a) MOnarCH robot. (b) ISRoboNet@Home testbed.

Fig. 2: Experiment setup, a service robot (2a) and a testbed with a motion capture system (2b).

A. The testbed

The lab is an apartment-like environment with a Motion Capture System (MCS), designed to benchmark service robots. The Testbed has, among many other sensors and actuators, 12 OptiTrack® "Prime 13" cameras (1.3 MP, 240 FPS).

The system provides real-time tracking data of rigid bodies with sub mm precision in 6 dimensions with low latency (4.2 ms).

B. The Robot

The MOnarCH robot[4], originally designed to interact with children in hospitals [13], has an omni-directional mobile base equipped with four mecanum wheels actuated by four independent motors. It also features 5/30 m laser rage finders installed 13.5 cm above ground level and two on-board computers with i7 processor, one PC is dedicated for navigation and one for human robot interaction. The robot has a weight of 24 Kg and a maximum velocity of 2.5 m/s with 1 m/s^2 acceleration. The 19 mm diameter, 5 marker pattern located on top of the head, allows to track the robot in real time by using the motion capture system earlier described. A more detailed robot specification can be consulted here [14].

C. Experiment description

The robot is teleoperated around a previously mapped environment, while ground truth information (with the pose of the robot), odometry and laser scanner readings are recorded. The collected open sourced data set[5] is later on used to (offline) tune the localization algorithm.

The recorded sensor log (280 sec.) was divided into training and test data set (30 sec.) by selecting different intervals of the file for playback. The AMCL pose estimate is provided based on the first available MCS pose. The algorithm was trained in approximately 4.2 hours with a single instance, limited to 500 runs max.

In Fig. 3 we show a graphic representation of the collected sensor data on RViz. The error function calculates the average of all transformations between the robot (base_link) and the ground truth frame provided by MCS.

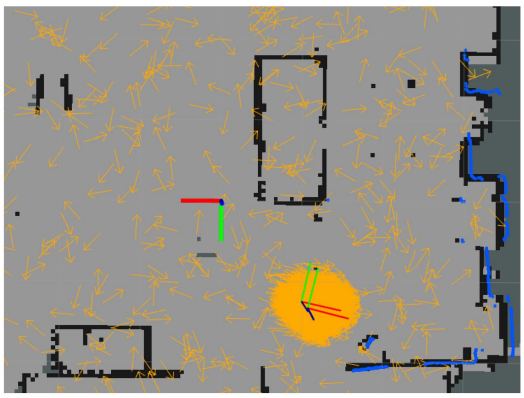


Fig. 3: Processed sensor data visualization,
frames : global frame, ground truth and robot estimated pose,
blue dots : laser scanner readings,
orange arrows : AMCL particles.

D. Parameter description

A subset of all configurable parameters was carefully chosen from the total amount (22 out of 47), with intervals selected based on dynamic reconfigure ranges (suggested by the algorithm author in the code). The complete code used for the experiment is available in this[6] website.

In table I. we show the 22 selected configurable parameters for our experiment with their admissible ranges and default values. The remaining 25 parameters were fixed to their default values. A brief description of all AMCL parameters and their default values can be found here[7]

[3]http://welcome.isr.tecnico.ulisboa.pt/isrobonet/
[4]http://monarch-fp7.eu/
[5]http://www.github.com/oscar-lima/localization_dataset
[6]http://www.github.com/oscar-lima/autom_param_optimization
[7]http://wiki.ros.org/amcl

TABLE I: AMCL selected parameter ranges to configure

Parameter	**Type**	**Min. value**	**Max. value**	**Default**
laser_max_beams	int	5	100	30
min_particles	int	5	980	100
max_particles	int	1000	10000	5000
kld_err	real	0.001	0.5	0.01
kld_z	real	0.5	0.999	0.99
odom_alpha1	real	0.001	10	0.005
odom_alpha2	real	0.001	10	0.005
odom_alpha3	real	0.001	10	0.010
odom_alpha4	real	0.001	10	0.005
odom_alpha5	real	0.001	10	0.003
laser_z_hit	real	0.001	10	0.95
laser_z_short	real	0.001	10	0.1
laser_z_max	real	0.001	10	0.05
laser_z_rand	real	0.001	10	0.05
laser_sigma_hit	real	0.001	10	0.2
laser_lambda_short	real	0.001	10	0.1
laser_likelihood_max_dist	real	0.001	20.0	2.0
update_min_d	real	0.001	0.5	0.2
update_min_a	real	0.001	1.0	0.5236
resample_interval	int	1	5	2
recovery_alpha_slow	real	0.001	0.5	0.001
recovery_alpha_fast	real	0.001	1.0	0.1

V. Experiment Results

A. AMCL consistency check

An algorithm is candidate for automatic configuration, only if there are pairs of configurations for which the quality score function is able to obtain statistically significant different scores across a number of runs.

In Fig. 2a, we show an example of this condition satisfied. In this experiment we randomize the algorithm parameters and perform several runs (10 different parameter sets, tested 50 times), looking for consistency and variance.

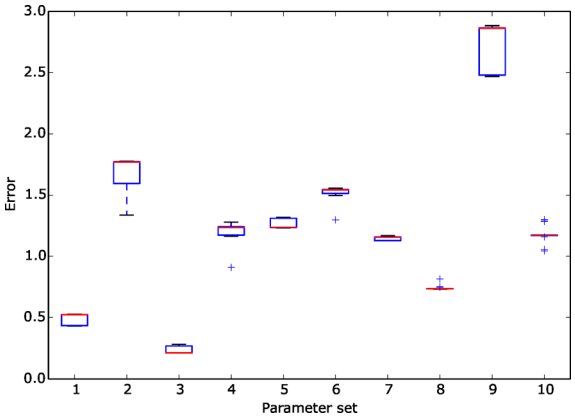


Fig. 4: AMCL consistency check experiment, for several runs with random parameters different quality scores are obtained.

B. Automatic algorithm configuration results

In Fig. 5a we present the results of the automatic AMCL optimization for 500 runs using SMAC.

The automatically configured parameters outperform the default set. At first the tuning software went for the obvious choice: to increase the number of particles in the filter, at the expense of an increase in the computational load. To be fair with the default set, we fixed the max. amount of particles to the default value (5000) and we tuned for the remaining parameters. The result still shows better performance on both training and test data set with similar CPU time consumption.

In Fig. 6 we show a plot of all the configurations tried out by the optimization algorithm against their associated error. Although it was limited to 500 runs, due to the non-deterministic nature of AMCL, only 134 different configurations were selected.

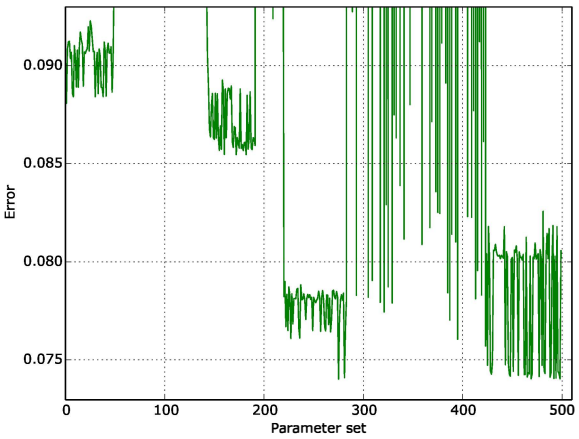


Fig. 6: AMCL optimization trajectory for 5000 particles max. Figure is trimmed to show the region of interest.

A robot position path plot is shown in Fig. 7. with a visual comparison of which parameter set is closer to the ground truth path.

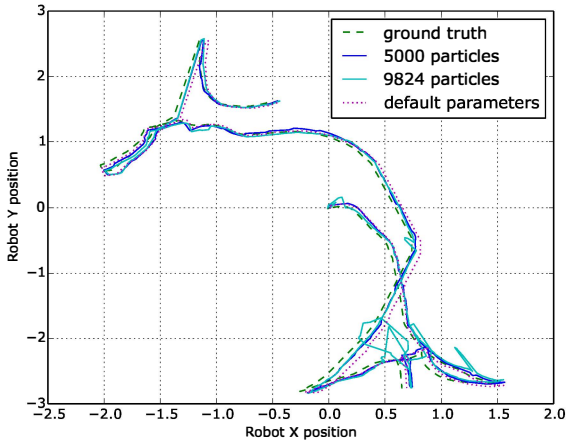
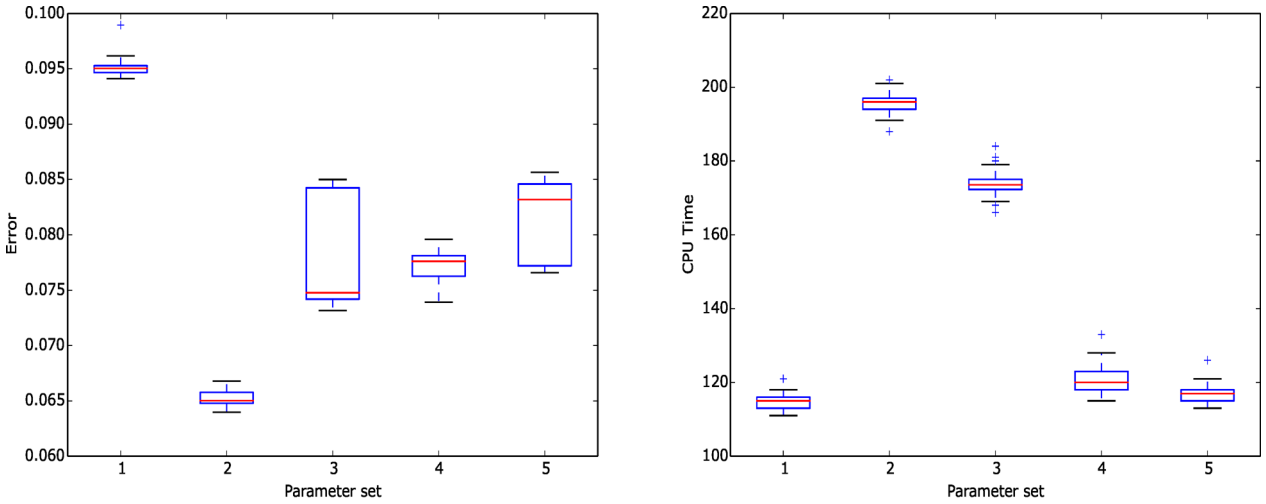


Fig. 7: Robot path comparison on test data set for different parameters.



(a) Error.

(b) CPU Time.

Fig. 5: Comparison between AMCL default parameter set and automatically optimized (2, 4) for 500 runs. (1) default parameters, (2) 9824 particles, (4) 5000 particles, (3, 5) trained parameters on test dataset.

VI. Discussion

The optimized parameters (see Figure 5a param. set 2, 4) outperform the default set. However results could be better, we deliberately did not include expert knowledge into the system nor extra model features (only the default parameters).

Although SMAC supports many instances, it was fed with a single training instance with mostly a linear trajectory. The results show an over fitting training, the obtained parameters are good for linear motions but partially fail at rotation (see Figure 7 at coordinates 0.5, -2).

Some of the challenges of the automated tuning rise from the fact that we are running algorithms in real time with offline data, in other words while playing the previously recorded sensor data an online instance of AMCL is processing the offline readings. For this exercise it is crucial to have the time synchronized between all involved PC's (MCS & robot) while recording the sensor data. Furthermore a second time synchronization needs to be performed while running the algorithm to configure.

We cannot assume sample normality for default set ($W(50) = 0.71508, p < 0.001$) and SMAC trained parameters ($W(50) = 0.91762, p < 0.001$) from the Shapiro-Wilk W test. The analysis of default set vs. SMAC trained parameters shows a statistically significant difference between default set and the parameters trained by SMAC ($U = 2500, 50$ dof, $p < 0.001$), in two-tailed Mann-Whitney U test. The group with the lower error is the one with 9820 particles max (see Figure 5a).

For future work we would like to investigate on multi-criteria optimization, for instance CPU Time and Error, training with different instances and trying other optimization algorithms.

VII. Conclusions

In this paper we have demonstrated by example the applicability of automatic algorithm configuration methods originally designed for SAT solvers into the field of mobile robotics, in particular to the well known algorithm AMCL.

Our experiments show that the result of the automatic configuration outperform the default set, having statistically significant better results with a p value less than 0.001 (99 % confidence).

The algorithms involved in the process (SMAC and AMCL) were treated as a black box and expert knowledge was not applied in order not to bias the experiments.

We contribute with a open-source ROS based Framework[8] to offline tune ROS based mobile robot algorithms. We present AMCL as a use case that demonstrates how the system can be used to automatically configure algorithm parameters.

In our experience optimization methods can be used to fine tune a particular algorithm by using ranges instead of fixed values. This allows for uncertainty during manual tuning, the author might not be sure about a specific parameter value but is sure that is confined in a certain range.

Acknowledgment

This work was supported by the FCT project [UID/EEA/50009/2013]. This work was supported by project [PTDC/EEI-SII/4698/2014].

[8] http://www.github.com/oscar-lima/autom_param_optimization

REFERENCES

[1] F. Hutter, H. H. Hoos, K. Leyton-Brown, and T. Stützle, "Paramils: an automatic algorithm configuration framework," *Journal of Artificial Intelligence Research*, vol. 36, no. 1, pp. 267–306, September 2009.

[2] D. Fox, "Kld-sampling: Adaptive particle filters," in *Neural Information Processing Systems*, vol. 14. NIPS, December 2001, pp. 713–720.

[3] M. Birattari, Z. Yuan, P. Balaprakash, and T. Stützle, "F-race and iterated f-race: An overview," in *Experimental methods for the analysis of optimization algorithms*. Springer, October 2010, pp. 311–336.

[4] F. Hutter, H. H. Hoos, and K. Leyton-Brown, "Sequential model-based optimization for general algorithm configuration," in *International Conference on Learning and Intelligent Optimization*. Springer, January 2011, pp. 507–523.

[5] C. Ansótegui, M. Sellmann, and K. Tierney, "A gender-based genetic algorithm for the automatic configuration of algorithms," in *International Conference on Principles and Practice of Constraint Programming*. Springer, September 2009, pp. 142–157.

[6] T. Bäck and H.-P. Schwefel, "An overview of evolutionary algorithms for parameter optimization," in *Evolutionary computation*, vol. 1, no. 1, March 1993, pp. 1–23.

[7] M. Vallati, C. Fawcett, A. Gerevini, H. Hoos, and A. Saetti, "Generating fast domain-specific planners by automatically configuring a generic parameterised planner," in *Workshop on Planning and Learning*. ICAPS, June 2011.

[8] J. Cheu, "Improving predictive power of the replica exchange monte carlo algorithm for protein folding by application of more complex energy and lattice models," Ph.D. dissertation, University of British Columbia, May 2009.

[9] L. Breiman, "Random forests," in *Machine learning*, vol. 45, no. 1. Springer, October 2001, pp. 5–32.

[10] A. Burchardt, T. Laue, and T. Röfer, "Optimizing particle filter parameters for self-localization," in *RoboCup 2010: Robot Soccer World Cup XIV*. Springer, 2011, pp. 145–156.

[11] J. Kennedy, "Particle swarm optimization," in *Encyclopedia of machine learning*. Springer, March 2011, pp. 760–766.

[12] S. Zickler, T. Laue, O. Birbach, M. Wongphati, and M. Veloso, "Ssl-vision: The shared vision system for the robocup small size league," in *Robot Soccer World Cup*. Springer, June 2009, pp. 425–436.

[13] J. S. Maria Isabel Aldinhas Ferreira, "Designing a robotic interface for children: The monarch robot example," in *Proceedings of the 19th International Conference on 19th International Conference on Climbing and Walking Robots*. CLAWAR, October 2016.

[14] J. Messias, R. Ventura, P. Lima, J. Sequeira, P. Alvito, C. Marques, and P. Carriço, "A robotic platform for edutainment activities in a pediatric hospital," in *Autonomous Robot Systems and Competitions (ICARSC), 2014 IEEE International Conference on*. IEEE, May 2014, pp. 193–198.