The 2010 IEEE/RSJ International Conference on
Intelligent Robots and Systems
October 18-22, 2010, Taipei, Taiwan

Fast Path Planning using Multi-Resolution Boundary Value Problems

Renato Silveira, Edson Prestes and Luciana Nedel
Instituto de Informática
Universidade Federal do Rio Grande do Sul
Porto Alegre - RS - Brazil
{rsilveira,prestes,nedel}@inf.ufrgs.br

***Abstract*— BVP Path Planners generate potential fields through a differential equation whose gradient descent represents navigation routes from any point of the environment to a goal position. Resulting paths are smooth and free from local minima. In spite of these advantages, this kind of planners consumes a lot of time to produce a solution. In this paper, we present a new approach that combines our BVP Path Planner with the Full Multigrid Method, which solves elliptic partial differential equations using a hierarchical strategy. Our new approach, called Hierarchical BVP Path Planner, enables real-time performance on large environments. Results show that our proposal spends less than 1% of the time needed to compute a solution using our original planner in several environments.**

I. INTRODUCTION

One of the main goals of robotics is to create robots that can act autonomously in the environment. This goal is very complex and requires advances in several domains, as localization, mapping, and planning. This paper focuses on the path planning problem which consist of finding a viable collision-free path for a robot to reach a goal. The path planner should produce a fast answer, particularly in problems that involve risk to human life, like rescue operations during natural disasters, for instance. In these cases, robots are expected to act efficiently and spending a minimum time in path planning.

Most path planners are generally based on graph searching algorithms, as the A^* [5] and D^* [15]. The A^* algorithm operates in an environment represented as a regular grid and finds the least-cost path from a given initial cell to a target cell using a heuristic function. D^* behaves like A^* except that, if the costs change during the execution, the algorithm adds the information to its map and replans a new path efficiently. Due to the environment discretization, in both algorithms the path generated is not smooth and the robot may have to pass close to obstacles, requiring thus some post-processing to smooth the path and ensure a high quality navigation.

A solution to improve the performance of path planners is to reduce the complexity of the problem by means of a hierarchical approach [9], [18], [6], [7]. The environment is represented in different resolutions and used as input to the path planners, that will decide what is the best resolution to be used for a given situation using criteria such as time restriction or memory limitations. A common hierarchical representation is the quadtree decomposition [8], where the environment is first represented using a coarse grid which is then subdivided generating patches with different resolutions. After that, any graph search algorithm may be used to obtain the path. The combination of A^* or D^* with a hierarchical decomposition gives rise to algorithms like Hierarchical A^* [4] or Hierarchical D^* [2]. Another way to generate paths hierarchically is using wavelet functions for environment decomposition [17]. The wavelet transform is a very fast approach that allows the decomposition of the environment at different levels of resolution. The smooth path is achieved using the information provided by the coefficients in the wavelet expansion.

In this work we extend our Path Planner on Boundary Value Problem, called BVP Path Planner (or BVP PP) [16], using the numeric solution of a multi-resolution BVP. The original BVP PP allows the control of the steering behavior of robots in dynamic environments using a potential field formalism. The resulting paths represent a compromise between length and safety, since the paths reduce the obstacle hitting probability[3]. The new path planner, called Hierarchical BVP Plath Planner (or HBVP PP), inherits all properties of the original BVP, but is much more efficient, usually spending less than 1% of the time needed for the original method to compute a solution. Like the original planner, the new approach can be adapted and used in mapping and localization problems [10], [14], [16], [13], and in path planning in non-homogeneous terrains[11]. In the latter, the planning considers the traveling preference to guide the robot towards the goal.

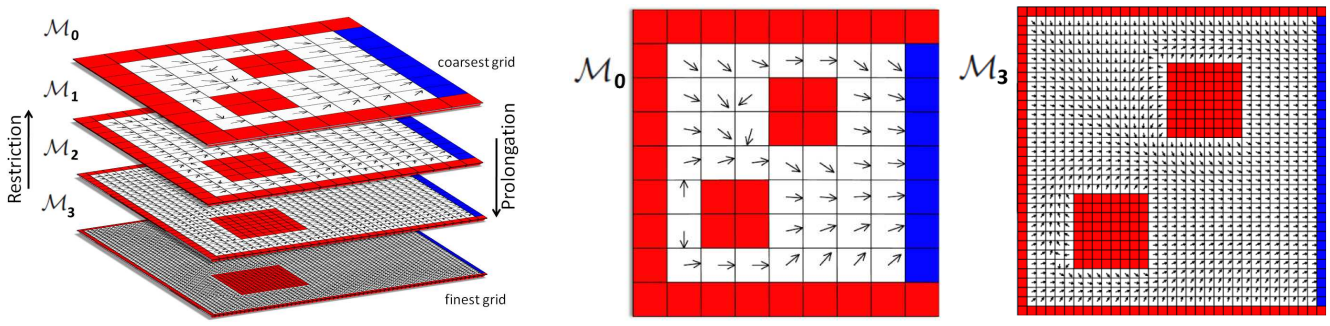
The paper is divided as follows. Section II describes the theoretic aspects of BVP PP. In Section III, we introduce the HBVP PP. In Section IV, we present some results in 2D environments and compare the performance of the HBVP PP with the original BVP PP and A^* algorithm. Section V discusses some aspects of our approach and Section VI presents our conclusions.

II. BVP PATH PLANNER

The original BVP path planner [16] generates paths using the potential information computed from the numeric solution of

$$\nabla^2 \, p(\mathbf{r}) \; = \; \epsilon \mathbf{v} \cdot \nabla p(\mathbf{r}) \; , \tag{1}$$

with Dirichlet boundary conditions, where $\mathbf{v} \in \Re^2$ and $|\mathbf{v}| = \mathbf{1}$ corresponds to a vector that inserts a perturbation in the potential field; $\epsilon \in \Re$ corresponds to the intensity of



(a) (b) (c)

Fig. 1. Hierarchical environment representation. (a) the environment is represented by a hierarchy of 4 grids $\mathcal{M}_k$ with different resolution. Red and blue cells correspond to obstacles and goals, respectively, while the arrows illustrate the vector field. (b) and (c) show the coarsest and finest mesh, respectively.

the perturbation produced by $\mathbf{v}$; and $p(\mathbf{r})$ is the potential at position $\mathbf{r} \in \Re^2$, respectively. Both $\mathbf{v}$ and ϵ must be defined before computing this equation. The gradient descent on these potentials represents navigational routes from any point of the environment to the goal position. Trevisan et al. [16] shows that this equation does not produce local minima and generates smooth paths.

To solve numerically a BVP, we can consider that the solution space is discretized in a regular grid. Each cell (i, j) is associated to a squared region of the real environment and stores a potential value $p(i, j)$. Using the Dirichlet boundary conditions, the cells associated to obstacles in the real environment store a potential value of 1 (*high potential*) whereas cells containing the target store a potential value of 0 (*low potential*).

A high potential value prevents the robot from running into obstacles whereas a low value generates an attraction basin that pulls the robot. The relaxation methods usually employed to compute the potentials of free space cells are Gauss-Seidel (GS) and SOR (successive over-relaxation). The GS method updates the potential of a cell c through:

$$p_c = \frac{p_b + p_t + p_r + p_l}{4} + \frac{\epsilon((p_r - p_l)v_x + (p_b - p_t)v_y)}{8} \quad (2)$$

where $\mathbf{v} = (v_x, v_y)$, and p_c, p_b, p_t, p_r and p_l are illustrated in Figure 2.

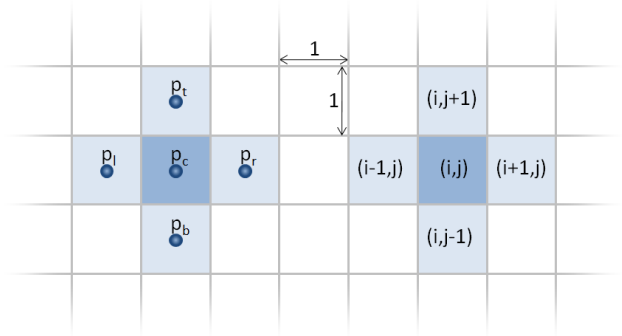


Fig. 2. Representation of p_c, p_b, p_t, p_r and p_l on the grid.

The SOR method updates the potential of a cell c is hrough:

$$p_c = p_c^t + \omega\left(\frac{p_b + p_t + p_r + p_l - 4p_c^t}{4} + \frac{\epsilon[(p_r - p_l)v_x + (p_b - p_t)v_y]}{8}\right) \quad (3)$$

where $\omega = \frac{4}{2+\sqrt{4-c^2}}$ with $c = \cos\frac{\pi}{m} + \cos\frac{\pi}{n}$ where m and n are the grid dimensions.

SOR is an extension of the GS method with an accelerator factor, which usually makes the potential convergence using SOR faster than GS. However, the error produced by SOR often grows before the convergence sets in, resulting in oscillatory potential fields during the calculation. On the other hand, the error produced by the GS method monotonically decays during the computation of the potential. This makes the GS more useful in tasks like robotic exploration, since the robot can use partial results as an approximation of the potential field [12].

III. Hierarchical BVP Path Planner

Our proposal, the Hierarchical BVP Path Planner, uses as core the combination of the BVP Path planner and the Full Multigrid Method (FMG), proposed by Brandt [1]. This method solves elliptic PDEs efficiently through a combination of solutions at several resolution levels. Basically, it takes an instance of the problem on a grid of pre-specified fineness and generates coarser grids containing a cruder problem representation. The method solves the problem in the coarsest grid, which is easy and cheaper, and obtains successive solutions on finer and finer grids.

The HBVP PP works as follows. The entire environment is represented by a hierarchy of homogeneous meshes, $\{\mathcal{M}_k\}$, where each mesh $\mathcal{M}_k$ has $L_x^k \times L_y^k$ cells, denoted by $\{c_{i,j}^k\}$. Each cell $c_{i,j}^k$ corresponds to a squared region centered in environment coordinates $r = (r_i, r_j)$ and stores a particular potential value $p_{i,j}^k$ and an error value $e_{i,j}^k$. Figure 1(a) shows the hierarchy of grids, where red and blue cells are associated to obstacles and targets, respectively, and the arrows represent the vector field. The potential fields in all grids were computed using the FMG method. Figure 1(b)

and (c) show the vector field on the coarsest grid and on the finest grid, respectively.

To use the FMG method in our planner, we consider that the Eq. 1 is solved in an uniform grid with cell size h. Then, Eq. 1 becomes

$$\nabla^2 \, p^h(\mathbf{r}) + \epsilon \mathbf{v}.\nabla p^h(\mathbf{r}) \; = \; f^h \tag{4}$$

with $f^h = 0$, for better understanding the algorithm.

Let the linear elliptical operator $\left(\nabla^2 + \epsilon \mathbf{v}.\nabla\right)$ be $\mathcal{A}^h$ and Eq. 1 becomes

$$\mathcal{A}^h p^h = f^h. \tag{5}$$

Assuming that $\tilde{p}^h$ is an approximate solution to Eq. 5 and p^h is the exact solution, we can define the error $e^h = p^h - \tilde{p}^h$ and the residual $r^h = \mathcal{A}^h \tilde{p}^h - f^h$. With this information, we can rewrite Eq. 5 as

$$\mathcal{A}^h(\tilde{p}^h + e^h) = 0 \tag{6}$$

which means that

$$\mathcal{A}^h e^h = -\mathcal{A}^h \tilde{p}^h.$$

Using the residual r^h, we obtain the Residual Equation

$$\mathcal{A}^h e^h = -r^h \tag{7}$$

and the Correction Equation

$$p^h = \tilde{p}^h + e^h \tag{8}$$

which are the fundamental equations of the FMG method.

To propagate the information in the grid hierarchy, we must define two operators: the restriction operator R, and the prolongation operator P.

$$R = \begin{bmatrix} \frac{1}{16} & \frac{1}{8} & \frac{1}{16} \\ \frac{1}{8} & \frac{1}{4} & \frac{1}{8} \\ \frac{1}{16} & \frac{1}{8} & \frac{1}{16} \end{bmatrix} \text{ and } P = \begin{bmatrix} \frac{1}{4} & \frac{1}{2} & \frac{1}{4} \\ \frac{1}{2} & 1 & \frac{1}{2} \\ \frac{1}{4} & \frac{1}{2} & \frac{1}{4} \end{bmatrix}$$

The operator R takes the information in the fine grid, level i in the hierarchy, and restricts it to the coarse grid, level $i+1$ in the hierarchy. The level i represents the grid with discretization h while the level $i+1$ represents the grid with discretization $2h$. The operator P takes the information in the coarse grid, level $i+1$, and interpolates it to the fine grid, level i. Figure 1(a) illustrates the direction of information propagation of these operators on the grid hierarchy.

The HBVP Path Planner algorithm, shown in Algorithm 1 works as follows. Initially, all grids $\{\mathcal{M}_1, \mathcal{M}_2, .., \mathcal{M}_N\}$ will represent the environment in a specific resolution, where the $\mathcal{M}_N$ is the coarsest representation. The algorithm receives as input the hierarchy of grids, and then it computes the potential in the grid $\mathcal{M}_N$ (Line 2) using any relaxation method (GS or SOR). That results in a coarse representation of the potential field.

The potential of the other grids $\mathcal{M}_i$ is prolongated to $\mathcal{M}_{i-1}$, with $N \geq i \geq 2$ (Line 5). This initial approximation is smoothened α_1 times using only GS method (Line 7). SOR method can not be used since it eliminates the smooth components of the error what is crucial to the algorithm convergence. After, we compute the residual of the grid $\mathcal{M}_{i-1}$ and restrict it to the grid $\mathcal{M}_i$ (Line 8). The error of this grid is reseted (Line 9) and the V-cycle, illustrated in Algorithm 2, is executed (Line 10). The potential of the fine grid $\mathcal{M}_{i-1}$ is corrected through the error computed from the execution of V-cycle (Line 11), and smoothened α_1 times (Line 12). The grid $\mathcal{M}_{i-1}$ is then ready to be used in the navigation process. The steps in the lines 7-12 are executed η times or until the error of the potential convergence reaches a desired value.

The V-cycle algorithm receives as input the grid $\mathcal{M}_i$ and acts in the error of the potential convergence for each grid $\mathcal{M}_k$, with $i \leq k \leq N$. Initially, the V-cycle minimizes the error associated to the grid $\mathcal{M}_k$ through the relaxation process that is executed α_1 times, not necessarily until the potential convergence (Line 1). If the current grid is the coarsest grid $\mathcal{M}_N$ (Line 2) then the algorithm updates the potential using the residual (Line 11). Otherwise, the residual is computed in the grid $\mathcal{M}_k$ (Line 5) and restricted to the grid $\mathcal{M}_{k+1}$ (Line 6). The error from the grid $\mathcal{M}_{k+1}$ is reseted (Line 7) and the V-cycle algorithm is recursively called receiving as input the grid $\mathcal{M}_{k+1}$ (Line 8). The potential of the grid $\mathcal{M}_{k+1}$ is updated with the error computed from the coarse grid (Line 10) and smoothened α_2 times.

Observe that the restriction and the prolongation operators are applied only to the free space cells. Besides, when the potential converges on the coarse grids the robot can use its vector field to navigate.

Algorithm 1 HBVP

1: {*Make sure all the grids have the correct environment representation.*}
2: Solve exactly the coarsest grid $\mathcal{M}_N$. {*GS or SOR*}
3: Set $\mathcal{M}_N$ as the finer grid ready to use.
4: **for** i$\leftarrow$ N to 2 **do**
5: Set initial guess $\tilde{p}^{i-1} \leftarrow I.\tilde{p}^i$
6: **for** j $\leftarrow$ 1 to η **do**
7: Relax α_1 times on $\mathcal{A}^{i-1}\tilde{p}^{i-1} = 0$.
8: $r^i \leftarrow R.r^{i-1}$
9: $\tilde{e}^i \leftarrow 0$
10: calls V-cycle($\mathcal{M}_i$)
11: Correct $\tilde{p}^{i-1} \leftarrow \tilde{p}^{i-1} + P.\tilde{e}^i$
12: Relax α_2 times on $\mathcal{A}^{i-1}\tilde{p}^{i-1} = 0$.
13: **end for**
14: Set $\mathcal{M}_{i-1}$ as the finer grid ready to use.
15: **end for**

IV. RESULTS

This section presents several results that compare the HBVP PP with the original BVP PP, as well as with the A^* algorithm. These results have been obtained in 10 different environments with arbitrary obstacles and goal

Algorithm 2 V-cycle

1: Relax α_1 times on $\mathcal{A}^i\tilde{e}^i = -r^i$.
2: **if** $\mathcal{M}_i$ is the coarsest grid **then**
3: Go to line 11.
4: **else**
5: $r^i \leftarrow -\mathcal{A}^i\tilde{e}^i$.
6: $r^{i+1} \leftarrow R.r^i$
7: $\tilde{e}^{i+1} \leftarrow 0$
8: calls V-cycle($\mathcal{M}_{i+1}$)
9: **end if**
10: Correct $\tilde{p}^i \leftarrow \tilde{p}^i + P.\tilde{e}^{i+1}$
11: Relax α_2 times on $\mathcal{A}^i\tilde{p}^i = r^i$.

configurations. In all cases, the parameters α_1 and α_2 used in Algorithm 1 and 2, were set as 3 and 4, respectively. Whereas, the parameter η was calculated based on the error $\|e\|_\infty \leq 10^{-3}$.

A. Performance evaluation

Table I shows the performance of HBVP PP, original BVP PP (using GS and SOR), and the A^* algorithm in an Intel Quad Core 2.4 GHz with 4 GB RAM. The environments have different sizes and were represented using grids with 17×17, 33×33, 65×65, 129×129 and 257×257 cells. We can observe that the HBVP PP far surpasses the original BVP using GS and SOR. It is also important to stress that the time spent by the HBVP PP is the time required to compute the potential plus the time to generate all the maps from a given resolution. For a map with 129×129 cells, we need to generate all the maps in the hierarchy, i.e., the maps with 9×9, 17×17, 33×33, 65×65, and 129×129 cells.

Most experiments show that the HBVP PP performance is about 100 to 700 times faster than the original BVP PP. As the HBVP is computationally efficient, any changes in the environment can be mapped into the grids and the potential field is quickly recomputed. Despite the time required to compute the potential field in a large environment (as the grid with 513×513 cells) is not acceptable for real-time simulation, the HBVP guarantees the real-time property. Since, the robot can move immediately when the potential field in coarsest grid converges. The potential field in this grid is always generated in real-time.

In the most cases, for a single robot, the A^* algorithm is faster than the HBVP PP. However, for very small maps, the HBVP PP is faster. The reason is that the A^* computes only one path from the robot current position to its goal, while the HBVP computes all paths from any free position of the environment to the goal. This allows the planning of different paths for many robots simultaneously and with no additional cost. Moreover, in the HBVP PP the robot starts moving immediately when the coarsest grid is computed. In other words, we can say that the coarsest grid would be computed while the A^* was being processed. Furthermore, the HBVP PP always generates smooth paths, whereas A^* needs a post-processing step to smooth the path generated in order to ensure a high quality navigation.

B. Path quality

We compared the paths produced by the HBVP PP with those produced by the BVP PP using a hierarchy of grids with resolution 17×17, 33×33, 65×65 and 129×129 cells. In all experiments we used the parameters ($\alpha_1 = 4$, $\alpha_2 = 4$), and η, allowing the error $\|e\|_\infty \leq 10^{-3}$.

Figure 3(a)-(l) shows the quality of the path produced by the BVP PP. In this example, we generate 3 arbitrary environments with different obstacles and goal configurations. Each line of the Figure 3 presents one of these environments. In low resolution grids – with 17×17 or 33×33 cells – the potential field converges quickly but the path produced has low-quality. In high resolution grids – 65×65 or 129×129 cells – we achieved high-quality paths.

Figure 4 shows the path produced by 3 levels in the hierarchy of grids in the HBVP PP in 3 different environments. These environments are the same of the Figure 3. Green, blue and black illustrate the path segments followed by the robot using the grids with 33×33, 65×65 and 129×129 cells, respectively. When the potential field of a coarse map converges, it becomes available to be used by the robot. Meanwhile, the potential field in other grids are being calculated and, when it converges, the robot can use it. The better the resolution of the grid, the better the path quality. We can see that this path combination is smooth, continuous and has almost the same quality of the path computed using the finest grid resolution (Figure 3(d,h,l)).

V. Discussion

There are three important points that must be commented about the HBVP PP.

Map size. In this paper, we are only considering maps with size equal to $2^i + 1$, where i is the $i-th$ map on the hierarchy. The technique is not restricted to this map size. However, according to Brandt [1], this size is more convenient and economic in the prolongation processes and accelerates the convergence than any other sizes.

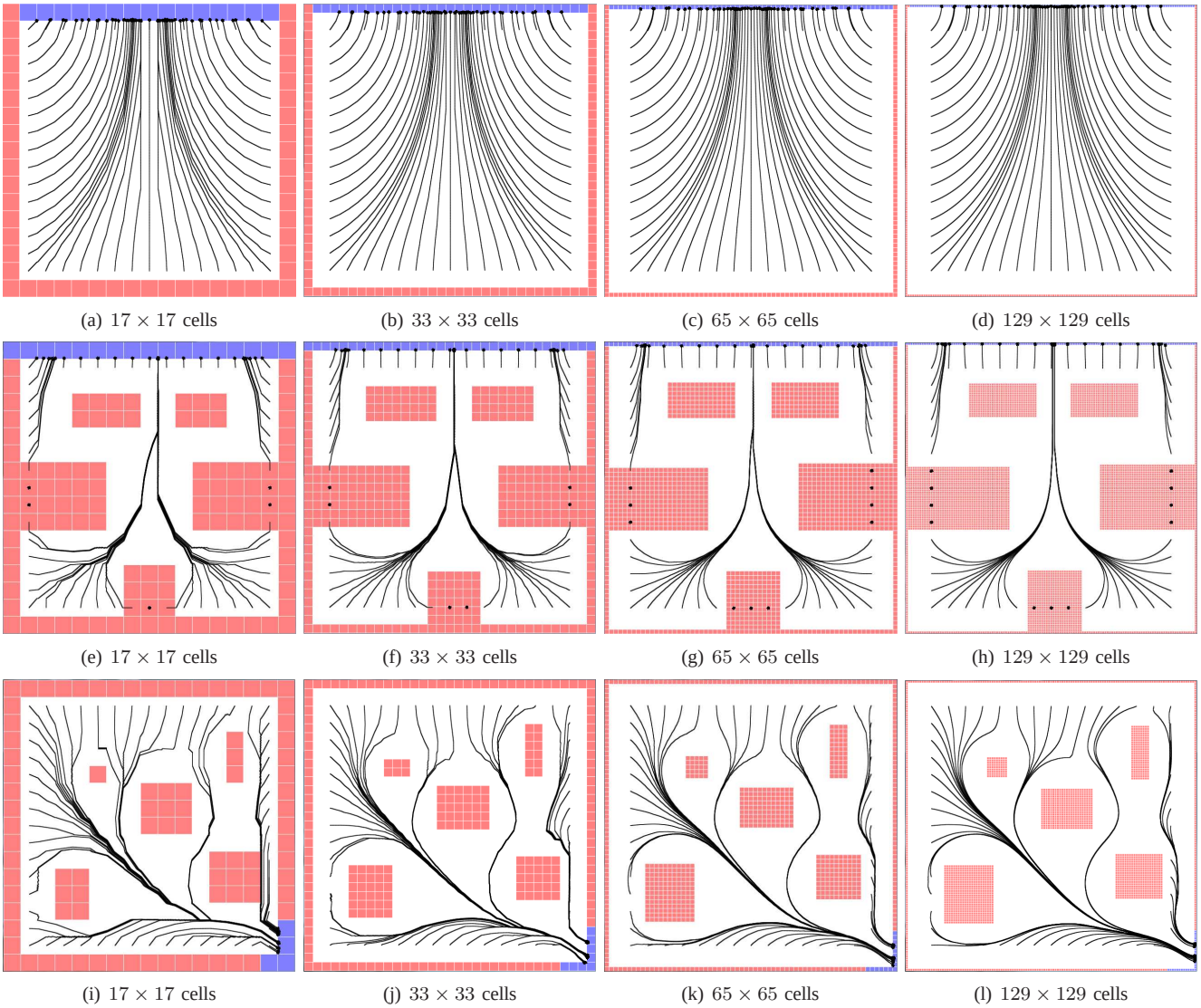
Obstacle representation. The obstacles must be represented in all maps. In our implementation we defined the obstacles on the finest map and assumed that the reduction operator will naturally propagate it to other maps during the algorithm execution. It is important to observe that the coarse map has been discretized enough to represent all obstacles, ensuring a consistent and smooth path during navigation.

Borders. The borders of the maps were represented only to delimit the environment and it is essential to the operation of the algorithm. The border on the coarse map occupies a larger area in the environment than the border on the finest map. The discretization size of the coarse map has to be chosen, ensuring that the whole environment will be represented within the map.

TABLE I

Performance evaluation. Comparing the HBVP Path Planner with the BVP Path Planner using SOR, Gauss-Siedel, and the A algorithm. To compute the A* performance, we considered the largest path found in each environment simulation.*

Resolution	Time (seconds)			
	HBVP PP	BVP PP (SOR)	BVP PP (GS)	A^*
9×9	0.00002924	0.00204756	0.00201016	0.00006588
17×17	0.00023715	0.00210896	0.00361216	0.00021092
33×33	0.00124391	0.00552464	0.031185	0.00055768
65×65	0.0151873	0.0353229	0.488865	0.00170312
129×129	0.0264888	0.290752	7.94992	0.0053646
257×257	0.23992	2.56322	130.323	0.0195956



(a) 17×17 cells (b) 33×33 cells (c) 65×65 cells (d) 129×129 cells

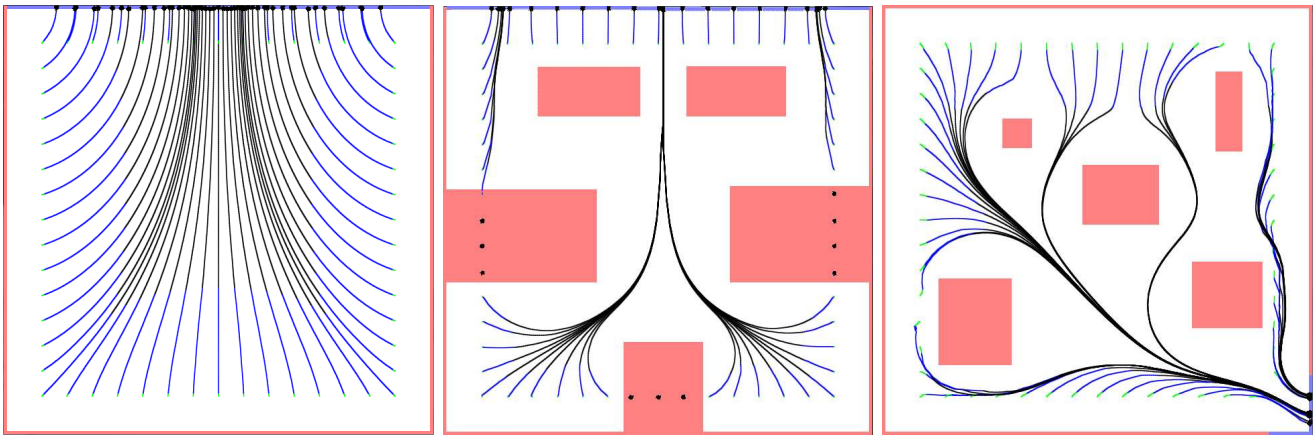
(e) 17×17 cells (f) 33×33 cells (g) 65×65 cells (h) 129×129 cells

(i) 17×17 cells (j) 33×33 cells (k) 65×65 cells (l) 129×129 cells

Fig. 3. Paths produced by BVP Path Planner in 3 different environments.

VI. Conclusions and Future Works

This paper proposed a new hierarchical path planner (HBVP PP) that combines our BVP Path Planner with the Full Multigrid method. The BVP PP generates paths that are smooth and free from local minima. The main weakness of this planner is the computation cost to find a solution. On the other hand, the Full Multigrid method solves elliptic



(a) (b) (c)

Fig. 4. Paths produced by the HBVP Path Planner on the same environments of Figure 3.

PDEs efficiently – as Laplace's Equation – through a combination of solutions at several resolution levels. Basically, this method takes an instance of the problem on a grid of pre-specified fineness and generates coarser grids containing a cruder problem representation. The method solves the problem in the coarsest grid, which is easy and cheaper, and obtains successive solutions on finer and finer grids.

This combination improves the efficiency of the BVP Path Planner expressively. Results in simulation shown that the HBVP PP spends less than 1% of the time needed to compute a solution using the original planner with GS method in several environments. For instance, in an environment with 513×513 cells, the HBVP PP spends only $2s$ to calculate the potential, while the BVP PP spends $22s$ using SOR and $\approx 2,223s$ using GS method. On the other hand, the traditional A^* computes the solution in $0.07s$. However, A^* computes only the path from the current robot cell towards the goal cell, whereas the HBVP computes simultaneously all paths from any environment cell towards the goal cell. Furthermore, the paths produced by HBVP are already smooth, while the paths produced by theA^* need to be smoothed in post-processing, some of them passing very close to obstacles.

In the same way of the BVP PP, the HBVP PP can be adapted and used efficiently in mapping and localization problems [10], [14], [16], [13], as well as in path planning in non-homogeneous terrains[11].

VII. Acknowledgments

The authors gratefully thank the financial funds provided by FAPERGS, and CNPq projects 142485/2008-0, 481292/2008-0, 309092/2008-6, and 303271/2009-4.

References

[1] Brandt, A.: Multi-level adaptive solutions to boundary-value problems. Mathematics of Computation **31**, 330–390 (1977)

[2] Cagigas, D.: Hierarchical d* algorithm with materialization of costs for robot path planning. In: Robotics and Autonomous Systems, pp. 190–208 (2005)

[3] Connolly, C.I.: Harmonic functions and collision probabilities. Int. J. of Robotic Research pp. 497–507 (1994)

[4] Harabor, D., Botea, A.: Hierarchical Path Planning for Multi-Size Agents in Heterogeneous Environments. In: IEEE Symposium on Computational Intelligence and Games, pp. 258–265. Perth, Australia (2008)

[5] Hart, P.E., Nilsson, N.J., Raphael, B.: A formal basis for the heuristic determination of minimum cost paths. In: S.S. Iyengar, A. Elfes (eds.) Autonomous Mobile Robots, pp. 375–382. IEEE Computer Society Press, Los Alamitos, CA (1991)

[6] Hyun, W.K., Suh, I.H.: A hierarchical collision-free path planning algorithm for robotics. In: International Conference on Intelligent Robots and Systems, p. 2488. IEEE Computer Society, Washington, DC, USA (1995)

[7] Jung, D., Ratti, J., Tsiotras, P.: Real-time implementation and validation of a new hierarchical path planning scheme of uavs via hardware-in-the-loop simulation. J. of Intell. and Robotic Systems **54**(1-3), 163–181 (2009)

[8] Nolborio, H., Naniwa, T., Arimoto, S.: A quadtree-based path-planning algorithm for a mobile robot. In: J. of Robotic Systems, vol. 7, pp. 555–574 (1990)

[9] Pai, D., , Pai, D.K., Reissell, L.: Multiresolution rough terrain motion planning. IEEE Trans. on robotics and automation **14** (1994)

[10] Prestes, E., Engel, P.M., Trevisan, M., Idiart, M.A.: Exploration method using harmonic functions. Robotics and Autonomous Systems **40**(1), 25–42 (2002)

[11] Prestes, E., Idiart, M.A.: Sculping potential fields in the bvp path planner. IEEE Int. Conf. on Robotics and Biomimetics (2009)

[12] Prestes, E., Idiart, M.A., Engel, P.M., Trevisan, M.: Exploration technique using potential field calculated from relaxation methods. Intelligent Robots and Systems **4**, 2012–2017 (2001)

[13] Prestes, E., Ritt, M., Fuhr, G.: Improving monte carlo localization in sparse environments using structural environment information. In: Int. Conf. on Intelligent Robots and Systems (2008)

[14] e Silva Junior, E.P., Idiart, M.A., Engel, P.M., Trevisan, M.: Autonomous learning architecture of environmental maps. J. of Intell. and Robotic Systems (2004)

[15] Stentz, A., Mellon, I.C.: Optimal and efficient path planning for unknown and dynamic environments. Int. J. of Robotics and Automation **10**, 89–100 (1993)

[16] Trevisan, M., Idiart, M.A.P., Prestes, E., Engel, P.M.: Exploratory navigation based on dynamic boundary value problems. J. of Intell. and Robotic Systems **45**(2), 101–114 (2006)

[17] Tsiotras, P., Bakolas, E.: A Hierarchical On-Line Path-Planning Scheme Using Wavelets. In: European Control Conference, pp. 2807–2812. Kos, Greece (2007)

[18] Wu, C.P., Lee, T.T., Tsai, C.R.: Obstacle avoidance motion planning for mobile robots in a dynamic environment with moving obstacles. Robotica **15**(5), 493–510 (1997)