12

Composite Models for Mobile Robot Offline Path Planning

Ellips Masehian, M. R. Amin-Naseri
Tarbiat Modares University
Iran

1. Introduction

As new technological achievements take place in the robotic hardware field, an increased level of intelligence is required as well. The most fundamental intelligent task for a mobile robot is the ability to plan a valid path from its initial to terminal configurations while avoiding all obstacles located on its way.

The robot motion planning problem came into existence in early 70's and evolved to a vast and active research discipline as it is today. Numerous solution methods have been developed for robot motion planning since then, many of them being variations of a few general approaches: Roadmap, Cell Decomposition, Potential Fields, mathematical programming, and heuristic methods. Most classes of motion planning problems can be solved using these approaches, which are broadly surveyed in (Latombe, 1991), (Hwang & Ahuja, 1992), and (Choset et al., 2005).

This chapter introduces two new offline path planning models which are founded on the Roadmap and Potential Fields classic motion planning approaches. These approaches have their unique characteristics and strategies for solving motion planning problems. In fact, each one has its own advantage that excels others in certain aspects. For instance, the Visibility Graph yields the shortest path; but its computational time exceeds other methods. Or, while the Voronoi Diagram plans the safest path and is easy to calculate in 2D, it often produces overly lengthy paths, and yields poor results in higher space dimensions. On the other hand, Potential Fields are easy to compute and are suitable for high dimensional problems, but they suffer from the local minima problem, and the oscillating paths generated near narrow passages of configuration space reduce their efficiency. A brief review on these underlying methods is given in this section.

In order to benefit from the strong aspects of these classic path planning methods and compensate their drawbacks, a policy of combining these basic approaches into single architectures is adopted. In devising the new planners it is intended to aggregate the superiorities of these methods and work out efficient and reliable composite algorithms for robot motion planning.

1.1 Roadmap Methods

The Roadmap approach involves retracting or reducing the robot's free Configuration space (C_{free}) onto a network of one-dimensional lines (i.e. a graph). Motion planning is then reduced to a graph-searching problem. At first, two paths are constructed from the start and

Source: Mobile Robots: Perception & Navigation, Book edited by: Sascha Kolski, ISBN 3-86611-283-1, pp. 704, February 2007, Plv/ARS, Germany Open Access Database www.i-techonline.com

goal positions to the roadmap, one for each. Then a path is planned between these points on the roadmap. The correctness of the solution strongly depends on the connectivity of the roadmap representing the entire C-space. If the roadmap does not represent the entire C-space, a solution path may be missed.

The *Visibility Graph* is the collection of lines in the free space that connects a feature of an object to that of another. In its principal form, these features are vertices of polygonal obstacles, and there are $O(n^2)$ edges in the visibility graph, which can be constructed in $O(n^2)$ time and space in 2D, where n is the number of features (Hwang & Ahuja, 1992).

The *Reduced Generalized Visibility Graph* can be constructed in $O(n^3)$ time and its search performed in $O(n^2)$ time. The shortest path can be found in $O(n^2 \log n)$ time using the A* algorithm with the Euclidean distance to the goal as the heuristic function (Latombe, 1991). Works such as (Oommen et al., 1987) and (Yeung & Bekey, 1987) have employed this approach for path planning.

The *Voronoi Diagram* is defined as the set of points that are equidistant from two or more object features. Let the set of input features be denoted as $s_1, s_2, \ldots, s_n$. For each feature s_i, a distance function $D_i(x) = \text{Dist}(s_i, x)$ is defined. Then the *Voronoi region* of s_i is the set $V_i = \{x \mid D_i(x) \leq D_j(x)\ \forall\ j \neq i\ \}$. The Voronoi diagram partitions the space into such regions. When the edges of convex obstacles are taken as features and the C-space is in $\Re^2$, The Voronoi diagram of the C_{free} consists of a finite collection of straight line segments and parabolic curve segments, referred to as *Medial Axis*, or more often, *Generalized Voronoi Diagram* (*GVD*).

In an $\Re^k$ space, the *k-equidistant face* is the set of points equidistant to objects $C_1, \ldots, C_k$ such that each point is closer to objects $C_1, \ldots, C_k$ than any other object. The *Generalized Voronoi Graph* (*GVG*) is the collection of m-equidistant faces (i.e. generalized Voronoi edges) and m+1-equidistant faces (i.e. generalized Voronoi vertices, or, *meet points*). The GVD is the locus of points equidistant to *two* obstacles, whereas the GVG is the locus of points equidistant to m obstacles. Therefore, in $\Re^m$, the GVD is m–1-dimensional, and the GVG, 1-dimensional. In planar case, the GVG and GVD coincide (Aurenhammer & Klein, 2000).

The Voronoi diagram is attractive in two respects: there are only $O(n)$ edges in the Voronoi diagram, and it can be efficiently constructed in $\Omega(n \log n)$ time, where n is the number of features. The Voronoi diagram can be searched for the shortest path in $O(n^2)$ time by using the Dijkstra's method. Another advantage of Voronoi method is the fact that the object's initial connectedness is directly transferred to the diagram (Hwang & Ahuja, 1992). In (Canny, 1985) and (Choset & Burdick, 2000) the Voronoi diagram is used for planning robot paths.

For higher-dimensional spaces than 2D, both the Visibility graph and the Voronoi diagram have higher complexities, and it is not obvious what to select for the features. For example, the Voronoi diagram among polyhedra is a collection of 2D faces, which is not a 1D roadmap (Agarwal et al., 1998).

The *Silhouette* method has been developed at early stages of the motion planning discipline, and is complex to implement. Its time complexity is in $O(2^m)$, where m is the dimension of the C-space, and is mostly used in theoretical algorithms analyzing complexity, rather than developing practical algorithms. A path found from the silhouette curves makes the robot slide along obstacle boundaries (Canny, 1988).

Probabilistic Roadmaps use randomization to construct a graph in C-space. Roadmap nodes correspond to collision-free configurations of the robot. Two nodes are connected by an

edge if a path between the two corresponding configurations can be found by a 'local planning' method. Queries are processed by connecting the initial and goal configurations to the roadmap, and then finding a path in the roadmap between these two connection points (Kavraki et al., 1996).

1.2 The Potential Fields Method

A robot in Potential Fields method is treated as a point represented in configuration space, and as a particle under the influence of an artificial potential field U whose local variations reflect the 'structure' of the free space (Khatib, 1986). In order to make the robot attracted toward its goal configuration while being repulsed from the obstacles, U is constructed as the sum of two elementary potential functions; attractive potential associated with the goal configuration q_{goal} and repulsive potential associated with the C-obstacle region. Motion planning is performed in an iterative fashion. At each iteration, the artificial force induced by the potential function at the current configuration is regarded as the most appropriate direction of motion, and path planning proceeds along this direction by some increment.

The most serious problem with the Potential Fields method is the presence of local minima caused by the interaction of attractive and repulsive potentials, which results in a cyclic motion. The routine method for getting free is to take a random step outwards the minimum well. Other drawbacks are (Koren & Borenstein, 1991):

- No passage between closely spaced obstacles.
- Oscillations in the presence of obstacles or in narrow passages.
- Non-smooth movements of the robot when trying to extricate from a local minimum.
- Overlapping of different obstacles' repulsive potentials when they are adjacent to each other.
- Difficulty in defining potential parameters properly.

Nevertheless, the Potential Fields method remains as a major path-planning approach, especially when high degrees of freedoms are involved. This approach has improved later through a number of works such as (Sato, 1993), (Brook & Khatib, 1999) and (Alvarez et al., 2003) to overcome the problem of getting trapped in local minima.

The next sections of this chapter introduce two new composite models for robot path planning, called *V-P Hybrid*, and *V-V-P Compound*. They are apt to cover the shortcomings of their original methods and are efficient both in time complexity and path quality. Although originally devised for two-dimensional workspaces, they can be extended straightforwardly to 3D spaces. Experiments have shown their strength in solving a wide variety of problems.

2. The V-P Hybrid Model

In this section we present a new algorithm, called *V-P Hybrid*, where the concepts of Voronoi diagram and Potential fields are combined to integrate the advantages of each. In this approach, the initial path planning problem is decomposed to a number of smaller tasks, having intermediate milestones as temporary start and goal points. Through this iterative process the global path is incrementally constructed.

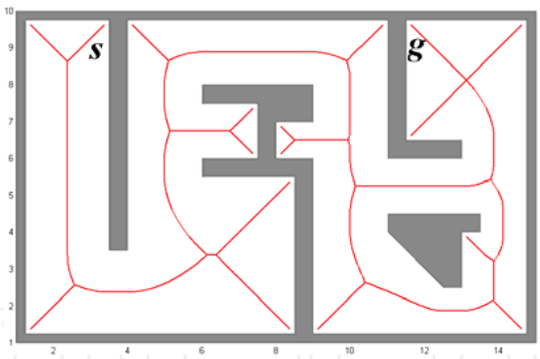
For the path planning task, a number of assumptions are made: (*i*) the map of workspace is known a priori, (*ii*) the obstacles are static, and (*iii*) the robot is considered a point. For real world applications, the latter assumption can be attained by expanding the obstacles using the Minkowski Set Difference method.

The algorithm's major steps are:
(1) *Preprocessing Phase*; consisted of constructing a *Pruned Generalized Voronoi Graph* of the workspace, and then applying a Potential Field to it. This operation yields a network of Voronoi valleys (Sec. 2.1).
(2) *Search Phase*; consisted of implementing a bidirectional steepest descent – mildest ascent search method to navigate through the network of Voronoi valleys. The search phase is designed to progressively build up a start-to-goal path (Sec. 2.2).
Before explaining the details of the composite model, a mathematical representation of some variables is given:

- n : Total number of obstacles' vertices.
- s : The Start configuration.
- g : The Goal configuration.
- $G = (V, E)$: The Generalized Voronoi Graph (GVG) of the C_{free} with the set of vertices (nodes) $V(G)$ and edges $E(G)$.
- $E(v, w)$: The edge connecting vertices v and w , $\forall\, v, w \in V(G)$.
- $N(v) = \{w \mid E(v, w) \neq \varnothing\}$: Neighboring vertices of the vertex v.
- $E(v)$: The set of all edges at vertex v.
- $d(v) = |E(v)|$: The degree of vertex v, equal to the number of passing edges.

2.1 Preprocessing Phase

The V-P Hybrid model starts solving the problem by constructing the Generalized Voronoi Graph (GVG) of the C-space. The Start and Goal configurations are then connected to the main Voronoi graph through shortest lines which are also included in the diagram. Fig. 1(a) provides an example of GVG.



Procedure **PRUNE**(*G, s, g*)
  $P=\{ v \mid v \in V(G) \setminus \{s, g\}, d(v) = 1 \}$
  if ($P = \varnothing$) then Stop
  $V(G) \leftarrow V(G) \setminus P$
  $E(G) \leftarrow E(G) \setminus E(v, N(v)), v \in P$
  PRUNE(G, s, g)
end

Fig. 1. (a) Generalized Voronoi Graph (GVG). (b) Algorithm for pruning the GVG.

The main reason for incorporating the Voronoi concept in the Hybrid algorithm is its property of lying on the maximum clearance from the obstacles. This property helps the robot to navigate at a safe distance from obstacles, making it less prone to be trapped in local minimum wells.
The next step is to exclude redundant or unpromising edges from the GVG. This is done through the *pruning* operation, where the Voronoi edges which either touch obstacle boundaries or have vertices with a degree ($d(v)$) equal to 1 are iteratively truncated. The pruning procedure is explained in Fig. 1(b). Also, the result of this operation performed on

the example of Fig. 1(a) is portrayed in Fig. 2. The resulting subgraph is called *Pruned Generalized Voronoi Graph*, or simply PGVG.

Note that the hypersensitivity of Voronoi diagram to minor inaccuracies in workspace definition which may lead to redundant edges (as in the lower-right disjoint obstacle in Fig. 2(a)) is resolved after running the pruning procedure.

The pruning operation is an important stage in the Hybrid algorithm since it truncates all paths toward collision with obstacles and dead-end traps, and therefore reduces the search space drastically. The resulting graph is a 'lean' network of interconnected Voronoi vertices, including the Start and Goal nodes.

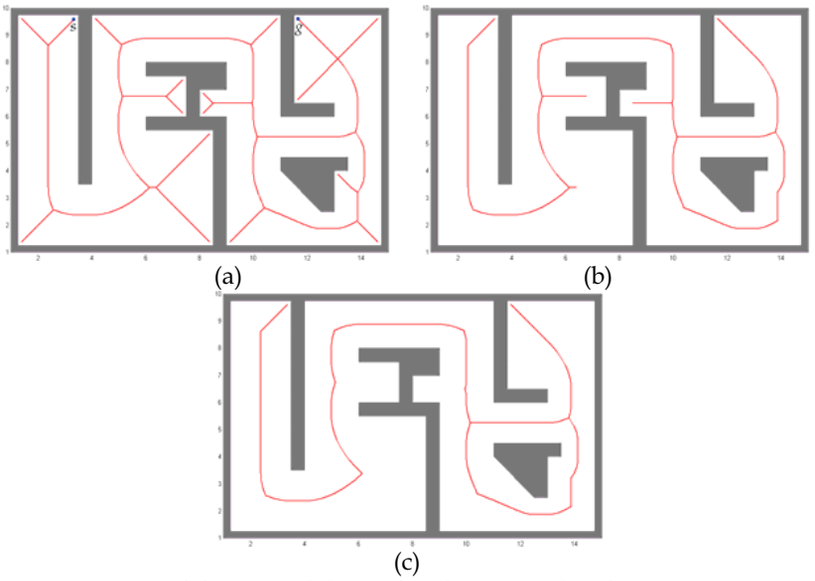


Fig. 2. The construction of the Pruned Generalized Voronoi Graph in two iterations.

The last step of the preprocessing phase is constructing a potential field for guiding the robot toward its goal. Unlike the conventional Potential Fields concept where there are two kinds of attractive and repulsive potentials associated with goal and obstacles respectively, the V-P hybrid algorithm makes use of only attractive potentials, related to the goal and the PGVG. By this, we avoid some known problems of the standard Potential Fields method concerning the calculation of repulsive forces for each obstacle and their integration into a single function, which usually gives rise to complexities due to overlapping and parameter setting (Koren & Bornstein, 1991). This reduces the computational time and memory significantly. Moreover, the problem of narrow corridors, where most Potential Field algorithms give way is fixed in this version.

To apply these potentials, we graduate the configuration space into a grid of fine-enough resolution. For every grid point (x_i, y_i) the potential can then be numerically calculated in a very short time.

As mentioned, the path planning process is decomposed into intermediate stages. So, each stage has its own temporary goal point, g_{temp}. The attractive potential of the goal is exerted through a paraboloid function with a nadir at the temporary goal by:

$$U_{g_{temp}}(x,y) = \xi\left[\left(x - x_{g_{temp}}\right)^2 + \left(y - y_{g_{temp}}\right)^2\right], \quad (1)$$

where ξ is a scaling factor.

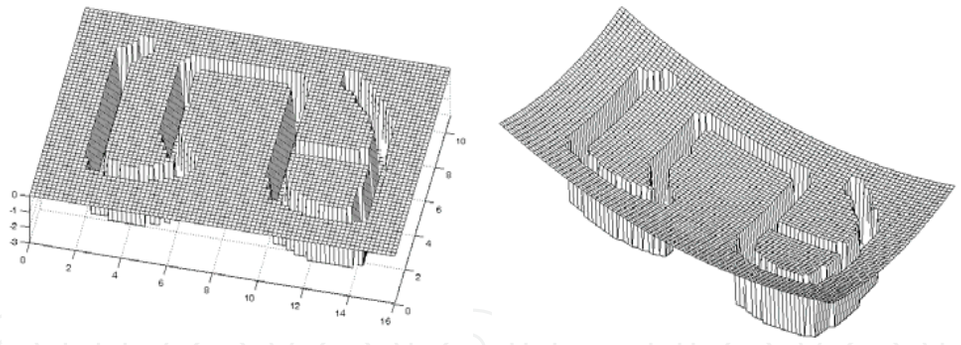
The next attractive potential applies to the PGVG. Because the GVG keeps a safe distance from obstacles the robot will hardly collide with them. Besides, since we prune the GVG such that all Voronoi edges toward obstacles (mainly leading to dead-ends) are eliminated from the graph, the possibility of the robot to get trapped in local minima reduces drastically. So we try to "encourage" the robot to move along the edges of PGVG. This is done by associating an attractive potential with the points on PGVG, which generates a network of deep "valleys" located at the maximum distance from obstacles, with a width of one gridpoint (Fig. 3(a)). The (virtual) robot will safely navigate at the bottom of PGVG valleys. The following function gives the desired result, in which *s* is the depth of valley:

$$U_{PGVG}(x_i, y_i) = \begin{cases} -s & \text{if } (x_i, y_i) \in \text{PGVG} \\ 0 & \text{otherwise.} \end{cases} \quad (2)$$

The U_{PGVG} field is calculated only once and remains constant till the end of the path planning. Instead, the attractive potential of the (temporary) goal is calculated at each iteration and is added to the U_{PGVG} to yield the total potential used for the Search phase by

$$U_{Total} = U_g + U_{PGVG} \quad (3)$$

The resulting manifold is depicted in Fig. 3(b) for a typical temporary goal point. Note that due to the numerical nature of the model, working with these complex functions is extremely easy, and just a simple addition of corresponding grid values is sufficient.



(a) (b)

Fig. 3. (a) PGVG potential valleys for the sample problem (here the width of canals are intentionally aggrandized for a better view). (b) the total potential manifold as the sum of PGVG valleys and goal attractive potentials.

Since the PGVG is a connected roadmap, a path connecting the Start and Goal points (which are located at the bottom of PGVG valleys) certainly exists.

This combination of potentials provides a straightforward and guaranteed attraction from start to goal point. The potential associated with the goal absorbs every point to itself, as the gradient direction at every configuration points to the goal. Note that repulsive potentials are not calculated and consequently all the problems related to them are avoided.

The parameters of the functions such as the valley depth and concavity of the paraboloid should be selected carefully to make sure that the robot will not "escape" from valleys and surmount the obstacles, which are implicitly defined by their high potentials compared to the deeper valleys.
It should be mentioned that the obtained total potential field may still have local minima (e.g. the V-shaped channel left to the center in Fig. 3(b)), but due to the applied search method they are resolved.

2.2 Search Phase

To search within the potential manifold, a bidirectional approach is adopted. First, two trajectory sets, *Traj*(*s*) and *Traj*(*g*), spanned from the Start (*s*) and Goal (*g*) points respectively, are initialized to keep the track of planned paths. Then through an iterative process, the PGVG valleys are being navigated alternately by *Traj*(*s*) and *Traj*(*g*). At each iteration first *Traj*(*s*) and then *Traj*(*g*) extend toward the endpoints of each other. Whenever a trajectory reaches a junction (i.e. a Voronoi vertex) it stops extending more, and the expansion is shifted to the other trajectory. The trajectories meet on the halfway and are concatenated into a single start-to-goal trajectory.
The bidirectional nature of the search requires that for each iteration, the PGVG manifold be numerically added to a paraboloid centered on an intermediate goal point. For instance, when extending *Traj*(*s*), the temporary goal is to reach the endpoint of *Traj*(*g*), which is located on a junction of PGVG valleys.
To maintain the movement of the robot in each iteration, the method of descent search is employed, which is the simplest and fastest searching method in numerical contexts.
The neighborhood of each cell is defined to be 2-neighbors, that is, the points lying in the range of (x±1, y±1) for the point (x, y). The number of neighbors of a cell is thus $3^2 - 1 = 8$. For a k-dimensional space, it would be $3^k - 1$.
The searching begins at Start point, with examining all its neighboring gridpoints. The descent search selects a neighboring cell with the lowest potential among all neighbors as the next configuration. The simple steepest descent method, however, is prone to stop at a local minimum. To cope with this problem, taking ascending steps (or, "hill climbing") is devised for exiting from local minimums. The amount of ascension is kept minimal. Therefore, the concept used here is a "steepest descent, mildest ascent" motion. The hill climbing movement is comparable to the random walk in the randomized planning (Barraquand et al., 1992). Upon reaching a junction, the next edge to navigate is the one having the lowest potential value at that point.
In order to prevent the robot from looping (i.e. infinitely fluctuating between two neighboring cells), we assign to all visited grid cells a relatively higher potential, but still lower than the potentials of points not on the PGVG. Therefore, the robot will not return immediately to a local minimum after it has been once there, simply because it is not a local minimum anymore. The height to which a visited point is elevated is suggested to be less than 1/3 of the valley depth (Fig. 4). This will allow traversing an edge for three times (as in correcting a wrong route) without diverting from the PGVG edges.
The process of the steepest descent - mildest ascent search applied to the example in Fig. 2(c) is shown in Figs. 5(a)-(d). Fig. 5(b) shows iteration 1, navigating from Start toward Goal. The *Traj*(*s*) stops at the first encountered junction (or Voronoi vertex). Fig. 5(c) shows iteration 1, navigating from the Goal point towards the temporary goal, which is now the endpoint of *Traj*(*s*). The *Traj*(*g*) stops at the first encountered junction, which becomes the new

temporary goal. Fig. 5(d) illustrates iteration 2, navigating from endpoint of *Traj*(*s*) toward the temporary goal. The two trajectories *Traj*(*s*) and *Traj*(*g*) are now get connected, and the Search phase is completed. Note the changes in depth of valleys as they are being filled.

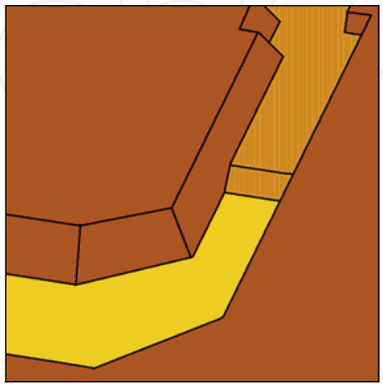


Fig. 4. Valley-filling operation: the potential valley is being filled as the trajectory proceeds.

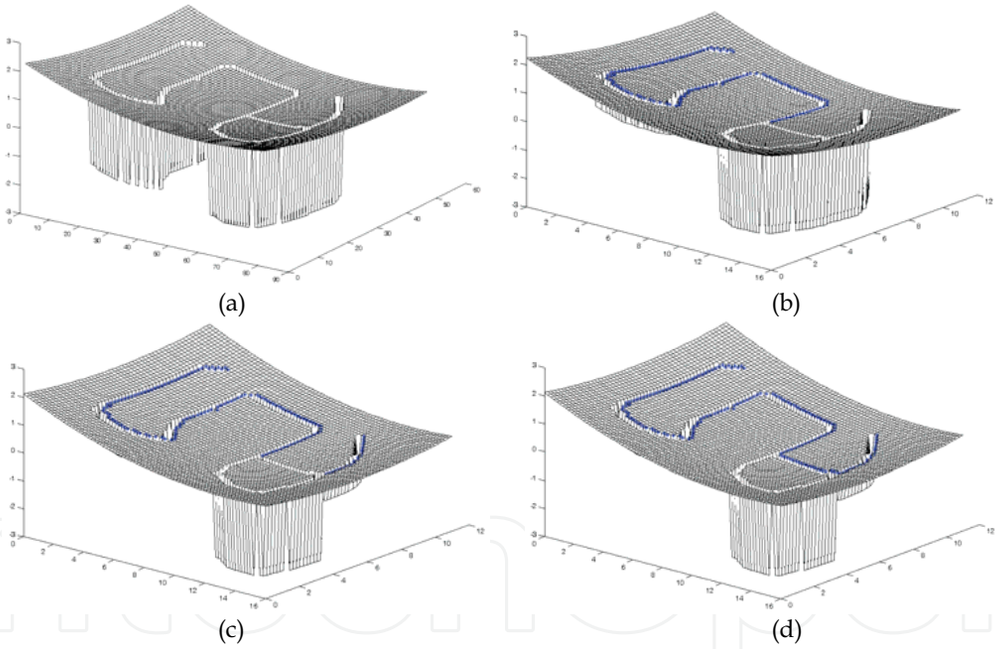


Fig. 5. The process of searching in the V-P Hybrid model is completed in two iterations.

2.3 Experiments

In order to test and evaluate the V-P Hybrid algorithm, 20 problems with obstacles differing in number and shape (including convex, concave, and maze-like problems) were designed and solved by three different methods: the V-P Hybrid, the classical Potential Fields, and the A* Search. Experiments were run on a PC with a 1.4 GHz processor using MATLAB.

Table 1 shows the average values of path lengths (in equal units), CPU time (in seconds) and the number of evaluated grid points computed for the test problems via different approaches. The average length of optimal paths was 27.46 units.

Parameter \ Model	Potential Field Algorithm	A* Search Algorithm	V-P Hybrid Algorithm
Path Length	33.49	30.67	33.62
Search CPU Time	1.0375	19.40	0.0715
Total Examined Grid points	2513	2664.2	331.8

Table 1. Experimental results.

An advantage of the V-P Hybrid algorithm over the classical Potential Fields method is its completeness. While the Potential Fields approach is not guaranteed to generate a valid path (Latombe, 1991), the V-P algorithm is *exact*, i.e. it finds a path if one exists. Since the Goal should be connected to the PGVG at the Preprocessing phase, the algorithm will report any failure in this stage, and so is *complete*.

The V-P Hybrid algorithm has also resolved a number of problems inherent in the conventional Potential Fields method. The local minimum problem is settled by implementing the steepest descent – mildest ascent search method and utilizing the PGVG. Problems due to obstacle potentials and narrow passages are totally fixed.

The Voronoi diagram-Potential Field Hybrid algorithm averagely spent much less time for searching the C-space than the Potential Field method (around 15 times faster). Also the number of examined grid-points was reduced about 7.5 times for the Hybrid algorithm. We ascribe these results to the efficient abstraction of workspace due to the pruning procedure where most local minimum wells are excluded from the search space. The number of Voronoi vertices is also reduced effectively. The pruning procedure together with the fast searching of Voronoi valleys made the V-P model successful in solving complex and labyrinthine, maze-like workspaces. In sparse environments the Potential Fields found slightly shorter paths, but for maze-like problems the Hybrid algorithm outperformed.

The time complexity of A* search is $O(n^2)$ (Latombe, 1991). A* is complete and optimal, but its space complexity is still prohibitive. The A* search employs a heuristic function for estimating the cost to reach the goal. For our experimentation a Euclidean straight-line distance was used as the heuristic. The Hybrid algorithm searched the grid space very much faster than A* search (270 times), examining around 8 times less points than it. This is because of the lower time complexity order of the Hybrid method compared to the $O(n^2)$ of A*. However, the quality of the path generated by A* is better than the Hybrid model by %10. The Hybrid algorithm also outperforms the Dijkstra's algorithm which has an $O(n^2)$ time complexity. The time complexity of the V-P Hybrid algorithm is discussed below.

2.4 Time Complexity Analysis

For a time complexity analysis of the V-P Hybrid algorithm, its two phases must be analyzed separately. Time complexities of constructing and pruning the Voronoi graph, as

well as the potential field calculation determine the computational burden of the Preprocessing phase. To evaluate this, we first need to study the problem's size. The following two lemmas deal with this issue:

Lemma 1. The Voronoi diagram has $O(n)$ many edges and vertices, in which n is the number of Voronoi sites.

Lemma 2. The average number of edges in the boundary of a Voronoi region is bounded by 6.

Proofs to these lemmas are provided in (Aurenhammer & Klein, 2000). The proofs are originally developed for the case of points or convex objects taken as Voronoi sites. However, since due to the pruning procedure any non-convex obstacle is located in a unique connected Voronoi region, the above lemmas hold true for non-convex cases as well.

The direct consequence of the Lemma 1 is that the Hybrid algorithm must perform $O(n)$ neighborhood checks for pruning the Voronoi Graph. Therefore, considering that the construction of the Generalized Voronoi Diagram takes $O(n\log n)$ time, we conclude that the Pruned Generalized Voronoi Diagram is built in $O(n\log n)$ time.

For the potential field calculation, since we do not need to calculate the potential values for *all* gridpoints, save for those located on the PGVG, it is essential to have an estimate for the number of gridpoints on the PGVG.

Assuming that after graduating the C-space the PGVG edges are divided into small intervals of size λ, each PGVG edge with vertices v and w will have grid points equal to $e = \frac{|E(v,w)|}{\lambda}$. Considering the $O(n)$ edges of the C-space, the number of all grid points would be $O(e \times n) \equiv O(n)$, which also gives the complexity of potential field calculation.

For obtaining an average-space complexity, the average length of the PGVG edges should be computed. Let m be the total number of configuration gridpoints, o the number of configuration gridpoints occupied by obstacles, and b the number of obstacles. Then the average number of C-points around an obstacle (Voronoi region) is $(m-o)/b$. Since the average number of edges around each obstacle is bounded by 6 (Lemma 2), we will assume that the typical shape of the region is hexagonal, with the surface area of $S = 3\sqrt{3}a^2/2$, where a is the edge of the hexagon (Fig. 6). By setting this surface area equal to the average number of C-points in a Voronoi region, we get

$$a = \frac{1}{3}\sqrt{\frac{2\sqrt{3}(m-o)}{b}} \cong 0.62\sqrt{\frac{(m-o)}{b}}. \tag{4}$$

Since $o < m$ in (4), we conclude that the average length of a Voronoi edge in terms of its number of gridpoints is in $O(\sqrt{m})$. This means that the number of points whose potentials are to be computed is in $O(\sqrt{m})$, where m is the total number of gridpoints.

The above space complexity can also be used for calculating the time complexity of the Search phase. Since only the gridpoints on the PGVG need to be searched, and the average number of these points is $O(\sqrt{m})$, the Search phase averagely will take $O(\sqrt{m})$ time to navigate the PGVG and accomplish the search. This result is superior to the conventional Potential Field's search which contains a neighborhood checking operation and is carried on in $O(m)$, m being the number of C-points (Latombe, 1991).

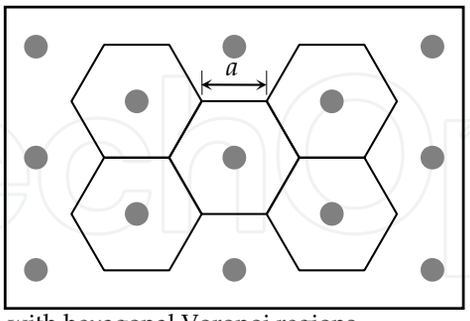


Fig. 6. A typical problem with hexagonal Voronoi regions.

To conclude, the Preprocessing phase of the algorithm takes $O(n\log n)$ time (n being the total number of obstacle vertices), which is due to construction of the GVG. The remaining components of the algorithm, i.e. the pruning, potential calculation, and potential search procedures all have linear or sub-linear time complexities. Since these components are executed sequentially, the most time-consuming operation will be bound to $O(n\log n)$ time, which is the total time complexity.

3. The V-V-P Compound Model

Since the paths generated by the V-P Hybrid model are a subset of the Generalized Voronoi Graph of the workspace, they have lengths identical to the ones generated by the Voronoi Diagram method. The Voronoi paths are longer than the optimal Visibility Graph-based paths, especially in sparse environments. Aiming to improve the quality of generated paths, another composite algorithm is proposed (Masehian & Amin-Naseri, 2004) where three methods of Voronoi Diagram, Visibility graph, and Potential Fields are integrated in a single architecture, called *V-V-P Compound* model.

The Compound model provides a parametric tradeoff between the safest and shortest paths and generally yields shorter paths than the Voronoi and Potential field methods, and faster than the Visibility graph. In the proposed model, positive attributes of these three path planning techniques have been combined in order to benefit from the advantages of each. To accomplish this, they are tailored and associated with a number of complementary procedures to generate a valid and high quality path. Hence, the Compound algorithm borrows its name, V-V-P, from these basic techniques, although the outcome is a new and different model as a whole.

An overview of the model is as follows: after constructing the PGVG, a network of broad freeways is developed through a new concept based on medial axis, named *MID*. A potential function is then assigned to the freeways to form an obstacle-free network of valleys. Afterwards we take advantage of a bidirectional search, where the Visibility Graph and Potential Field modules execute alternately from both Start and Goal configurations. A steepest descent – mildest ascent search technique is used for local planning and avoiding local minima. The assumptions on which the model is principally developed are the same as for the V-P Hybrid model; that is, the workspace is considered two-dimensional, and the map of workspace is known a priori. Similar to the Hybrid model, the Compound model has also two major stages: the Preprocessing phase and the Search phase. The Search phase contains two modules: *Visibility*, and *Potential Field*, which are executed alternately, as illustrated in Fig. 7.

The main differences between the V-V-P Compound and V-P Hybrid models are the width of the potential valleys and their filling technique. Additionally, the V-V-P model employs a Visibility module to obtain shorter paths than the V-P model. The description of algorithm's phases is presented in the next subsections.

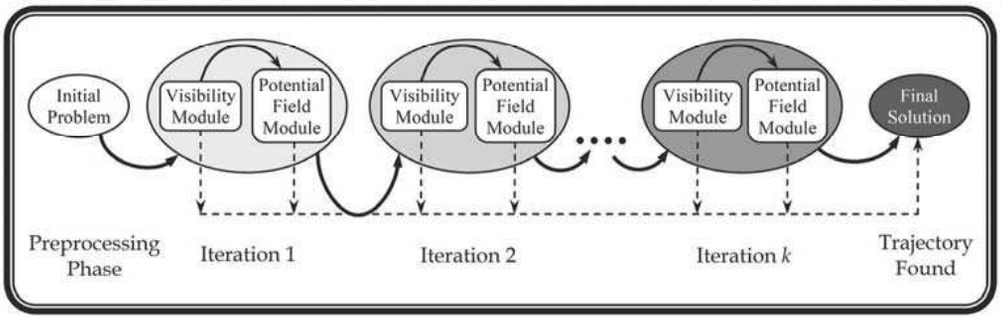


Fig. 7. The overall process of problem solving in the V-V-P path planning model. Each iteration in search phase is comprised of two sequentially executed modules, Visibility and Potential Field. The gradually darkening shades imply the completion of a solution.

3.1 Preprocessing Phase

This phase establishes an obstacle-free area for robot navigation. The main steps are:

P1) Constructing the PGVG of the workspace (as described in Sec. 2.1).

P2) Forming an obstacle-free C-space region based on PGVG points.

P3) Associating an attractive (negative) potential to that region. The result is an obstacle-free network of valleys as the robot's navigation area.

As noted in Sec. 1.1, the Generalized Voronoi Graph is also known as *Medial Axis* (*MA*). Voronoi diagram lies on the maximum clearance of objects. Although this property offers some advantages regarding to path safety, it makes the path longer, especially in workspaces where the obstacles are located quite far from each other. Besides, the generated path usually has sharp angles at Voronoi vertices, making it ineffective for robots with nonholonomic or rotational constraints.

In order to compensate these shortcomings, unlike the 1-pixel-wide valleys in the V-P model, a network of "wider" channels is built based on PGVG. These channels are "dilated" Voronoi edges that provide sufficient space for the robot to plan shorter paths and maneuver freely. Due to the varying sizes of inter-obstacle free spaces, the widths of these channels must vary from region to region.

For constructing this obstacle-free network of channels the *Maximal Inscribed Disc* (*MID*) concept is incorporated. First some definitions are presented:

A *Locally Maximal Disc* (*LMD*) of the point $x \in C_{free}$ is the set of points such that:

$$LMD(x) = \left\{ q \;\middle|\; \|x - q\| \le \mathrm{Min}\|x - \partial C_{free}\|,\ q \in C_{free} \right\}, \tag{5}$$

and denotes a disc centered at x and tangent to the nearest obstacle boundary (∂C_{free}).

The *Maximal Inscribed Disc* (*MID*) is defined as:

$$MID(x)=\left\{LMD(x)\middle|\, r_{LMD(x)}>r_{LMD(y)},\ x\in MA\wedge y\in N(x)\right\}, \tag{6}$$

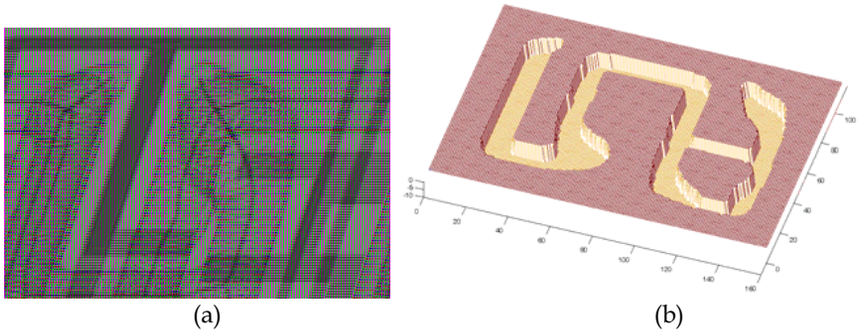
in which the $r_{LMD(x)}$ is the radius of the $LMD(x)$, and $N(x)$ is the neighborhood of x.
For the Compound model, only the radii of all *MID*s centered on PGVG points are calculated. Also, in order to maintain a safe distance from obstacle borders, *MID*s radii are multiplied by a lessening factor α (α ∈ [0, 1]), to produce α*MID*s defined as:

$$\alpha MID(x)=\left\{LMD(x)\middle|\, r_{LMD(x)}=\alpha\times r_{MID(x)},\ x\in MA, 0\le\alpha\le 1\right\} \tag{7}$$

All α*MID*s are integrated in a connected region called *Region*(α*MID*). The *Region*(α*MID*) of a C-space is the union of all α*MID*s centered on the medial axis:

$$Region(\alpha MID)=\left\{\bigcup_{\forall x\in MA}\alpha MID(x)\right\} \tag{8}$$

The *Region*(α*MID*) is obstacle-free and non-convex, and reflects the topology of the C_{free}. An interesting property of the α is that it offers a balance between the Roadmap and full C_{free} concepts. If we set α=0, the *Region*(α*MID*) will turn into the medial axis roadmap. For α =1, the region's borders will be tangent to obstacles. Based on experiments, we recommend α ∈ [0.5, 0.8].
The *Region*(α*MID*) for the workspace of Fig. 2(c) is calculated and depicted in Fig. 8(a). Fig. 8 also indicates the property of *Region*(α
MID) in smoothening the Voronoi roadmap's sharp corners and local irregularities.



(a) (b)

Fig. 8. (a) The *Region*(α*MID*) is comprised of α*MID*s centered on the medial axis. Here the α is set to 0.6. (b) Attractive potentials associated with the *Region*(α*MID*).

Similar to the Hybrid model, the Compound model also creates a network of navigable valleys. It assigns attractive potentials to the points lying in *Region*(α*MID*):

$$U(x_i,y_i)=\begin{cases}-s & \text{if } (x_i,y_i)\in Region(\alpha MID)\\ 0 & \text{otherwise.}\end{cases} \tag{9}$$

The preprocessing phase terminates with the construction of potential valleys.

3.2 Search Phase

This phase is designed to progressively build up a Start-to-Goal path. The initial problem is decomposed to a number of smaller path planning tasks, having intermediate milestones as

temporary start and goal points. Through this iterative process the solution path is incrementally constructed, and the algorithm becomes capable to resolve more complex problems.

Similar to the V-P Hybrid, the global search process is performed bidirectionally. Again we initialize the two trajectories *Traj*(*s*) and *Traj*(*g*), and set *Traj*(*s*) = {*s*} and *Traj*(*g*) = {*g*} for the beginning.

The main modules included in this phase are *Visibility* and *Potential Field*, which are executed iteratively until the construction of the final path. The termination condition is satisfied when *Traj*(*s*) and *Traj*(*g*) are either being seen or get in touch with each other. We characterize 'being seen' as being able to draw a straight line in free space to connect the two trajectories' endpoints.

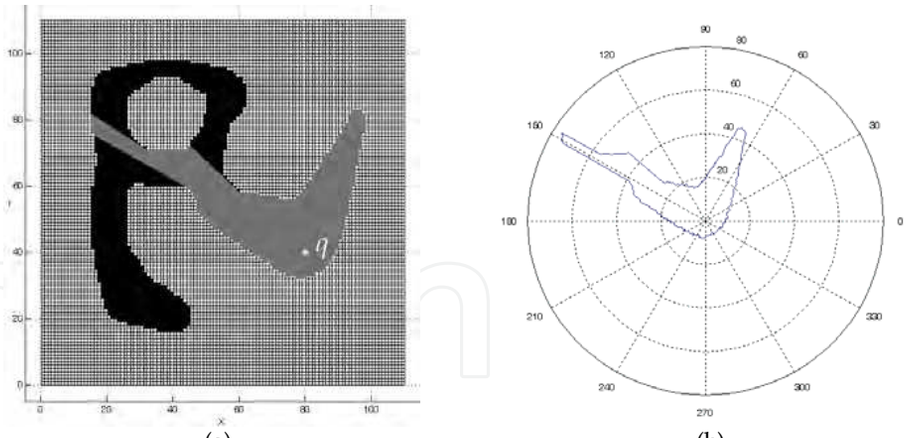
The following subsections describe the Visibility and Potential Field modules.

3.2.1 Visibility Module

Each iteration of the Search phase starts with a *Visibility scan* performed concurrently for both endpoints of *Traj*(*s*) and *Traj*(*g*). For this purpose, a "ray sweeping" technique is used to collect information about the surrounding valley borders and probably the opposite trajectory.

The aim of this procedure is to determine whether the opposite trajectory is visible from the current point or not. If it is visible, then the Search phase is over. If not, we have to find the boundary vertices as seen from the current point, as described below.

By applying a polar coordinate system with the origin defined on the vantage point (e.g. endpoint of *Traj*(*s*)), the radial Euclidean distances to valley borders (∂C_{free}) are calculated for [0, 2π] and integrated in an array (i.e. *Visibility Polygon*). Fig. 9(a) shows the C_{free} valleys and the point (*q*) considered for visibility scan in a sample problem. Fig. 9(b) shows the distance of that point from its surroundings.



(a) (b)

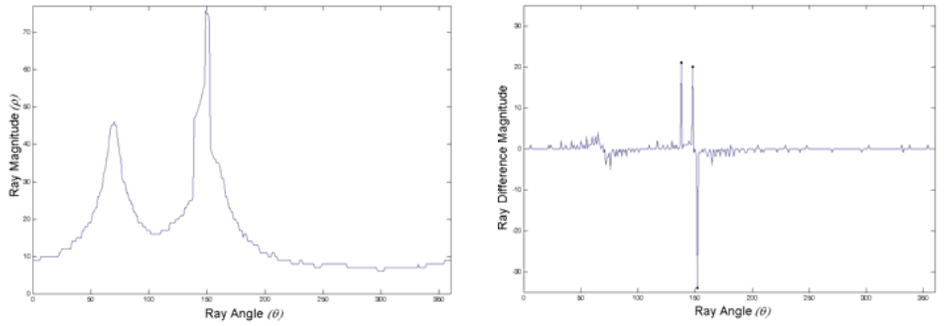
Fig. 9. (a) Visible configurations (visibility polygon) as a result of a visibility scan performed for the point *q*. (b) The polar representation of radial distances (i.e. ray magnitudes) of the point *q* from C_{free} boundary (∂C_{free}).

Subsequent to the calculation of distances (ρ) between the vantage point and ∂C_{free} for any angle ($\theta \in [0, 2\pi]$), this data is mapped into Cartesian coordinates (Fig. 10(a)).

Since the C_{free} boundary generally has a complex geometrical shape and lacks definite vertices as in polygonal objects, we take advantage of the ray sweeping data to determine the boundary points being tangent to any ray emanated from the vision source point. A ray is *tangent* to ∂C_{free} if in the neighborhood of their contact point the interior of C_{free} lies entirely on a single side of it.

In order to find the tangent rays and their touching boundary points, we apply a difference function for successive adjacent rays. We define the *Ray Difference* variables as $\Delta\rho_\theta = \rho_{\theta+1} - \rho_\theta$ for $\theta \in [0, 2\pi]$ and collect them in an array plotted in Fig. 10(b). By applying a notch filter, the peaks of the Ray Difference array are determined. These peaks imply abrupt and large differences in successive ray magnitudes and therefore indicate the points where sweeping rays leave (positive peaks) or meet (negative peaks) a convex contour on ∂C_{free}, based on anticlockwise rotation of rays.

The boundary points corresponding to the tangent rays are treated as boundary vertices visible from the vantage point, q. These points are called *Critical points* and form the set $R(q)$ (see step S1(d)). The tangent rays and critical points are shown in Fig. 11.



(a) (b)

Fig. 10. (a) The Cartesian representation for the ray magnitudes of Fig. 9. (b) Magnitude difference of sweeping rays for successive angles. The three peaks show tangent rays.

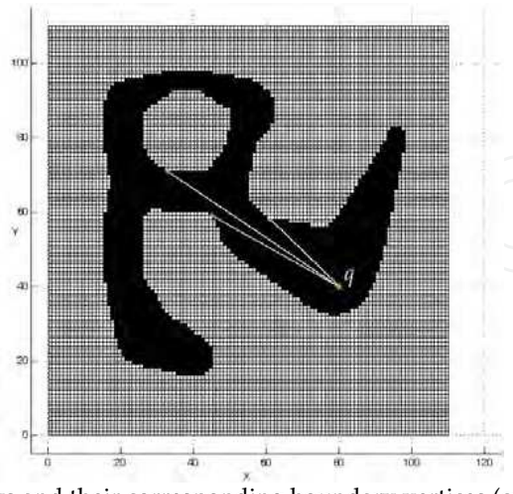
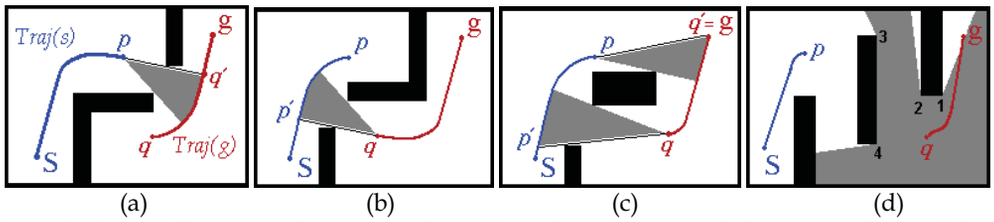


Fig. 11. The tangent rays and their corresponding boundary vertices (critical points).

By concurrently implementing the visibility scan for both ends of *Traj*(*s*) and *Traj*(*g*), we discover that either there exists a line which connects the two trajectories (and lies entirely in C_{free}), or none of them is within the scope of the other's endpoint. If the first case holds then the search phase terminates. For the latter case, critical points of the two sets *R*(*p*) and *R*(*q*) are calculated and matched to find the closest pair, one point from each. These points determine the two positions which the two trajectories must extend toward.
The following steps are taken for the Visibility module:

S1) Performing *Visibility scan*. The scan is concurrently implemented for the endpoints of both *Traj*(*s*) and *Traj*(*g*).

Suppose that the visibility scan operation is performed from *p* and *q*, the endpoints of *Traj*(*s*) and *Traj*(*g*), respectively. Consequently, four incidences may occur (Fig. 12):



(a) (b) (c) (d)

Fig. 12. Four different combinations of *Traj*(*s*) and *Traj*(*g*) in visibility scan. The visibility envelope is shown in grey.

(*a*) A subset of points in *Traj*(*g*) is visible from *p*, but no point from *Traj*(*s*) is visible from *q* (Fig. 12(a)). In this case, by a straight line, connect *p* to a visible point in *Traj*(*g*), say *q*', which is nearest to the Goal (i.e. has the smallest ordinal rank in *Traj*(*g*) among the visible points), and truncate all elements in *Traj*(*g*) located after *q*'. Note that the Goal point might be visible itself, which in that case point *p* is directly connected to the *g* (Fig. 12(c)).

(*b*) A subset of points in *Traj*(*s*) is visible from *q*, but no point from *Traj*(*g*) is visible from *p* (Fig. 12(b)). This is the reverse of the previous case, so act similarly, but swap *p* and *q*, and also *Traj*(*s*) and *Traj*(*g*).

(*c*) Subsets of points in both *Traj*(*g*) and *Traj*(*s*) are visible from *p* and *q*, respectively (Fig. 12(c)). In this case, define the following criterion *C* as:

$$\begin{aligned} C &= \mathrm{Min}\left\{\left|spq'g\right|,\left|gqp's\right|\right\} \\ &= \mathrm{Min}\left\{\left|Traj(s)\right|+\left\|p-q'\right\|+\left|q'\in Traj(g)\right|,\left|Traj(g)\right|+\left\|q-p'\right\|+\left|p'\in Traj(s)\right|\right\} \end{aligned} \tag{10}$$

where $\left|Traj(s)\right|$ means the cardinality (or length) of *Traj*(*s*), $\left\|p-q'\right\|$ is the Euclidean distance of *p* and *q*', and $\left|q'\in Traj(g)\right|$ indicates the ordinal position of *q*' in *Traj*(*g*) (i.e. the distance of *q*' to *g* via the *Traj*(*g*)). Among *pq*' and *qp*', the line providing the minimum value for the above criterion will be selected to connect *Traj*(*s*) and *Traj*(*g*). Again truncate the elements of the trajectory located after the connection point *p*' or *q*', according to the drawn line.

(*d*) If none of the *Traj*(*s*) and *Traj*(*g*) are visible to each other's endpoints, then for both *p* and *q*, determine those rays that are tangent to visible C_{obs} boundary. Note that this boundary is at a safe distance from actual obstacles' edges. The intersection of these rays and the free space's boundary produces two sets of *Critical Points*, *R*(*p*)

and $R(q)$. Fig. 12(d) shows the result of visibility scan from q, which consequently renders 4 visible obstacle vertices in $R(q) = \{1, 2, 3, 4\}$.
Now among all combinations of the elements of $R(p)$ and $R(q)$, select the closest x and y pair meeting the following condition:

$$\{(x, y) \mid \forall\, x, u \in R(p);\, y, v \in R(q);\, \|x - y\| \leq \|u - v\|\}, \tag{11}$$

where $\|\bullet\|$ shows Euclidean distance. The total number of combinations to be evaluated is $|R(p)| \times |R(q)|$, where $|\bullet|$ is the cardinality of sets. This operation determines the mutually best points that $Traj(s)$ and $Traj(g)$ must extend toward via two straight lines.

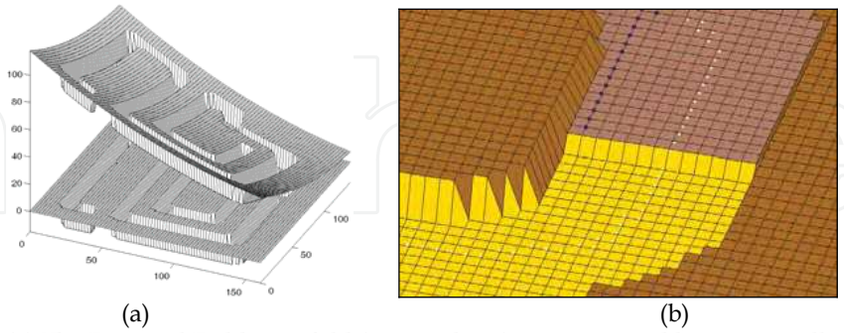
S2) Map the line segment(s) found in step S1 to the configuration space grid. Through a fine-enough discretizing operation, new points are added to $Traj(s)$ and/or $Traj(g)$.

If any of the cases (*a*), (*b*), or (*c*) in step S1 holds, then terminate the Search phase and go to step S10 (Sec. 3.2.2). For the case (*d*) continue with the next step.

S3) Since all the points in $Traj(s)$ and $Traj(g)$ lie on the bottom of roadmap valleys, in order to mark the valleys as traversed, increase the potentials of trajectory points and their surroundings to 'fill' the width of valleys (Sec. 3.2.2). This is an effective operation for preventing the planner from searching the C_{free} exhaustively.

3.2.2 Potential Field Module

The bidirectional nature of the V-V-P algorithm requires that for each iteration, the valley potentials manifold be numerically added to a paraboloid with a nadir on a temporary goal point (see step S4). For instance, when extending $Traj(s)$, the temporary goal is the endpoint of $Traj(g)$, and vice versa. To apply the paraboloid potential, we graduate the configuration space in a fine-enough resolution, then assigning every grid cell as (x_i, y_i), the potential is calculated numerically. Fig. 13(a) shows the Potential Field manifold superimposed on the 'flat' valley potentials manifold.
As soon as new points are appended to the trajectories, the navigated valleys must be distinguished by 'elevating' their potentials in order to prevent the robot to re-traverse them later (Fig. 13(b)).

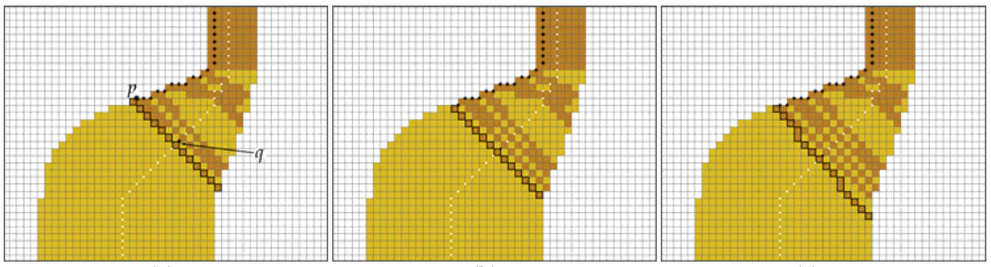


(a) (b)

Fig. 13. (a) The Potential Field manifold (upper object) is constructed by numerically adding a paraboloid function defined in (1) to the valley potentials manifold (lower object). (b) A scene from an intermediate iteration in potential search. Trajectory points are shown black and the medial axis points are in white.

The valley filling technique is somehow a "micro-visibility" process; it marks the neighboring configurations as 'seen', and excludes them from the search space. This process is analogous to walking in a long corridor while trying to get out by reaching an open door or a junction. Naturally one does not consider the tiles across the corridor and near his feet as promising cells leading to a desired destination. Rather, he deems those points as traversed (though physically not indeed), and continues his wall-following motion. This is done in filling technique by 'elevating' the potentials of those cells, making them less attractive. Since in a steepest descent context the robot occupies the cell with the least potential value across the valley, the filling procedure does not affect the path length adversely.

The filling procedure is applied immediately after a new point is appended to a trajectory. So it is performed in a layer-by-layer manner. Suppose that a point p is just being added to an existing trajectory array (Fig. 14(a)). In order to 'mark' and elevate the potentials of visited cells across the C_{free} valley, we must find a line passing from p and perpendicular to the local direction of the channel. To do this, the point p must be connected to its nearest point q on the medial axis (skeleton) of the valley. By interpolation and extrapolation, the cells along this line are found and increased in potential. The amount of this increase is proposed to be about 1/3 of the valley depth (i.e. s in (9)). Fig. 14 shows three consecutive iterations of filling operation.



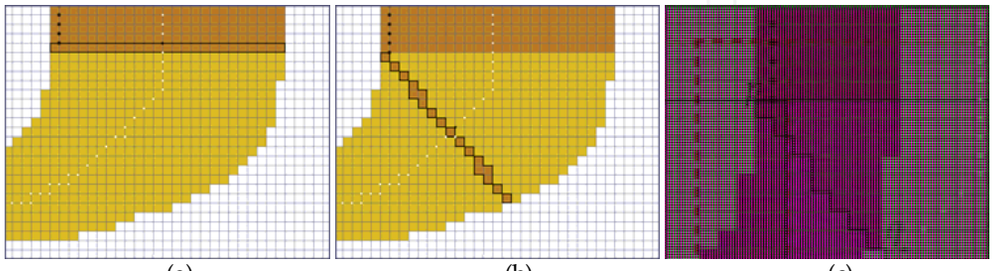
(a) (b) (c)

Fig. 14. Three iterations from the valley filling process. As new points (black dots) are appended to the trajectory, the cells across the channel are elevated in potential, so that the planner is encouraged to move along the valley's main direction. Points on the medial axis are shown white, except for the point q which is nearest to trajectory's endpoint p (shown in black). The elevated rack is highlighted in each iteration.

For a better understanding of the role of this process, imagine that an attractive potential (i.e. a local minimum) is located in the upper-end of the narrow channel in Fig. 14(a). Then according to the *steepest descent* search, the trajectory points should move towards it, which is of course hopeless. However, the elevated barrier created in each iteration blocks this motion, and forces the planner to take a *mildest ascent* step and run off the fatal situation.

For channels of uniform width this method fills the cells thoroughly and compactly, but it may cause porosities in curved and bent valleys, or leave unfilled areas behind, as in Figs. 14 or 15(b). The case in Fig. 15(b) arises from the fact that for two successive trajectory points their respective nearest medial axis points are not adjacent. Although this does not cause a serious problem most of the time, we will present a variation to this procedure to overcome such conditions:

First a square (or rectangular) frame with a symmetrical center on the medial point q is defined (the dashed line in Fig. 15(c)). This frame is partitioned into two hyper-planes by the connecting line pq. The hyper-plane that contains the penultimate trajectory point is therefore the 'backward' region which may contain some unfilled cells. Then, the potentials of the cells confined within the frame and valley border are elevated. The magnitude of this frame can be set such that all the unfilled cells can be covered. However, a size equal to the valley width in that point suffices. The still unfilled area at the right of Fig. 15(c) will not cause any problem since it is far from trajectory points.



(a) (b) (c)

Fig. 15. An unfilled area is originated from the fact that for two successive trajectory points, their respective nearest medial axis points are not adjacent. To resolve this problem, a frame is defined around the medial point q (drawn by dashed line), and the unfilled area confined within this frame is elevated in potential.

The implemented valley filling routine provides some advantages for the model:

(1) It reduces the potential searching time significantly by discarding the configurations in C_{free} which have normal vectors pointing toward a local minimum, and so obviates the random or 'Brownian' movements.

(2) This technique enables the planner to perform a 'hill climbing' operation for coping with the attraction of a nearby local minimum, and as such, is a subtle way to avoid exhaustively filling up dead-end or saddle point regions and the consequent path smoothing operations (Barraquand et al., 1992).

For more clarification, suppose that the planner incrementally builds up a search tree and adopts a 'best-first' strategy to find the goal point. This task becomes time-consuming when the tree has many branches. The valley filling process curtails most of the non-promising branches and directs the planner along an effective branch leading to another valley. In other words, this technique converts a 'breadth-first' or 'best-first' search into a 'depth-first' search.

Experiments showed that the valley filling process aids the robot considerably especially in departing from deep local minimum wells.

Now the Potential Field module is executed according to the following steps. It is applied in two directions: first the $Traj(s)$ is extended (steps S4 to S6), then $Traj(g)$ is stretched out (step S7 to S9).

S4) Setting the endpoint of $Traj(g)$ as the temporary goal (g_{temp}), construct an attractive field by the paraboloid function introduced in (1). Then add this potential to the potential of $Region(\alpha MID)$ calculated in step P3 (Sec. 3.1).

S5) Now the steepest descent – mildest ascent search is performed with setting the endpoint of $Traj(s)$ as temporary start and the endpoint of $Traj(g)$ as temporary goal

point. This step contains a gradient search for selecting the next gridcell to proceed. New points are appended to *Traj*(*s*). Also, in order to provide a mechanism for escaping from local minima, perform the valley filling procedure.

S6) Repeat the step S5 until one of the following situations take place:

(*a*) If before the occurrence of case (*b*) below, the endpoint of *Traj*(*s*) meets any point in opposite trajectory *Traj*(*g*), the search phase is completed. First truncate the elements of *Traj*(*g*) located after the connection point, then go to step S10.

(*b*) The gridcell wavefront distance between the endpoint of *Traj*(*s*) and the free space boundary, ∂C_{free}, exceeds a certain limit, i.e. $|\text{END}(Traj(s)) - \partial C_{free}| > d$. Through experimentations $d = 3$ was found appropriate.

The steepest descent search for *Traj*(*s*) is now terminated and the searching process is shifted to steps S7 to S9, where *Traj*(*g*) is being extended towards *Traj*(*s*).

S7) This step is similar to step S4, except that the paraboloid which is added to the *Region*(α*MID*) valleys has a minimum on the endpoint of *Traj*(*s*).

S8) Setting the endpoints of *Traj*(*g*) and *Traj*(*s*) as temporary start and goal points respectively, perform a steepest descent – mildest ascent search, as well as the valley filling procedure, as described in step S5.

S9) Repeat the step S8 until either of the following cases happen:

(*a*) If before the occurrence of case (*b*) below, the endpoint of *Traj*(*g*) meets any point in *Traj*(*s*), the search phase is completed. Truncate the elements of *Traj*(*s*) located after the connection point, then go to step S10.

(*b*) If the gridcell wavefront distance between the endpoint of *Traj*(*g*) and the ∂C_{free} exceeds a certain limit, i.e. $|\text{END}(Traj(g)) - \partial C_{free}| > 3$, terminate the Potential Field module and start the next iteration from step S1, the Visibility module.

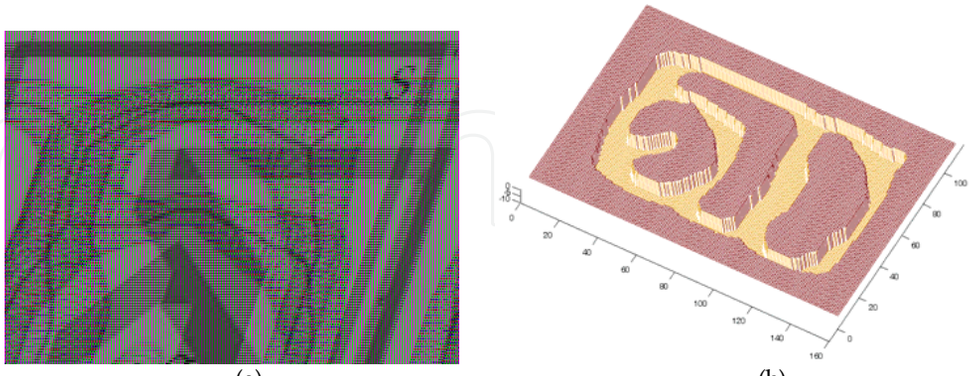
S10) Reverse the order of elements in *Traj*(*g*) and concatenate it to the endpoint of *Traj*(*s*). As a result, a single start-to-goal trajectory is achieved which is the final output of the V-V-P algorithm.

3.3 An Example

Now the algorithm's path planning technique is demonstrated through solving a problem illustrated in Fig. 16(a).

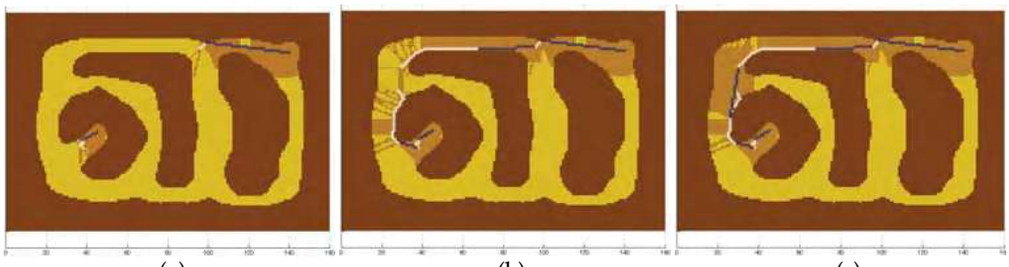
After preparing the valley potentials (Fig. 16(b)), the Search phase is accomplished in 3 iterations. The bidirectional progression of trajectories is clearly shown in Figs. 17(a)-(c). The C_{free} region is light-colored, and the 'filled' area has a darker shade. Fig. 17(a) indicates the development of *Traj*(*s*) (upper-right), and *Traj*(*g*) (lower-left) trajectories in iteration 1, by first performing a visibility scan, then a Potential Field search. The visibility scan matches with case S1(*d*), where none of the two trajectories is in the scope of another. Hence, 6 possible pairs of critical points ((2 for *g*) × (3 for *s*)) are evaluated and the closest pair is selected as the destination of trajectories. The filling procedure is then implemented for the drawn lines (darker area in C_{free}) according to step S3.

The Potential Field module now starts with performing a steepest descent – mildest ascent search from the endpoint of *Traj*(*s*) toward the endpoint of *Traj*(*g*), the temporary goal. This requires a superimposition of a paraboloid function with a minimum on END(*Traj*(*g*)) on the 'flat' potential manifold in Fig. 16(b) (as described in step S4). This search generates points directed to the temporary goal, elevates the potentials across the current valley, and stops after a few repetitions upon detaching enough from the ∂C_{free} (case S6(*b*)). These points are appended to *Traj*(*s*).



(a) (b)

Fig. 16. (a) The PGVG and *Region*(α*MID*) (Step P2). (b) Obstacle-free network of valley potentials (Step P3).



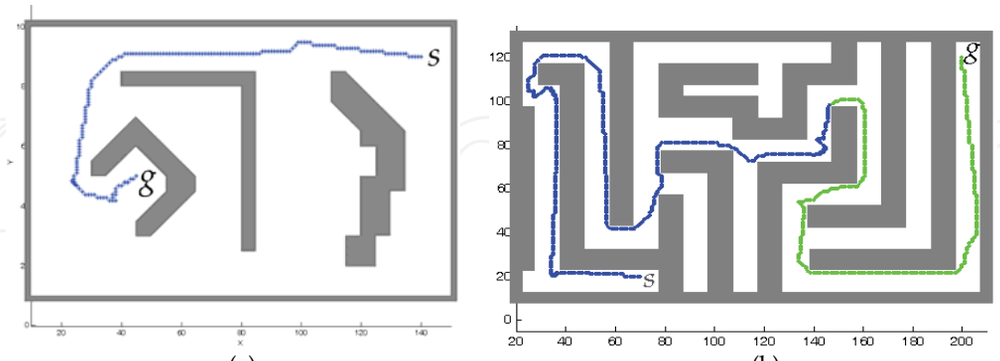
(a) (b) (c)

Fig. 17. The first, second and third iterations of the Search phase. The black lines show tangent rays for visibility scan, and white points are generated by potential search.

The same operation is carried on from END(*Traj*(*g*)) to the new endpoint of *Traj*(*s*), which now includes recently added potential search points. Note that in Fig. 17(a), due to the filling operations executed before and during the Potential Field module, the steepest descent search does not fill the nearby minimum well, and thus avoids entrapment in the local minimum around the Goal point. Rather, it utilizes the *mildest ascent* concept, and exhibits a hill climbing behavior. This case shows the importance and effectiveness of the filling procedure, which helps the planner substantially through the whole process. Fig 17(b) illustrates the second iteration, which is performed in the same fashion as the first iteration. Note the wall-following function of the potential module before detachment from C_{free} border.

Fig. 17(c) displays the case S1(*c*) occurred in the third iteration, where both trajectories are being seen by each other's endpoints. By applying the criterion (10) it becomes evident that the endpoint of *Traj*(*s*) must be connected to a visible point in *Traj*(*g*) closest to *g*. The remaining points to the end of *Traj*(*g*) are truncated afterwards. Eventually the reversely-ordered *Traj*(*g*) is concatenated to the *Traj*(*s*) and yields the final path from Start to Goal (Fig. 18(a)).

Another example is presented in Fig. 18(b) to display the shape of the generated path for a maze-like problem. The meeting point of the approaching trajectories is shown by a color contrast. The search took 7 seconds and five iterations.



(a) (b)

Fig. 18. (a) The final start-to-goal path. (b) Maze-like problem solved by the V-V-P algorithm.

3.4 Time Complexity

As discussed in the Sec. 2.4, the time complexity of constructing the PGVG is $O(n\log n)$. The time required for establishing the *Region*(□*MID*) depends on the total length of PGVG edges, which is in $O(n)$. The time required to calculate the valley potentials is constant for each gridpoint lying in *Region*(□*MID*). Hence, the total time complexity for the preprocessing phase is in the order of $O(n\log n)$.

The Search phase has the Visibility and Potential Field modules which are executed for k iterations. In the worst-case, k is bounded by half the number of all edges, which is in $O(n/2) \equiv O(n)$. During the Search phase, the visibility radial sweep operation has constant time complexity and depends on the number of radial rays. The number of potential valleys is in $O(n)$, which is affected by the $O(n)$ number of Voronoi edges, n being the total number of obstacle vertices. The time complexity for the Potential Field searching operation is $O(m)$ in the total number of gridpoints (m), and is independent of the number and shape of the obstacles (Latombe 1991). Therefore, the time complexity of the Search phase is in the order of $O(m)$.

3.5 Comparisons

In order to compare the V-V-P model with the Visibility Graph, Voronoi diagram, and Potential Fields methods, we solved the 20 problems mentioned in Sec. 2.3 by these methods and calculated the lengths of produced paths. Path lengths were normalized via a uniform scale to set up a proper benchmark for comparison. The value of □ in V-V-P algorithm was set to 0.7. The Preprocessing phase of the Compound model took about 9 seconds averagely, and the Search phase finished within 6 seconds on average. The experiments were run in MATLAB using a 1.4 GHz processor. A comparison of path lengths, as well as time complexities of the preprocessing and search procedures of all tested methods is provided in Table 2.

The results show that the V-V-P Compound takes advantage of the superiorities of its parent methods; that is, low construction time from the GVG, low search time from the PF, and short paths from the VG. It provides an effective balance between computational speed and path quality. The extent of this tradeoff is determined by selecting different values for $\alpha \in (0, 1)$, after which the V-V-P method assumes the properties of either the Visibility, or Voronoi methods, or an intermediate state.

Path planning method	Preprocessing Time complexity	Searching		Relative path length
		Time complexity	Search method	
Voronoi Diagrams	$O(n\log n)$	$O(n^2)$	Dijkstra on graph nodes	125.7
Potential Fields	$O(n)$	$O(m)$	Improved numerical navigation function	128.0 [a]
Visibility Graph	$O(n^2)$	$O(n^2)$	A* on graph nodes	100.0
V-P Hybrid	$O(n\log n)$	$O(\sqrt{m})$	Steepest descent – mildest ascent	126.8
V-V-P Compound	$O(n\log n)$	$O(m)$	Steepest descent – mildest ascent	114.3

[a] After post-processing and path smoothing

Table 2. Time complexity and path quality comparison for five path planning approaches.

It is worth noting that similar to the V-P Hybrid method (Sec. 2.4), the V-V-P Compound algorithm has the property of completeness.

3.6 Extension to Higher Spaces

The V-V-P algorithm has the potential to be extended to three and higher dimensional spaces. Though the full *n*-dimensional implementation of the algorithm is among our future research, we will briefly discuss here the possibility of its extension to 3D.

Recall that the Generalized Voronoi Graph (GVG) in *n*-D space is the locus of points being equidistant from *n* or more obstacle features. Figs. 19(a)-(b) demonstrate a 3D environment and its GVG. The GVG is constructed incrementally using an algorithm which is the 3D version of our work presented in (Masehian et al., 2003).

Due to the one-dimensional nature of the GVG roadmap, the pruning procedure is still applicable to 3D context. Fig. 19(c) depicts the result of pruning the GVG in Fig. 19(a), after fixing Start and Goal positions. Similar to the 2D case, the pruning procedure reduces the search space considerably in 3D.

The Maximal Inscribed Discs can easily be generalized to 3D space, resulting in *Maximal Inscribed Balls* (*MIB*s), which are spheres centered on the GVG and tangent to 3 or more obstacle boundaries. In the same manner, we can extend the concept of αMID to αMIB, and the concept of *Region*(αMID) to *Region*(αMIB). The *Region*(αMIB) is a network of "tube-like" obstacle-free navigable channels. Fig. 19(d) illustrates the *Region*(□*MIB*) with $\alpha = 0.5$. Greater values for α cause "fatter" tubes, and freer space for robot's maneuvering.

The visibility scan in 3D can be applied via "sweep surfaces" instead of sweep rays in the 2D method. The robot should scan the space inside the *Region*(αMIB) to find "tangent surfaces". The Potential calculations for gridpoints is still tractable in 3D workspace, and the

search phase can be performed similar to the 2D V-V-P method; the Visibility and Potential Field modules will execute alternately, and the valley filling procedure will change to "tube filling". Therefore, the V-V-P and V-P models are extendable to at least 3D C-spaces.

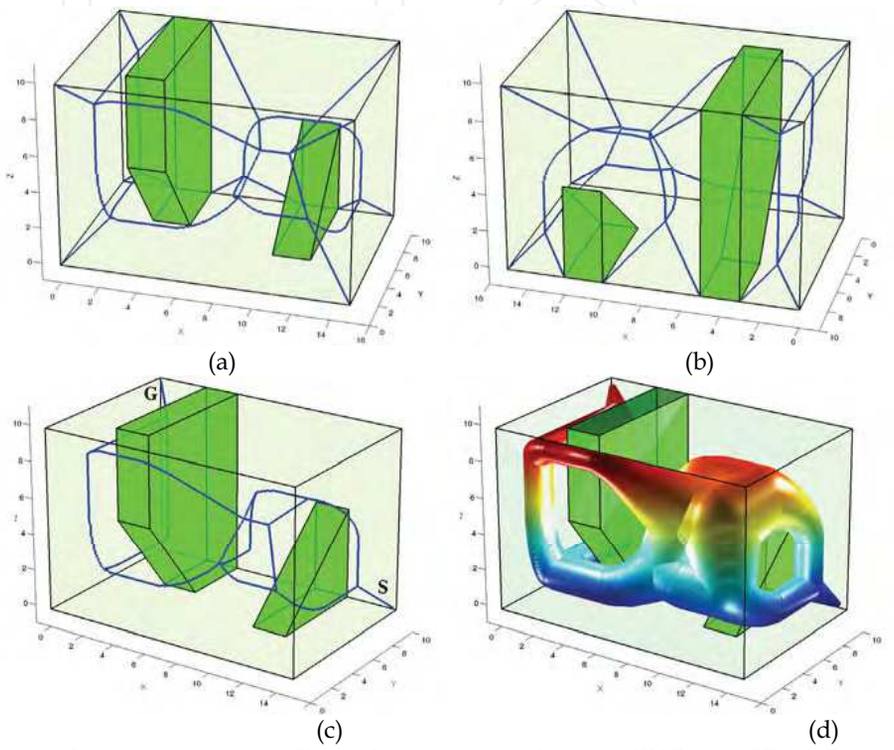


Fig. 19. (a) Front view, and, (b) Back view of the medial axis (GVG) of a 3D workspace. (c) The PGVG of the same workspace. (d) The *Region*(α*MIB*).

4. Summary and Future work

This chapter introduces two new offline path planning models which are based on the Roadmap and Potential Fields classic motion planning approaches. It is shown that how some relatively old methods can combine and yield new models.

The first path planning model is established based on two traditional methods: the Voronoi Diagrams and Potential Fields, and so is called V-P Hybrid model. The model integrates the advantages of Voronoi diagram's safest distance and Potential Fields' search simplicity properties. After constructing the Generalized Voronoi Graph roadmap for the workspace, it is reduced to the Pruned Generalized Voronoi Graph (PGVG) through a pruning procedure. The PGVG decreases the search time effectively. An attractive potential is then applied to the resulting roadmap, which yields a new version of Potential Fields method, since it implicitly models the obstacles by attractive potentials rather than repulsive ones. The search technique developed for finding the trajectory is a bidirectional steepest descent – mildest ascent stage-by-stage method, which is complete, and performs much faster than the classical Potential Fields or Dijkstra's methods.

The second model is a generalization of the V-P Hybrid model: it integrates three main approaches: Voronoi Diagrams, Visibility Graph, and Potential Fields, and is called V-V-P Compound path planner. After constructing the PGVG roadmap, a broad freeway net (called *Region*(□*MID*)) is developed based on the Maximal Inscribed Discs concept. A potential function is then assigned to this net to form an obstacle-free network of valleys. Afterwards, a bidirectional search technique is used where the Visibility Graph and Potential Fields modules execute alternately from both start and goal configurations. The steepest descent – mildest ascent search method is used for valley filling and local planning to avoid local minima. This Compound model provides a parametric tradeoff between safest and shortest paths, and generally yields shorter paths than the Voronoi and Potential Fields methods, and faster solutions than the Visibility Graph.

Different implementations of the presented algorithms exhibited these models' competence in solving path planning problems in complex and maze-like environments. Comparisons with classical Potential Fields and A* methods showed that composite methods usually perform faster and explore far less grid-points.

The developed composite path planning models can however be extended in numerous directions to accommodate more general assumptions. Here we mention two possible extensions which are achievable in the future versions of the models:

(1) Both methods are basically developed for point robots. This assumption is not realistic and requires an extra preprocessing step for obstacle expanding through the Minkowski Set Difference technique. Moreover, the robot is bound to have mere translational movements, and not rotational. The models can be modified to accommodate arbitrary-shaped robots with rotational ability.

(2) The developed models handle single-robot problems. The potential valleys in both V-P and V-V-P models may provide a framework for multiple robots motion planning. Especially, the Visibility component of the Compound model can be readily applied to mobile robots teams with vision capabilities.

5. References

Agarwal, P.K.; de Berg, M.; Matousek, J. & Schwarzkopf, O. (1998). Constructing levels in arrangements and higher order Voronoi diagrams, *SIAM Journal on Computing*, Vol. 27, 1998, pp. 654-667.

Alvarez, D.; Alvarez, J.C.; & González, R.C. (2003). Online motion planning using Laplace potential fields, *Proceedings of IEEE International Conference on Robotics and Automation (ICRA 2003)*, pp. 3347-3352, Taiwan, September 2003, IEEE, Taipei.

Aurenhammer, F. & Klein, R. (2000). Voronoi diagrams, In: *Handbook of Computational Geometry*, Sack, J. & Urrutia, G. (Eds.), pp. 201-290, Elsevier/North-Holland, ISBN: 0444825371, Amsterdam.

Barraquand, J.; Langlois, B. & Latombe, J.C. (1992). Numerical potential field techniques for robot path planning, *IEEE Trans. Systems, Man, and Cybernetics*, Vol. 22, No. 2, (March/April 1992), pp. 224-241.

Brock, O. & Khatib, O. (1999). High-speed navigation using the global dynamic window approach, *Proceedings of IEEE International Conference on Robotics and Automation (ICRA '99).*

Canny, J.F. (1985). A voronoi method for the piano movers' problem, *Proceedings of IEEE International Conference on Robotics and Automation (ICRA '85).*

Canny, J.F. (1998). *The Complexity of Robot Motion Planning*, MIT Press, ISBN: 0262031361, Cambridge, Mass.

Choset, H. & Burdick, J. (2000). Sensor-based exploration: The hierarchical generalized Voronoi graph, *International Journal of Robotics Research*, Vol. 19, No. 2, 2000, pp. 96-125.

Choset, H.; Lynch, K.M.; Hutchinson, S.; Kantor, G.; Burgard, W.; Kavraki, L.E. & Thrun, S. (2005). *Principles of Robot Motion: Theory, Algorithms, and Implementations*, MIT Press, ISBN: 0262033275, Boston.

Hwang, Y.K. & Ahuja, N. (1992). Gross motion planning-A survey, *ACM Computing Surveys*, Vol. 24, No. 3, (September 1992), pp. 219-291.

Kavraki, L.E.; Svestka, P.; Latombe, J.C. & Overmars, M. (1996). Probabilistic roadmaps for path planning in high dimensional configuration spaces, *IEEE Transactions on Robotics and Automation*, Vol. 12, No. 4, (August 1996), pp. 566-580.

Khatib, O. (1986). Real-time obstacle avoidance for manipulators and mobile robots, *International Journal of Robotics Research*, Vol. 5, No. 1, pp. 90-98.

Koren, Y. & Borenstein, J. (1991). Potential field methods and their inherent limitations for mobile robot navigation, *Proceedings of IEEE International Conference on Robotics and Automation (ICRA '91)*, April 1991, pp. 1398-1404.

Latombe, J.C. (1991). *Robot Motion Planning*, Kluwer Academic Publishers, ISBN: 0792391292, Boston.

Masehian, E.; Amin-Naseri, M.R. & Khadem, S.E. (2003). Online motion planning using incremental construction of medial axis, *Proceedings of IEEE International Conference on Robotics and Automation (ICRA 2003)*, (September 2003), Taiwan, pp. 2928-2933.

Masehian, E. & Amin-Naseri, M.R. (2004). A Voronoi diagram – visibility graph – potential fields compound algorithm for robot path planning, *Journal of Robotic Systems*, Vol. 21, No. 6, (June 2004), pp. 275-300.

Oommen, J.B.; Iyengar, S.S.; Rao, N.S.V. & Kashyap, R.L. (1987). Robot navigation in unknown terrains using visibility graphs: Part I: The disjoint convex obstacle case, *IEEE Journal of Robotics and Automation*, *Vol. RA-3*, (1987), pp. 672-681.

Sato, K. (1993). Deadlock-free motion planning using the Laplace potential field, *Advanced Robotics*, Vol. 7, No. 5, 1993, pp. 449-462.

Yeung, D.Y. & Bekey, G.A. (1987). A decentralized approach to the motion planning problem for multiple mobile robots, *Proceedings of IEEE International Conference on Robotics and Automation (ICRA '87)*, pp. 1779-1784.

Mobile Robots: Perception & Navigation

Edited by Sascha Kolski

ISBN 3-86611-283-1

Hard cover, 704 pages

Publisher Pro Literatur Verlag, Germany / ARS, Austria

Published online 01, February, 2007

Published in print edition February, 2007

Today robots navigate autonomously in office environments as well as outdoors. They show their ability to beside mechanical and electronic barriers in building mobile platforms, perceiving the environment and deciding on how to act in a given situation are crucial problems. In this book we focused on these two areas of mobile robotics, Perception and Navigation. This book gives a wide overview over different navigation techniques describing both navigation techniques dealing with local and control aspects of navigation as well es those handling global navigation aspects of a single robot and even for a group of robots.

How to reference

In order to correctly reference this scholarly work, feel free to copy and paste the following:

Ellips Masehian and M. R. Amin-Naseri (2007). Composite Models for Mobile Robot Offline Path Planning, Mobile Robots: Perception & Navigation, Sascha Kolski (Ed.), ISBN: 3-86611-283-1, InTech, Available from: http://www.intechopen.com/books/mobile_robots_perception_navigation/composite_models_for_mobile_robot_offline_path_planning

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen