Robotica (1997) volume 15, pp 421–434. Printed in the United Kingdom © 1997 Cambridge University Press

A 3-D Potential panel method for robot motion planning*

Yanjun Zhang and Kimon P. Valavanis

Robotics & Automation Laboratory, A-CIM Center & CACS, The University of Southwestern Louisiana, Lafayette, LA 70504 (USA)

(Received in Final Form: September 3, 1996)

SUMMARY

A potential panel method is proposed to solve the collision-free path planning problem for a free-flying robot operating in an obstacle filled 3-D environment. The problem is solved in three steps: (1) transform the arbitrary shaped obstacles in the 3-D workspace into simple convex polyhedrons; (2) generate an artificial potential field using the 3-D panel method in the 3-D workspace; (3) generate a streamline from the starting position towards the goal position in the artificial potential field. The computational complexity of the pertinent algorithms justify the efficiency of the approach and its applicability in real-time. Simulation results illustrate the potential of the proposed approach. The reported research is the outgrowth of the 2-D method, already published.[1]

KEYWORDS: Sensor-based motion planning; Collision avoidance; Potential panel method; Potential field.

1. INTRODUCTION

The research reported in this paper has been motivated by the challenge to solve the free-flying robot motion planning problem in an obstacle-filled 3-D workspace. The main objective of this paper is to advance the potential field motion planning approach from the theoretical idea level to the practical algorithms level, by deriving a set of practical, computationally efficient formulas and algorithms to solve the free-flying robot navigation problem in an obstacle filled 3-D environment in real-time. The proposed potential panel method solves the free-flying robot motion planning problem in a *static* environment in three steps:

(i) Transform the arbitrary shaped obstacles in the 3-D workspace into simple convex polyhedrons, which tightly enclose the obstacle bodies, based on a set of sample points, which are gathered by various sensors, over the boundary surface of the obstacles.
(ii) Solve the strength values of the potential sources over each panel on the boundary surface of the simple convex polyhedral obstacles according to the boundary conditions of the potential field and the potential sink at the goal position, in order to generate an artificial potential field.
(iii) Generate a streamline from the starting position towards the goal position with a sequence of small straight-line displacements in the artificial potential field. Each displacement is guided by the velocity vector, which is the negated gradient at the starting point of the displacement in the potential field.

† This work has been partially funded by an NSF grant, BES 95-06771. I wish to thank Timothy Hebert, the Graduate assistant who helped to run the simulations.

However, the presence of moving obstacles in a *dynamic* environment requires the generation of a sequence of motion subpaths, each intended to reach some intermediate goal position, while the whole subpath sequence eventually leads to the final goal position. Thus, a reactive motion planning algorithm suitable for on-line dynamic motion planning of a sensor-based free-flying robot is also presented. To derive the reactive motion planning algorithm it has been assumed that sensor data are available and registered in world coordinates to allow construction of the convex polyhedra (it is beyond the scope of this paper to discuss problems related to sensor fusion and sensor noise data).

A summary of related work following the roadmap approach and the cell decomposition approach may be found in reference 2. Related work using the potential field approach has been done by Khatib,[3] Krogh,[4] Koditschek and Rimon,[5–9] Khosla and Volpe,[10] Barraquand and Latombe,[12,13] Megherbi and Wolovich,[14] Kim and Khosla,[15,16] Chuang.[17] More details, including a summary of the computational complexity of related approaches in the 2-D and 3-D case, may also be found in reference 2.

The rest of the paper is organized as follows: Section 2 introduces the computational method of the 3-D potential panel method; Section 3 presents the algorithms for implementing the potential panel path planning method; Section 4 introduces the sensor-based path planning method in a dynamic environment; Section 5 presents the simulation results; Section 6 concludes the paper and discusses the limitations of the introduced algorithms.

2. COMPUTATIONAL METHOD OF 3-D POTENTIAL PANEL APPROACH

According to Green's identity, the potential at an arbitrary point P in a region, which consists of convex

https://doi.org/10.1017/S0263574797000520 Published online by Cambridge University Press

polyhedral obstacles with boundary surface S_B and a potential sink at a goal position, can be represented as:[18]

$$\phi_P = \int_{S_B} q\phi_S dS + Q_q \phi_{s_G} \tag{1}$$

where ϕ_s is the potential at P due to a unit-strength potential source on the surface S_B and q is the actual strength of the potential source, ϕ_{s_G} is the potential at P due to a unit-strength potential source at the goal position and Q_g ($Q_g < 0$) is the actual strength of the potential source. The negated gradient of ϕ_P is the velocity vector of the potential flow at P; thus, the basic equation for the 3-D potential source-based panel method is;

$$\boldsymbol{V}_P = -\nabla\phi_P = -\int_{S_B} q\nabla\phi_s dS - Q_g \nabla\phi_{s_G} \tag{2}$$

If there exist k disjointed obstacles in a 3-D workspace, the boundary of the k obstacles can be approximated by k disjointed convex polyhedrons which enclose the k obstacles, respectively, and each planar polygon on the k convex polyhedrons can be defined as a potential "panel". Assume that the boundary surface of the k polyhedrons consist of a total of N planar polygons, where the N planar polygons are numbered as 0, 1, $2, \ldots, N-1$ panels in an arbitrary order. Assume that potential sources with constant strength are uniformly distributed on each panel, but with different unknown strength values $(q_0, q_1, q_2, \ldots, q_{N-1})$ on the N panels; and a potential sink with known strength Q_g is located at the goal position within the 3-D workspace.

As shown in Figure 1, the goal position is denoted by G and its coordinates in the world coordinate system are (x_g, y_g, z_g); the potential function of the potential sink with strength Q_g ($Q_g > 0$), and its velocity components are:

$$\begin{aligned}
\phi_G(x, y, z) &= Q_g \phi_{s_G} \\
&= \frac{Q_g}{4\pi} \cdot \frac{1}{\sqrt{(x-x_g)^2 + (y-y_g)^2 + (z-z_g)^2}} \\
u_g(x, y, z) &= -Q_g \cdot \frac{\partial \phi_{s_G}}{\partial x},
\end{aligned} \tag{3}$$

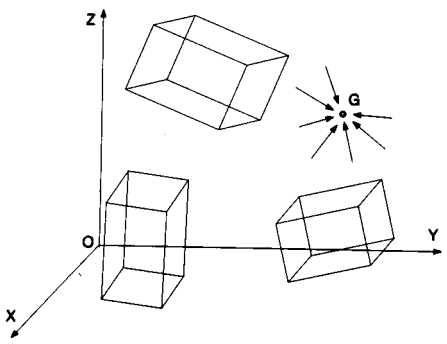


Fig. 1. The configuration of the polyhedral object.

$$v_g(x, y, z) = -Q_g \cdot \frac{\partial \phi_{s_G}}{\partial y},$$

$$w_g(x, y, z) = -Q_g \cdot \frac{\partial \phi_{s_G}}{\partial z}$$

According to the basic equation (2) of the potential source-based panel method, the velocity at any point in the potential field is:

$$\begin{aligned}
\mathbf{V} &= -\int_{S_B} q\nabla\phi_s \, dS - Q_g \nabla\phi_{s_G} \\
&= -\sum_{j=0}^{N-1} q_j \int_{\text{panel } j} \nabla\phi_s \, dS - Q_g \nabla\phi_{s_G} \\
&= \sum_{j=0}^{N-1} q_j (U_x^j \hat{\mathbf{i}} + U_g^j \hat{\mathbf{j}} + U_z^j \hat{\mathbf{k}}) + (u_g \hat{\mathbf{i}} + v_g \hat{\mathbf{j}} + w_g \hat{\mathbf{k}})
\end{aligned} \tag{4}$$

where U_x^j, U_y^j and U_z^j are the velocity components due to a unit-strength sources distributed on the jth panel. Let the coordinates of the centroid point of the ith panel in the world coordinate system be $(\bar{x}_i, \bar{y}_i, \bar{z}_i)$, where:

$$\begin{aligned}
\bar{x}_i &\equiv \frac{1}{M_i} \sum_{k=0}^{M_i - 1} x_{ik}, \\
\bar{y}_i &\equiv \frac{1}{M_i} \sum_{k=0}^{M_i - 1} y_{ik}, \\
\bar{z}_i &\equiv \frac{1}{M_i} \sum_{k=0}^{M_i - 1} z_{ik} \\
i &= 0, 1, 2, \ldots, N-1
\end{aligned} \tag{5}$$

M_i is the total number of vertices of the ith polygonal panel, (x_{ik}, y_{ik}, z_{ik}) are the coordinates of the kth vertex of the ith panel. Let:

$$\begin{cases} U_{xi} \equiv U_x(\bar{x}_i, \bar{y}_i, \bar{z}_i) \\ U_{yi} \equiv U_y(\bar{x}_i, \bar{y}_i, \bar{z}_i) \\ U_{zi} \equiv U_z(\bar{x}_i, \bar{y}_i, \bar{z}_i) \end{cases} \quad \begin{cases} u_{gi} \equiv u_g(\bar{x}_i, \bar{y}_i, \bar{z}_i) \\ v_{gi} \equiv v_g(\bar{x}_i, \bar{y}_i, \bar{z}_i) \\ w_{gi} \equiv w_g(\bar{x}_i, \bar{y}_i, \bar{z}_i) \end{cases} \tag{6}$$

The velocity components at the centroid point of the ith panel can be represented according to equation (4) as:

$$\begin{aligned}
U_{xi} &= \sum_{j=0}^{N-1} q_j U_{xi}^j + u_{gi} \\
U_{gi} &= \sum_{j=0}^{N-1} q_j U_{yi}^j + v_{gi} \\
U_{yi} &= \sum_{j=0}^{N-1} q_j U_{zi}^j + w_{gi} \\
i &= 0, 1, 2, \ldots, N-1
\end{aligned} \tag{7}$$

where U_{xi}^j, U_{yi}^j and U_{zi}^j are the velocity components at the centroid point of the ith panel due to the unit-strength sources distributed on the jth panel.

According to the boundary condition of the potential field, the velocity vector at the centroid point of each panel must be tangent to the panel. Let the normal unit vector of the ith panel be represented as:

$$\hat{\mathbf{n}}_i = n_{xi}\hat{\mathbf{i}} + n_{yi}\hat{\mathbf{j}} + n_{zi}\hat{\mathbf{k}} \tag{8}$$

https://doi.org/10.1017/S0263574797000520 Published online by Cambridge University Press

then, the following equation holds for each panel:

$$\hat{\mathbf{n}}_i \cdot \mathbf{V}_i = U_{xi}n_{xi} + U_{yi}n_{yi} + U_{zi}n_{zi} = 0 \qquad i = 0, 1, 2, \ldots, N-1 \tag{9}$$

By substituting equation (7) into (9), the flow tangency condition (9) becomes:

$$\sum_{j=0}^{N-1} A_{ij}q_j = B_i, \qquad i = 0, 1, 2, \ldots, N-1 \tag{10}$$

where

$$\begin{aligned} A_{ij} &= U^j_{xi}n_{xi} + U^j_{yi}n_{yi} + U^j_{zi}n_{zi} \\ B_i &= -(u_{gi}n_{xi} + v_{gi}n_{yi} + w_{gi}n_{zi}) \\ i, j &= 0, 1, 2, \ldots, N-1 \end{aligned} \tag{11}$$

The values of U^j_{zi}, U^j_{yi} and U^j_{zi} can be computed by using the formulas (equation (39)) presented later in this section. Thus the N unknown potential strength values $(q_0, q_1, q_2, \ldots, q_{N-1})$ can be calculated by solving N linear equations using the Gauss elimination method, which can be easily implemented by a computer program.

Therefore, the velocity vector at a point (x, y, z) in the potential field may be computed by the following equations:

$$\mathbf{V}(x, y, z) = u(x, y, z)\hat{\mathbf{i}} + v(x, y, z)\hat{\mathbf{j}} + w(x, y, z)\hat{\mathbf{k}}$$

where

$$\begin{aligned} u(x, y, z) &= \sum_{j=0}^{N-1} q_j U^j_x(x, y, z) + u_g(x, y, z) \\ v(x, y, z) &= \sum_{j=0}^{N-1} q_j U^j_y(x, y, z) + v_g(x, y, z) \\ w(x, y, z) &= \sum_{j=0}^{N-1} q_j U^j_z(x, y, z) + w_g(x, y, z) \end{aligned} \tag{12}$$

By continuously following the direction of the velocity vector $\mathbf{V}$ at the current point (x, y, z) in the artificial potential field, a smooth and collision-free streamline (path) towards the goal position can be generated. However, one may observe that the magnitude of flow velocity along a path in the potential field generated by the potential source-based panel method has a dramatic jump as the path penetrates the boundary surface formed by potential panels. This characteristic of the path in the artificial potential field may be used to detect the collision/penetration of a generated path with the obstacles in the workspace.[2]

2.1. Computation of the potential due to a polygonal potential panel

Let S denote a planar polygon. The potential, due to the potential sources distributed on S, observed at an arbitrary point E in the 3-D workspace is;

$$\phi^s_E = \int_S \frac{1}{4\pi R}\, dS' \tag{13}$$

where R is the distance from the observation point E to the potential sources on S. Evaluation of the potential integral is accomplished by employing the Gauss divergence theorem to transform the integration over S to one over the boundary of S; thus, the integral over the polygon S may be transformed into a sum of integrals over the set of edges of the polygon S. It is critical to define a proper local coordinate system for each edge of the polygon S so that the potential integral has the simplest closed form solution and can be computed by a computer program in a straightforward way. The computational method reported in reference 19 is used to solve the potential integral problems as follows:

If the polygon is represented as S, then the boundary of S can be denoted as ∂S, and the ith edge of the polygon S can be denoted as $\partial_i S$. Consider that the coordinate system $XYZO$ whose origin is at O is the world coordinate system, as shown in Figure 2; the panel S is on a spacial plane B, and an arbitrary spacial position E is defined as the observation point. Let O' and E' denote the projections of O and the point E onto the plane B, $\hat{\mathbf{n}}_s$ denote the normal unit vector of the plane B, $\mathbf{r}$ denote the position vector from O to E, while $\mathbf{r}'$ be the vector from O to a source point in S, ρ and ρ' represent the projection vectors of $\mathbf{r}$ and $\mathbf{r}'$ onto B, $\mathbf{r}^+$ and $\mathbf{r}^-$ designate the vectors to the end points of the straight-line segment $\partial_i S$ (the ith edge of the polygon S). Then, a local coordinate system $X^s_i Y^s_i Z^s_i E'$ can be defined for the straight-line segment $\partial_i S$, as also shown in Figure 2. The origin of the local coordinate system is at E' on B, and the three unit vectors that determine the directions of x^s_i, y^s_i and z^s_i axis are:

$$\begin{aligned} \hat{\mathbf{u}}^s_i &= (u_{ix}, w_{iy}, u_{iz}) = \hat{\mathbf{v}}^s_i \times \hat{\mathbf{n}}^s_i \\ \hat{\mathbf{v}}^s_i &= (v_{ix}, v_{iy}, v_{iz}) = \frac{\mathbf{r}^+_i - \mathbf{r}^-_i}{|\mathbf{r}^+_i - \mathbf{r}^-_i|} \\ \hat{\mathbf{n}}^s_i &= (n_{ix}, n_{iy}, n_{iz}) = \hat{\mathbf{n}}_s \end{aligned} \tag{14}$$

Let R^+_i and R^-_i denote the distances from the observation point E to the two end points of the line segment $\partial_i S$; let d denote the absolute height of the observation point E above the plane of S; P^0_i denote the perpendicular distance from the point E' to the straight-line segment $\partial_i S$ or its extension; l^+ and l^- denote the perpendicular distances from the x^s_i axis to

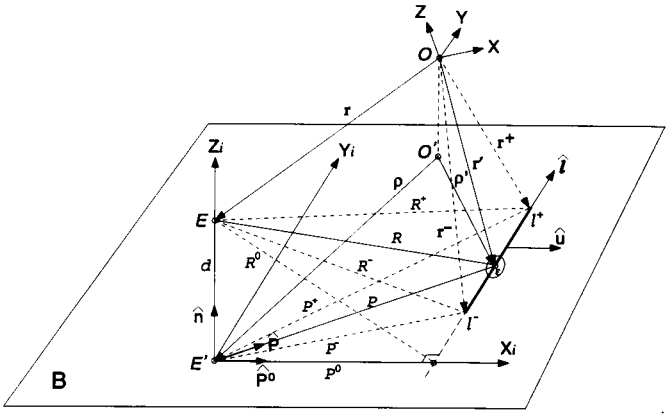


Fig. 2. The local coordinate system of a polygon edge.

https://doi.org/10.1017/S0263574797000520 Published online by Cambridge University Press

the two end points of the line segment $\partial_i S$, which are measured in the positive direction of y_i^s axis; these distance variables can be calculated as follows:

$$\begin{aligned} R_i^{\pm} &= |(\mathbf{r}_i^{\pm} - \mathbf{r})| \geq 0 \\ d &= |\hat{\mathbf{n}}_i^s \cdot (\mathbf{r}_i^{\pm} - \mathbf{r})| \geq 0 \\ P_i^0 &= |\hat{\mathbf{u}}_i^s \cdot (\mathbf{r}_i^{\pm} - \mathbf{r})| \geq 0 \\ (R_i^0)^2 &= (P_i^0)^2 + d^2 \geq 0 \\ l_i^{\pm} &= \hat{\mathbf{v}}_i^s \cdot (\mathbf{r}_i^{\pm} - \mathbf{r}) \end{aligned} \tag{15}$$

As shown in Figure 2, the local coordinates of an arbitrary point on $\partial_i S$ with respect to the local coordinate system $X_i^s Y_i^s Z_i^s E'$ can be expressed as $(x_i^s, y_i^s, 0)$, and the local coordinates of the point E can be expressed as $(0, 0, z_i^s)$ where $z_i^s \geq 0$. Thus:

$$\begin{aligned} P_i^0 = x_i^s, \quad d = z_i^s, \\ P = \sqrt{x_i^{s2} + y_i^{s2}}, \quad R = \sqrt{x_i^{s2} + y_i^{s2} + z_i^{s2}} \end{aligned} \tag{16}$$

Since the following equation holds:

$$\begin{aligned} \nabla_s \left(\frac{R}{P} \cdot \hat{\mathbf{P}} \right) &= \nabla_{xy} \left(\frac{R}{P} \cdot \frac{x_i^s}{P} \cdot \hat{\mathbf{i}} + \frac{R}{P} \cdot \frac{y_i^s}{P} \cdot \hat{\mathbf{j}} \right) \\ &= \frac{\partial}{\partial x_i^s} \left(\frac{R \cdot x_i^s}{P^2} \right) + \frac{\partial}{\partial y_i^s} \left(\frac{R \cdot y_i^s}{P^2} \right) = \frac{1}{R} \end{aligned} \tag{17}$$

the integral on $S - S_\varepsilon$ can be solved by employing the Gauss divergence theorem as explicitly shown in reference 2.

Based on equation (16), the closed form solution of the potential at point E due to the uniform distributed sources on S can be represented in terms of the local coordinates of $\partial_i S$, that is:

$$\phi_E^s = \frac{1}{4\pi} \sum_i \left[\varphi_1(x_i^s, y_i^s, z_i^s) - z_i^s \varphi_2(x_i^s, y_i^s, z_i^s) \right] \Big|_{y_i^s = l_i^-}^{y_i^s = l_i^+} \tag{18}$$

where:

$$\begin{aligned} \varphi_1(x_i^s, y_i^s, z_i^s) &= x_i^s \ln \left(y_i^s + \sqrt{x_i^{s2} + y_i^{s2} + z_i^{s2}} \right) \\ \varphi_2(x_i^s, y_i^s, z_i^s) &= \tan^{-1} \left(\frac{x_i^s y_i^s}{x_i^{s2} + z_i^{s2} + z_i^s \sqrt{x_i^{s2} + y_i^{s2} + z_i^{s2}}} \right) \end{aligned} \tag{19}$$

and (x_i^s, y_i^s, z_i^s) is the coordinates of the polygon edge $\partial_i S$ in the local coordinate system $X_i^s Y_i^s Z_i^s E'$.

2.2. Computation of the velocity due to a polygonal potential panel

In order to find the velocity components at the observation point E in the x, y, z directions with respect to the world coordinate system $XYZO$, the variables of the functions, $\varphi_1(x_i^s, y_i^s, z_i^s)$ and $\varphi_2(x_i^s, y_i^s, z_i^s)$, which are defined with respect to the local coordinate system of $\partial_i S$, need to be defined with respect to the world coordinate system. Since the transformation between the local coordinate system $X_i^s Y_i^s Z_i^s E'$ and the world coordinate system $XYZO$ can be expressed by the three unit vectors $(\hat{\mathbf{u}}_i^s, \hat{\mathbf{v}}_i^s, \hat{\mathbf{n}}_i^s)$, which determine the orientation of the local coordinate system $X_i^s Y_i^s Z_i^s E'$, by the following equation:

$$\begin{bmatrix} x_i^s \\ y_i^s \\ z_i^s \\ 1 \end{bmatrix} = \begin{bmatrix} u_{ix} & u_{iy} & u_{iz} & c_1 \\ v_{ix} & v_{iy} & v_{iz} & c_2 \\ n_{ix} & n_{iy} & n_{iz} & c_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{20}$$

The velocity vector at the point E, which is the negated gradient of the potential function ϕ_E^s, with respect to the world coordinate system $XYZO$ can be induced based on the closed form solution of the potential function ϕ_E^s and the following coordinate transformation equations;

$$\begin{aligned} x_i^s(x, y, z) &= u_{ix} x + u_{iy} y + u_{iz} z + c1 \\ y_i^s(x, y, z) &= v_{ix} x + v_{iy} y + v_{iz} z + c2 \\ z_i^s(x, y, z) &= n_{ix} x + n_{iy} y + n_{iz} z + c3 \end{aligned} \tag{21}$$

Let $\mathbf{V}_E^s$ denote the velocity vector at E due to the potential sources distributed on the panel S, U_x^s, u_y^s, U_z^s denote the velocity components of $\mathbf{V}_E^s$ in the x, y, z directions of the world coordinate system; then:

$$\begin{aligned} \mathbf{V}_E^s &= U_x^s \hat{\mathbf{i}} + U_y^s \hat{\mathbf{j}} + U_z^s \hat{\mathbf{k}} \\ &= -\nabla \phi_E^s = -\nabla \left(\frac{1}{4\pi} \sum_i \left[\varphi_1(x_i^s, y_i^s, z_i^s) - z_i^s \varphi_2(x_i^s, y_i^s, z_i^s) \right] \right) \end{aligned} \tag{22}$$

The next step is to calculate the derivatives of φ_1, φ_2 with respect to the variables x, y, z in the world coordinate system. This is a computationally intensive task, and all calculations are explicitly shown in reference 2. Without a loss of generality, only the final result is presented in this paper.

The velocity components at point E in the x, y, z directions of the world coordinate system, due to the uniformly distributed sources on the panel S can be written as:

$$\begin{aligned} U_x^s &= -\frac{\partial \phi_E^s}{\partial x} = -\frac{1}{4\pi} \sum_i \left[\frac{\partial \varphi_1}{\partial x} - n_{ix} \varphi_2 - d \frac{\partial \varphi_2}{\partial x} \right] \Big|_{y_i^s = l_i^-}^{y_i^s = l_i^+} \\ U_y^s &= -\frac{\partial \phi_E^s}{\partial y} = -\frac{1}{4\pi} \sum_i \left[\frac{\partial \varphi_1}{\partial y} - n_{iy} \varphi_2 - d \frac{\partial \varphi_2}{\partial y} \right] \Big|_{y_i^s = l_i^-}^{y_i^s = l_i^+} \\ U_z^s &= -\frac{\partial \varphi_E^s}{\partial z} = -\frac{1}{4\pi} \sum_i \left[\frac{\partial \varphi_1}{\partial z} - n_{iz} \varphi_2 - d \frac{\partial \varphi_2}{\partial z} \right] \Big|_{y_i^s = l_i^-}^{y_i^s = l_i^+} \end{aligned} \tag{23}$$

Since the distance variable d $(z_i^s = d)$ is taken as an absolute value, the velocity components, which are computed from the above equations, in the two sides of a polygonal panel along its normal direction are in the uniform direction. In order to apply the panel method, the velocity directions in the two sides of a panel along the normal direction should be opposite. Thus, if the observation point E is in the other side of a panel plane as the side pointed by the normal vector of the panel, the velocity components U_x^s, U_y^s, U_z^s obtained from the above equations need to be further transformed as follows:

(i) Transform the velocity vector at E with respect to the world coordinate system $XYZO$ into a vector

https://doi.org/10.1017/S0263574797000520 Published online by Cambridge University Press

with respect to the local coordinate system $X_i^s Y_i^s Z_i^s E'$ of an arbitrary edge $\partial_i S$ of S, i.e.:

$$\begin{bmatrix} U_{x_i}^s \\ U_{y_i}^s \\ U_{z_i}^s \\ 1 \end{bmatrix} = \begin{bmatrix} u_{ix} & u_{iy} & u_{iz} & 0 \\ v_{ix} & v_{iy} & v_{iz} & 0 \\ n_{ix} & n_{iy} & n_{iz} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} U_x^s \\ U_y^s \\ U_z^s \\ 1 \end{bmatrix} \tag{24}$$

(ii) Reverse the direction of the velocity component in the normal direction of the panel, i.e.:

$$U_{z_i}^s = -U_{z_i}^s \tag{25}$$

(iii) Transform the velocity vector at E with respect to the local coordinate system $X_i^s Y_i^s Z_i^s E'$ back to the vector with respect to the world coordinate system $XYZO$, i.e.:

$$\begin{bmatrix} U_x^s \\ U_y^s \\ U_z^s \\ 1 \end{bmatrix} = \begin{bmatrix} u_{ix} & v_{ix} & n_{ix} & 0 \\ u_{iy} & v_{iy} & n_{iy} & 0 \\ u_{iz} & v_{iz} & n_{iz} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} U_{x_i}^s \\ U_{y_i}^s \\ U_{z_i}^s \\ 1 \end{bmatrix} \tag{26}$$

If the observation point E is on the panel plane, the direction of the velocity at this point is by default in the positive normal direction.

3. 3-D POTENTIAL PANEL-BASED PATH PLANNING ALGORITHM

3.1. Obstacle deformation algorithm

Based on the basic geometric modelling algorithm for generating the intersection polyhedron from two intersecting polyhedrons, an algorithm is developed to transform arbitrary shaped obstacles in the 3-D workspace into a set of simple convex polyhedrons based on the sample data gathered by sensors from the boundary surface of the obstacles. The boundary surface of each generated polyhedron consists of 24 polygons, which can be defined as the potential panels. The deformation of the obstacles yields several significant benefits for the panel-based path planning method:

(i) It significantly reduces the amount of data related to the obstacle geometry, thus it reduces the computational complexity of applying the panel method to the obstacles.
(ii) It eliminates the possibilities of generating the local potential minima at the inward corners of the concave polyhedral obstacles in the artificial potential field.
(iii) It generates the most efficient and smooth collision-free path around the arbitrary shaped obstacle instead of the zig-zag path along the contour of the obstacle.

The obstacle deformation algorithm is presented as follows:

Algorithm 1. Generate a set of convex polyhedrons whose boundary surface encloses the arbitrary shaped obstacles, respectively, in the 3-D workspace.
Input:

(i) The total number of the arbitrary shaped obstacles in the 3-D workspace, which is represented by an integer m.
(ii) The total number of the sample points on the boundary surface of each obstacle in the 3-D workspace, which is represented as an integer list, $\{N_{s_0}, N_{s_1}, \ldots, N_{s_{m-1}}\}$. The sample points on the boundary surface of an obstacle are chosen to represent the shape of the obstacle. For example, the sample points of a polyhedral obstacle can be the vertices of the polyhedron.
(iii) The world coordinates of the sample points of all obstacles in the 3-D workspace. The sample points over the boundary surface of the ith arbitrary shaped obstacle in the 3-D workspace can be represented by a non-ordered coordinates list S_i, and the sample points of all obstacles in the 3-D workspace can be represented by a non-ordered list, S, of the sample point list S_i, i.e.:

$$S = \{S_0, S_1, \ldots, S_{m-1}\}$$

where

$$S_i = \{s_{i0}, s_{i1}, \ldots, s_{i(N_{s_i}-1)}\}$$

and

$$s_{ij} = \{x_{ij}, y_{iy}, z_{ij}\} \tag{27}$$

$$i = 0, 1, 2, \ldots, m-1, \quad j = 0, 1, 2, \ldots, N_{s_i} - 1$$

(iv) The clearance between the boundary surface of the generated polyhedron to the boundary surface of the obstacle, which is denoted by a real d.
(v) The world coordinates of the goal position G, (x_g, y_g, z_g), in the 3-D workspace.

Output:

(i) The boundary representation of the m generated convex polyhedrons, whose boundary surface encloses the m arbitrary shaped obstacles in the 3-D workspace, respectively. Assume that the vertices of the jth polygon on the ith generated polyhedron in the 3-D workspace is labeled sequentially in the order, which follows the right hand principle with respect to the outer normal direction of the polygon, as $(v_{ij0}, v_{ij1}, \ldots, v_{ij(k_{ij}-1)})$; the jth polygon on the ith generated polyhedron is represented by an ordered vertex list:

where

$$V_{ij} = \{v_{ij0}, v_{ij1}, \ldots, v_{ij(k_{ij}-1)}, v_{ij0}\}$$
$$v_{ijk} = (x_{ijk}, y_{ijk}, z_{ijk}), \tag{28}$$

The ith generated polyhedron is represented by a non-ordered polygon list:

$$V_i = \{V_{i0}, V_{i1}, \ldots, V_{i(j_i-1)}\}, \quad i = 0, 1, \ldots, m-1, \tag{29}$$

and the m generated polyhedrons in the 3-D workspace are represented by a non-ordered polyhedron list:

$$\{V_0, V_1, \ldots, V_{m-1}\} \tag{30}$$

Algorithm Steps:
Assume that the ith obstacle in the 3-D workspace is the

https://doi.org/10.1017/S0263574797000520 Published online by Cambridge University Press

current obstacle to be transformed, and let $\hat{\mathbf{u}}_{ik}$, $\hat{\mathbf{v}}_{ik}$, $\hat{\mathbf{n}}_{ik}$ denote the three unit vectors which determine the orientations of the X_{ik}, Y_{ik} and Z_{ik} axis of the local coordinate system $X_{ik}Y_{ik}Z_{ik}O_i$; the algorithm processes the ith obstacle in terms of the following steps:

(i) Compute the coordinates of the centred point O_i, $(x_{o_i}, y_{o_i}, z_{o_i})$, of the arbitrary shaped obstacle based on the coordinates of the sample points on the boundary surface of the obstacle, i.e.:

$$\begin{aligned} x_{o_i} &\equiv \frac{1}{N_{s_i}} \sum_{j=0}^{N_{s_i}-1} x_{ij}, \\ y_{o_i} &\equiv \frac{1}{N_{s_i}} \sum_{j=0}^{N_{s_i}-1} y_{ij}, \\ z_{o_i} &\equiv \frac{1}{N_{s_i}} \sum_{j=0}^{N_{s_i}-1} z_{ij} \end{aligned} \tag{31}$$

(ii) Determine the orientation of the local coordinate system $X_{i1}Y_{i1}Z_{i1}O_i$ based on the vector $\overrightarrow{O_iG}$ and an arbitrary vector $\hat{\mathbf{a}}$ (say $\hat{\mathbf{a}} = (1, 0, 0)$, which must be different from the vector $\overrightarrow{O_iG}$, as follows:

$$\begin{gathered} \hat{\mathbf{n}}_{i1} = \frac{\overrightarrow{O_iG}}{|\overrightarrow{O_iG}|}, \quad \hat{\mathbf{u}}_{i1} = \frac{\hat{\mathbf{n}}_{i1} \times \hat{\mathbf{a}}}{|\hat{\mathbf{n}}_{i1} \times \hat{\mathbf{a}}|}, \\ \hat{\mathbf{v}}_{i1} = \hat{\mathbf{n}}_{i1} \times \hat{\mathbf{u}}_{i1} \end{gathered} \tag{32}$$

Generate the boundary representation of the bounding block, U_{i1}, with respect to the local coordinate system $X_{i1}Y_{i1}Z_{i1}O_i$. The bounding planes of U_{i1} in the $\pm Z_{i1}$ direction are determined by specifying an artificial maximum value, MAXVAL, as follows:

$$\begin{aligned} Z_{upper} &= MAXVAL, \\ Z_{lower} &= -MAXVAL \end{aligned} \tag{33}$$

As in Figure 3, the boundary representation of the block U_{i1} is as follows:

$$V = \{V_0, V_1, V_2, V_3, V_4, V_5\}$$

where

$$\begin{aligned} V_0 &= \{a, d, c, b, a\}, \quad V_1 = \{a, e, h, d, a\}, \\ V_2 &= \{A, b, f, e, a\}, \quad V_3 = \{g, h, e, f, g\}, \\ V_4 &= \{g, f, b, c, g\}, \quad V_5 = \{g, c, d, b, g\}. \end{aligned} \tag{34}$$

(iii) Determine the orientation of the local coordinate system $X_{i2}Y_{i2}Z_{i2}O_i$ based on the orientation of the local coordinate system $X_{i1}Y_{i1}Z_{i1}O_i$ as follows:

$$\begin{aligned} \hat{\mathbf{n}}_{i2} &= \hat{\mathbf{n}}_{i1}, \\ \hat{\mathbf{u}}_{i2} &= \frac{\hat{\mathbf{u}}_{i1} + \hat{\mathbf{v}}_{i1}}{|\hat{\mathbf{u}}_{i1} + \hat{\mathbf{v}}_{i1}|}, \\ \hat{\mathbf{v}}_{i2} &= \hat{\mathbf{n}}_{i2} \times \hat{\mathbf{u}}_{i2} \end{aligned} \tag{35}$$

Apply the polyhedron intersection algorithm to the block U_{i1} and the bounding block with respect to the local coordinate system $X_{i2}Y_{i2}Z_{i2}O_i$ to generate a new polyhedron U_{i2}, which points to the goal position G, as shown in Figure 4.

(iv) Determine the orientation of the local coordinate system $X_{i3}Y_{i3}Z_{i3}O_i$ based on the orientation of the local coordinate system $X_{i1}Y_{i1}Z_{i1}O_i$ as follows:

$$\begin{aligned} \hat{\mathbf{n}}_{i3} &= \hat{\mathbf{v}}_{i1}, \\ \hat{\mathbf{u}}_{i3} &= \frac{\hat{\mathbf{u}}_{i1} + \hat{\mathbf{n}}_{i1}}{|\hat{\mathbf{u}}_{i1} + \hat{\mathbf{n}}_{i1}|}, \\ \hat{\mathbf{v}}_{i3} &= \hat{\mathbf{n}}_{i3} \times \hat{\mathbf{u}}_{i3} \end{aligned} \tag{36}$$

Apply the polyhedron intersection algorithm to the block U_{i2} and the bounding block with respect to the local coordinate system $X_{i3}Y_{i3}Z_{i^3}O_i$ to generate a new polynedron U_{i3}, as shown in Figure 5.

(v) Determine the orientation of the local coordinate system $X_{i4}Y_{i4}Z_{i4}O_i$ based on the orientation of the local coordinate system $X_{i1}Y_{i1}Z_{i1}O_i$ as follows:

$$\begin{aligned} \hat{\mathbf{n}}_{i4} &= -\hat{\mathbf{u}}_{i1}, \\ \hat{\mathbf{u}}_{i4} &= \frac{\hat{\mathbf{v}}_{i1} + \hat{\mathbf{n}}_{i1}}{|\hat{\mathbf{v}}_{i1} + \hat{\mathbf{n}}_{i1}|}, \\ \hat{\mathbf{v}}_{i4} &= \hat{\mathbf{n}}_{i4} \times \hat{\mathbf{u}}_{i4} \end{aligned} \tag{37}$$

Apply the polyhedron intersection algorithm to the block U_{i3} and the bounding block with respect to the local coordinate system $X_{i4}Y_{i4}Z_{i4}O_i$ so as to generate a new polyhedron U_{i4}.

(vi) Determine the orientation of the local coordinate system $X_{i5}y_{i5}Z_{i5}O_i$ based on the orientation of the local coordinate system $X_{i2}Y_{i2}Z_{i2}O_i$ as follows:

$$\begin{aligned} \hat{\mathbf{n}}_{i5} &= \hat{\mathbf{v}}_{i2}, \\ \hat{\mathbf{u}}_{i5} &= \frac{\hat{\boldsymbol{u}}_{i2} + \hat{\mathbf{n}}_{i2}}{|\hat{\mathbf{u}}_{i2} + \hat{\mathbf{n}}_{i2}|}, \\ \hat{\mathbf{v}}_{i5} &= \hat{\mathbf{n}}_{i5} \times \hat{\mathbf{u}}_{i5} \end{aligned} \tag{38}$$

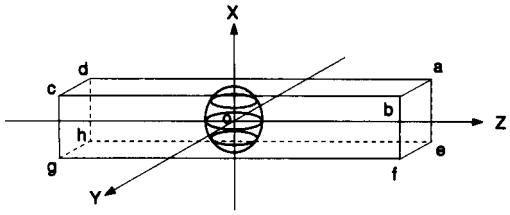


Fig. 3. The initial bounding block of an obstacle.

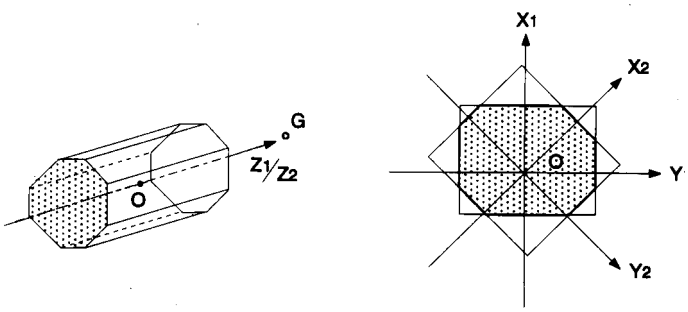


Fig. 4. Generation of the polyhedron U_{i2}.

https://doi.org/10.1017/S0263574797000520 Published online by Cambridge University Press

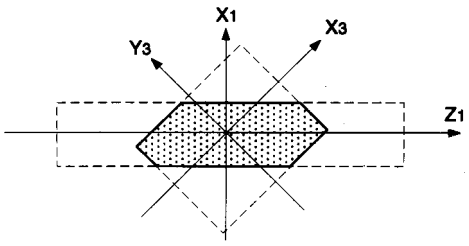


Fig. 5. Generation of the polyhedron U_{i3}.

Apply the polyhedron intersection algorithm to the block U_{i4} and the bounding block with respect to the local coordinate system $X_{i5}Y_{i5}Z_{i5}O_i$ so as to generate a new polyhedron U_{i5}.

(vii) Determine the orientation of the local coordinate system $X_{i6}Y_{i6}Z_{i6}O_i$ based on the orientation of the local coordinate system $X_{i2}Y_{i2}Z_{i2}O_i$ as follows:

$$\begin{aligned} \hat{\mathbf{n}}_{i6} &= -\hat{\mathbf{u}}_{i2}, \\ \hat{\mathbf{u}}_{i6} &= \frac{\hat{\mathbf{v}}_{i2} + \hat{\mathbf{n}}_{i2}}{|\hat{\mathbf{v}}_{i2} + \hat{\mathbf{n}}_{i2}|}, \\ \hat{\boldsymbol{v}}_{i6} &= \hat{\mathbf{n}}_{i6} \times \hat{\mathbf{u}}_{i6} \end{aligned} \tag{39}$$

Apply the polyhedron intersection algorithm to the block U_{i5} and the bounding block with respect to the local coordinate system $X_{i6}Y_{i6}Z_{i6}O_i$ so as to generate a new polyhedron U_{i6}. Output the boundary representation of the polyhedron U_{i6}, whose boundary surface consists of 24 polygons and tightly encloses the arbitrary shaped obstacle.

(viii) Check if all the obstacles in the 3-D workspace have been processed,
   a. If there exist more obstacles in the 3-D workspace which have not been processed by the algorithm, then let the $(i+1)$th obstacle be the current obstacle and go back to Step 1.
   b. If all obstacles in the 3-D workspace have been processed, then exit the algorithm.

3.2. Artificial potential field generation algorithm

Based on the boundary representation of the simple convex polyhedrons whose boundary surface enclose the arbitrary shaped obstacles in the 3-D workspace, the potential source-based panel method can be applied to generate a global artificial potential field in the 3-D workspace, in which the streamlines flow around the obstacles and go towards the goal position. The algorithm to implement the panel method is presented as follows.

Algorithm 2. Generate an artificial potential field based on the geometry of a 3-D workspace.

Input:

(i) The total number of obstacles in the 3-D workspace, which is represented by an integer m.

(ii) The world coordinates of the vertices of each polygonal panel over the boundary surface of the simple convex polyhedrons in the 3-D workspace, (the output of Algorithm 1):

$$\{V_0, V_1, \ldots, V_{m-1}\}$$

where

$$V_i = \{V_{i0}, V_{i1}, \ldots, V_{i(j_i-1)}\}$$

where

$$V_{ij} = \{\boldsymbol{v}_{ij0}, \boldsymbol{v}_{ij1}, \ldots, \boldsymbol{v}_{ij(k_{ij}-1)}, \boldsymbol{v}_{ij0}\} \tag{40}$$

where

$$\boldsymbol{v}_{ijk} = (x_{ijk}, y_{ijk}, z_{ijk}) \quad \text{and} \quad i = 0, 1, \ldots, m-1,$$

(iii) The world coordinates of the goal position **G**, (x_g, y_g, z_g), in the 3-D workspace.

Output:

1. The set of the strength values of potential source on all panels in the 3-D workspace, which is represented by a non-ordered list, $\{q_0, q_1, \ldots, q_{n-1}\}$, where n is the total number of the polygonal panels in the 3-D workspace.

Algorithm Steps:

Assume that the total number of the potential panels over the boundary surface of the m convex polyhedrons in the 3-D workspace is n.

(i) Allocate the memory block for a $n \times (n+1)$ matrix M. The most convenient data structure for the $n \times (n+1)$ matrix is a two-dimensional array M[0 . . n − 1][0 . . n] in which M[i][i] contains the element at the ith row and jth column of the matrix.

(ii) Compute the parameter B_i of equation (10) according to equation (11) and (3), and store the parameters B_i $(i = 0, 1, \ldots, n-1)$ into the $(n+1)$th column of the $n \times (n+1)$ matrix, M[0 . . n − 1][n].

(iii) Compute the parameter A_{ij} of equation (10) according to equation (11) and (39), and store the n^2 parameters A_{ij} $(i, j = 0, 1, \ldots, n-1)$ into the first n columns of the $n \times (n+1)$ matrix M, i.e. $M[i][j] = A_{ij}$ for $i, j = 0, 1, 2, \ldots, n-1$.

(iv) Solve the strength value of potential source on each panel, q_j $(j = 0, 1, \ldots, n-1)$, from the linear equations (10) using the Gauss elimination algorithm. Since the coefficients of the linear equations have been stored in the two-dimensional array M[0 . . n − 1][0 . . n], the computational procedure of the Gauss elimination algorithm can be expressed as:

$$\mathrm{M} = \begin{bmatrix} A_{00} & A_{01} & \ldots & \ldots & A_{0(n-1)} & B_0 \\ A_{10} & A_{11} & \ldots & \ldots & A_{1(n-1)} & B_1 \\ A_{20} & A_{21} & \ldots & \ldots & A_{2(n-1)} & B_2 \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ A_{(n-1)0} & A_{(n-1)1} & \ldots & \ldots & A_{(n-1)(n-1)} & B_{n-1} \end{bmatrix}$$

https://doi.org/10.1017/S0263574797000520 Published online by Cambridge University Press

$$\Rightarrow \begin{bmatrix} 1 & 0 & \dots & \dots & 0 & q_0 \\ 0 & 1 & \dots & \dots & 0 & q_1 \\ 0 & 0 & \dots & \dots & 0 & q_2 \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & \dots & 1 & q_{n-1} \end{bmatrix} \quad (41)$$

(v) Output the calculated strength values of potential source, $\{q_0, q_1, \dots, q_{n-1}\}$, release the memory block for the $n \times (n-1)$ matrix M, then exit the algorithm.

3.3. Streamline (path) generation algorithm

After the artificial potential field in the 3-D workspace has been generated, the motion path starting from any point to the goal position in the 3-D workspace can be obtained by generating the particular streamline in the artificial potential field. Since the velocity vector at any point along a streamline is tangent to the streamline in a potential field, the streamline can be generated by a sequence of small straight-line displacements, whose directions are determined by the velocity vectors at the starting points of each displacement. The streamline generation algorithm is presented in the following.

Algorithm 3. Generate a collision-free streamline starting from a given position towards the goal position in the 3-D workspace.

Input:

(i) The total number of obstacles in the 3-D workspace, which is represented by an integer m.

(ii) The world coordinates of the vertices of each polygonal panel over the boundary surface of the simple convex polyhedrons in the 3-D workspace (the output of Algorithm 1):

$$\{V_0, V_1, \dots, V_{m-1}\}$$

where

$$V_i = \{V_{i0}, V_{i1}, \dots, V_{i(j_i-1)}\}$$

where (42)

$$V_{ij} = \{v_{ij0}, v_{ij1}, \dots, v_{ij(k_{ij}-1)}, v_{ij0}\}$$

where

$$v_{ijk} = (x_{ijk}, y_{ijk}, z_{ijk}) \quad \text{and} \quad i = 0, 1, \dots, m-1,$$

(iii) The world coordinates of the path starting position S, (x_s, y_s, z_s), in the 3-D workspace.

(iv) The world coordinates of the goal position G, (x_g, y_g, z_g), in the 3-D workspace.

(v) The list of the strength values of potential source on all panels in the 3-D workspace, $\{q_0, q_1, \dots, q_{n-1}\}$, where n is the total number of the panels in the 3-D workspace. (The output of Algorithm 2)

Output:

(i) The world coordinates of a sequence of points on a collision-free motion path from the starting position S to the goal position G in the 3-D workspace if the collision-free path exists. The motion path can be represented by a list of point coordinates, $\{P_0, P_1, P_2, \dots, P_k\}$, in which P_0 is the coordinates of the starting point S while P_k is the coordinates of the goal position G, and the $P_1, P_2, \dots, P_{k-1}$ are the coordinates of the intermediate points on the path, as shown in Figure 6.

Algorithm Steps:

Assume that the current position on the partially generated path starting from the point S is P_i.

(i) Compute the velocity components at the current position P_i due to the potential sink at the goal position G ($u_g(x, y, z)$, $v_g(x, y, z)$ and $w_g(x, y, z)$ in equation (12)) according to equation (3).

(ii) Compute the velocity components at the current position P_i due to the potential sources over all polygonal panels in the potential field ($\sum_{j=0}^{N-1} q_j U_x^j(x, y, z)$, $\sum_{j=0}^{N-1} q_j U_y^j(x, y, z)$ and $\sum_{j=0}^{N-1} q_j U_z^j(x, y, z)$ in equation (12)) according to equation (39).

(iii) Add the calculated velocity components at the current position P_i together to obtain the total velocity vector $\mathbf{V}_{P_i}$ according to equation (12). Check the absolute magnitude of the velocity vector, $|\mathbf{V}_{P_i}|$,

a. If the magnitude of the velocity is zero, i.e. $|\mathbf{V}_{P_i}| = 0$, which indicates the current position P_i is a stagnation point in the potential field, then change the current position P_i with a small displacement ΔP,

$$P_i = P_i + \Delta P \quad (43)$$

go back to Step 1 and continue the algorithm.

b. If the magnitude of the velocity has a dramatic jump with respect to the magnitude of the velocity of the previous step displacement, i.e. $|\mathbf{V}_{P_i}| \gg |\mathbf{V}_{P_{i-1}}|$, which indicates the current position P_i on the path is inside the boundary surface of an obstacle in the 3-D workspace and the partially generated path fails, then either exit the algorithm and report failure, or begin proper recovery procedure to continue the path generation.

c. If the magnitude of the velocity is close to the magnitude of the velocity of the previous step displacement, i.e. $\|\mathbf{V}_{P_i}\| - |\mathbf{V}_{P_{i-1}}\| \le \Delta V$, then continue the algorithm to the next step.

(iv) Output the current position P_i as a point on the generated path, and obtain the new current position P_{i+1} by adding a small displacement ($c > 0$) to the point P_i in the direction of its velocity vector $\mathbf{V}_{P_i}$, i.e.

$$P_{i+1} = P_i + c\mathbf{v}_{P_i}, \quad \text{where} \quad \mathbf{v}_{P_i} = \frac{\mathbf{V}_{P_i}}{|\mathbf{V}_{P_i}|} \quad (44)$$

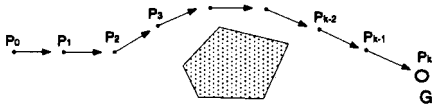


Fig. 6. Streamline generation.

https://doi.org/10.1017/S0263574797000520 Published online by Cambridge University Press

(v) Compare the new current position on the path P_{i+1} with the goal position G in the 3-D workspace to verify if the generated path has already reached the goal position G,
   a. If $(P_{i+1} = G)$ then exit the algorithm.
   b. If $(P_{i-1} \neq G)$ then go back to Step 1 and begin the next loop with the new current position P_{i+1}.

3.4. Computational complexity

For Algorithm 1, assume that the ith obstacle in the 3-D workspace has N_{s_i} sample points on its boundary surface. Since the major steps of the algorithm are just the recursive applications of the polyhedron intersection algorithm to the fixed polyhedrons, which have limited number of vertices, edges, and polygons, the computational complexity of processing the ith obstacle in the 3-D workspace is $O(N_{s_i} + c_1)$, and the computational complexity of processing the m obstacles in the 3-D workspace is $O(\sum_{i=0}^{m-1} N_{s_i} + c_1 m)$ where c_1 is a constant. In the practical situation of the sensor-based motion planning, the number of the sample points on the boundary surface of each obstacle is limited, i.e. $N_{s_i} \leq c_2$ where c_2 is a constant, thus, the computational complexity of Algorithm 1 can be represented as:

$$O\left(\sum_{i=0}^{m-1} N_{s_i} + c_4 m\right) = O(c_2 m + c_1 m) = O(m) \qquad (45)$$

For Algorithm 2, let n denote the total number of the potential panels in the 3-D workspace. Since the Gauss elimination algorithm is well known to be an algorithm with cubic polynomial complexity, the computational complexity of Algorithm 2, which solves n variables from n linear equations using the Gauss elimination algorithm, is $O(n^3)$. Since the obstacle deformation algorithm (Algorithm 1) transforms each arbitrary shaped obstacle into a convex polyhedron which has twenty four planar polygons (panels) on its boundary surface, i.e. $n = 24m$, therefore, the computational complexity of Algorithm 2 can also be represented as:

$$O(n^3) = O(24^3 m^3) = O(m^3) \qquad (46)$$

For Algorithm 3, the computational complexity of generating one step displacement along the collision-free path is $O(n)$. Assume that the entire motion path generated by Algorithm 3 consists of k step displacements, the computational complexity of Algorithm 3 is:

$$O(kn) = O(k \cdot 24m) = O(km) \qquad (47)$$

Therefore, the computational complexity of generating a collision-free path starting from any position to the goal position in a 3-D static environment, which includes Algorithm 1, Algorithm 2, and Algorithm 3, can be represented as:

$$O(c_1 m + c_2 m^3 + c_3 km) \qquad (48)$$

Obviously, if the total number of the step displacements, k, in the genrated path is less than or equal to the square of the obstacle number, m^2, i.e. $k \leq m^2$, the computational complexity of generating a collision-free motion path in a 2-D static environment is $O(m^3)$; otherwise, the computational complexity is $O(km)$. Anyway, this path planning algorithm is extremely fast and perfectly suitable for on-line path planning in a static environment.

4. REACTIVE MOTION PLANNING IN A DYNAMIC ENVIRONMENT

There always exist uncertainties during the robot motion, such as moving obstacles in the workspace, inaccurate sensor signals due to noise, under-activation of robot joints, actuator failures, etc. Such uncertainties are called *unbounded uncertainties.*

The motion planning problem with bounded uncertainty may be stated as follows [20]:

Given an initial region I *and a goal region* G *in the robot's configuration space, generate a motion plan whose execution guarantees the robot to reach a configuration in* G *if it starts from any* (*unknown*) *configuration in* I, *despite bounded uncertainty in control, sensing, and model.*

Since emphasis is put in obstacle avoidance during robot motion, only uncertainty caused by moving obstacles in the robot workspace is considered here. A reactive motion planning strategy, which is suitable for sensor-based free-flying robots to implement on-line motion planning in a dynamic environment with moving obstacles, is developed. It is assumed that all necessary sensor data are available and registered in world coordinates.

Algorithm 4. Navigate a free-flying robot towards a given goal while avoiding collision with randomly moving obstacles in the 3-D workspace.

Input:

(i) The world coordinates of the starting position S in the 3-D workspace.
(ii) The world coordinates of the goal position G in the 3-D workspace.
(iii) The world coordinates of the sample points on the boundary surface of the obstacles which enter the sensing range of the range sensor.

Output:

(i) A sequence of on-line motion commands which instruct the robot moving towards the goal position while avoiding collision with the randomly moving obstacles. The robot motion path can be represented by a list of point coordinates iwth the timing to reach these points, i.e.:

$$\{P_0(t_0), P_1(t_0 + \Delta t), P_2(t_0 + 2 \cdot \Delta t), \ldots, P_k(t_0 + k \cdot \Delta t)\} \qquad (49)$$

where $P_0(t_0)$ represents that the robot starts its motion from the starting point S at time t_0; Δt is the time interval required by the robot to complete sensing, planning, and executing one step motion displacement; $P_k(t_0 + k \cdot \Delta t)$ represents that the robot reaches

https://doi.org/10.1017/S0263574797000520 Published online by Cambridge University Press

the goal position G after k step motion displacements; and the $P_1, P_2, \ldots, P_{k-1}$ represent the intermediate points on the robot motion path.

Algorithm Steps:

Let the robot motion towards the goal position G start at time t_0, and the initial robot position $P_0(t_0)$ be the starting position S. Assume that the range sensors mounted on the mobile robot can provide the sample point positions of obstacles within its sensing range in real-time.

(i) Adjust the direction of the robot's range sensor so that the center line of the sensing range goes through the goal position G, as shown in Figure 7. Then acquire the sample point positions of the obstacles within the sensing range through the sensors.
(ii) Apply Algorithm 1 to generate the convex shaped polygons, which enclose the arbitrary shaped obstacles, based on the sample points over the boundary surface of the obstacles.
(iii) Apply Algorithm 2 to generate an artificial potential field within the sensing range of the range sensor.
(iv) Apply Algorithm 3 to obtain the velocity vector $\mathbf{V}_{P_i}$ at the current robot position P_i in the artificial potential field.
(v) Generate the next robot position P_{i+1} according to the computed velocity vector $\mathbf{V}_{P_i}$, by adding a small displacement ($c > 0$) to the point P_i in the direction of its velocity vector $\mathbf{V}_{P_i}$, i.e.

$$P_{i+1} = P_i + c\mathbf{v}_{P_i}, \quad \text{where} \quad \mathbf{v}_{P_i} = \frac{\mathbf{V}_{P_i}}{|\mathbf{V}_{P_i}|} \tag{50}$$

(vi) Drive the robot to complete the computed step motion displacement, $P_i \rightarrow P_{i+1}$ within the time interval Δt by executing the appropriate motion command. Thus the current robot position/time is $P_{i+1}(t_i + \Delta t)$.
(vii) Compare the new current position P_{i+1} with the goal position G in the robot workspace to verify if the robot has already reached the goal position G,
   a. If $(P_{i+1} = G)$ then exit the algorithm.
   b. If $(P_{i+1} \neq G)$ then go back to Step 1 and begin the next loop of sensing, planning and executing with the new current robot position P_{i+1}.

Computational complexity: As proved in the previous section, the computational complexity of generating one step displacement towards the goal position is $O(m^3)$, where m is the number of obstacles within the current sensing range. Assume that the free-flying robot uses k step displacements to reach the goal position, the numbers of obstacles within the k different sensing ranges along the robot motion path are obviously random. Let the average number of obstacles within the k different sensing ranges be $\bar{m}$, then the computational complexity of generating the entire motion path towards the goal position is $O(k\bar{m}^3)$. The average number of obstacles, $\bar{m}$, within the different sensing ranges can be easily controlled by the size of the sensing range.

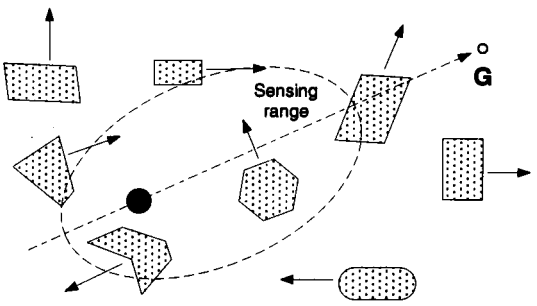


Fig. 7. Sensor based motion planning among moving obstacles.

5. SIMULATION RESULTS

Figures 8, 9 and 10 show some simulation results of the 3-D potential panel-based path planning algorithms introduced in this paper. Each figure shows the streamlines in a particular artificial potential field generated by the obstacles and the goal. The small circles at one end of each streamline indicates that the streamline starts from this point and goes towards the goal position (indicated by the big circle), so that the direction of the potential flow in the particular potential field is clearly shown. Since the collision-free motion paths are generated in the 3-D workspace, six viewpoints are chosen for each simulation case so that the 3-D collision-free paths can be shown.

6. CONCLUSIONS

The motion planning method presented in this paper combines the advantages of model-based and sensor-based motion planning approaches, thus, it is applicable to both global motion planning and local motion planning in the 3-D space, and provides a novel perspective for designing practical motion planning algorithms for free-flying robot.

The advantages of the proposed motion planning method are:

(i) The generated motion path in the artificial potential field has no local minima but one global minimu mat the goal position. Thus, the robot is guaranteed to reach the goal position as long as it follows the streamline passing through the starting position.
(ii) The path planning method is analytical as opposed to numerical, which makes the method highly efficient and applicable to the dynamic environment with moving obstacles.
(iii) The path planning algorithms are designed to base only on sensory information and basic geometric modeling techniques, other than complex representation of arbitrary shaped obstacles, thus, the algorithms are perfectly suitable for sensor-based real-time path planning.
(iv) The generated paths by the potential panel method are smooth curves, which enable the steering mechanism of a robot to follow the path easily whlie maintaining a reasonable fast motion speed, and

https://doi.org/10.1017/S0263574797000520 Published online by Cambridge University Press

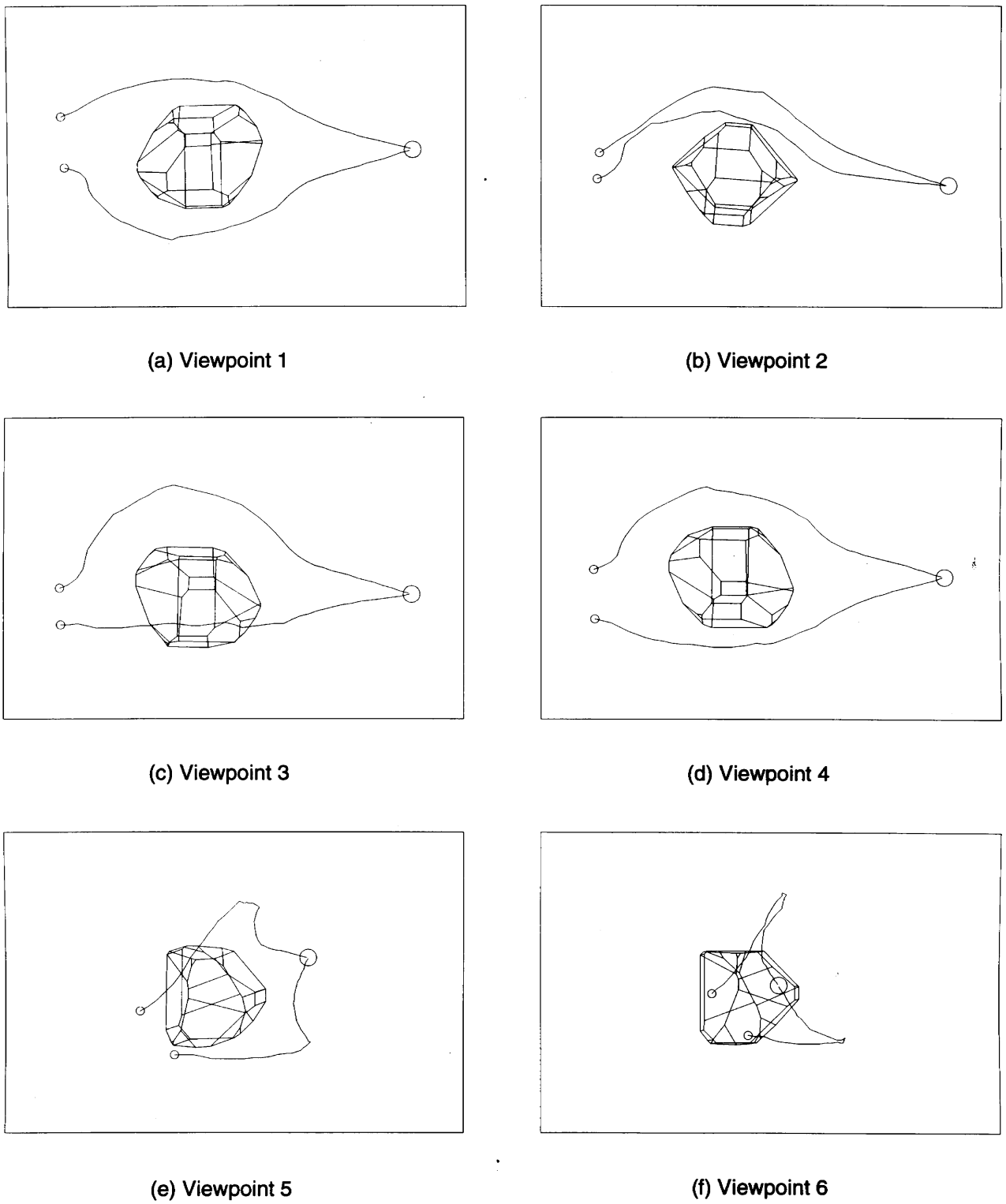


Fig. 8. Simulation of path planning in a one-obstacle workspace.

avoid the jerky stop and sharp turn which are out of the kinematic and dynamics constraints of the robot.

Although the proposed potential panel method is an attractive approach to the motion planning problem because of its simplicity and efficiency, it has some inherent limitations. The limitations of the potential panel method are:

(i) This motion planning method is not very effective in an environment with densely spaced obstacles. Since the generated convex polyhedrons are easily overlapped, the generated path is unable to pass through the obstacles.
(ii) This motion planning method can not be used to find the path in a maze environment, because the ideal-fluid flow can not flow back and forth in the narrow passages of the maze.

https://doi.org/10.1017/S0263574797000520 Published online by Cambridge University Press

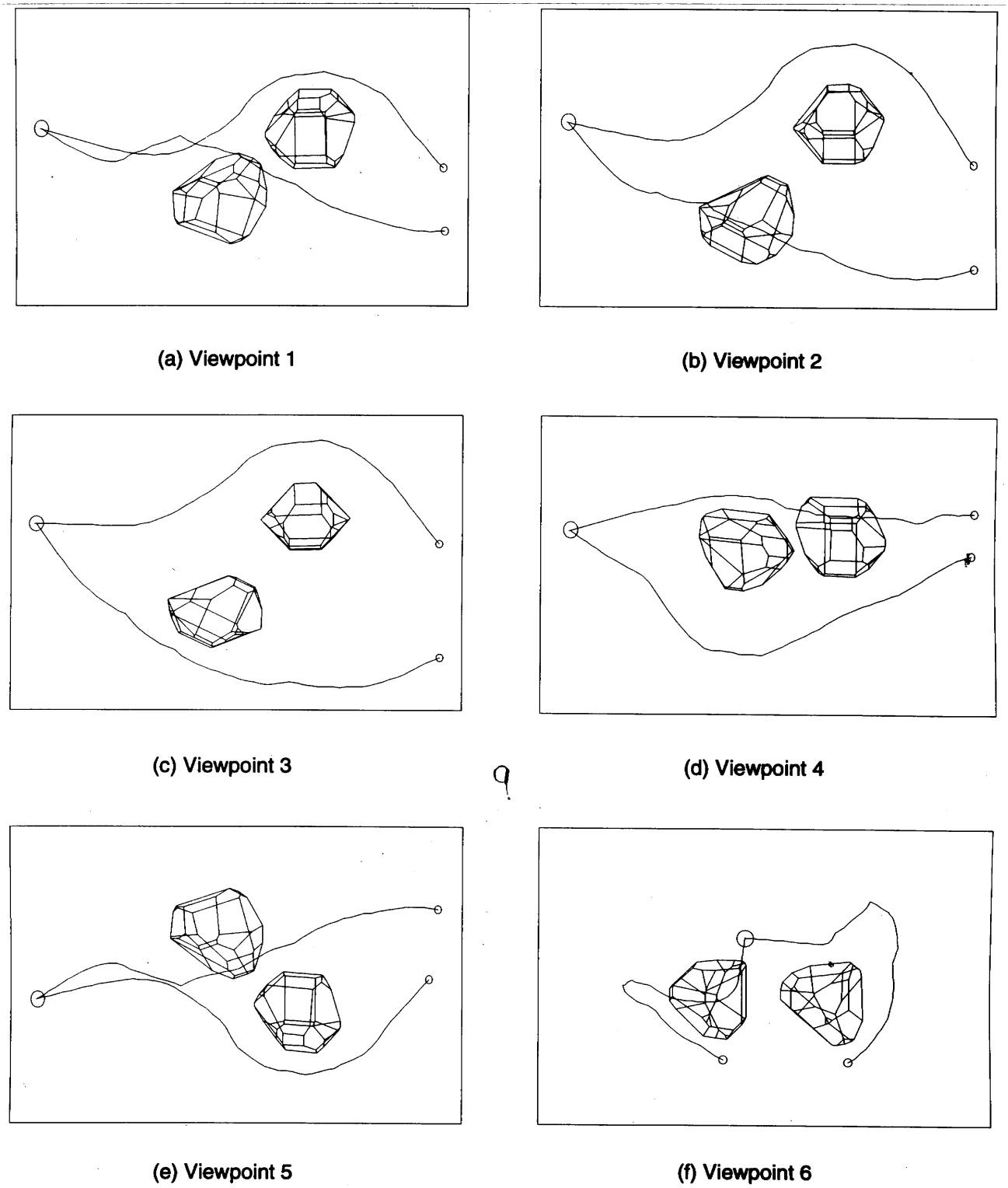
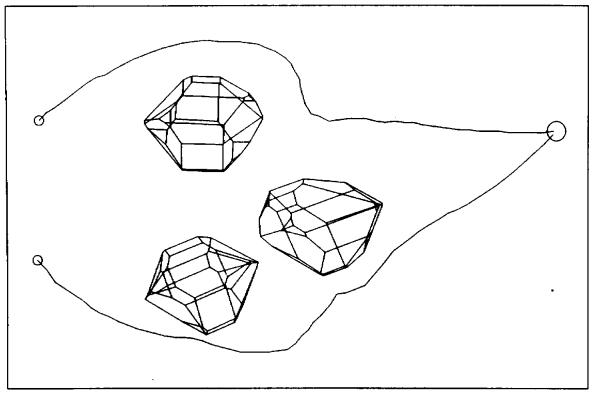
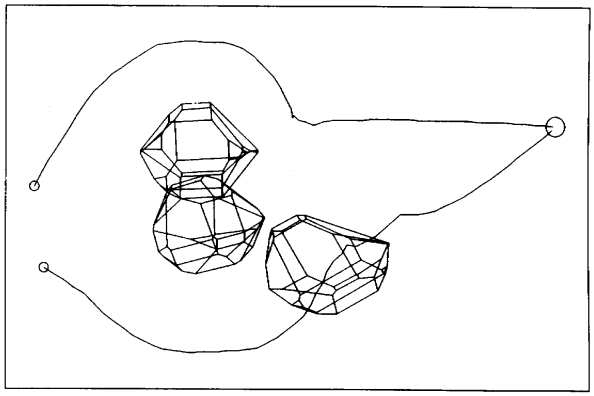


Fig. 9. Simulation of path planning in a two-obstacle workspace.

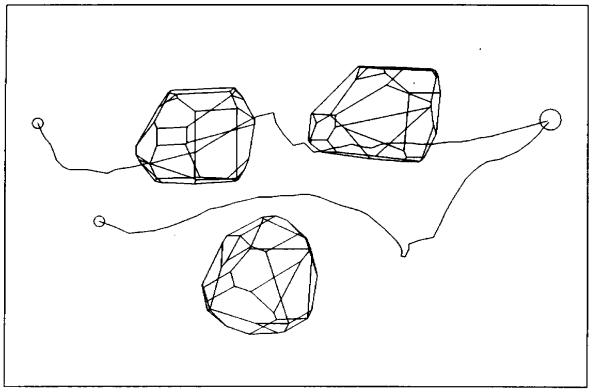
https://doi.org/10.1017/S0263574797000520 Published online by Cambridge University Press



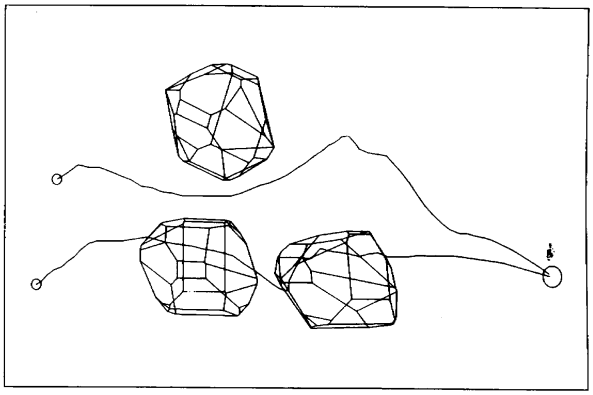
(a) Viewpoint 1



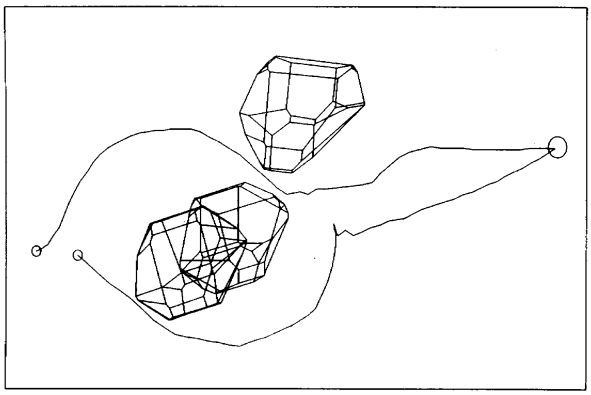
(b) Viewpoint 2



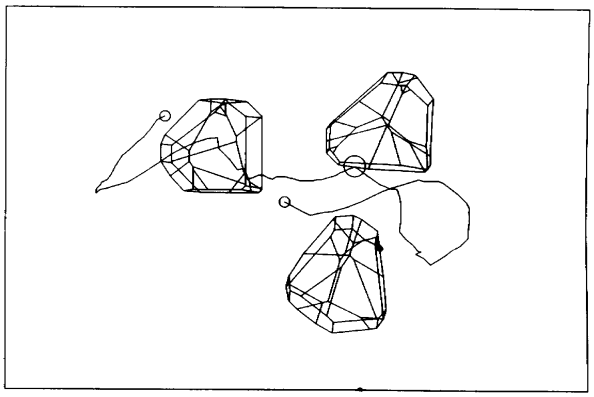
(c) Viewpoint 3



(d) Viewpoint 4



(e) Viewpoint 5



(f) Viewpoint 6

Fig. 10. Simulation of path planning in a three-obstacle workspace.

https://doi.org/10.1017/S0263574797000520 Published online by Cambridge University Press

Overall the proposed algorithms are implementable in real-time and applicable for free-flying robot autonomous motion planning in undersea or outer space environment.[2]

7. Bibliography

1. Y. Zhang and K.P. Valavanis, "Sensor-Based 2-D Potential Panel Method for Robot Motion Planning" **Robotica, 14,** Part 1, 81–89 (1996).
2. Y. Zhang, "Sensor-based Potential Panel Method for Robot Motion Planning" *Ph.D. thesis* CACS, The University of Southwestern Louisiana, 1995).
3. O. Khatib, "Real-time Obstacle Avoidance for Manipulators and Mobile Robots" *Proceedings of the IEEE International Conference on Robotics and Automation,* St. Louis, MO (1985) pp. 500–505.
4. B. Krogh, "A Generalised Potential Field Approach to Obstacle Avoidance Control" *Robotics Research: the next five years and beyond, SME Conference Proceedings* (August 1984) pp. 64–69.
5. D.E. Koditschek, "Exact Robot Navigation by Means of Potential Functions: Some Topological Considerations" *Proceedings of the IEEE International Conference on Robotics and Automation,* Raleigh, NC (1987) pp. 1–6.
6. E. Ritman and D.E. Koditschek, "Exact Robot Navigation using Cost Functions: The Case of Distinct Spherical Boundaries in En" *Proceedings of the IEEE International Conference on Robotics and Automation,* Philadelphia, PA (1988) pp. 1791–1796.
7. E. Riman and D.E. Koditschek, "The Construction of Analytic Diffeomorphisms for Exact Robot Navigation on Star Worlds" *Proceedings of the IEEE International Conference on Robotics and Automation,* Scottsdale, AZ (1989) pp. 21–26.
8. E. Riman and D.E. Koditschek, "Exact Robot Navigation in Geometrically Complicated but Topologically Simple Spaces" *Proceedings of the IEEE International Conference on Robotics and Automation,* Cincinnati, OH (1990) pp. 1973–1942.
9. E. Rimen, "A Navigation Function for a Simple Rigid Body" *Proceedings of the IEEE International Conference on Robotics and Automation,* Sacramento, CA (1991) pp. 546–551.
10. P. Khosla and R. Volpe, "Superquadric Artificial Potentials for Obstacle Avoidance and Approach" *Proceedings of the IEEE International Conference on Robotics and Automation,* Philadelphia, PA (1988) pp. 648–649.
11. R. Volpe and P. Khosla, "Manipulator Control With Superquadric Artificial Potential Functions: Theory and Experiments" *IEEE Transactions on Systems, Man and Cybernetics* **20**(6), 1423–1436 (1990).
12. J. Barraquand and J.C. Latombe, "A Monte-Carlo Algorithm for Path Planning With Many Degrees of Freedom" *Proceedings of the IEEE International Conference on Robotics and Automation,* Cincinnati, OH (1990) pp. 1712–1717.
13. J. Barraquand and J.C. Latombe, "Robot Motion Planning: A Distributed Representation approach" *Int. J. Robotics Research* **10**(6), 628–649 (1991).
14. D. Megherbi and W.A. Wolovish, "Real-Time Velocity Feedback Obstacle Avoidance Via Complex Variables and Conformal Mapping" *Proceedings of the IEEE International Conference on Robotics and Automation,* Nice, France (1992) pp. 206–213.
15. J.O. Kim and P.K. Khosla, "Real-Time Obstacle Avoidance Using Harmonic Potential Functions" *Proceedings of the IEEE International Conference on Robotics and Automation,* Sacramento, CA (1991) pp. 1728–1733.
16. J.O. Kim and P.K. Khosla, "Real-time Obstacle Avoidance Using Harmonic Potential Functions" *IEEE Transactions on Robotics and Automation,* **8**(3), 338–349 (1992).
17. J.H. Chuang, "Potential Based Modeling of Three Dimensional Workspace for Obstacle Avoidance" *Proceedings of the IEEE International Conference on Robotics and Automation,* Atlanta, GA (1993) pp. 19–24.
18. R.B. Tilove, "Local Obstacle Avoidance for Mobile Robots Based on the Method of Artificial Potentials" *Proceedings of the IEEE international Conference on Robotics and Automation,* Cincinatti, OH (1990) pp. 566–571.
19. D.R. Wilton, S.M. Rao, A.W. Glisson, D.H. Schuabert, O.M. Al-Bundak and C.M. Butler, "Potential Integrals for Uniform and Linear Source Distributions on Polygonal and Polyhedral Domains" *IEEE Transactions on Antennas and Propagation,* AP-32(3), 276–281 (March 1984).
20. J.C. Latombe, *Robot Motion Planning* (Kluwer Academic Publishers, Norwell, MA, 1991).

https://doi.org/10.1017/S0263574797000520 Published online by Cambridge University Press