Research Article

INTERNATIONAL JOURNAL OF ADVANCED ROBOTIC SYSTEMS

International Journal of Advanced Robotic Systems
January-February 2018: 1–11

DOI: 10.1177/1729881418757046
journals.sagepub.com/home/arx
SAGE

Human experience–inspired path planning for robots

Wenyong Gong[1,2], Xiaohua Xie[2,3] and Yong-Jin Liu[4]

Abstract

In this article, we present a human experience–inspired path planning algorithm for service robots. In addition to considering the path distance and smoothness, we emphasize the safety of robot navigation. Specifically, we build a speed field in accordance with several human driving experiences, like slowing down or detouring at a narrow aisle, and keeping a safe distance to the obstacles. Based on this speed field, the path curvatures, path distance, and steering speed are all integrated to form an energy function, which can be efficiently solved by the A* algorithm to seek the optimal path by resorting to an admissible heuristic function estimated from the energy function. Moreover, a simple yet effective fast path smoothing algorithm is proposed so as to ease the robots steering. Several examples are presented, demonstrating the effectiveness of our human experience–inspired path planning method.

Keywords

Path planning, human experience, A* search algorithm, path smoothing

Date received: 5 July 2017; accepted: 5 January 2018

Topic: Robot Manipulation and Control
Topic Editor: Andrey V Savkin
Associate Editor: Yongping Pan

Introduction

The path planning, which refers to designing a path for navigating an agent toward a desired location from a start position, has been widely applied in robotics community and other real application fields,[1] such as window cleaning,[2] exploration of Mars, and video game.[3] Among these applications, path planning for service robots plays a central role in complex indoor and outdoor environments.[4]

In general, path planning is to find a sequence of location transition actions that transform a start position to a goal position, where each transition action has an associated cost, and the sum of costs of all transition actions represents some measurement for the path. Designing transition actions for path planning generally concerns on following aspects: (i) travel time between the start position and the goal position (also referred as the time factor); (ii) energy spent of an agent traveling a path; (iii) robots do not collide with other objects; and (iv) smoothness of a path is desired to ease steering of robots. Currently, main path planning methods aim at designing an optimization model which considers one or more of the above-mentioned aspects, and then performing a minimization procedure to achieve an optimal path.[1,4–7] For example, the shortest path, minimum time-consuming path, minimal energy cost, and coverage path planning are, respectively, studied in the literature[6,8,9] for a given navigation task. Path planning

[1] Department of Mathematics, Jinan University, Guangzhou, China
[2] Key Laboratory of Machine Intelligence and Advanced Computing, Sun Yat-sen University, Ministry of Education, Guangzhou, China
[3] School of Data and Computer Science, Sun Yat-Sen University, Guangzhou, China
[4] TNList, Department of Computer Science and Technology, Tsinghua University, Beijing, China

Corresponding author:
Xiaohua Xie, School of Data and Computer Science, Sun Yat-sen University, Guangzhou Higher Education Mega Center, 510006 Guangzhou, China.
Email: xiexiaoh6@mail.sysu.edu.cn

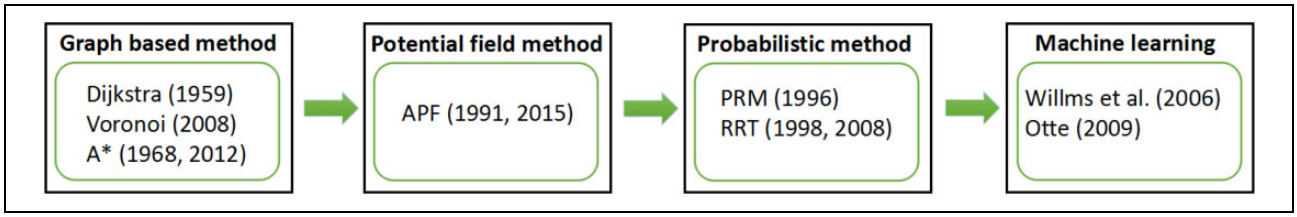

Figure 1. The developing process of path planning.

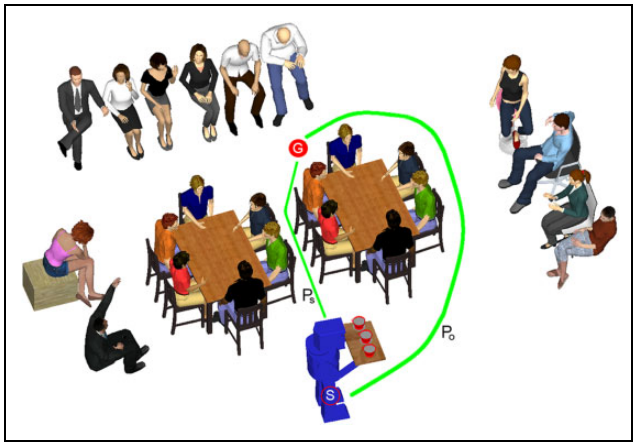

Figure 2. Illustration of human experience–inspired path planning. In this scene, a service robot carrying several cups of coffee on a plate is serving at a party and is required to traverse from the current position S to the target position G. The shortest path P_s is selected by the traditional algorithms, and the path P_o is selected by the human experience–inspired algorithm.

involves four developing stages: graph-based methods (e.g. Dijkstra, Voronoi,[10] A* and its variants[5,11,12]), artificial potential method,[7,13] probabilistic methods (e.g. probabilistic roadmap method (PRM),[14] rapidly exploring random tree (RRT)[15,16]), and machine learning-based methods.[17,18] Tthe developing process of path planning is shown in Figure 1.

While existing methods give available solutions for practical applications, neither of them has considered the safety of navigation. In practice, the safety requirement is very important. For example, a relatively narrow space between a robot and persons may easily cause collision between them especially when persons are playing regular activities (e.g. hand waving), resulting in an uneasy environment for humans. Therefore, a safety requirement is explicitly considered along with the distance and smoothness of a path for service robots in this article.

Our scheme is partly inspired by the human experience, because humans seem to be experts in path selection and could well analyze and synthesize diverse factors (e.g. distance, smoothness, and safety) in mind to make a proper decision for path planning. We therefore design a new path planning algorithm for robots by learning from human experience especially in terms of safe driving. Take the scene in Figure 2 as an example, where a service robot is ordered to fetch several cups of coffee for participants of a party. While a traditional algorithm may plan the path P_s, the human experience–inspired algorithm tends to select the path P_o which should be more safe although taking a little longer distance.

In fact, a large number of robots has been built that explicitly mimic biological navigation behaviors for obstacle avoidance, such as the ones emulating an annual migration of seabirds[19,20] and ant-like behavior navigation model.[21] Inspired by the social interactions in human crowds or animal swarms, Savkin and Wang[22] proposed an efficient obstacle avoidance algorithm in dynamic environments by integrating representation of the information about the environment. Typically, the development of robotics has been directly or indirectly affected by human's experiences and behaviors.[23–25] However, these methods[24–27] mainly focus on the motion modeling of robot's body parts (e.g. arms) and the interactive applications between robots and humans. In this article, we emphasize and learn from human driving experiences. Let's take some discussion on the human's driving experiences. Generally, a person prefers reducing the driving speed when passing neighboring areas of obstacles, while speeding up in wide areas with safe enough distance to obstacles. Furthermore, in front of a very narrow aisle, a person like to select another wide path even with a little longer distance. Yet, she/he may determine to continue the narrow path if the wide one is too much long compare with the narrow one. Passive human walking has been studied by many researchers,[28] and it simulates human's behavior by considering the potential energy consuming when human walking. In a word, path planning by humans is not only related to the time cost, path distance, and ease steering but also relevant to the safety.

Inspired by human's experiences, we develop a safety-related path planning method. We construct a speed field that can efficiently simulate slowing down and detouring behaviors of humans around obstacles and then integrate the distance and smoothness factors with the speed field to form a human experience–inspired energy function model. Based on this model, the optimal path can be efficiently achieved by the well-known A* algorithm. Specially, the lower bound of the energy function, namely the admissible function in A* algorithm, can be easily estimated. We also propose a fast yet effective post-processing to smooth paths generated by the A* method. In summary, the main contributions of this article include:

i. A human experience–inspired path planning algorithm that considers the safety factor is presented.
ii. A dynamic speed field is constructed to control robot's speed, which can also adapt to dynamic scenes.
iii. An effective post-processing approach is utilized to smooth paths for ease steering of robots.

The rest of this article is organized as follows. In "Related work" section introduces some related works. In "The proposed model" section, an energy functional model of path planning is presented to simulate human's experience, and speed field computation and a practical feasible curvature computation scheme are also detailed. In "Path planning scheme" section, A* algorithm is applied to the energy function to search an optimal path, and a path smoothing method is also presented. Several path evaluation criteria are introduced in "Path evaluation" section. Experimental results are shown in "Implementation and results" section. Finally, conclusions are made in "Conclusions and future works" section.

Related work

Path planning has been widely studied by robot communities and other researchers, and plenty of methods have been proposed to deal with practical problems. In this section, we mainly focus on representative path planning algorithms in two dimensional scenarios, and more references can be found in the study by LaValle.[1]

Visibility graph[29,30] views obstacles in configuration spaces as polygons and then constructs a graph using the start position, the goal position, and vertices of polygons. Then the path is obtained by graph search methods, for example, Dijkstra's algorithm.

Artificial potential field (APF)[7,13,31,32] is another important method for path planning in robots. In general, obstacles are modeled as repulsive fields in APF, while the goal position is considered as an attractive field. Thus, the repulsive fields avoid agents approaching to obstacles, and the attractive fields move agents to the goal. Paths generated by APF are generally smooth, but the main drawback of APF is that a local minimizer is trapped possibly.

Probabilistic path planning algorithm is an effective method, and its two representative methods are the PRM[14] and the RRT.[15,16,33] The basic idea of probabilistic path planning is to randomly select non-collision points in free motion space and then connect them to get a path. Probabilistic methods are probabilistically complete, and the path generated by them for the same problem is also not unique.

A* method[5,11] is a widely used path planning method since it has many good properties: (1) given start and goal positions in a scene, the path achieved by A* is unique; (2) in a finite time, A* can always return a result even there is no solution; and (3) a good admissible function can lead to an acceptable time-consuming even for a large map. Now many researchers have developed various variants of A* to deal with different situations and tasks, such as D* Lite,[34] any-angle A*,[3] and Field D*.[12]

Gradient-based methods[35,36] are a two-step algorithm of designing an optimal path. It first uses a straight line to connect the start and target points (even pass through obstacle regions) and then takes points out of obstacles in the gradient directions. Generally, the raw paths generated by the process are not smooth, thus a post smoothing operation is desired.

When an agent faces a completely unknown environment and equips with some sensors, it requires to deal with all kinds of situations according to the sensed information. In the case, machine learning-based methods[17,18] are a good technology to deal with path planning tasks.

Comparing with other path planning algorithms, A* algorithm is a deterministic and flexible approach, and it is very convenient to search an optimal solution for a concrete path planning problem. In this article, we propose using the A* algorithm to optimize the proposed path planning model.

Our method is partially similar to Dolgov et al.'s method,[10] and both methods consider the practical safe requirement, but their method does not consider the distance factor and causes the generated paths often along with central lines of roads. As compared, our method can flexibly adjust the path by controlling parameters of the speed field.

The proposed model

In this section, we introduce the energy function of our path planning method according to human experiences.

Energy function

The driving path of a person is related to both the smoothness and the psychological speed of the path. Generally, people prefer driving on a wide road with high speed, which motivates us to design the energy function E inversely proportional to the speed v. However, when v is too small, $\frac{1}{v}$ would tend to infinity and be the dominant contribution for path planning, which is not desired in practice. Therefore, we set

$$E \propto M - v \tag{1}$$

where $M > 0$ is a large enough constant.

On the other hand, humans are prevailing to smooth paths with little turnings. An ideal case is that the driving road is almost straight. In fact, the steering experience implies two factors: the driving time is as short as possible, and the driving road is as straight and smooth as possible. Let s be the driving distance, and κ is the path smoothness parameter. We also consider that the energy function E is proportional to the distance s and the parameter κ.

According to the above analysis, the energy cost along a path C is defined as follows

$$E(C) = \lambda_1 s(C) + \lambda_2 \kappa(C) + \lambda_3 \int_C M \, \mathrm{da} - v(C) \tag{2}$$

where da represents the line element; $\lambda_1 > 0$, $\lambda_2 > 0$, and $\lambda_3 > 0$ are three trade-off parameters. The energy function relates to two initial conditions, the start position p_s and the target position p_g. The inequality $v \leq V_{\max}$ is also required, where v and $V_{\max}$ are the scalar values of velocity $v(p)$ and maximal velocity, respectively. We then denote them as the speed and the maximal speed, respectively. $\kappa(C)$ represents the sum of curvatures along a path C, and $v(C)$ represents the speed sum along a path C.

The objective of this article is to seek an optimal path C^* which minimizes the energy cost function defined in equation (2). We also denote the minimal energy as $E(C^*)$. Supposing a path $C = \{p_1 = p_s, p_2, \ldots, p_n = p_g\}$, then the corresponding discrete energy function is defined as

$$\begin{aligned} E(C) = \; & \lambda_1 \sum_{i=1}^{n-1} \| \, p_{i+1} - p_i \, \| + \lambda_2 \sum_{i=1}^{n} \kappa(\, p_i) \\ & + \lambda_3 \sum_{i=1}^{n} (M - v(\, p_i)) \end{aligned} \tag{3}$$

where $\kappa(\, p)$ and $v(\, p)$ are the curvature and speed at point p, respectively.

So far, we have three unknowns $s(C)$, $\kappa(\, p)$, and $v(\, p)$. In the following sections, we give methods to determine the last two quantities. The distance $s(C)$ can be determined by the algorithm automatically.

In the following two subsections, we mainly give details how to compute the speed field $v(\, p)$ and the computation of curvatures $\kappa(\, p)$ at every point p. Here, we just concern the scalar values of $v(\, p)$ and $\kappa(\, p)$.

Dynamic speed field

We assume the map for a robot traversing is a white and black image. The black patches represent obstacles that robots cannot reach, and white parts are available areas that robots can reach.

As we have pointed out, the driving speed of humans is related to driver's psychology. The speed is high if there are no obstacles around, otherwise it is slow. Thus, the speed is related to the distance between vehicles and obstacles. Therefore, we use the Laplacian equation defined in equation (4) to estimate an impact coefficient map u for an input map. Then u can be viewed as a smooth control function which can simulate the speed field with the maximal speed $V_{\max}$

$$\Delta u(x,y) = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0 \tag{4}$$

where $u(x,y)$ has the following approximation scheme

$$\begin{aligned} u^k(x,y) = \; & \frac{1}{4}(u^k(x+1,y) + u^k(x-1,y) \\ & + u^k(x,y+1) + u^k(x,y-1)) \end{aligned} \tag{5}$$

where k is the iterative step. The initial conditions are boundaries of obstacles with value 0. The coefficient map is calculated by equation (5) using iterative optimization. In our experiments, the number of iterations is set to 40.

Assuming the maximal speed of a robot as $V_{\max}$, the speed field of an input map for a robot is defined as follows

$$\tilde{v}(x,y) = \omega V_{\max} \cdot u(x,y) \tag{6}$$

where $0 < \omega \leq 1$ is a trade-off coefficient which can control the speed of a robot further.

In practical situations, it is required that the speed of robots is not too slow even it is turning. Thus, the speed field is finally defined by

$$v'(x,y) = \max\Big(\tilde{v}(x,y), \varepsilon V_{\max}\Big) \tag{7}$$

where $0 < \varepsilon < 1$ is a constant that control the minimal speed of robots. Essentially, the truncated definition ensures the speed of a robot not too small. Due to the smoothness of Laplacian interpolation, there is not enough discrimination for neighbors of a point in speed map v'. Therefore, a logarithm transformation is performed, and the final speed map is $v(x,y) = \log\Big(v(x,y)\Big)$.

Dynamic updating. We have presented a static speed field in scenes as earlier, but in practice, dynamic scenes are ubiquitous (e.g. diverse activities of humans) and thus a dynamic updating scheme is required for modern robots with sensors.

For a given map with obstacles O, we first compute the Voronoi diagram V_d of O. If robots detect a new obstacle O_s, we then update V_d for O_s locally. Denote C_s the Voronoi cell of O_s, then the speed field $v(x,y)$ is also updated locally in C_s by the Laplacian equation (5) with zero boundaries of O_s.

Curvature computation

As shown in Figure 3, p and q are two points on a curve (blue solid curve), and q is also a neighbor point of p. l'_1 and l_2 (two green vectors in Figure 2) are two unit tangent lines at p and q, respectively. Translating l'_1 to q gets another vector l_1, and $\delta T = l_1 - l_2$ (red vector) is the difference between l_1 and l_2. From the view of tangent, the curvature $\kappa(p)$ at p has the following definition

$$\kappa(p) = \lim_{q \to p} \frac{\delta T}{\| \, p - q \, \|} \tag{8}$$

where $\| \, p - q \, \|$ is the Euclidean distance of p and q. In this article, we use tangents to approximate curvatures.

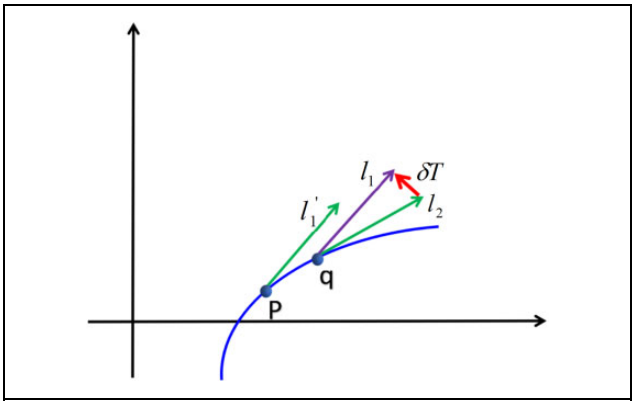

Figure 3. Curvature definition by tangent. l'_1 and l_2 are the unit tangents of p and q, and translating l'_1 to q generates the purple vector l_1. δT is the difference between l_1 and l_2.

Supposing a robot has been moved i steps which generates i points in the motion space, which are denoted by $\{p_1, \ldots, p_{i-1}, p_i\}$. According to the definition in equation (8), the curvatures can be approximated at these discrete points by

$$\kappa(p_i) = \frac{2 \parallel \overrightarrow{\overline{p_{i-1}}\overline{p_i}} - \overrightarrow{\overline{p_i}\overline{p_g}} \parallel}{\parallel \overline{p_{i-1}}p_g \parallel} \tag{9}$$

where $\overrightarrow{\overline{pq}} = q - p$ represents a vector pointing from p to q, and $\parallel pq \parallel$ is the Euclidean distance between p and q. It is worthy to pointing out that $\overline{p_i}$ is the weighted mean value of previous points $p_1, p_2 \ldots, p_i, \ldots$ that is

$$\overline{p_i} = \frac{1}{2}(p_i + \overline{p_{i-1}})$$

A benefit of the definition of $\overline{p_i}$ is that it measures the amount of $\overline{p_i}$ deviated from the previous short time path.

Path planning scheme

Based on the proposed model, we search a path by using the A* method that can navigate a robot from any initial position to the target position avoiding collisions.

Formulation in motion space

In this article, robots are assumed to be disk-shaped objects, and obstacles are convex or concave polygons. Assuming a robot centered at $x \in R^2$ with radius $R > 0$ operates in a closed compact space $S \subset R^2$, and M polygonal obstacles $\{O_i\}_{i=1}^{M}$ also locate in S. We define the free motion space F of a robot as follows

$$F = \{x \in S : \overline{B(x,R)} \subset S \backslash \bigcup_{i=1}^{M} O_i \oplus B(\partial O_i, R) \cup B(\partial S, R)\} \tag{10}$$

where $B(x,r)$ is an open ball $B(x,R) = \{y \in S : \parallel y - x \parallel < R\}$, $\overline{B(x,R)}$ represents the closure, $\parallel \cdot \parallel$ denotes the 2-norm, ∂S is the boundary of S, and $\oplus$ is the dilation operation along every obstacle's boundary. We also assume that the distance of any two polygonal obstacles in F satisfies

$$d_M(p_i, p_j) > 2R, \quad \forall i \neq j \tag{11}$$

where $d_M(p_i, p_j)$ is the minimal distance of two sets, that is $d_M(p_i, p_j) = \min\{\parallel x - y \parallel : x \in p_i, y \in p_j\}$

Overall, a robot in theory can walk freely in its free motion space, however, in practice, a more wider space is needed for robots especially when they traverse some obstacles. For example, in Figure 2, rail-mounted robots in an unmanned dinning room generally preserve a certain distance from tables and chairs for safety considerations.

A*-based path planning

Given a target location $p_g \in F$ and a robot locates at $p_s \in F$, we need to find a path that can navigate a robot from p_s to p_g without collisions. The task can be accomplished by getting a position vector $P_n = (p_1, p_2, \ldots, p_n)$ such that $p_1 = p_s$ and $p_n = p_g$.

The A* method selects a path that minimizes $f(p) = g(p) + h(p)$, where p is the last position of a path, and $g(p)$ is the cost of the path from the start position p_s to p, and $h(p)$ is an admissible heuristic function that measures the lower bound of the cost from p to the target position p_g. The key of A* search algorithm is to design an admissible heuristic function $h(p)$, where the use of $h(p)$ is to avoid overestimating the actual cost to arrive at the target position. In this article, we define the heuristic function $h(p)$ as follows

$$h(p) = \lambda_1 \parallel p - p_g \parallel + \lambda_2 \kappa(p) + \lambda_3 \Big(M - v(p)\Big) \tag{12}$$

where $\parallel \cdot \parallel$ is the 2-norm. The heuristic function $h(p)$ designed here is admissible obviously.

On the other hand, if let $p_k = p$, then the function $g(p)$ is defined as

$$g(p) = \lambda_1 s_s(p) + \lambda_2 \sum_{i=1}^{k} \kappa(p_i) + \lambda_3 \sum_{i=1}^{k} \Big(M - v(p_i)\Big) \tag{13}$$

where $s_s(p)$ is the length of a polyline $\{p_1, p_2, \ldots, p_k\}$.

Based on aforementioned formulations, the optimal path can be produced using the standard A^* algorithm.

Path smoothing

In most cases, the raw paths generated by A* algorithm are possibly not smooth enough, even the smoothness has been considered. Therefore, a post smooth process is performed to ease driving for robots. Specifically, the path is smoothed by minimizing an energy function under following constraints: (1) paths are smooth enough, (2) point

positions in the smoothed path after smoothing are not too far away from original corresponding ones, and (3) smoothed path should also have a strong safe distance to the obstacles.

Assuming the path $C = \{p_1 = p_s, p_2, \ldots, p_n = p_g\}$ is a polyline, and $C_s = \{q_1 = p_s, q_2, \ldots, q_{n-1}, q_n = p_g\}$ is its smoothed version. Then, the optimal energy function for path smoothing is defined as follows

$$E(C_s) = \sum_{i=2}^{n-3} \frac{\| q_i - B_i \|^2}{2} + \beta_1 \sum_{i=2}^{n-1} \frac{\| q_i - p_i \|^2}{2} + \beta_2 \sum_{i=2}^{n-1} F_{\text{rep}}(q_i) \tag{14}$$

where $\beta_1, \beta_2 > 0$ both are real numbers, q_i^b is an attractive point, and $F_{\text{rep}}(q_i)$ is the repulsive function defined in APF,[13] that is

$$F_{\text{rep}}(q_i) = \begin{cases} \frac{1}{2}\left(\frac{1}{\rho(q_i, p_{\text{obs}})} - \frac{1}{\rho_0}\right)^2 & \rho(q_i, p_{\text{obs}}) \geq \rho_0 \\ 0 & \rho(q_i, p_{\text{obs}}) < \rho_0 \end{cases}$$

where $\rho(\cdot, \cdot)$ is the distance function, p_{obs} represents obstacles, and $\rho_0 > 0$ is a constant given by users. The second term of equation (14) guarantees that smoothed paths cannot deviate from original ones too far. The third term guarantees the strong safe distance. The first term attracts the point q_i to a Bezier point B_i, which can control the smoothness of the path. Specially, B_i is a Bezier point interpolated by four neighbor points $(q_{i-2}, q_{i-2}, q_{i+1}, q_{i+2})$ of q_i

$$B_i = \frac{1}{8} q_{i-2} + \frac{3}{8} q_{i-1} + \frac{3}{8} q_{i+1} + \frac{1}{8} q_{i+2} \tag{15}$$

This formula indicates that B_i is from a cubic Bezier curve. Since Bezier curves are generally smooth, we can desire paths processed by this method are also smooth. In fact, Bezier-based smoothing path generation methods are extensively used in path planning.[37,38]

Using the gradient descent algorithm, the energy functional shown in equation (14) can be optimized by the following iterative scheme

$$q_i^{k+1} = B_i^k + \lambda_1 p_i + \alpha_i^k (q_i^k - O_i^k)/(1 + \lambda_1) \tag{16}$$

where α_i^k is defined as

$$\alpha_i^k = \begin{cases} \alpha\left(\frac{1}{\rho(q_i^k, O_i^k)} - \frac{1}{\rho_0}\right) \frac{1}{\left(\rho(q_i^k, O_i^k)\right)^3} & \rho(q_i^k, O_i^k) \leq \rho_0 \\ 0 & \rho(q_i^k, O_i^k) > \rho_0 \end{cases}$$

where $O_i^k \in p_{\text{obs}}$ represents the closest point to p_i^k.

Path evaluation

Path Evaluation is very important in path planning study. In the previous work, researchers usually used the following three criteria to measure a path's quality: the running time T of algorithm, the number of turning points TPN, and the length of a path PL. However, all three aspects do not related to safety. Thus we design following two criteria to measure safety:

1. the minimal distance between a path P and obstacles O: $\text{MD} = \min\{\| x - o \| : x \in P, o \in O\}$.
2. the safety coefficient of a path P and obstacles O: $\text{SC} = \frac{\sum_{x \in P} d(x,O)}{\text{PL}}$, where $d(x, O) = \min\{ \| x - o \| : o \in O\}$, and O represents obstacles.

MD reflects the worst case of a path passing through obstacle regions. SC is used to measure the safety performance which is not considered by previous works to the best of our knowledge. In practice, humans generally desire the path length PL as short as possible and distances to obstacles as long as possible. Therefore, a bigger SC reports a better safety performance.

Implementations and results

In this section, we give implementation details of our algorithm and show the results from our approach. All examples shown in this article are performed on a laptop running on a 3.40 GHz laptop computer with 4 GB of memory. Generally, the path planning algorithm, including A* search and path smoothing, can be accomplished in 0.3 s for an indoor layout map (1026 × 800) and the Paris rendering map (907 × 728) shown in Figures 5 and 6, respectively.

In our implementation, the A^* method searches eight directions (8-neighbors) of current position each time, and therefore the step length of every action is 1 or $\sqrt{2}$. We normalize curvatures and speeds of all positions into an interval [0.0, 1.0] to eliminate the influence of different dimensions. Furthermore, we set $\omega = 0.8$ and $\varepsilon = 0.1$. If without statement, in path smoothing module, we set $\alpha = 3$ and $\rho_0 = 10$. In path planning module, we set $\lambda_1 = 0.4, \lambda_2 = 0.4$, and $\lambda_3 = 0.6$ if without special statement.

The proposed optimal path planning algorithm is built on the platform of Visual C++ with Qt 5.3.2 GUI. In the system as shown in Figure 4, we implement the shortest path method and our method, respectively, which are both optimized by the A* search algorithm. On the left working area of the system interface, maps and paths are shown, and the start and target positions S and G can be chosen interactively. On the right panel of the system interface, several widgets are designed to adjust parameters. Once a start

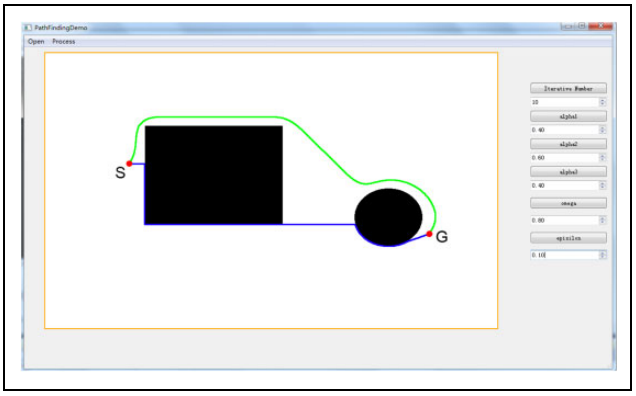

Figure 4. The interface of our optimal path planning system. The green and blue curves represent our optimal path and the shortest path, respectively.

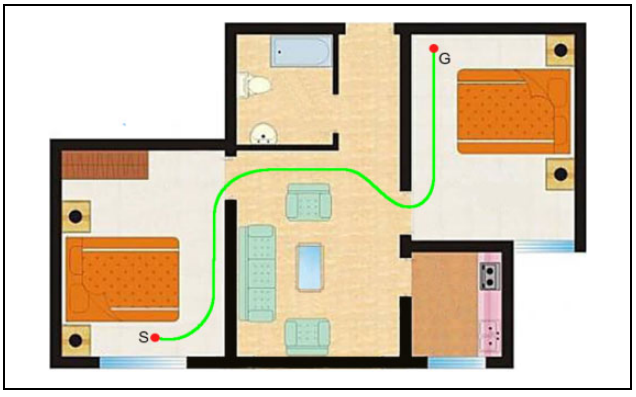

Figure 5. Illustration of path planning by our method at an indoor layout scene.

position and a target position are chosen, we can choose an algorithm in the process menu to generate a path. For example, in Figure 4, we use our method to generate a path (green curve) that connects S and G.

Experimental results

For given start and target nodes $p_s = 248, 110$ and $p_g = 669, 653$, the example for an indoor scene is shown in Figure 5. The result indicates that our path planning method is feasible for indoor environments. The path (green curve in Figure 5) generated by our method is smooth and extends along boundaries of obstacles, which is very similar to human's walking habit. Furthermore, the path indeed has a certain distance with boundaries of obstacles, which guarantees strong safety for robot steering. Therefore, it is suitable for robot steering along the path.

We have also tested the path planning on large-scale scenes. We download a three-dimensional (3-D) Paris model from Internet, and then the model is projected onto the $x - z$ plane orthogonally. The projection is shown in the left of Figure 6. Then the projection is binarized with color clustering operations and manual refinement to extract the transportation road networks of Paris. As shown in Figure 6, red dots indicate the start position $(p_s = 154, 579)$ and the target position $(p_g = 853, 159)$, and the green and blue curves represent our optimal path and the shortest path, respectively. Note that in this example, the path smoothing is not performed since the width of the road networks is very thin. Note that our path (green curve in

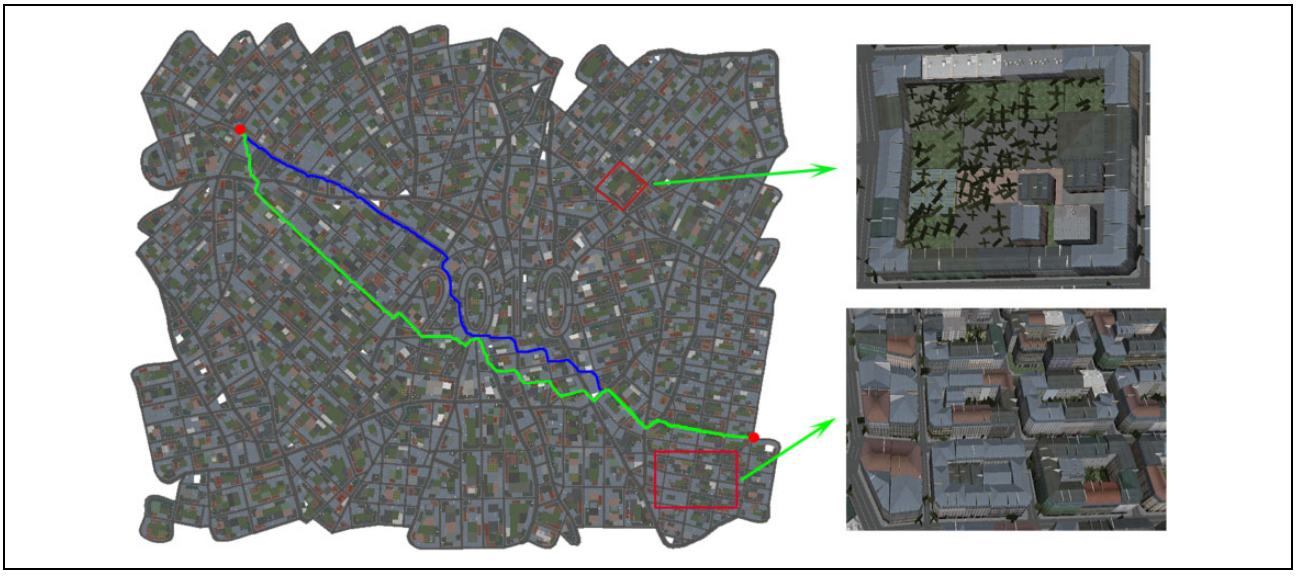

Figure 6. Path planning on a large-scale scene. The left is the render map of Paris 3-D model with several path planning results, where the green and blue curves represent our optimal path planning result and the shortest path planning result, respectively. The right-top and the right-bottom are two local zoom-in images.

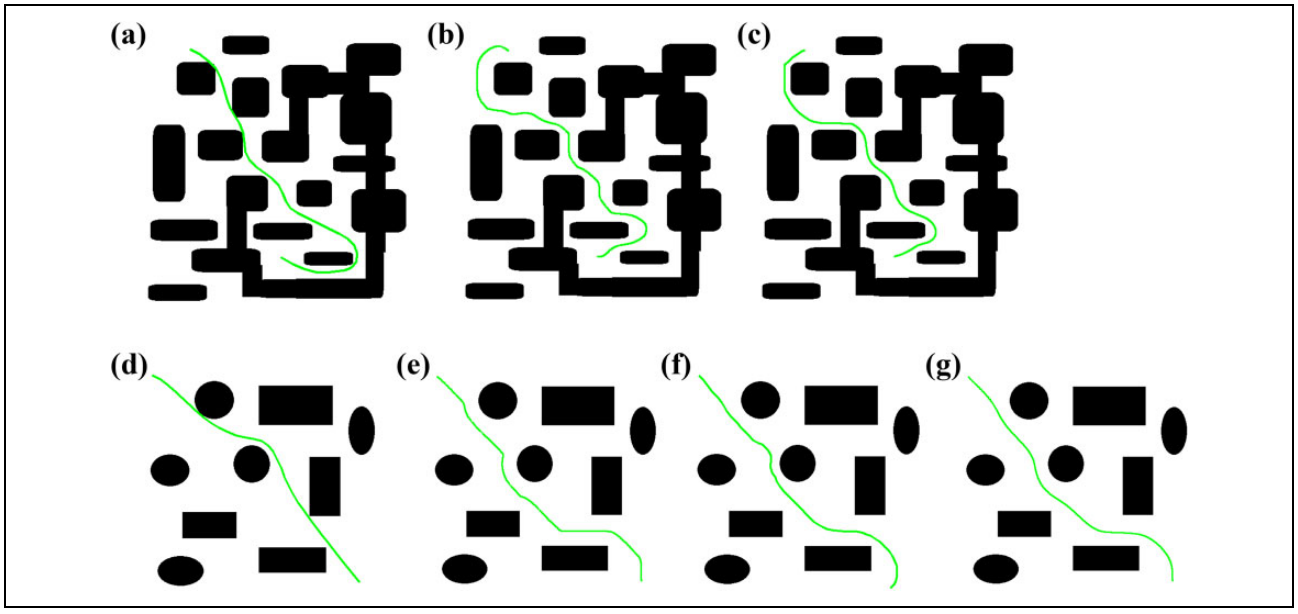

Figure 7. Path planned by different methods. (a) RRT[16]; (b) practical method[10]; (c) our path planning result; (d) RRT[16]; (e) practical method[10]; (f) fuzzy logic[39]; and (g) our path planning result. RRT: rapidly exploring random tree.

Figure 6) contains a long and straight segment comparing with the shortest path (red curve in Figure 6).

Path evaluation

We evaluate different paths generated by different path planning algorithms, including RRT method,[16] practical method,[10] fuzzy logic (FL) method,[39] and our method. RRT[16] and the practical search (PS) method[10] are implemented using Matlab [version 2010], and the FL method[39] is implemented by C++. Note that in Figure 7, we set $\lambda_3 = 2.0$. Paths generated by other methods are smoothed except for RRT, thus a simple mean value filter is used for RRT. Path planning results of these algorithms in maps with size 500×500 are shown in Figure 7.

To evaluate path's quality, five different criteria (see "Path evaluation" section) are computed for paths generated by different algorithms, and the statistical results are reported in Tables 1 and 2. Since RRT is a random algorithm, it is tested ten times in the two static maps, and only one of the results are shown in Figure 7. As shown, RRT method[16] has less turning points comparing to other three algorithms but performs worse with respect to other criteria, especially getting lower safe coefficient. FL method[39] shows a mediocre performance for these criteria. The practical method[10] and our method have a similar safe performance, but our method always have a shorter path length than the practical method. In summary, the method proposed in this article can balance different requirements.

We have also analyzed influence of different parameters λ_1, λ_2, and λ_3 for optimal path planning, as shown in Figure 8. In this example, we generally set $\lambda_1 = 0.01$, $\lambda_2 = 0.01$, and $\lambda_3 = 0.01$ if without special explanation. The paths shown in Figure 8(a) to (d) tend to be a shorter one with increasing λ_1, which agrees with our model designing. With the increasing of λ_2, paths shown in Figure 8 (e) to (h) have obvious variations. In fact, since curvature characterizes local shapes of geometry, and it is also very sensitive, thus different λ_2 lead diverse results. There is a little special for λ_3. When it is bigger than λ_1 and λ_2, paths shown in Figure 8 (j) to (l) do not show obvious variations. Therefore λ_3 is not sensitive if it has a bigger value. Hence choices of λ_1, λ_2, and λ_3 are pivotal for concrete applications. We could set a larger λ_1 for a shorter distance requirement. Note that λ_2 controls the weight of curvatures of paths which are a local geometrical features. Therefore, we advise a large λ_2 if a local smoothness is desired. However, a very large λ_2 is not a good choice for smoothness (e.g. Figure 8(g) and (h)). Generally, we suggest a large value of λ_3 and a similar value of λ_1 and λ_2 for a safe path application.

Dynamic scenes

A dynamic path planning example is tested, and the result is shown in Figure 9. A path is generated and connects S and G in a static scene shown in Figure 9(a). But in Figure 9(b), when robots move to the node M, a new obstacle H is detected, and then the speed field is updated, and the path is also re-computed. The final path is shown in Figure 9(b). The example illustrates that the dynamic updating scheme is useful for dynamic scenes.

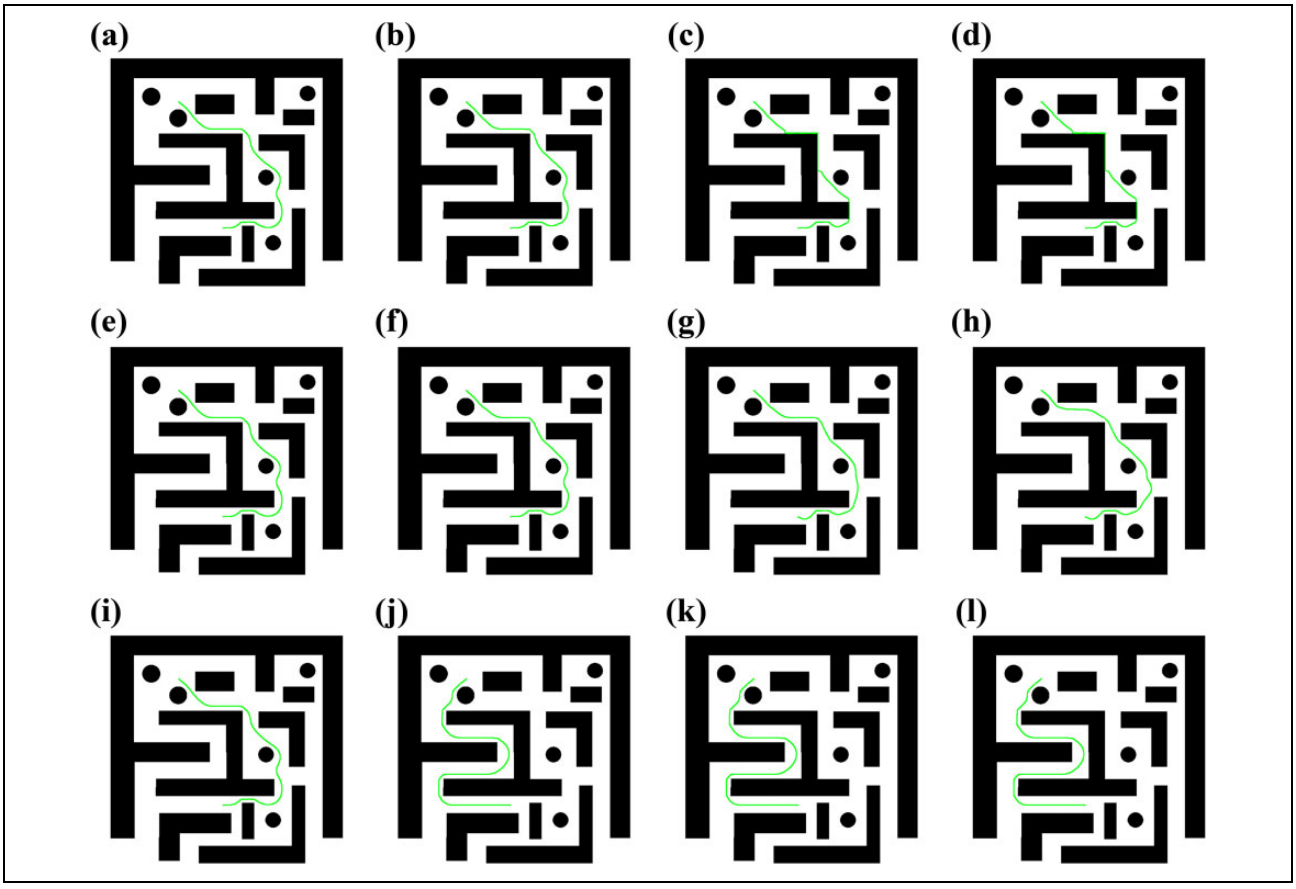

Figure 8. The path planning results by our method with different parameters. (a) $\lambda_1 = 0.01$; (b) $\lambda_1 = 0.05$; (c) $\lambda_1 = 0.5$; (d) $\lambda_1 = 3$; (e) $\lambda_2 = 0.01$; (f) $\lambda_2 = 0.05$; (g) $\lambda_2 = 0.5$; (h) $\lambda_2 = 3$; (i) $\lambda_3 = 0.01$; (j) $\lambda_3 = 0.05$; (k) $\lambda_3 = 0.5$; and (l) $\lambda_3 = 3$.

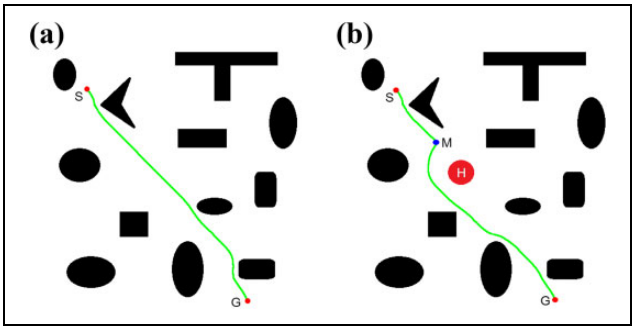

Figure 9. Illumination of path planned in dynamic scene by our method. Compared with (a), a new obstacle *H* appears in (b).

Table 1. Statistical results of paths are shown in Figure 7(a) to (c), where — means the algorithm fails for the map with given start and target nodes.

	T (s)	TPN	PL	MD	SC
RRT	3.088	436	541.75	0.307	9.224
FL	—	—	—	—	—
PS	0.062	573	665.439	10.6538	20.0372
Human experience–inspired path planning method	0.107	564	590.783	9.07	14.6226

RRT: rapidly exploring random tree; FL: fuzzy logic; PS: practical search; TPN: number of turning points; PL: length of a path; MD: minimal distance; SC: safety coefficient.

Examples shown in this section demonstrate that our optimal path planning algorithm for robots is practically feasible. Specially, these results also validate that our model can simulate human's experiences.

Conclusions and future works

In this article, we propose a model to simulate human's path planning. Specially, the safety of robot navigation is considered. The major technical contributions of the article include the following aspects:

1. A dynamic speed field is proposed, which is incorporated with multiple factors, such as safety, path curvatures and path distance, to model human driving experiences. By adapting parameters $(\lambda_1, \lambda_2, \lambda_3)$, the model can generate diverse paths according to specific requirements.

Table 2. Statistical results of paths are shown in Figure 7(d) to (g).

	T (s)	TPN	PL	MD	SC
RRT	2.27	461	617.1	6.082	25.374
FL	1.14	508	630.46	15.94	29.98
PS	0.01	524	637.193	28.6314	32.95
Human experience–inspired path planning method	0.109	527	621.276	22.6872	33.8662

RRT: rapidly exploring random tree; FL: fuzzy logic; PS: practical search; TPN: number of turning points; PL: length of a path; MD: minimal distance; SC: safety coefficient.

2. An efficient but easy-to-implement path smoothing algorithm is presented. The smoothed path is more suitable for steering of robots.
3. Experiments on indoor scene and large-scale outdoor scene have demonstrated the effectiveness of our path planning method.

In the future, we would like to develop our method in several aspects, such as smartly selecting parameters λ_1, λ_2, and λ_3, and adapting to usual maps (e.g. maps with non-smoothed obstacles). We also would like to develop our method assisted by scene sematic analysis especially identifying the static objects and persons.

Acknowledgements

We would like to give our thanks to the artist Johnny who shares the 2-D indoor layout map (Figure 5), and Konstantin who shares the 3-D Paris model (Figure 6).

Declaration of conflicting interests

The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding

The author(s) disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: This project was partially supported by the Natural Science Foundation of China (no. 61672544, 61725204, 61661130156), Guangdong Natural Science Foundation (no. 2015A030311047), Tip-top Scientific and Technical Innovative Youth Talents of Guangdong special support program (no. 2016TQ03X263), the Fundamental Research Funds for the Central Universities (no. 11617348), Beijing Higher Institution Research Center of Visual Media Intelligent Processing and Security, and Foundation of Key Laboratory of Machine Intelligence and Advanced Computing of the Ministry of Education (no. MSC-201701A), and Shenzhen Innovation Program (no. JCJY2015040114529008).

References

1. LaValle SM. *Planning algorithms*. Cambridge: Cambridge University Press, 2006.
2. Farsi M, Ratcliff K, Johnson JP, et al. Robot control system for window cleaning. In: *Proceedings of American control conference*, Baltimore, Maryland, 29 June–1 July 1994, pp. 994–995.
3. Yap P, Burch N, Holte RC, et al. Any-angle path planning for computer games. In: *Proceedings of the seventh AAAI conference on artificial intelligence and interactive digital entertainment*, Stanford, California, USA, 10–14 October 2011, pp. 201–207. AIIDE'11, AAAI Press.
4. Liu M. Robotic online path planning on point cloud. *IEEE Trans Cybern* 2015; 46(5): 1217–1228.
5. Peter H, Nilsson N, and Raphael B. A formal basis for the heuristic determination of minimum cost paths. *IEEE Trans Syst Sci Cybern* 1968; SSC–4(2): 100–107.
6. Mei Y, Lu YH, and Hu YC. Energy-efficient motion planning for mobile robots. In: *IEEE international conference on robotics and automation (ICRA)*, New Orleans, LA, USA, 26 April–1 May 2004, pp. 4344–4349.
7. Khansari-Zadeh SM and Khatib O. Learning potential functions from human demonstrations with encapsulated dynamic and compliant behaviors. *Auton Robot* 2015; 41(1): 45–69.
8. Goldberg AV and Harrelson C. Computing the shortest path: A* search meets graph theory. In: *SIAM symposium on discrete algorithms (SODA)*, Vancouver, Canada, 23–25 January 2005, pp. 156–165.
9. Galceran E and Carreras M. A survey on coverage path planning for robotics. *Robot Auton Syst* 2013; 61(12): 1258–1276.
10. Dolgov D, Thrun S, Montemerlo M, et al. Practical search techniques in path planning for autonomous driving. In: *Proceedings of the first international symposium on search techniques in artificial intelligence and robotics (STAIR-08)*, Chicago, USA, 13–14 July 2008, AAAI.
11. Rina D and Pearl J. Generalized best-first search strategies and the optimality of A*. *J ACM* 1985; 32(3): 505–536.
12. Nash A. *Any-angle path planning*. PhD Thesis, Department of Computer Science, University of Southern California, 2012.
13. Koren Y and Borenstein J. Potential field methods and their inherent limitations for mobile robot navigation. In: *IEEE international conference on robotics and automation (ICRA)*, Sacramento, USA, 9–11 April 1991, pp. 1398–1404.
14. Kavraki LE, Svestka P, Latombe JC, et al. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Trans Robot Autom* 1996; 12(4): 566–580.
15. Lavalle SM. *Rapidly-exploring random trees: a new tool for path planning*. Technical report, Computer Science Dept, Iowa State University, 1998.
16. Jaillet L, Cortes J, and Simeon T. Transition-based RRT for path planning in continuous cost spaces. In: *International conference on intelligent robots and systems (IROS)*, Nice, France, 22–26 September 2008, pp. 2145–2150.
17. Willms A and Yang SX. An efficient dynamic system for realtime robot-path planning. *IEEE Trans Syst Man Cybern B (Cybernetics)* 2006; 36(4): 755–766.
18. Otte MW. *A survey of machine learning approaches to robotic path-planning*. Cs.Colorado.Edu, 2015.
19. Franz MO and Mallot HA. Biomimetic robot navigation. *Robot Auton Syst* 2000; 30: 133–153.

20. Milford M and Schulz R. Principles of goal-directed spatial robot navigation in biomimetic models. *Philos Trans Royal Soc B* 2014; 369(1655). DOI: 10.1098/rstb.2013.0484.
21. Mathews Z, Lechon M, Calvo JMB, et al. Insect-like mapless navigation based on head direction cells and contextual learning using chemo-visual sensors. In: *IEEE/RSJ international conference on intelligent robots and systems*, St. Louis, USA, 10–15 October 2009, pp. 10–15.
22. Savkin A and Wang C. Seeking a path through the crowd: robot navigation in unknown dynamic environments with moving obstacles based on an integrated environment representation. *Robot Auton Syst* 2014; 62(10): 1568–1580.
23. Chen H and Sun D. Moving groups of microparticles into array with a robot-tweezers manipulation system. *IEEE Trans Robot* 2012; 28(5): 1069–1080.
24. Xiao Y, Zhang Z, Beck A, et al. Human-robot interaction by understanding upper body gestures. *Presen Teleop Virt Environ* 2014; 23(2): 133–154. DOI: 10.1162/PRES-a-00176.
25. Zhang Z, Beck A, and Thalmann N. Human-like behavior generation based on head-arms model for robot tracking external targets and body parts. *IEEE Trans Cybern* 2015; 45(8): 1390–1400. DOI: 10.1109/TCYB.2014.2351416.
26. Zhang Z, Li Z, Zhang Y, et al. Neural-dynamic-method-based dual-arm cmg scheme with time-varying constraints applied to humanoid robots. *IEEE Trans Neural Netwk Learn Syst* 2015; 26(12): 3251–3262. DOI: 10.1109/TNNLS.2015.2469147.
27. Zhang Z, Lin Y, Li S, et al. Tri-criteria optimization-coordination-motion of dual redundant robot manipulators for complex path planning. *IEEE Trans Control Syst Technol* 2017; 99: 1–13. DOI: 10.1109/TCST.2017.2709276.
28. Corral E, Meneses J, Castejon C, et al. Forward and inverse dynamics of the biped pasibot. *Int J Adv Robot Syst* 2014; 11(7): 109. DOI: 10.5772/58537.
29. Lozano-Pérez T and Wesley MA. An algorithm for planning collision-free paths among polyhedral obstacles. *Commun ACM* 1979; 22(10): 560–570. DOI: 10.1145/359156.359164.
30. Mark B, Marc K, Mark O, et al. *Computational Geometry*, 2nd ed. Berlin, Heidelberg: Springer-Verlag, 2000.
31. Kuric I, Bulej V, Saga M, et al. Development of simulation software for mobile robot path planning within multilayer map system based on metric and topological maps. *Int J Adv Robot Syst* 2017; 14(6). DOI: 10.1177/1729881417743029.
32. Arslan O and Koditschek DE. Exact robot navigation using power diagrams. In: *IEEE international conference on robotics and automation (ICRA)*, Stockholm, Sweden, 16–21 May 2016, pp. 1–8.
33. Viseras A, Losada RO, and Merino L. Planning with ants: efficient path planning with rapidly exploring random trees and ant colony optimization. *Int J Adv Robot Syst* 2016; 13(5). DOI: 10.1177/1729881416664078.
34. Koenig S and Likhachev M. D*lite. In: *Eighteenth national conference on artificial intelligence*, 28 July–1 August 2002, pp. 476–483. Menlo Park, CA, USA: American Association for Artificial Intelligence, ISBN: 0-262-51129-0.
35. Zucker M, Ratliff N, Dragan A, et al. Chomp: Covariant Hamiltonian optimization for motion planning. *Int J Robot Res* 2013; 33(9–10): 1164–1193.
36. John S, Yan D, Jonathan H, et al. Motion planning with sequential convex optimization and convex collision checking. *Int J Robot Res* 2014; 33(9): 1251–1270.
37. Choi JW, Renwick C, and Gabriel E. Path planning based on Bézier curve for autonomous ground vehicles. In: *Proceedings of the advances in electrical and electronics engineering—IAENG special edition of the world congress on engineering and computer science*, San Francisco, California, 22–24 October 2008, pp. 158–166.
38. Jolly KG, Sreerama KR, and Vijayakumar R. A bezier curve based path planning in a multi-agent robot soccer system without violating the acceleration limits. *Robot Auton Syst* 2009; 57(1): 23–33.
39. Hong TS, Nakhaeinia D and Karasfi B. Application of fuzzy logic in mobile robot navigation. In: Dadios E (ed.) *Fuzzy Logic-Controls, Concepts, Theories and Applications*. 2012, ISBN: 978-953-51-0396-7.