A Decentralized Method Using Artificial Moments for Multi-Robot Path-Planning

Wang-bao Xu\textsuperscript{1,2,*}, Xue-bo Chen\textsuperscript{2}, Jie Zhao\textsuperscript{3} and Tian-yun Huang\textsuperscript{1,2}

\textsuperscript{1} School of Control Science and Engineering, Dalian University of Technology, Dalian, P. R. China
\textsuperscript{2} School of Electronics and Information Engineering, Liaoning University of Science and Technology, Anshan, P. R. China
\textsuperscript{3} State Key Laboratory of Robotics and System, Harbin Institute of Technology, Harbin, P. R. China

* Corresponding author E-mail: xuwangbao@ sina.com

Received 21 Mar 2012; Accepted 15 Nov 2012

DO: 10.5772/55091

© 2013 Xu et al.; licensee InTech. This is an open access article distributed under the terms of the Creative Commons Attribution License (http://creativecommons.org/licenses/by/3.0), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Abstract For the local path-planning of multi-robots, a decentralized method is presented where each robot plans its own path in the following steps for each iteration. Firstly, an \textit{optimal way representative point} (OWRpoint) is obtained for guiding the robot to move along a shorter path. Then, the robot moves a step under the control of its own motion controller, which is designed based on artificial moments. In the motion controller, attractive and repulsive moments are used to move robots closer to their OWRpoints and away from obstacles, while coordinated moments are used to resolve the conflicts between robots. Two simulations are given to test the method and the results indicate that the method is valuable as it meets the requirements of the real-time property while optimizing the performance measure of each robot: namely, the path travelled to reach the robot’s target.

Keywords Multiple Mobile Robots, Path-Planning, Motion Control, Collision, Artificial Moments

1. Introduction

As remarked upon in [1], there has been a growing interest in multi-robot systems in recent years and motion planning is of primary importance in the design of multi-robot systems. This paper focuses on a version of the problem that concerns the path-planning of multiple autonomous mobile robots. The environment is populated with various obstacles. No robot has \textit{a priori} knowledge about the environment or other robots. The objective is to simultaneously bring each robot from an initial position to an independent target. In addition to ensuring collision avoidance, each robot has a performance measure - the path travelled to reach its target - to be optimized.

As no robot has \textit{a priori} knowledge, the problem is a form of local or online path-planning for multi-robots [2]. As individual performance measures are not combined into a single scalar criterion, the problem differs from that discussed in [3], [4], where the criterion is to minimize the time taken by the last robot to reach its target. As remarked upon in [5], when individual performance measures are combined, certain information about potential solutions and alternatives are lost. For example, the degree of sacrifice that each robot makes in order to avoid other robots is not usually taken into account.
Many approaches have been developed for multi-robot path-planning, which are often categorized as centralized or decentralized [5], [6], [7]. With centralized methods, a central planner designs the motion plan for all of the robots based on full knowledge about the environment. Decentralized approaches require that each robot plans its own path based only on the locally available information - e.g., the positions of neighbouring robots - so that it requires less computational resources and also ensures the scalability of the system.

Obviously, only decentralized approaches can be used for the local path-planning of multi-robots as no robot has full knowledge about the environment or other robots.

In [2], [8], [9], a decentralized method based on evolutionary algorithms (EAs) or differential evolution algorithms is discussed, where n EAs are used for n robots and the i-th EA determines the next position for the i-th robot during each iteration, satisfying the necessary constraints associated with that robot and certain cooperation objectives related with the others. In [10], [11], prioritized planning is discussed, which works as follows. First, priorities are assigned to the robots. Then, in order of decreasing priority, the robots are picked. For each picked robot, a path is planned, avoiding collisions with obstacles as well as with the previously picked robots. The drawback of the two methods is the requirement of a considerable amount of communication.

In recent years, the artificial potential field (APF) method is extended to multi-robot path-planning [12], [13]. Here, a repulsive potential field to force a robot away from obstacles or other robots and an attractive field to drive the robot close to a target are employed to generate a force. The force equals the negative gradient of the total potentials and makes the robot move from a position with higher potentials to one with lower potentials. This method requires only a simple calculation and no pre-processing of the environment, but powerful results and elegance of output are generated in a short time [12], [14]. However, this method suffers from many problems [12], [13], [14], [15], such as the “local minimum” problem. When it is used for multi-robot path-planning, coordination between robots is difficult to obtain through the pure APF method as robots cannot be regarded as simply obstacles, and as such some other techniques, such as priorities, have to be used [12], [13].

The configuration space method [12], [16], [17] and the free space method [16], [18], [19] are the methods directly used not only for multi-robot path-planning but also as the basis of other methods, such as the probabilistic roadmap method [20]. In the configuration space, the original problem of planning the motion of a robot through a space of obstacles is transformed into an equivalent - but simpler - problem of planning the motion of a point through a space of enlarged configuration space obstacles or pseudo-obstacles. The free space method is to search the free space directly without first transforming the problem into a configuration space. A representation of a free space can be obtained using generalized Voronoi diagrams [16], a tangent graph [18] or a visibility graph [19]. The drawback of the two techniques is that in local path-planning the configuration space or the free space is frequently required to be recomputed, considering the safe distance and the sizes of the robots. Again, a great deal of computation is needed.

In fact, and to the best of our knowledge, although many algorithms have been developed for the local path-planning of multi-robots, very few can not only meet the requirements of a real-time property but also optimize each robot’s performance measure well. Activated by the limitation noted above, a decentralized method is proposed for multi-robot path-planning, which is an extension of the artificial moment method for multi-robot formation control [21], [22].

In the proposed method, each robot which has not reached its target or which is required to cooperate with other robots uses a path planner, which works as follows. At each sampling time, an optimal way representative point (OWRpoint), which can guide the robot in moving along a shorter path, is obtained firstly, ignoring the presence of other robots. Then, a motion controller based on artificial moments causes the robot to move a step to the next position. The above process is repeated until the task is fulfilled. The proposed method is valuable as it not only meets the requirements of a real-time property, but it also optimizes the independent performance measure of each robot well.

The rest of this paper is organized as follows. In section 2, the problem under discussion is formulated and some concepts are defined. In section 3, the techniques and algorithms for computing OWRpoints are presented. In section 4, a motion controller based on artificial moments is designed. In section 5, the algorithm for the path-planning is given. Several simulations are given and analysed in section 6 and certain conclusions are drawn in section 7.

2. Problem formulation and basic definitions

2.1 Problem Formulation

This paper focuses on the path-planning of multi-robots in a bounded planar environment which is populated with static polygonal obstacles.

A robot model is a square, as shown in Fig. 1, which has a principal motion direction line (PMDline) such as that in [21], [22] and two coordinated segments. The PMDline of a
robot is a ray starting from the robot’s position (the
centre) and can indicate the robot’s posture or motion
direction.

A target model is a static point with a PMDline, as shown
in Fig. 1. The PMDline of a target indicates the target’s
posture.

![Figure 1. Associated model](image)

We rotate a directed line (ray, directed segment) around
its start-point until its direction is the same as that of the
X-axis of the global coordinate system. As such, the angle
formed by the rotation is the direction angle of the line if
the angle’s absolute value is not larger than $\pi$. Assume
that $\beta$ and $\beta$ are the direction angles of directed lines $l_i$ and $l_j$ and that $n$ is an integer and function:

$$agl(x)=\begin{cases} x - 2(n-1)\pi \text{sign}(x) & 2(n-1)\pi \leq x \leq 2n\pi - \pi \\ x - 2n\pi \text{sign}(x) & 2n\pi - \pi \leq x < 2n\pi \end{cases}$$

(1)

Then, the angle $\beta_{agl(\beta-\beta)}$ is the angle from $l_i$ to $l_j$.

For convenience, certain notations which will be used
throughout this paper are listed as follows.

- $D_s$: the side length and longest step length of a robot;
- $D_v$: the valid radius of a robot’s sensor;
- $D_i$: a constant which is larger than $2D_v+D_s$ and is
  associated with the robots’ safety;
- $D_j$: a positive constant less than $D_i$;
- $\delta$: a constant which is less than $\pi/2$ but not $\pi/4$;
- $\delta$: a positive constant far less than $\delta$;
- $\lambda_i$, $\lambda_j$: two constants satisfying $0<\lambda_i<\lambda_j<1$;
- $\lambda$: a constant whose value is $\pi/D_i$;
- $R_i$, $T_i$: the $i$-th robot and the target of the $i$-th robot;
- $\beta_i$, $\beta_j$: the PMDline direction angles of $R_i$ and $T_i$;
- $P_i$: the position (the centre) of $R_i$;
- $(x_i, y_i)^T$: the position coordinates of $T_i$;
- $A_iA_j$: the line segment with end points $A_i$ and $A_j$;
- $DS(A_i, A_j)$: the directed line segment from $A_i$ to $A_j$;
- $\beta(A_i, A_j)$: the direction angle of $DS(A_i, A_j)$;
- $RL(A_i, A_j)$: a ray starting from $A_i$ and passing through $A_j$;
- $|W|$: the length of a continuous curve $W$;
- $\delta(W, -A_i, ..., -A_j)$: points on $W$ except for points $A_i$, ..., $A_j$;
- $\delta(RL(A_i, A_j), -A_iA_j)$: points on $RL(A_i, A_j)$ except for those
  on $A_iA_j$;
- $A_iA_j...A_n$: a polyline with vertices $A_i, A_j, ..., A_n$.

For an arbitrary variable $V$, $V(k)$ denotes its value at time
$t_k$ and $V$ denotes it at the present.

A robot’s coordinated segments are two directed segments
whose lengths are $D_s+D_v$, the start-points are the robot’s
position while angles to the PMDline are $\pm\pi$, as shown
in the model of $R_i$ in Figure 1. The coordinated segments
are important in resolving the conflicts between robots.

The robot sensors are omni-directional. For a point $F,$
only when $P(F) \in D_s$ does not pass through any obstacle
(including robots) and $|P(F)| \leq D_v$, can $F$ be detected by $R_i$.
For two robots, only when the segment between their
positions does not pass through any obstacle and is
shorter than $D_v$ can they be detected and their positions
and PMDlines be known by each other.

For example, in Figure 1 only $R_i$ and its PMDline, as well
as the points within the region with a grey colour or on
the boundaries of the region, can be detected by $R_i$ at
present. The point $A_i$ can be detected by $R_i$ at $R_i$’s
initial position but cannot be detected at $R_i$’s current position.

As it is important to reduce the communication required
between robots to the absolute minimum (otherwise real-
time dynamic planning will not be possible) there is no
explicit communication between robots in this paper.
That is to say, a robot has no knowledge about other
robots until it detects them and, then, it knows - and only
knows - their positions and PMDlines at the present. A
robot also has no a priori knowledge about its workspace.

**Assumption 1.** At each sampling time, each robot knows
the position and the PMDline’s direction angle of its
target through communication or through other means.

**Assumption 2.** For each robot, there is at least one
feasible way between its target and its initial position
which is not narrower than $2D_s$ (ignoring other robots).
Once the models, assumptions, sensing capabilities and
communication protocols highlighted above have been
detailed, the problem reduces to finding out a motion
controlled robot algorithm that will enable each robot to avoid
potential collisions with the remaining robots and any
other obstacles present, reaching its target in a finite time.
When a robot $R_i$ reaches $T_i$, $R_i$ is required to achieve the
same posture as that of $T_i$ as much as possible while its travelling path is also required to be as short as possible.

**Remark 1.** When $R_i$ reaches $T_i$, the requirement for $R_i$ to achieve the same posture as that of $T_i$ is the requirement for $\beta_{\infty}$ to be as small as possible, which is beneficial for certain applications, such as formation control [21], [22].

### 2.2 Basic Definitions

**Definition 1.** Assume $\Sigma$ is a polyline on the boundary of an obstacle. If all of the points on $\Sigma$ have been detected by $R_i$ at the current time or at previous times, and $\Sigma^+$ has at least one point which has never been detected by $R_i$ when $\Sigma^+\to\Sigma$, then $\Sigma$ is a knowledge obstacle wall of $R_i$.

For example, in Figure 1, $A_1A_2A_3A_4$ is a knowledge obstacle wall of $R_i$ and $A_1, A_2, A_3, A_4$ are its vertices; $A_1A_2A_3A_4$ is not so because some points on it, such as $A_5$, have never been detected by $R_i$.

Each robot will memorize and update all of its knowledge obstacle walls.

**Definition 2.** Assume that two points $F_1$ and $F_2$ are the ends of a continuous curve $W$ and that $S$ is a set such that its members are all continuous curves with ends $F_1$ and $F_2$, and that all points on its members are on or very close to $W$. $\Sigma$ is a knowledge obstacle wall of $R_i$. 1) If for any curve $W^+$ in $S$, $\partial(W^+, -F_1, -F_2)$ shares at least one point with $\Sigma$, then $W$ is blocked by $\Sigma$. 2) If $\Sigma$ blocks $P_iT_i$ then $\Sigma$ is a block wall of $R_i$. 3) If a continuous curve from $R_i$ to $T_i$ cannot be blocked by $R_i$'s block walls, it is a free block-wall way of $R_i$.

For example, in Figure 1, $A_1A_2A_3A_4$ is a block wall of $R_i$; $P_iA_1T_i$ and $P_iA_2A_3A_4T_i$ are free block-wall ways of $R_i$, but $P_iA_2A_3T_i$ is not because it is blocked by $A_1A_2A_3A_4$.

The shortest free block-wall way is important as a shorter, feasible way can be easily obtained through it. Furthermore, compared with the shortest feasible way obtained by such methods as the tangent graph method and the configuration space method, the shortest free block-wall way is easier to obtain as there is no requirement to consider the safe distance of robots and no requirement to pre-process the environment.

As obstacles are all polygons, the shortest free block-wall way of $R_i$ must be a polyline. Assume it is $P_iA_1A_2\ldots A_nT_n$; then, the first vertex $A_1$ but not the full way is needed to be obtained for guiding $R_i$ to move. The reasons for this are that, as the shortest free block-wall way of $R_i$ at present is often significantly different from that at the next in local path-planning, it is effective only at the present.

Moreover, at the current time, $A_1$ has the same effect as the full way for optimizing $R_i$'s path but is easier to obtain and requires less memory.

**Definition 3.** The first vertex $A_i$ of the shortest free block-wall way of $R_i$ is the optimal way representative point of $R_i$ (OWRpoint) at present.

For example, when $P_iT_i$ is not blocked by any knowledge obstacle wall of $R_i$, $T_i$ is the OWRpoint.

**Definition 4.** For two robots $R_i$ and $R_j$, if $R_i$ can be detected by $R_j$ and $|P_iA_j|<D_2+s2D_3$, then $R_i$ is a coordinated companion of $R_j$.

For example, in Figure 1, $R_i$ and $R_j$ are coordinated companions of each other.

**Definition 5.** For a point $G$ on a knowledge obstacle wall $\Sigma$ of $R_i$, if $G$ can be detected by $R_j$ at present, $|P_iG|<D_2$, and a constant $\delta_0$ exists such that, for any point $A$ on $\Sigma$, $|AG|<\delta_0$ implies $|P_iA|<|P_iG|$, then $G$ is a dangerous wall representative point of $R_i$.

Obviously, a dangerous wall representative point of $R_i$ is the point that is the shortest distance from $P_i$ in a local region of a dangerous knowledge obstacle wall. Accordingly, for $R_i$, several such points may exist on the same knowledge obstacle wall, which can prevent $R_i$ from colliding with one side while avoiding another side of the wall.

**Definition 6.** Assume that $F$ is a point on a free block-wall way of $R_i$ and that $A_i$ is a point on a knowledge obstacle wall $\Sigma$ of $R_i$. 1) If $\delta(RL(F, A_1), -FA_1)\Sigma$ is null and $RL(F, A_1)$ is not blocked by $\Sigma$, then $\delta(RL(F, A_i))$ is a single-direction tangent of $R_i$ from $F$ to $\Sigma$ and $A_i$ is the tangent point. 2) When $\delta(RL(F, A_i), -FA)\Sigma$ is not null, assume that $A_i$ is the point that is the shortest distant from $A_i$ in the set. If $FA_i$ is not blocked by $\Sigma$ and if a closed curve by which $T_i$ and $F$ are separated can be formed by $A_1A_2$ and a part of $\Sigma$, then $RL(F, A_i)$ is a close tangent of $R_i$ from $F$ to $\Sigma$ and $A_i$ and $A_2$ are the tangent and close point respectively.

For example, in Figure 2, $RL(P_i, A_i)$ is a single-direction tangent from $P_i$ to $\Sigma$ and $A_i$ is the tangent point. $RL(P_i, A_i)$ is a close tangent from $P_i$ to $\Sigma$ ($\Sigma$ and $\Sigma'$ are merged into one wall by an artificial obstacle segment $\Sigma\Sigma'$) and $A_1$ and $F_1$ are the tangent and the close point respectively. $RL(A_i, A_2), RL(A_2, A_3)$ are close tangents of $R_i$.

### 3. Method for OWRpoints

In this section, a method is presented to determine a series of OWRpoints which guide each robot to move along a shorter path. The first step of this method is to set artificial obstacle segments according to the rules given as
follows until no such segments can be set. Since such segments can block these ways onto which the OWRpoints are not suitably placed, the computation of the OWRpoints is decreased by setting such segments.

3.1 Rules for Setting Artificial Obstacle Segments

There are two types of artificial obstacle segments, namely narrow-types (N-type) and back-types (B-type).

The rule for N-type artificial obstacle segments is as follows. For two points \( A_i \) and \( A_2 \) on knowledge obstacle walls of \( R_i \), if \( |A_iA_2| \) is less than \( 2D_b \) and a free block-wall way of \( R_i \) passes through \( A_iA_2 \) once and once only, then \( A_iA_2 \) can be set as an N-type artificial obstacle segment. For example, in Figure 2, \( V_iV_4 \) is an N-type artificial obstacle segment of \( R_i \).

N-type artificial obstacle segments are used to prevent a robot from walking along ways narrower than \( 2D_b \), as if two robots encounter each other in a way that is narrower than \( 2D_b \), they may be blocked by each other such that their convergence is lost. As remarked upon in [6], the loss of convergence in this case is not a matter of a good or a bad algorithm - it is due to the decentralized control model.

The rules for B-type artificial obstacle segments are as follows. For \( R_i \), assume \( \Sigma_1 \) and \( \Sigma_2 \) are two different knowledge obstacle walls and that \( \Sigma_1 \) blocks \( P_bT_i \). 1) If \( \Sigma \) also blocks \( P_bT_i \), then \( A_iA_2 \) can be set as such a segment, where \( A_1 \) and \( A_2 \) are the intersections of \( P_bT_i \) with \( \Sigma_1 \) and \( \Sigma_2 \) respectively. 2) Assume that \( A_i \) is the tangent point of a close or single-direction tangent from \( P_b \) to \( \Sigma_i \). If \( \Sigma_i \) blocks \( P_bA_i \) and \( A_i \) is an intersection of \( P_bA_i \) with \( \Sigma_2 \), then \( A_iA_2 \) can be set as such a segment.

For example, \( V_3V_4 \) in Figure 2 is a B-type artificial obstacle segment of \( R_i \). Obviously, for \( R_i \), if \( A_iA_2 \) can be set as a B-type artificial obstacle segment at present, then no point (except for \( A_1 \) and \( A_2 \)) on \( A_iA_2 \) can be detected at present; however, \( A_iA_2 \) may have been detected at some previous time because \( A_1 \) and \( A_2 \) are all on knowledge obstacle walls.

That is to say, \( R_i \) has moved from a position with a shorter distance from \( A_iA_2 \) to the current one with a longer distance. Thus, walking towards \( A_iA_2 \) will result in walking backwards, which is the reason for calling such segments B-type artificial obstacle segments.

Setting B-type artificial obstacle segments has two main benefits. One is to guarantee the global convergence to a certain extent by preventing a robot from walking backwards. The second is to decrease the total number of knowledge obstacle walls of a robot by merging two different walls into one wall.

**Conclusion 1.** If no artificial obstacle segment can be set by \( R_i \) at present, then: 1) \( P_bT_i \) is blocked by at most one knowledge obstacle wall; 2) assume that \( \Sigma \) is a block wall of \( R_i \) and that \( A_i \) is the tangent point of a close or single-direction tangent of \( R_i \) from \( P_b \) to \( \Sigma \). Then, \( P_bA_i \) will not be blocked by any knowledge obstacle wall.

**Artificial obstacle segments** are dealt with as a part of knowledge obstacle walls in general. However, if a closed curve by which \( R_i \) and \( T_i \) are separated is formed by a knowledge obstacle wall of \( R_i \), then all B-type artificial obstacle segments on the wall will be removed.

3.2 Algorithm to Determine OWRpoints

For \( R_i \), after setting artificial obstacle segments, an OWRpoint will be obtained, which is based on the following conclusions.

**Conclusion 2.** If \( \Sigma \) is a block wall of \( R_i \), no close tangent from \( P_b \) to \( \Sigma \) exists, and if \( A_i \) is the tangent point of a single-direction tangent from \( P_b \) to \( \Sigma \) then a polyline \( P_bA_1A_2...A_nT_i \) represented with \( FBW_{A_1} \) can be obtained by algorithm 1.

**Algorithm 1** Obtaining the vertices on \( FBW_{A_1} \).

1: Let \( A_0 = P_b \), \( A_1 = A_i \) and, then, go to step 2.
2: If \( A_{i+1} \) is not blocked by \( \Sigma \), then stop. Otherwise, go to step 3.
3: If a close tangent from \( A_i \) to \( \Sigma \) exists, then the tangent point is \( A_{i+1} \). Let \( A_{i+1} = A_{i+1} \) and then return to step 2. Otherwise, go to step 4.
4: If \( A_{i+1} = A_i \), then \( A_{i+2} \) is the tangent point of the single-direction tangent from \( A_i \) to \( \Sigma \) such that \( A_iA_{i+1} \) is blocked by \( \Sigma \), where \( A_{i+1} \) is a point on \( A_iA_2 \) and close to \( A_i \). Otherwise, \( A_{i+1} \) is the tangent point of the single-
direction tangent from $A_i$ to $\Sigma$ that shares no point with $RL(A_i-2, A_i)$. Let $A_i = A_{i+1}$; then return to step 2.

**Conclusion 3.** Assume that no artificial obstacle segment can be set by $R_i$ at present and that $\Sigma$ is a block wall of $R_i$. For $R_i$, if a close tangent from $P_{k+1}$ to $\Sigma$ exists, then the tangent point is the OWRpoint; otherwise, $A_i$ or $Q_i$ is the OWRpoint as $FBW_{A1}$ or $FBW_{O1}$ is the shortest free block-wall way, where $A_i$ and $Q_i$ are the tangent points of the two single-direction tangents from $P_{k+1}$ to $\Sigma$.

Based on previous discussion, an algorithm for the OWRpoint is given as follows.

**Algorithm 2 Obtaining the OWRpoint at present.**

1: If $R_i$ has no block wall, then $T_i$ is the OWRpoint; otherwise, go to step 2.
2: Assume that $\Sigma$ is a block wall of $R_i$. If a close tangent from $P_{k+1}$ to $\Sigma$ exists, then its tangent point is the OWRpoint; otherwise, go to step 3.
3: Obtain the tangent points $A_i$ and $Q_i$ of the two single-direction tangents from $P_{k+1}$ to $\Sigma$.
4: Obtain $FBW_{A1}$ and $FBW_{O1}$ respectively.
5: Substitute $P_{k+1}$ in $FBW_{A1}$ and $FBW_{O1}$ with $P_{k+0}$ such that another two ways represented with $FBW_{A1}$ and $FBW_{O1}$ are obtained.
6: If $|FBW_{A1}| < |FBW_{O1}|$, then $A_i$ is the OWRpoint; otherwise $Q_i$ is.

**Remark 2.** In step 6 of algorithm 2, the method to determine OWRpoints is to compare $|FBW_{A1}|$ with $|FBW_{O1}|$ but not $|FBW_{A1}|$ with $|FBW_{O1}|$. The reasons for this are that if the OWRpoint is determined by comparing $|FBW_{A1}|$ with $|FBW_{O1}|$, the OWRpoint at present may be significantly different from that in the last time even if the difference between $|FBW_{A1}|$ and $|FBW_{O1}|$ is very small. As such, a great change in $R_i$'s PMDLine direction is frequently needed in some cases. To avoid the above problem, $|FBW_{A1}|$ and $|FBW_{O1}|$ are used.

For algorithm 2, the necessary computation in the worst case is to compute $|FBW_{A1}|$ and $|FBW_{O1}|$ using algorithm 1. Assume that $m$ is the vertex number of the knowledge obstacle wall of $R_i$ with most vertices; then, the computational complexity of algorithm 2 is $O(2m)$ as the number of vertices on $FBW_{A1}/FBW_{O1}$ cannot be more than $m$.

4. Motion controller based on artificial moments

A motion controller based on artificial moments is designed in this section, which is an extension of the artificial moment method for multi-robot formation control [21], [22]. Its basic idea is that at each sampling time $R_i$ may be influenced by three types of artificial moments - that is, the attractive moment by the OWRpoint, the repulsive moments by obstacles and the coordinated moments by other robots. The gradient of each artificial moment generates an expected vector for $R_i$ to change position and the PMDLine so that the moment can increase as quickly as possible. Additionally, $R_i$ has a motion vector along its PMDLine which is not in general determined by artificial moments. The total vector determines the next position and the PMDLine direction of $R_i$ and, afterwards, the above process is repeated until the task is fulfilled.

Obviously, the artificial moment method is similar to the APF method in some aspects. Therefore, they have shared advantages. However, there are certain important differences between them.

One of these is that in the artificial moment method each robot has a motion vector along its PMDLine which is nearly uninfluenced by artificial moments. Let $(S_y(k+1), S_x(k+1))$ represent the expected vector of $R_i$ along its PMDLine in $(t_i, t_{i+1})$. If the OWRpoint at the current time $t_i$ is $T_i$ and the distance from $P_{k+1}$ to $T_i$ is not larger than $D_s$, then $(S_y(k+1), S_x(k+1))= (0, 0)$; otherwise:

$$
\begin{align}
S_y(k+1) &= S_R \cos(\beta_R(k+1)) \\
S_x(k+1) &= S_R \sin(\beta_R(k+1))
\end{align}
$$

where $\beta_R(k+1)$ is $R_i$'s PMDLine’s direction angle in $(t_i, t_{i+1})$.

The motion vector $(S_y(k+1), S_x(k+1))$ causes $R_i$ to have a high moving speed even if the total moment’s gradient is zero and, as such, it is difficult for $R_i$ to be trapped even in complicated environments.

A second difference is that a unique robot model is used in the artificial moment method, where each robot has a PMDLine and two coordinated segments. According to the robot model, a coordinated moment - which can resolve the conflicts between robots (almost) optimally - is designed as follows.

4.1 Coordinated Moments

Each coordinated companion of $R_i$ will - and only such robots can - impose a coordinated moment on $R_i$ so that the conflict between them can be solved.

Assume that $R_i$ is a coordinated companion of $R_i$ and that $CM(k)$ is the coordinated moment generated by $R_i$ at the current time $t_i$. The function $cmt(x)$ is defined as (3) and its derivative $dcmt(x)$ as (4):

$$
cmt(x)=\begin{cases}
\cos(x) & |x| \leq \pi/2 \\
\pi/2 - |x| + \cos(x-\pi/2)\text{sign}(x) & |x| > \pi/2
\end{cases}
$$

$$
dcmt(x)=\begin{cases}
-\sin(x) & |x| \leq \pi/2 \\
\text{sign}(x) - \sin(x-\pi/2)\text{sign}(x) & |x| > \pi/2
\end{cases}
$$
When \( \text{lag}([\beta_k-R(P_{50}, P_{56})]) < \delta_R \) and \( \text{lag}([\beta_{k-R}(P_{56}, P_{50})]) < \delta_R \) (the case where \( R_i \) and \( R_j \) have a marked trend to move towards each other), \( CM(k) \) is required to cause \( R_i \) to move to a side of \( R_j \) so that \( R_i \) can bypass \( R_j \) and so that no collision occurs between them. Furthermore, according to the PMD lines of \( R_i \) and \( R_j \) at present, it can be concluded that the two robots are moving towards \( P_{Mj} \) and \( P_{Mi} \) respectively. Accordingly, \( CM(k) \) is required to make \( R_i \) move towards a position that is away from \( P_{Mj} \) but close to \( P_{Mi} \) so that their influence upon the motion of each other can be minimized.

Let \( M \) represent the end-point of the directed segment whose start-point is \( P_{56} \); the direction angle is the same as that of \( DS(P_{50}, P_{56}) \) and the length is \( D + D_{6} \) (as shown in Figure 1); \( (x_M, y_M)^T \) represents the coordinates of \( M \). As such, \( M \) is an ideal point towards which \( R_i \) moves as \( M \) satisfies all of the requirements mentioned above. Thus, \( CM(k) \) is designed as (5) in this case:

\[
CM(k) = \lambda x \cdot \text{cmt}(\text{ag}([\beta_k-R(P_{50}, C_5)])) + \\
\lambda y \cdot \text{cmt}(\text{ag}([\beta_k-R(P_{50}, C_5)])) + \\
(cmt(\lambda_x(x_M-x_0)) + cmt(\lambda_y(y_M-y_0))) / (\lambda)^2 
\]

(5)

where \( C_5 \) and \( C_5 \) are the end-points of the left and right coordinated segment of \( R_i \) respectively; \( \lambda = DS/(P_{50}C_5 + D) \) and \( \lambda = DS/(P_{50}C_5 + D) \).

In (5), only when \( P_{50} \) is at \( M \) does \( CM(k) \) have the potential to be the greatest. Thus, \( CM(k) \) will cause \( R_i \) to move towards \( M \). However, when \( PM_{50} \) is parallel to \( P_{50}P_{6j} \), \( R_i \)’s motion towards \( M \) and \( R_i \)’s similar motion will lead the two robots to move in parallel and, then, a deadlock arises. To avoid the deadlock situation, the first term, \( \lambda x \cdot \text{cmt}(\text{ag}([\beta_k-R(P_{50}, C_5)])) \) and the second, \( \lambda y \cdot \text{cmt}(\text{ag}([\beta_k-R(P_{50}, C_5)])) \) are designed. The first term is to make \( R_i \)’s PMD line point to \( C_5 \) as the term will be the greatest when \( R_i \)’s PMD line point to \( C_5 \); the second is to make \( R_i \)’s PMD line point to \( C_5 \) for the same reasons. As \( \lambda < \lambda \), the influences of the two terms are not the same in general. As a result, \( PM_{50} \) cannot be parallel to \( P_{50}P_{6j} \) at next time, even if \( PM_{50} \) is parallel to \( P_{50}P_{6j} \) at present. Accordingly, the deadlock situation is avoided.

The coefficient \( \lambda = DS/(1P_{50}C_5 + D) \) will increase with the decrease of \( 1P_{50}C_5 \) since the smaller that \( 1P_{50}C_5 \) is, the greater the first term is expected to be so that \( R_i \)’s PMD line can point to \( C_5 \) and \( R_i \) can move towards \( C_5 \). For the same reason, the coefficient \( \lambda = DS/(1P_{50}C_5 + D) \) will increase with the decrease of \( 1P_{50}C_5 \).

When \( \text{lag}([\beta_k-R(P_{50}, P_{56})]) \geq \delta_R \) or \( \text{lag}([\beta_{k-R}(P_{56}, P_{50})]) \geq \delta_R \) (the case where \( R_i \) and \( R_j \) have no marked trend to move towards each other), \( CM(k) \) is only required to ensure that \( R_i \) maintains a safe distance from \( R_j \). Thus, \( CM(k) \) is designed as (6) in this case:

\[
CM(k) = cmt(\lambda_x(x_M-x_0))cmt(\lambda_y(y_M-y_0))(\lambda)^2 
\]

(6)

Let \( (\Delta x_{\beta}(k), \Delta y_{\beta}(k)) \) denote the expected vector to increase \( CM(k) \); then, it is the gradient of \( CM(k) \). When \( \text{lag}([\beta_k-R(P_{50}, P_{56})]) \geq \delta_R \) and \( \text{lag}([\beta_{k-R}(P_{56}, P_{50})]) \geq \delta_R \):

\[
\begin{align*}
\Delta x_{\beta}(k) &= \lambda_x \cdot \text{dcm}(\beta_{k-R}(P_{50}, P_{56})) \\
&= \lambda_x \cdot \text{dcm}(\beta_{k-R}(P_{50}, P_{56}))/\lambda x \\
\Delta y_{\beta}(k) &= \lambda_y \cdot \text{dcm}(\beta_{k-R}(P_{50}, P_{56}))/\lambda y \\
&= \lambda_y \cdot \text{dcm}(\beta_{k-R}(P_{50}, P_{56}))/\lambda y \\
\end{align*}
\]

(7)

When \( \text{lag}([\beta_k-R(P_{50}, P_{56})]) \leq \delta_R \) or \( \text{lag}([\beta_{k-R}(P_{50}, P_{56})]) \leq \delta_R \) (the case where \( R_i \) and \( R_j \) have no marked trend to move towards each other), \( CM(k) \) is only required to ensure that \( R_i \) maintains a safe distance from \( R_j \). Thus, \( CM(k) \) is designed as (6) in this case:

\[
CM(k) = cmt(\lambda_x(x_M-x_0))cmt(\lambda_y(y_M-y_0))(\lambda)^2 
\]

(6)

4.2 Attractive Moments

A third difference between the artificial moment method and the APF method is the difference between artificial moment functions and artificial potential functions. Artificial potential functions are always designed in terms of the positions, velocities and accelerations of robots, targets and obstacles (or relative ones between them). Artificial moment functions, however, are always designed in terms of the angles from robots’ PMD lines to OWR points and obstacles and, in most cases, artificial moments influence only robots’ PMD lines but not their positions and velocities. As a result, many problems encountered in the APF - such as there being no passage between closely-spaced obstacles or goals being non-reachable with obstacles nearby - are solved by the proposed motion controller.

Assume the function \( \text{amt}(x) \) is (9); then, its derivative is (10):

\[
\text{amt}(x) = \begin{cases} 
\cos(\delta_R) + (\delta_R^2 - x^2)/2 & |x| \leq \delta_R \\
\pi/2 - |x| + \cos(\pi/2)\text{sign}(x) & |x| > \pi/2 
\end{cases} 
\]

(9)

\[
\text{d amt}(x) = \begin{cases} 
\cos(\delta_R) - x & |x| \leq \delta_R \\
-\sin(x) & |x| > \delta_R 
\end{cases} 
\]

(10)

Let \( AM(k) \) denote the attractive moment imposed by the OWR points at the current time \( t_i \); then, \( AM(k) \) is designed as follows.
If the OWRpoint is not \( T_i \) or else if its distance from \( p_b \) is larger than \( D_b \), then \( AM(k) \) is only required to make \( R \)'s PMDline face the OWRpoint. As with when \( R \)'s PMDline is facing the OWRpoint, the objective for \( R \) in moving closely to the OWRpoint can be fulfilled by the motion vector \((S_x(k+1), S_y(k+1))^T\). Thus, \( AM(k) \) is designed as (11) in this case:

\[
AM(k) = \left( \frac{\lambda_t}{2} \right) (amt(agl(\beta_k - \beta(G, \text{OWRpoint}))))^2
\]  

If the OWRpoint is \( T_i \) and its distance from \( p_b \) is not larger than \( D_b \), \( AM(k) \) is required to influence both \( R \)'s PMDline and position so that \( R \) can reach \( T_i \) exactly while having as similar a posture to \( T_i \) as is possible. Thus, \( AM(k) \) is designed as (12) in this case:

\[
AM(k) = \text{amt}(agl(\beta_k - \beta(G, \text{OWRpoint}))) + \text{amt}(\lambda_k(x_{k+1} - x_i)) \frac{\lambda_k}{2}
\]  

Let \((\Delta \beta(k), \Delta x(k), \Delta y(k))\) denote the expected vector to increase \( AM(k) \); then, it is the gradient of \( AM(k) \). When the OWRpoint is not \( T_i \) or its distance from \( p_b \) is larger than \( D_b \):

\[
(\Delta \beta(k), \Delta x(k), \Delta y(k))^T = (-\lambda_k agl(\beta_k - \beta(G, \text{OWRpoint}))), 0, 0)^T
\]  

When the OWRpoint is \( T_i \) and its distance from \( p_b \) is not larger than \( D_b \):

\[
\begin{bmatrix}
\Delta \beta_{R_i}(k) \\
\Delta x_{R_i}(k) \\
\Delta y_{R_i}(k)
\end{bmatrix} = \begin{bmatrix}
\text{damt}(agl(\beta_k - \beta(G, \text{OWRpoint}))) \\
damt(\lambda_k(x_{k+1} - x_i))/\lambda_k \\
damt(\lambda_k(y_{k+1} - y_i))/\lambda_k
\end{bmatrix}
\]  

4.3 Repulsive Moments and Motion Controller

Suppose that \( G \) is a dangerous wall representative point of \( R \) at the current time \( t \) and that \( PM(k) \) is the repulsive moment generated by \( G \) at \( t \).

When \( R \) has no coordinated companion at present, \( PM(k) \) is only required to cause \( R \)'s PMDline to move away from \( G \). As with when \( R \)'s PMDline is away from \( G \), the objective for \( R \) in moving from \( G \) can be fulfilled by the motion vector \((S_x(k+1), S_y(k+1))^T\). Thus, \( PM(k) \) is designed as (15) in this case:

\[
PM(k) = \lambda \eta agl(\beta_k - \beta(G, p_b)))^2
\]  

where \( \lambda = (D_s - 1)p_b / G \) for the reason that the lower that \( 1p_b G \) is, the larger that the repulsive moment is required to be, so that \( R \)'s PMDline can move away from \( G \) quickly.

When \( R \) has coordinated companions, \( PM(k) \) is required to influence \( R \)'s position but not the PMDline so that the influence of the coordinated moments on \( R \)'s position can be weakened and so that \( R \) will not move too closely to \( G \). Accordingly, \( PM(k) \) is designed as (16) in this case:

\[
PM(k) = \left( \frac{\lambda_t}{2} \right) (amt(\lambda_k(x_{k+1} - x_i) + cmt(\lambda_k(y_{k+1} - y_i))))(\lambda_k)^2
\]  

where \((x_{k+1}, y_{k+1})^T\) is the coordinates of the end-point of the directed segment with a start-point \( G \), a direction angle \( \beta(G, p_b) \) and a length \( D_b \).

Let \((\Delta \beta(k), \Delta x(k), \Delta y(k))\) denote the expected vector to increase \( PM(k) \); then, it is the gradient of \( PM(k) \). When \( R \) has no coordinated companion:

\[
(\Delta \beta(k), \Delta x(k), \Delta y(k))^T = (-\lambda \eta agl(\beta_k - \beta(G, p_b))), 0, 0)^T
\]  

When \( R \) has coordinated companions:

\[
\begin{bmatrix}
\Delta \beta_{R_i}(k) \\
\Delta x_{R_i}(k) \\
\Delta y_{R_i}(k)
\end{bmatrix} = \begin{bmatrix}
\Delta \beta_{R_i}(k) \\
\Delta x_{R_i}(k) \\
\Delta y_{R_i}(k)
\end{bmatrix} = \begin{bmatrix}
0 \\
\lambda \eta agl(\beta_k - \beta(G, p_b)) \\
\lambda \eta agl(\beta_k - \beta(G, p_b))
\end{bmatrix}
\]  

\((\Sigma \Delta \beta(k), \Sigma \Delta x(k), \Sigma \Delta y(k))\) denotes the sum of the gradients of all repulsive moments acting on \( R \), where when \( R \) has no dangerous wall representative point at \( t \), \((\Sigma \Delta \beta(k), \Sigma \Delta x(k), \Sigma \Delta y(k)) = (0, 0, 0)^T\).

The motion controller of \( R \) using artificial moments for multi-robot path-planning is designed as (19), (21), (22):

\[
\beta_{R_i}(k+1) = agl(\beta_k + \Sigma \Delta \beta_{R_i}(k) + \Delta \beta_{R_i}(k) + \Sigma \Delta \beta_{k}(k))
\]  

Let:

\[
\begin{bmatrix}
\Delta x_{R_i} \\
\Delta y_{R_i}
\end{bmatrix} = \begin{bmatrix}
S_{x_t} (k+1) + \Sigma \Delta x_{R_i}(k) + \Delta x_{R_i}(k) + \Sigma \Delta y_{R_i}(k) \\
S_{y_t} (k+1) + \Sigma \Delta y_{R_i}(k) + \Delta y_{R_i}(k) + \Sigma \Delta x_{R_i}(k)
\end{bmatrix}
\]  

If \( \sqrt{(\Delta x_{R_i})^2 + (\Delta y_{R_i})^2} \) is not larger than \( S_t \), then:

\[
\begin{bmatrix}
x_{R_i}(k+1) \\
y_{R_i}(k+1)
\end{bmatrix} = \begin{bmatrix}
x_{R_i}(k+1) \\
y_{R_i}(k+1)
\end{bmatrix} + \begin{bmatrix}
\frac{\Delta x_{R_i} S_t}{\sqrt{(\Delta x_{R_i})^2 + (\Delta y_{R_i})^2}} \\
\frac{\Delta y_{R_i} S_t}{\sqrt{(\Delta x_{R_i})^2 + (\Delta y_{R_i})^2}}
\end{bmatrix}
\]  

Otherwise:

\[
\begin{bmatrix}
x_{R_i}(k+1) \\
y_{R_i}(k+1)
\end{bmatrix} = \begin{bmatrix}
x_{R_i}(k+1) \\
y_{R_i}(k+1)
\end{bmatrix} + \begin{bmatrix}
\Delta x_{R_i} S_t \\
\Delta y_{R_i} S_t
\end{bmatrix}
\]  

From (21)-(22), we can conclude that the proposed motion controller can make a robot enter and pass through narrow passages. Although the effects of attractive and repulsive moments may be cancelled out by each other when \( R \) is close to a narrow passage, \( R \) can still enter the passage through the motion vector \((S_x(k+1), S_y(k+1))^T\).
The controller can also make \( R_i \) reach a target with obstacles nearby. As with when \( R_i \) is close to \( T_i \), \((S_{n}(k+1), S_{n}(k+1))\) is zero, which means that repulsive moments cannot influence \( R_i \)'s motion. As a result, \( R_i \) can reach \( T_i \) under the influence of the attractive moment generated by \( T_i \).

When \( R_i \) has no coordinated companion, the controller will make \( R_i \) move under the guidance of its OWRepoint at its full speed for almost of the time. As such, no problem similar to the “local minimum” problem exists in the controller as it is impossible for \( R_i \) to stay at a “local minimum” point or in a small region.

5. Algorithm for multi-robot path-planning

Although the proposed motion controller has many advantages, it still has certain problems.

One is that where the OWRepoint is not \( T_i \) or else its distance from \( P_{Io} \) is larger than \( D_{h} \), the controller may cause \( R_i \) to move away from the OWRepoint. If \( R_i \) has a dangerous wall representative point \( G \) satisfying (23), as shown in Figure 3(a):

\[ |\text{agl}(\beta(G, P_{Io})-\beta_{h})+\text{agl}(\beta_{h}-\beta(G, P_{Io}))| \geq \pi \]  \hspace{1cm} (23)

\[ \text{OWRepoint} \]

\[ \text{PMDline} \]

\[ \text{G} \]

\[ \text{PMDline} \]

\[ \text{G} \]

\[ \text{OWRepoint} \]

\[ (a) \]

\[ (b) \]

\[ (c) \]

Figure 3. (a) \( R_i \) moves away from the OWRepoint; (b) \( R_i \) cannot achieve the same posture as that of \( T_i \) as much as possible. (c) The conflict between \( R_i \) and \( R_j \) cannot be solved effectively.

The reasons for this are that if (23) is satisfied, \( R_i \)'s PMDline direction may not be changed by the controller since the sign of \( \text{agl}(\beta_{h}-\beta(P_{Io}, \text{OWRepoint})) \) will be opposite to that of \( \text{agl}(\beta_{h}-\beta(G, P_{Io})) \), as shown in Figure 3(a); as such, the effects of the attractive and repulsive moments may cancel each other out. Accordingly, \( R_i \) will move away from the OWRepoint: if \( |\text{agl}(\beta_{h}-\beta(P_{Io}, \text{OWRepoint}))| \geq \pi/2 \). In this situation, the method for avoiding the problem set out above is to let \( \beta_{h}(k) \) be \( \text{agl}(\beta_{h}(P_{Io}, \text{OWRepoint})+\delta_{h}) \) if the OWRepoint is \( T_i \) or else after letting \( \beta_{h}(k)=\text{agl}(\beta_{h}(P_{Io}, \text{OWRepoint})+\delta_{h}) \), \( R_i \)'s PMDline has no intersection with the knowledge obstacle wall on which the OWRepoint lies; otherwise, let \( \beta_{h}(k) \) be \( \text{agl}(\beta_{h}(P_{Io}, \text{OWRepoint})-\delta_{h}) \).

A second problem is that in the case where the OWRepoint is \( T_i \) and its distance from \( P_{Io} \) is not larger than \( D_{h} \), the controller may not cause \( R_i \) to achieve the same posture as that of \( T_i \) as much as possible, if \( R_i \) has a dangerous wall representative point \( G \) satisfying (24), as shown in Figure 3(b).

\[ |\text{agl}(\beta_{h}-\beta_{h})+\text{agl}(\beta_{h}-\beta(G, P_{Io}))| \geq \pi \]  \hspace{1cm} (24)

The reasons are similar to that of the first. In this case, the method for avoiding the problem is to let \( \beta_{h}(k) \) be \( \beta_{h}(k) \).

A third problem is that where the OWRepoint is \( T_i \) and the distance between \( P_{Io} \) and \( T_i \) is not larger than \( D_{h} \), if \( R_i \) has a coordinated companion of \( R_j \) and \( |\text{agl}(\beta_{h}-\beta_{h})| \leq \delta_{h} \), as shown in Figure 3(c), then the controller may not resolve the conflicts between \( R_i \) and \( R_j \). The reasons for this are that as the direction of \( R_j \)'s PMDline may be the same as that of \( T_i \)'s PMDline under the influence of the attractive moment generated by \( T_i \), it is concluded that there is no marked trend for them to move towards each other. As a result, the coordinated moment generated by \( R_j \) will only allow \( R_i \) to maintain a certain safe distance from \( R_j \) but it will not guide \( R_i \) in bypassing \( R_j \). In this situation, the method for avoiding the problem is to let \( \beta_{h}(k)=\text{agl}(\beta_{h}(k)+\pi) \).

After the above-mentioned pre-processes, the proposed controller performs well in various situations. The algorithm for the local path-planning of multi-robots is then given as follows:

**Algorithm 3 Path-planning of multi-robots**

1. Set the values of the parameters in the control system; initialize the shared environment and the sets of robots, targets and knowledge obstacle walls; let \( t = t_0 \).
2. For each robot that has not reached its target, remove or set artificial obstacle segments until no such work can be done.
3. For each robot \( R_j \) if it has not reached \( T_i \) or else has coordinated companions, then go to step 4. Otherwise, its position and PMDline will not be changed.
4. Obtain the OWRepoint and all dangerous wall representative points of \( R_j \).
5. In the case where the OWRepoint is \( T_i \) and its distance from \( P_{Io} \) is not larger than \( D_{h} \), if \( R_i \) has a coordinated companion \( R_j \) and \( |\text{agl}(\beta_{h}-\beta_{h})| \leq \delta_{h} \), then let \( \beta_{h}=\text{agl}(\beta_{h}+\pi) \); otherwise, if \( T_i \)'s PMDline direction is not at \( t_0 \), then let \( \beta_{h} \) be that at \( t_0 \).
6: If the OWRpoint is not Ti or else its distance from Pi is larger than D, then if Ri has a dangerous wall representative point G satisfying (23), let \[ b(k) = \alpha(\beta(P_i, OWRpoint) + \delta). \] Otherwise, if Ri has a dangerous wall representative point G satisfying (24), let \[ b(k) = \beta(k). \]

7: Move a step to the next position under the control of the proposed motion controller. Update the current time as \( t = t + 1 \).

8: If all of the robots have reached their targets, then stop; otherwise, update their knowledge obstacle walls and let \( t = t + 1 \); then, return to step 2.

6. Simulations and analysis

In order to demonstrate the feasibility of the proposed method, extensive simulations have been carried out and two of them are given in what follows. In the simulations, no robot has a priori knowledge about the environment or other robots. The parameters are: \( D_s = 0.4, D_v = 3.5, D_w = 1, D_w = S = 0.24, \lambda = 0.5, \lambda = 0.8, \delta = \pi/90 \) and \( \delta = \pi/3 \). A simulation will terminate automatically if the distance from the boundary of a robot to that of another one (or an obstacle) is less than 0.05.

![Figure 4. Path-planning of two robots with 130 steps](image)

The path-planning of two robots in a complicated environment is shown in Figure 4 where (including Figure 5) short segments on obstacles are N-type artificial obstacle segments. From Figure 4, we can see that the two robots encounter each other in a narrow passage whose width is only \( 2D_s = 2 \), but the conflict between them is solved quickly. The simulation verifies that the method is effective in solving the conflicts between robots in narrow passages and that robots will not be trapped in complicated environments.

The path-planning of three robots in a complicated environment is shown in Figure 5, where \( T_3 \) is in the middle of a narrow passage whose width is also \( 2D_s = 2 \), and \( R_5 \) is at \( T_3 \) at the initial time.

Figure 5 (a) and (b) present the situations in solving the conflicts between the robots at the entrance of a narrow passage and in the middle of the passage, respectively. The simulation verifies, again, that the method is effective for the path-planning of multi-robots in complicated environments.

From Figures 4 and 5, we can see that although the path travelled by each robot may not be the optimal one, it is sure to be a sub-optimal one. As such, the method can optimize the path travelled by each robot.

![Figure 5. Path-planning of three robots with 149 steps](image)
7. Conclusions

From the discussion and simulations given in this paper, we can arrive at the following conclusions.

1. The proposed method meets the requirements of the real-time property for the following reasons. Firstly, all of the computations involved in the method are spatially distributed and their complexity is bounded regardless of the number of robots. Secondly, it needs almost no pre-processing of the environment, no consideration of the safe distance of robots and no explicit communication. Thirdly, the computational complexity of OWRpoints is low.

2. Compared with the traditional APF method, the proposed motion controller has certain advantages as follows. Firstly, under its control, a robot will not be trapped in a complicated environment, can enter and pass through narrow passages, can reach a target with obstacles nearby, and can achieve the same posture as that of its target as much as possible when it reaches its target. Secondly, it can resolve the conflicts between robots while no other techniques - such as priorities and negotiation - are needed and it can minimize other robots’ influences on a given robot’s motion.

3. The proposed method can optimize the path travelled by each robot as each robot’s motion is under the guidance of its OWRpoints and other robots’ influences are minimized by coordinated moments.

A disadvantage is that the proposed method may find it difficult to solve the conflicts between robots in passages narrower than $2D_0$ or in a dynamical environment. In the future, we will try our best to overcome these disadvantages.

8. Acknowledgments

The research reported in this paper was supported by the NSF of China under Grant No. 60874017, the State Key Laboratory of Robotics and System (HIT) under Grant No. SKLRS-2011-MS-03 and the Special Research Foundation of Liaoning University of Science and Technology under Grant No. 2012RC01.

9. References

[1] Rigatos G. G. (2008) Multi-robot motion planning using swarm intelligence. *International Journal of Advanced Robotic Systems*, Vol. 5, No. 2, pp. 139-144.

[2] Chakraborty J., Konar A., Chakraborty U. K., Jain L. C. (2008) Distributed cooperative multi–Robot path planning using differential evolution. *Proceedings of IEEE Congress on Evolutionary Computation*, pp. 718-725.

[3] Bien Z., Lee J. (1992) A minimum-time trajectory planning method for two robots. *IEEE Trans. Robot. Autom.*, Vol. 8, pp. 414–418.

[4] Shin K. G., Zheng Q. (1992) Minimum-time collision-free trajectory planning for dual-robot systems. *IEEE Trans. Robot. Autom.*, Vol. 8, pp. 641–644.

[5] Lavalle S. M., Hutchinson S. A. (1998) Optimal motion planning for multiple robots having independent goals. *IEEE Trans. Robot. Autom.*, Vol. 14, No. 6, pp. 912-925.

[6] Lumelsky V. J., Harinarayan K. R. (1997) Decentralized motion planning for multiple mobile robots: the cocktail party model. *Autonomous Robots*, No. 4, pp. 121–135.

[7] Pallottino L., Scordio V. G., Bischi A., Frazzoli E. (2007) Decentralized cooperative policy for conflict resolution in multivehicle systems. *IEEE Trans. Robot.*, Vol. 23, No. 6, pp. 1170-1183.

[8] Zheng C., Li L., Xu F., Sun F., Ding M. (2005) Evolutionary route planner for unmanned air vehicles. *IEEE Trans. Robot.*, Vol. 21, No. 4, pp. 609–620.

[9] Besada-Portas E., Torre L., Cruz J. M., Andrés-Toro B. (2010) Evolutionary trajectory planner for multiple UAVs in realistic scenarios. *IEEE Trans. Robot.*, Vol. 26, No. 4, pp. 619–634.

[10] Bennewitz M., Burgard W., Thrun S. (2001) Optimizing schedules for prioritized path planning of multi-robot systems. *IEEE International Conference on Robotics and Automation*, pp. 271-276.

[11] Regge R., Levi P. (2006) Cooperative multi-robot path planning by heuristic priority adjustment. *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5954-5959.

[12] Kim S. H., Lee G., Hong I., Kim Y. J., Kim D. (2010) New potential functions for multi robot path planning: SWARM or SPREAD. *The 2nd international conference on computer and automation engineering*, pp. 557-561.

[13] Yan Y. J., Zhang Y. (2009) Collision avoidance planning in multi-robot based on improved artificial potential field and rules. *Proceedings of the 2008 IEEE International Conference on Robotics and Biomimetics*, pp. 1026-1031.

[14] Yin L., Yin Y. X., Lin C. J. (2009) A New Potential Field Method for Mobile Robot Path Planning in the Dynamic Environments. *Asian Journal of Control*, Vol. 11, No. 2, pp. 214-225.

[15] Ge S. S., Cui Y. J. (2002) Dynamic motion planning for mobile robots using potential field method. *Autonomous Robots*, No. 13, pp. 207–222.

[16] Takahashi O., Schilling R. J. (1989) Motion planning in a plane using generalized Voronoi diagrams. *IEEE Trans. Robot. Autom.*, Vol. 5, No. 2, pp. 143-150.

[17] Lozano-Perez T. (1983) Spatial planning: a configuration space approach. *IEEE Trans. Comput.*, Vol. 32, pp. 108-120.
[18] Liu Y. H., Arimoto S. (1991) Proposal of tangent graph and extended tangent graph for path planning of mobile robots. *Proceedings of IEEE International Conference on Robotics and Automation*, pp. 312-317.

[19] Huang H. P., Chung S. Y. (2004) Dynamic visibility graph for path planning. *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2813-2818.

[20] Svestka P., Overmars M. H. (1995) Coordinated motion planning for multiple car-like robots using probabilistic roadmaps. *Proceedings of IEEE International Conference on Robotics and Automation*, pp. 1631-1636.

[21] Xu W. B., Chen X. B. (2008) Artificial moment method for swarm-robot formation control. *Science in China Series F: Information Science*, Vol. 51, No. 10, pp. 1521-1531.

[22] Xu W. B., Chen X. B. (2009) A dynamical formation control approach based on artificial moments. *Control Theory & Applications* (in Chinese), Vol. 26, No. 11, pp. 1232-1238.