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

Tracking of Closed–Curve Trajectories for Multi–Robot Systems

Lorenzo Sabattini, Cristian Secchi, Cesare Fantuzzi and Daniel de Macedo Possamai

Abstract— **In this paper we address the trajectory tracking problem for groups of mobile robots. We consider trajectories described by completely arbitrary shaped closed curves. The proposed control strategy is a completely decentralized algorithm, and does not require any global synchronization. The desired behavior is obtained by means of some properly designed artificial potential functions.**

I. Introduction

In this paper we describe a control strategy for the trajectory tracking of groups of mobile robots. This control strategy has been developed for the control of Automated Guided Vehicles (AGVs) moving in an industrial environment. More specifically, our target applications are automatic systems for end–of–line operations, for example delivery of goods from the production machines to the warehouse.

Since industrial environments are typically highly cluttered environments, the admissible paths for the AGVs often assume very strange shapes. Thus, in this paper we introduce a trajectory tracking control strategy that is suitable for trajectories with completely arbitrary shapes.

In the literature, many control strategies have been proposed for trajectory tracking. Traditional approaches (see e.g. [3] and references therein) generally make the mobile robot follow a reference point that moves along the trajectory, by means of error feedback. Even though these strategies are very effective for a single vehicle to track a trajectory, it's not straightforward to extend them to the multi–vehicle case.

In the multi–vehicle case, each robot has to track the trajectory without colliding with the other ones, and maintaining a desired distance from them. For this purpose, traditional collision avoidance strategies, for example potential based ones [6], are not suitable. An adapted potential based control strategy is presented in [13] for automatic driving on highways. The composition of the artificial potentials makes the vehicles change the lane to overtake other vehicles, thus avoiding collisions. Conversely, we would like to address a single lane scenario, in which the vehicles never leave the trajectory, and synchronize their motion along it.

The deployment of a group of robots over a curve has been addressed in the field of boundary tracking algorithms [4], [5]. These algorithms aim at deploying a group of robots along the boundary of a certain zone, usually for environmental monitoring. However, generally boundaries are approximated by convex (or star–convex) curves [2], since a higher precision in the definition of the boundary is not needed for environmental monitoring. Furthermore, the aim of these algorithms is to spread the robots over the boundary and then to stop them [2], or to make them patrol a small segment of the boundary moving alternatively forward and backward [12].

In [11] a potential based control strategy has been introduced to obtain an arbitrary shaped formation of mobile robots. The composition of the artificial potential fields leads to the creation of a regular polygon formation. Exploiting a properly defined bijective coordinates transformation, it is possible to obtain formations with completely arbitrary shape.

Introducing one further coordinates transformation it is possible to make the coordinates system rotate, in order to make the robots move along the circumference. However, since it's not always possible to find a suitable coordinates transformation to relate a circumference with a completely arbitrary shaped curve, it is necessary to heavily modify the control strategy, in order to avoid the use of coordinates transformations.

Thus, in this paper we introduce an artificial potential based control strategy to make a group of robots track an arbitrary shaped trajectory, moving at a given desired speed. The algorithm is completely decentralized, and the coordination along the curve is obtained without any global synchronization. Artificial potential functions are used to make the robots reach the curve and move along it, and to obtain the desired spacing between each couple of robots. Furthermore, the number of robots involved in the trajectory tracking can change dynamically, allowing sudden addition or subtraction of robots. Clearly, this control strategy works also perfectly for the trajectory tracking performed by a single vehicle.

II. Paths described with implicit functions

In this paper we consider a group of n point mass holonomic agents characterized by the following dynamics:

$$\ddot{x}_i = v_i \quad i = 1, ..., n \tag{1}$$

where $x_i \in \mathbb{R}^2$ is the position of the i–th agent. The dynamic behavior we are considering is quite simple, but all the results obtained in the paper can be extended to nonholonomic vehicles. In fact, many strategies can be found (e.g. [9]) to feedback linearize several classes of nonholonomic vehicles. Furthermore, we suppose that the agents can localize themselves exactly.

L. Sabattini is with the Department of Electronics, Computer Sciences and Systems (DEIS), University of Bologna, Italy. lorenzo.sabattini2@unibo.it

C. Secchi and C. Fantuzzi are with the Department of Sciences and Methods of Engineering (DISMI), University of Modena and Reggio Emilia, Italy {cristian.secchi, cesare.fantuzzi}@unimore.it

D. Possamai is with the Department of Automation and Systems Engineering (DAS), Federal University of Santa Catarina, Brazil

In this section we introduce a control law that makes the robots move along an arbitrary shaped curve that can be described by means of an implicit function $f(x) = 0$, $x \in \mathbb{R}^2$.

To make a group of robots converge to the desired curve, we control them to perform a gradient descent of f^2 [7]. Thus we introduce the following control law:

$$v_i = -K\nabla f^2 + f_{ti} + f_{di} - b\dot{x}_i \tag{2}$$

where K and b are positive constants, and b implements a damping action.

The term $-K\nabla f^2$ is orthogonal to the curve $\mathcal{C}$ in every point of the space. The role of this term is to make the robots converge to the desired curve $\mathcal{C}$.

The role of the term f_{ti} is to make the i–th robot move along the curve $\mathcal{C}$ at the desired speed. To this aim, the force f_{ti} is tangent to the curve at every time. More specifically, f_{ti} is described as follows:

$$f_{ti} = \omega \cdot R_\theta \cdot \frac{-\nabla f^2}{\|\nabla f^2\|} \tag{3}$$

where $\omega \in \mathbb{R}$ is a constant, proportional to the desired speed for the robot along the curve, and R_θ is a rotation matrix. The rotation matrix R_θ is defined as follows:

$$R_\theta(x_i) = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \tag{4}$$

where:

$$\theta(x_i) = \begin{cases} -\pi/2 & \text{if } f(x_i) > 0 \\ \pi/2 & \text{otherwise} \end{cases} \tag{5}$$

In other words, $\theta(x_i) = -\pi/2$ if the i–th robot is outside the curve $\mathcal{C}$, and $\theta(x_i) = \pi/2$ if it is inside the curve. As shown for example in Fig. 1, this definition of the rotation matrix R_θ leads to a movement along the curve in counterclockwise direction if $\omega > 0$, and in clockwise direction if $\omega < 0$.

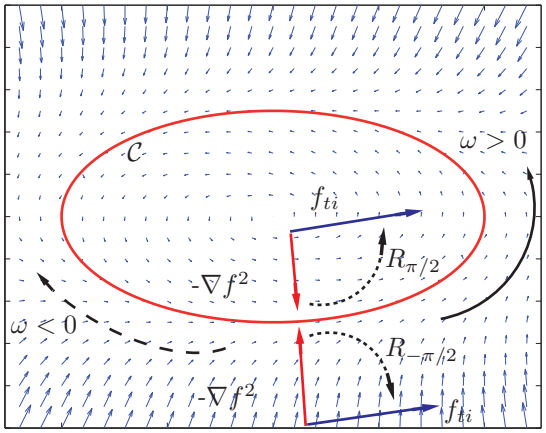

Fig. 1. The force f_{ti} is perpendicular to the negative gradient of f^2. Their composition makes the robots converge to the curve and move along it

Clearly Eq. (3) is not defined when $\|\nabla f^2\| = 0$. This condition is verified only when the robot is on the curve $\mathcal{C}$. In this case, we want the control action to drive the robot along the curve. In other words, the force f_{ti} needs to be still tangent to the curve $\mathcal{C}$. To obtain this, we slightly modify Eq. (3):

$$f_{ti}(x_i) = \begin{cases} \omega \cdot R_\theta(x_i) \cdot \dfrac{-\nabla f^2(x_i)}{\|\nabla f^2(x_i)\|} & \text{if } f(x_i) \neq 0 \\ \omega \cdot R_\theta(x_p) \cdot \dfrac{-\nabla f^2(x_p)}{\|\nabla f^2(x_p)\|} & \text{otherwise} \end{cases} \tag{6}$$

where x_p is an arbitrary point on the line perpendicular to the curve $\mathcal{C}$ passing through x_i. Fig. 1 shows the composition of the negative gradient of f^2 and of the force f_{ti}: the composition of these two actions drives the robots to converge to the curve $\mathcal{C}$, and then move along it.

The role of the term f_{di} is to take the robot i at the desired distance from the other robots. This force is given by the composition of two terms:

$$f_{di} = \sum_{j=1;j\neq i}^{n} f_{rij} + \sum_{j=1;j\neq i}^{n} f_{qij} \tag{7}$$

The term f_{rij} implements a repulsive action if robot i and robot j are closer than the safety distance d_s. The value of d_s is the minimum distance that ensures that collisions between two robots never happen. More specifically:

$$f_{rij} = -\nabla_{x_i} V_{rij}(x_i, x_j) \tag{8}$$

and

$$V_{rij}(x_i, x_j) = \begin{cases} 0.5K_r(d_{ij} - d_s)^2 & \text{if } d_{ij} \leq d_s \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

where $d_{ij}(t) = \|x_i(t) - x_j(t)\|$, and K_r is a positive constant.

To regulate the relative positions of the agents along the curve we introduce the term f_{qij}. In fact, the term f_{rij} regulates only the euclidean distances among the agents. The fact that the euclidean distances between each couple of agents are equal to the desired one doesn't imply at all that the agents are deployed along the curve as desired.

Let u be a curvilinear abscissa, defined on the curve $\mathcal{C}$. The term f_{qij} is active only when robot i and robot j are on the curve $\mathcal{C}$. This term implements a repulsive action based on the value of the curvilinear abscissa that corresponds to the positions of robot i and robot j on the curve $\mathcal{C}$. Given the position of the robot $x_i \in \mathbb{R}^2$, the corresponding curvilinear abscissa u_i is defined as the value of the curvilinear abscissa that corresponds to the point of the curve that is closest to x_i.

The force f_{qij} is tangent to the curve $\mathcal{C}$ in the position of robot i. This force implements a repulsive action if the distance between robot i and robot j. Let u_i and u_j be the value of the curvilinear abscissa corresponding to the positions of robot i and robot j respectively, and let $u_{ij} = |u_i - u_j|$. The magnitude of f_{qij} is defined as follows:

$$\|f_{qij}\| = \begin{cases} K_u|u_{ij} - u_d| & \text{if } u_{ij} \leq u_d \\ 0 & \text{otherwise} \end{cases} \tag{10}$$

where K_u is a positive constant. The direction of f_{qij} is tangent to $\mathcal{C}$ in the position of robot i, and the orientation is defined to implement a repulsive action.

To ensure collision avoidance, we assume that the forces f_{rij} and f_{qij} are much stronger than $-K\nabla f^2$ and f_{ti}. This is obtained by means of an appropriate choice of the parameters K_r and K_u.

We want to remark that the force f_{rij} should be active only for collision avoidance. This means that, to avoid interference between f_{rij} and f_{qij}, we must choose the parameter d_s so to define a region much smaller than the one defined by u_d. Furthermore, the curve must be defined such that its curvature do not cause any collision among the agents.

Proposition 1 *The robots asymptotically converge to the curve $\mathcal{C}$ and, after the transient, never leave it*

Proof: The motion of the robots can be considered as the composition of two components of motion:

- the motion in direction parallel to the curve $\mathcal{C}$,
- the motion in direction perpendicular to the curve $\mathcal{C}$.

Namely:

$$\dot{x}_i = \dot{x}_{i\perp} + \dot{x}_{i\parallel} \tag{11}$$

To prove the convergence of the motion to the curve $\mathcal{C}$, we are only interested in the perpendicular component, namely $\dot{x}_{i\perp}$.

As defined so far, the forces f_{ti} and f_{qij} don't have any component in the direction perpendicular to the curve $\mathcal{C}$. Thus, these forces do not influence the dynamics of $\dot{x}_{i\perp}$.

The force f_{rij} is active only for collision avoidance: this means that it can be different from zero only during the initial transient, when the robots start moving from their initial positions, and it can happen that two or more robots are closer than the safety distance. Therefore, f_{rij} can be considered zero after the initial transient.

Thus, from Eq. (1) and Eq. (2), we obtain the following dynamics:

$$\ddot{x}_{i\perp} = -K\nabla f^2 - b\dot{x}_{i\perp} \tag{12}$$

To prove that the robot converges to the curve, we need to prove the asymptotic stability of:

$$\begin{cases} x_i \in \mathcal{C} \\ \dot{x}_{i\perp} = 0 \end{cases} \tag{13}$$

Consider the following Lyapunov candidate function:

$$V(x_i) = Kf^2(x_i) + 0.5\left\|\dot{x}_{i\perp}\right\|^2 \tag{14}$$

which is trivially non–negative, and equal to zero only when the conditions in Eq. (13) are verified. The time derivative of this function is the following:

$$\dot{V}(x_i) = \left(K\nabla f^2 + \ddot{x}_{i\perp}\right)^T \dot{x}_{i\perp} \tag{15}$$

From Eq. (12) we obtain:

$$\dot{V}(x_i) = -b\left\|\dot{x}_{i\perp}\right\|^2 \tag{16}$$

which is always less than or equal to zero. The asymptotic stability can be proved by invoking LaSalle's principle. ■

Proposition 2 *After the transient, once on the curve the robots move along the curve at a constant speed*

Proof: With respect to the decomposition of the motion of the robot described in Eq. (11), in this case we are interested in the component of the motion which is parallel to the curve $\mathcal{C}$, namely $\dot{x}_{i\parallel}$.

By definition, the gradient of f^2 doesn't have any component in the direction parallel to $\mathcal{C}$. As stated before, the force f_{rij} can be considered zero after the initial transient.

Thus, from Eq. (1) and Eq. (2), we obtain the following dynamics:

$$\ddot{x}_{i\parallel} = f_{ti} + \sum_j f_{qij} - b\dot{x}_{i\parallel} \tag{17}$$

As stated before, the forces f_{qij} are much stronger than f_{ti}. Therefore, if the robots are on the curve and the distance between two neighbors is less than the desired one, the forces f_{qij} make them deploy along the curve as desired. Once the robots have deployed along the curve, the forces f_{qij} are no longer active, and Eq. (17) can be rewritten as follows:

$$\ddot{x}_{i\parallel}(t) = f_{ti}(t) - b\dot{x}_{i\parallel}(t) \tag{18}$$

Since we are considering only the dynamics in direction parallel to the curve, it follows from Eq. (6) that, along this direction, $f_{ti}(t) \equiv \omega$. Thus, Eq. (18) can be rewritten as follows:

$$\ddot{x}_{i\parallel}(t) = \omega - b\dot{x}_{i\parallel}(t) \tag{19}$$

The differential equation in Eq. (19) can be easily integrated, thus obtaining

$$\dot{x}_{i\parallel}(t) = (\omega/b) + ce^{-bt} \tag{20}$$

where c is an arbitrary constant. As time goes to infinity, we have

$$\lim_{t\to\infty} \dot{x}_{i\parallel}(t) = (\omega/b) = \text{constant} \tag{21}$$

This proves that, asymptotically, the robots move along the curve $\mathcal{C}$ at a constant speed proportional to ω. ■

We want to remark that, since all the terms of the control strategy are independent of the total number of robots, sudden addition or subtraction of robots is managed automatically, as shown in the experiments described in Section V.

The main drawback of the control strategy presented so far is that, even if many curves can be represented as implicit function, with this formulation it is not possible to represent completely arbitrary shaped curves.

III. Paths described with parametric functions

Generally speaking, a closed curve in $\mathbb{R}^2$ can be described by means of a parametric function $x = g(u)$, with $x \in \mathbb{R}^2$ and $u \in \mathbb{R}$. In the literature, many methods can be found to define these parametric functions. For example, arbitrary shaped closed curves can be defined by means of Bezier curves, B–splines or NURBS [10].

Since, in general, it is not always possible to obtain an implicit formulation of the curve $\mathcal{C}$ from its parametric formulation, we adapt our algorithm to avoid the use of the implicit formulation.

Let L be the length of the curve $\mathcal{C}$, i.e. the curvilinear abscissa $u \in [0, L]$. Since the curve $\mathcal{C}$ is closed, $g(0) = g(L)$.

We slightly modify the control law in Eq. (2), in order to be able to compute the forces without the expression of $f(x)$.

The gradient descent of f^2 is approximated as follows. Let x_i be the position of the i–th robot. At each time, the robot finds the closest point of the curve, i.e. the value u^* of the curvilinear abscissa such that

$$u^* = \arg\min_{u\in[0,L]} \|x_i - g(u)\| \tag{22}$$

It can happen that u^* is not uniquely defined, i.e. more than one point of the curve have the same minimum distance from the i–th robot. In particular, this can happen when the i–th robot is approaching the (non–convex) curve, and u^* defines the point where the robot enters the curve. In this case, u^* can be chosen randomly among the minimum distance points. Once the robot is on the curve, this ambiguity will not happen anymore.

In any point, the negative gradient of f^2 is perpendicular to the curve $\mathcal{C}$, and points towards it. Thus, we approximate it with a force that attracts the robot to $g(u^*)$, namely:

$$-K\nabla f^2 \approx -\nabla U_{att} \tag{23}$$

where

$$U_{att} = 0.5K \|x_i - g(u^*)\|^2 \tag{24}$$

Furthermore, we need to approximate the force f_{ti}. We approximate the composition of the negative gradient of f^2 and of the force f_{ti} as follows:

$$-K\nabla f^2 + f_{ti} \approx -\nabla U^{\omega}_{att} \tag{25}$$

with

$$U^{\omega}_{att} = 0.5K \|x_i - g((u^* + \omega)_{\mod L})\|^2 \tag{26}$$

where $(u)_{\mod L}$ is the reminder of the division of u by L.

This term approximates the composition of the negative gradient of f^2 and of the force f_{ti}, as described for example in Fig. 1. More specifically, the robot is not attracted to $g(u^*)$, but it is attracted to $g((u^* + \omega)_{\mod L})$, where $\omega \in \mathbb{R}$ is proportional to the desired speed along the curve.

- When the robot is not on the curve, it is attracted to the curve $\mathcal{C}$ as desired, since $g((u^* + \omega)_{\mod L})$ is clearly a point of $\mathcal{C}$.
- When the robot is on the curve, the point $g(u^*)$ is the robot's own position. Thus, being attracted to $g((u^* + \omega)_{\mod L})$ it is forced to move along the curve, at a speed proportional to ω.

The other terms of the control law are defined as described in the previous section.

Thus, the approximated control law introduced in this section implements both the actions perpendicular and parallel to the curve $\mathcal{C}$, making the robots converge to the curve and move along it.

We remark that the choice of the value of ω must be related to the shape of the curve, to guarantee a good tracking performance. In fact, if the curve, for instance, presents a sharp bend, a high value of ω will make the robots cross the bend according to a straight line, instead of following the curve as desired.

IV. Multiple objectives: logical layer of control

In this section we introduce a strategy to deal with multiple objectives, i.e. multiple curves to be tracked. More specifically, we want to make a group of mobile robots autonomously spread among a certain number of paths, to complete different tasks. For example, in an industrial end–of–line application, there can be different kinds of goods to be delivered from different machines to different locations in the warehouse. The objective is to obtain an emerging behavior by which the group of robots automatically spreads among the different tasks, given the number of robots needed by each task.

To this aim we introduce a higher control layer, that enables each robot to select the right task. More specifically, we introduce an algorithm based on the following assumptions and definitions:

1) The robots can communicate, by means of message passing, when their distance is less then a certain communication radius.
2) Each robot has a unique identifier (UID).
3) As stated in the previous sections, let L be the length of the curve, and let u_d be the desired distance between two neighboring agents on the curve. Then, $(L/u_d) = N \in \mathbb{N}$. In other words, u_d is defined so that so that an exact number N of robots is allowed to track the curve.
4) If the distance between two robots is less than or equal to u_d, they can communicate.
5) The k–th robot is attracted to the curve $\mathcal{C}$ by means of the control strategy presented so far, while n robot are already moving along the curve $\mathcal{C}$. Thus, if it finds that $n \geq (L/u_d)$, the k–th robot will move to a different task.
6) Let $\Delta_{j+1} = |u_{j+1} - u_j|$ and $\Delta_{j-1} = |u_{j-1} - u_j|$ be the distances, in terms of curvilinear abscissa, between the j–th robot and its neighbors along the curve.

Thus, the proposed algorithm is the following:

- The k–th robot sends to the j–th robot a message with its own UID, UID_k.
- The j–th robot starts Algorithm 1.

Algorithm 1 Reply of the j–th robot to the k–th robot request

if (Robot j is not on the curve) **then**
  Robot $j \rightarrow$ Robot k: $msg = [0, 0]$
else
  if $((\Delta_{j+1} > u_d)$ AND $(\Delta_{j-1} > u_d))$ **then**
    Robot $j \rightarrow$ Robot k: $msg = [1, 1]$
  else
    Robot $j \rightarrow$ Robot $j+1$: $msg = [UID_k, UID_j, 0]$
    Algorithm 2
  end if
end if

Algorithm 2 Message passing among the robots on the curve, to understand if one more robot is allowed to track the curve as well

msg_{in} = incoming message
msg_{out} = outgoing message
The i–th robot receives msg_{in}:
$msg_{in} = [UID_k, UID_j, m]$, $m = 0, 1$

1: **if** $msg_{in}(2) == UID_i$ **then**
2: Robot $i \rightarrow$ Robot k: $msg_{out} = [1, msg_{in}(3)]$
3: **else**
4: **if** $msg_{in}(3) == 1$ **then**
5: Robot $i \rightarrow$ Robot $i-1$: $msg_{out} = msg_{in}$
6: **else**
7: **if** $(\Delta_{i+1} > u_d)$ **then**
8: Robot $i \rightarrow$ Robot $i-1$: $msg_{out} = [UID_k, UID_j, 1]$
9: **else**
10: Robot $i \rightarrow$ Robot $i+1$: $msg_{out} = [UID_k, UID_j, 0]$
11: **end if**
12: **end if**
13: **end if**

Algorithm 1 describes how the j–robots computes the reply message to the request of the k–th robot. The reply message is a two–bit message, and its meaning is the following:

- $msg = [0, 0]$: the j–th robot is not on the curve $\mathcal{C}$.
- $msg = [1, 0]$: the j–th robot is on the curve $\mathcal{C}$, and $n \geq (L/u_d)$. The k–th robot must move to a different task.
- $msg = [1, 1]$: the j–th robot is on the curve $\mathcal{C}$, and $n < (L/u_d)$. The k–th robot can move along the curve $\mathcal{C}$.

If the condition in line 4 of Algorithm 1 is true, then the distance between the j–th robot and its neighbors is strictly greater than u_d. Thus, under Assumption 3, the number of robots on the curve $\mathcal{C}$ is less than the maximum allowed, and the k–th robot is allowed to move along the curve $\mathcal{C}$ as well (Fig. 2(a)).

Otherwise, if this condition is not verified for the j–th robot, we must check whether it is verified for another robot on the curve $\mathcal{C}$ (Fig. 2(b)). This is the purpose of Algorithm 2. In this case, the message is a vector with three components:

1) The first component is the UID of the k–th robot.
2) The second component is the UID of the j–th robot. This is the robot that started the message passing along the curve.
3) The last component can be 0 or 1:
   - it is set to 0 if the k–th robot is not allowed to move along the curve $\mathcal{C}$.
   - it is set to 1 if the k–th robot is allowed to move along the curve $\mathcal{C}$.

If the condition in line 7 of Algorithm 2 is true, then the distance between the i–th robot and its following neighbor is strictly greater than u_d. Thus, under Assumption 3, the number of robots on the curve $\mathcal{C}$ is less than the maximum allowed, and the k–th robot is allowed to move along the curve $\mathcal{C}$ as well (Fig. 2(c)). Thus, the third component of the outgoing message is set to one, and this message is sent back to the previous robot. The message is delivered to the j–th robot, that allows the k–th robot to move along the curve $\mathcal{C}$.

Conversely, if the condition in line 7 of Algorithm 2 is not verified for any robot on the curve $\mathcal{C}$, then the number of robots currently on the curve is greater then or equal to the maximum allowed (Fig. 2(d)). In this situation, the message passes through all the robots, and no one of them sets to 1 the third component of the message.

When the message comes back to the to the j–th robot, the condition in line 1 is true. The message passing among the robots on the curve ends, and the j–th robot sends to the k–th one the correct answer.

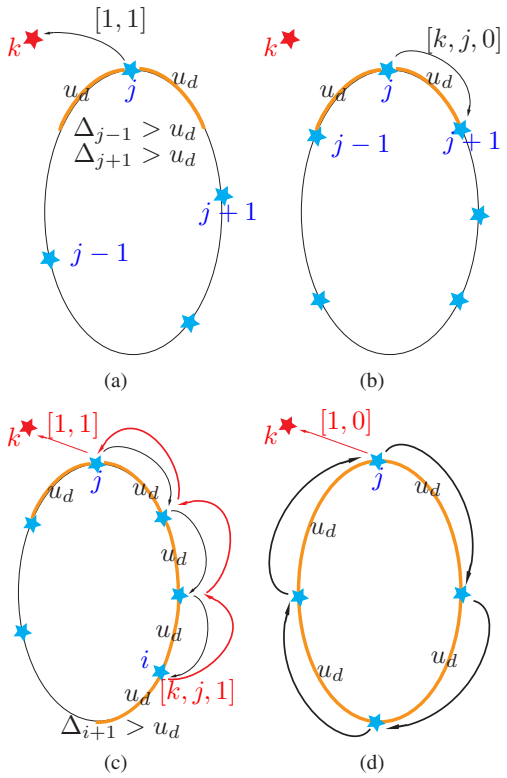

Fig. 2. Message passing algorithm

To quantify the complexity [1] of this algorithm, we consider the worst case: starting from the j–th robot, the message passes through all the robots until it reaches the $(j-1)$–th, and then goes back until it reaches the j–th one again. Let n be the number of robots currently on the curve $\mathcal{C}$. In the worst case, the message passing along the curve involves $2(n-1)$ messages. Furthermore, one message is sent from the k–th to the j–th robot, and the answer is sent back. Thus, the total number of messages exchanged in the worst case is $2n$. Hence, the communication complexity of this algorithm is linear with the number of robots involved.

V. Simulations and experiments

To validate our control strategy, we performed both Matlab simulations and experiments on real robots. Some examples are shown in the accompanying video clip.

In the simulations we considered point mass agents, tracking the desired curve $\mathcal{C}$, defined by means of the B–spline formulation. In the accompanying video clip some examples are presented. The agents, starting from random initial positions, reach the curve $\mathcal{C}$ and move along it. The speed of the agent is not uniform along the curve: this is obtained by means of a non–uniform discretization of the curve. This is useful to make the robots move faster in some zones and slower in some other zones. For example, this strategy can be exploited to slow down the robots while loading or unloading goods on them.

Furthermore, we performed several experiments on real robots, a group of E–Puck robots [8], moving in a $2.0m \times 1.5m$ arena. The positions of the robots are measured by means of a camera placed above the arena, and robots are tracked by means of colored markers. To deal with the fact that these robots are nonholonomic systems, we applied the feedback linearization technique presented in [9].

During the simulations and the experiments, the following values have been used for the parameters: $K = 100$, $K_u = K_r = 500$, $b = 15$, $\omega = 5$, with curves defined by 400 points.

Both simulations and experiments show the effectiveness of the control strategy described so far: in fact, after the transient, the robots correctly deploy along the curve and track it. The tracking is not perfect in some zones, where the agents move a little far from the curve: this is due to the discretization of the curve. In fact, in these zones the discretization is coarser than the rest of the curve. A non–uniform discretization is useful if in some zones a precise tracking is not needed (e.g. because in some zones of the environment there are no obstacles), because it reduces the number of points to be stored to describe the curve.

As shown in the video clip, collisions among the robots are avoided, and sudden addition or subtraction of robots is managed automatically.

VI. Conclusions and future works

In this paper we have presented a control strategy to make a group of mobile robots track a trajectory given by an arbitrary shaped desired curve.

This control strategy is a completely decentralized algorithm, since there is no need for any centralized controller or global synchronization.

By means of this control strategy, a group of mobile robots, starting from random initial positions, is able to reach the desired curve and then move along it. Once reached the curve, the robots never leave it. Furthermore, collision avoidance and desired spacing between neighboring robots is ensured.

With respect to previous works on trajectory tracking, the main advantage of this control strategy is the fact that it combines trajectory tracking with the coordination of multiple mobile robots. Furthermore, this result is obtained without any global controller, and without the need of knowing information about the whole group of robots. This ensures the correct behavior even in presence of sudden addition or subtraction of one or more robots.

In Section IV we have presented how to extend our control strategy to deal with multiple tasks, by means of message passing between the robots.

The application of this control strategy in the end–of–line industrial scenario is clearly not the only possible field of application. Another possible applications is the boundary tracking, for example around a nuclear plant. Generally speaking, this control strategy can be used to make mobile robots move around the perimeter of a given zone, for example for surveillance purposes.

Future work aims at studying the initial transient behavior, in order to control it by means of an appropriate tuning of the parameters. Furthermore, we want to extend the control strategy presented so far. More specifically, we are investigating how to realize decentralized control strategies to obtain more complex cooperative behaviors, which can be of interest in industrial environment.

References

[1] F. Bullo, J. Cortés, and S. Martínez. *Distributed Control of Robotic Networks*. Applied Mathematics Series. Princeton University Press, 2009. Electronically available at http://coordinationbook.info.

[2] Y. Cao and R. Fierro. Dynamic boundary tracking using dynamic sensor nets. In *Proceedings of the 45th IEEE Conference on Decision and Control*, 2006.

[3] M. Egerstedt, X. Hu, and A. Stotsky. Control of mobile platforms using a virtual vehicle approach. *IEEE Transactions On Automatic Control*, 11 2001.

[4] S. Jang, G. Song, and S. K. Hong. Dynamic boundary tracking in active sensor networks. In *Proceedings of the International Conference on Control, Automation and Systems*, 2007.

[5] Z. Jin and A. L. Bertozzi. Environmental boundary tracking and estimation using multiple autonomous vehicles. In *Proceedings of the 46th IEEE Conference on Decision and Control*, 2007.

[6] O. Khatib. Real–time obstacle avoidance for manipulators and mobile robots. *The International Journal of Robotics Research*, 1986.

[7] L. S. Marcolino and L. Chaimowicz. No robot left behind: Coordination to overcome local minima in swarm navigation. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 2008.

[8] F. Mondada, M. Bonani, X. Raemy, J. Pugh, C. Cianci, A. Klaptocz, S. Magnenat, J. C. Zufferey, D. Floreano, and A. Martinoli. The e-puck, a robot designed for education in engineering. In *Proceedings of the 9th Conference on Autonomous Robot Systems and Competitions*, pages 59–65, 2009.

[9] G. Oriolo, A. De Luca, and M. Vendittelli. WMR control via dynamic feedback linearization: Design, implementation, and experimental validation. *IEEE Transactions On Control Systems Technology*, 11 2002.

[10] L. Piege and W. Tiller. *The NURBS Book*. Springer–Verlag, 1995-1997.

[11] L. Sabattini, C. Secchi, and C. Fantuzzi. Potential based control strategy for arbitrary shape formations of mobile robots. In *Proceedings of the IEEE/RJS International Conference on Intelligent Robots and Systems*, 2009.

[12] S. Susca, F. Bullo, and S. Martínez. Synchronization of beads on a ring. In *Proceedings of the 46th IEEE Conference on Decision and Control*, 2007.

[13] M. T. Wolf and J. W. Burdick. Artificial potential functions for highway driving with collision avoidance. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 2008.