

# A trajectory generation method for mobile robot based on iterative extension-like process

Kuniaki Kawabata<sup>1</sup>

Received: 18 June 2016 / Accepted: 14 July 2016 / Published online: 28 July 2016  
© ISAROB 2016

**Abstract** In this paper, we propose a trajectory generation method for mobile robot based on iterative extension-like process. Due to use mobile robots in the real world, trajectory generation must be done depending on the faced situation on each occasion. Proposed method enables online iterative trajectory extension process based on a low-order polynomial curve named as trajectory segment. The waypoints on the existing trajectory segment and a waypoint designated every fixed interval are the constraints to trigger the trajectory extension. For maintaining the smooth continuity of the trajectory, the velocity state must be sustained at the connecting point. Resultantly, the trajectory segments are organized into a single smooth trajectory.

**Keywords** Trajectory generation · Mobile robot · Iterative extension-like process

## 1 Introduction

Intelligent mobile robotics can contribute to build a life partner agent such as a porter [1], a guide [2, 3], a home surveillance [4, 5] etc. Trajectory/path generation is an important factor for mobile robots. Polynomial-curve-based method is also a major approach [6–11]. The smoothness and diversity are determined by the redundancy on the degree of the polynomial. In [6], a polynomial is introduced to define the curvature primitive for planning motion of a vehicle. A cubic B-spline, is selected in [7]; the approach

discussed to generate a collision-free path with continuous curvature. Moreover, the Bezier-curve-based approach [8–11] is also reported. These works discussed single path generation when the constraints are given. Considering the long-term missions of the robot, there are the difficulties to design whole trajectory/path by only a single polynomial curve. Because in order to design the trajectory/path, appropriate high-order base curve and the number of the constraints must be prepared. An approach to compose a single complex path is to connect plural piecewise low order curves. Choi et al. [12] described a method for continuous curvature paths based on a set of low order Bezier curves. Zhou et al. [13] also proposed a method that cuts the Bezier curve into pieces and reconnects the pieces to constitute an improved Bezier curve. They assume that the shape of the original curve are given.

Rapidly-exploring Random Trees (RRT), [14] is a notable path planning approach and the extended algorithm [15] is also presented. A RRT method with closed-loop prediction by the forward simulation utilizing the vehicle's controller model was developed for smoothing the path [16]. Instead of such forward simulation, the spline interpolation between the sampled points was utilized for generating the path [17]. An anytime path planning method : RRT\* [18] can obtain asymptotic optimal solution by a rewritten process. However, a typical stop conditions for extending process by RRT and RRT\* is often that the path enters a certain area near the target and there is a weakness on the accuracy of reaching the target. Moreover, the shape of generated path by RRT is often jagged and meandering although the works attempt to smooth the path.

Considering the trajectory/path for long-term activities of the mobile robot, a possible and practical approach is to extend the trajectory on-line by maintaining continuity with the existing trajectory and satisfying the constraints

✉ Kuniaki Kawabata  
kawabata.kuniaki@jaea.go.jp

<sup>1</sup> Japan Atomic Energy Agency, 1-22, Yamadaoka, Naraha, Futaba, Fukushima 979-0513, Japan

on each occasion. Previous works [19] proposed a motion generation method that utilizes a velocity profile based on the cubic spline. can extend the velocity profile based on time shifts. However, this method could only work with the cubic spline polynomial. Our previous work has discussed trajectory extension method based on iteratively designated waypoint [20]. This method can derive trajectory segments sequentially and passing through designated waypoints. However, sometimes it cannot realize smooth continuity because the trajectory is derived with only position-based constraints of the waypoints. Therefore, we extended the method to maintain smooth continuity between the polynomial curve -based trajectory segments [21]. In this paper, we described practical trajectory generation algorithm and proposed method can realize to provide the types of the trajectory for the mobile robots.

### 2 Iterative trajectory extension-like strategy

Polynomial-based approaches have the advantage of expressing smooth curves with a small number of variables. Due to use mobile robots such as service purposes in the real world, online trajectory generation must be done depending on the faced situation on each occasion. Such situation information are usually acquired when the robot moves to a location. Moreover, it is required to generate the trajectory continuously for long-term activity. However, even when the polynomial-based approach is utilized, it is difficult to design a single curve for the whole trajectory during the mobile robot’s mission because high-order-based polynomial curves need to be estimated and a certain number of constraints are required to derive such a trajectory. Regarding practical use, generally the situation and environment in which the robot travels will possibly change due to the appearance of the obstacles, etc.. In such cases, online local trajectory re-planning is a suitable approach, but such re-planning must be done by considering the consistency with the previous trajectory like the continuity. One possible solution is to extend the trajectory gradually and iteratively based on the constraints. Figure 1 illustrates the concept of such trajectory extension approach. The approach generates another partial trajectory, called trajectory segment, online iteratively by referring to the current trajectory segment and additional waypoints. For realizing continuous trajectory extension, both the interpolation of all waypoints and the continuity between current and the next trajectory segments must be considered. As an example, Fig. 1 shows two trajectory segments and left one is the current trajectory segment. They have overlapped time

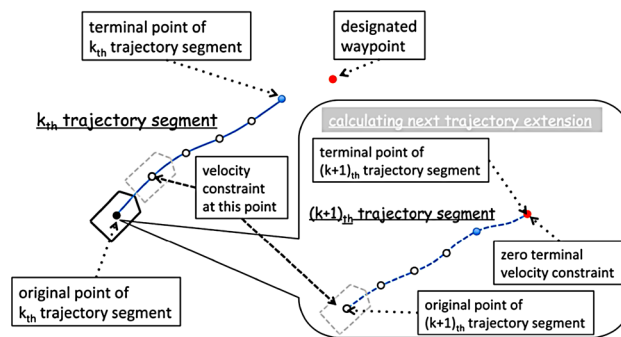


Fig. 1 Proposed online iterative trajectory extension scheme based on trajectory segment

duration and there are common waypoints between them. Concretely speaking, iterative trajectory extension process requires the propagation of the waypoints on the existing trajectory segment, newly designated waypoint and the smooth continuity between the trajectory segments as the constraints. Here, we assume that a newly designated waypoint is provided from like the environment recognition process every fixed time period. For maintaining the continuity between the trajectory segments, the state at such waypoint on the current trajectory segment must be maintained. Thus, the first waypoint is the connecting point for the current and the next trajectory segment and becomes the original point of the new trajectory segment. The passing time of each waypoints can also be propagated between the trajectory segments. Additionally, a primary differentiated state at the connecting point also must be maintained to realize the continuity between the trajectory segments As one more additional condition, it should be guaranteed to suspend extending the trajectory at anytime for realizing to design various types of the trajectory. The terminal velocity of every trajectory segment must be set to stop state. This condition contributes the robot can stop at the end of the final trajectory segment spontaneously. An application that this kind of approach can contribute is human following system like porter robots that required to generate the trajectory depending on master’s motion. In such application, environment recognition processing with the sensor like the camera and laser range finder can be considered to provide the candidate of additional waypoint.

### 3 Iterative calculation of trajectory segment

In this section, we describe iterative trajectory generation method. Initial trajectory segment is either given and

the waypoint for extension process is designated at a fixed interval. The equation of the  $k$ -th trajectory segment based on  $n$ -th polynomial curve is expressed.  $k$  refers to the number of iterative calculations.

$${}^k f(t) = \begin{bmatrix} {}^k x(t) \\ {}^k y(t) \end{bmatrix} = \sum_{i=0}^n {}^k A_i t^i \in \mathbf{R}^2 \tag{1}$$

$$t \in [k\Delta t, (k + m)\Delta t] \tag{2}$$

where  ${}^k A_i$  is the coefficient vector of the  $k$ -th trajectory segment and is valid under the conditions of Eq. (2). Thus,  ${}^0 f(t)$  is the initial trajectory segment.  $\Delta t$  is the sampling period for determining the next trajectory segment that passes through additional waypoints, and  $m\Delta t$  is the time period of each trajectory segment. As mentioned above, the iterative trajectory extension should be considered to design resultantly long term smooth trajectory. For this purpose, we first describe the conditions related to the waypoints. We can also obtain below equations of the  $(k + 1)$ -th trajectory segment.

$${}^{k+1} f(t) = \sum_{i=0}^n {}^{k+1} A_i t^i \in \mathbf{R}^2 \tag{3}$$

$$t \in [(k + 1)\Delta t, (k + m + 1)\Delta t] \tag{4}$$

When comparing Eqs. (3) with (1), an overlapped duration exists. Thus, there are common waypoints at  $t = (k + 1)\Delta t, (k + 2)\Delta t \cdots (k + m)\Delta t$  as follows.

$${}^k f((k + 1)\Delta t) = {}^{k+1} f((k + 1)\Delta t) \tag{5}$$

$${}^k f((k + 2)\Delta t) = {}^{k+1} f((k + 2)\Delta t) \tag{6}$$

$$\vdots \tag{7}$$

$${}^k f((k + m)\Delta t) = {}^{k+1} f((k + m)\Delta t)$$

${}^{k+1} f((k + m + 1)\Delta t)$  is an additional designated waypoint for extension-like process. Thus, from Eqs. (5)–(7) and waypoint  ${}^{k+1} f((k + m + 1)\Delta t)$ , we can obtain the following conditions.

$${}^k f((k + 1)\Delta t) = \sum_{i=0}^n {}^{k+1} A_i \{(k + 1)\Delta t\}^i \tag{8}$$

$${}^k f((k + 2)\Delta t) = \sum_{i=0}^n {}^{k+1} A_i \{(k + 2)\Delta t\}^i \tag{9}$$

$$\begin{aligned} & \vdots \\ {}^k f((k + m)\Delta t) &= \sum_{i=0}^n {}^{k+1} A_i \{(k + m)\Delta t\}^i \end{aligned} \tag{10}$$

$${}^{k+1} f((k + m + 1)\Delta t) = \sum_{i=0}^n {}^{k+1} A_i \{(k + m + 1)\Delta t\}^i \tag{11}$$

Like Eqs. (8)–(10), the waypoints of the  $k$ -th trajectory segment is transferred to the  $(k + 1)$ -th trajectory segment, maintaining the passing time. Additional designated waypoints at the time  $(k + m + 1)\Delta t$  is set in the row of the waypoints on  $(k + 1)$ -th trajectory segment. Moreover, we also know the relationship as follows.

$${}^{k+1} f((k + 1)\Delta t) = {}^{k+1} A_0 = {}^k f((k + 1)\Delta t) \tag{12}$$

These conditions ensure that trajectory segments pass through both the common waypoints and the added designated waypoint.

At next, we introduce the velocity constraint meant to guarantee a smooth trajectory extension. We apply the minimal velocity constraint and may focus on the smoothness of the curve at time  $(k + 1)\Delta t$  because there is transferring point between the neighboring trajectory segments. Here, a primary differentiation of the  $k$ -th trajectory segment  ${}^k f(t)$  is expressed in below equation.

$${}^k \dot{f}(t) = \begin{bmatrix} {}^k \dot{x}(t) \\ {}^k \dot{y}(t) \end{bmatrix} = \sum_{i=1}^n i \cdot {}^k A_i t^{i-1} \in \mathbf{R}^2 \tag{13}$$

$$t \in [k\Delta t, (k + m)\Delta t] \tag{14}$$

The velocity constraint at time  $(k + 1)\Delta t$  between  $k$ -th and  $(k + 1)$ -th trajectory segments is expressed as follows.

$${}^k \dot{f}((k + 1)\Delta t) = {}^{k+1} \dot{f}((k + 1)\Delta t) \tag{15}$$

$$= \sum_{i=1}^n i \cdot {}^{k+1} A_i ((k + 1)\Delta t)^{i-1} \tag{16}$$

Another velocity constraint is to set the terminal velocity of every trajectory segment to zero. This conditions is introduced to realize to finish extending the trajectory arbitrarily as follows.

$$\begin{aligned} {}^{k+1} \dot{f}((k + m + 1)\Delta t) &= \sum_{i=1}^n i \cdot {}^{k+1} A_i ((k + m + 1)\Delta t)^{i-1} \\ &= O_2 \end{aligned} \tag{17}$$

where,  $O_2 \in \mathbf{R}^2$  is a zero vector. This guarantees the terminal velocity of each trajectory segment is zero.

By summarizing the above-mentioned constraints (8)–(12), (16) and (17), we obtain below equation.

$$H = D \cdot \begin{bmatrix} {}^{k+1}A_1 \\ {}^{k+1}A_2 \\ \vdots \\ {}^{k+1}A_n \end{bmatrix} \in \mathbf{R}^{2(m+3)} \tag{18}$$

where

$$H = \begin{bmatrix} {}^k f((k+1)\Delta t) - {}^{k+1}A_0 \\ {}^k f((k+2)\Delta t) - {}^{k+1}A_0 \\ \vdots \\ {}^k f((k+m)\Delta t) - {}^{k+1}A_0 \\ {}^{k+1}f((k+m+1)\Delta t) - {}^{k+1}A_0 \\ {}^k \dot{f}((k+1)\Delta t) \\ O_2 \end{bmatrix} \tag{19}$$

$$D = \begin{bmatrix} (k+1)\Delta t I_2 & \{(k+1)\Delta t\}^2 I_2 & \dots & \{(k+1)\Delta t\}^n I_2 \\ (k+2)\Delta t I_2 & \{(k+2)\Delta t\}^2 I_2 & \dots & \{(k+2)\Delta t\}^n I_2 \\ \vdots & \vdots & \vdots & \vdots \\ (k+m)\Delta t I_2 & \{(k+m)\Delta t\}^2 I_2 & \dots & \{(k+m)\Delta t\}^n I_2 \\ (k+m+1)\Delta t I_2 & \{(k+m+1)\Delta t\}^2 I_2 & \dots & \{(k+m+1)\Delta t\}^n I_2 \\ I_2 & 2\{(k+1)\Delta t\} I_2 & \dots & n\{(k+1)\Delta t\}^{n-1} I_2 \\ I_2 & 2\{(k+m+1)\Delta t\} I_2 & \dots & n\{(k+m+1)\Delta t\}^{n-1} I_2 \end{bmatrix} \tag{20}$$

$\in \mathbf{R}^{2(m+3) \times 2n}$

Here,  $I_2 \in \mathbf{R}^{2 \times d}$  is an identity matrix. From Eq. (12), we obtain  ${}^k f((k+1)\Delta t) - {}^{k+1}A_0 = O_2$  that is for first factor of  $H$ . The solution of Eq. (18) is derived as follows.

$$\begin{bmatrix} {}^{k+1}A_1 \\ {}^{k+1}A_2 \\ {}^{k+1}A_3 \\ \vdots \\ {}^{k+1}A_n \end{bmatrix} = \begin{cases} D^{-1} \cdot H & \text{for } n = m + 3 \\ D^+ \cdot H & \text{for } n > m + 3 \end{cases} \tag{21}$$

where  $D^+$  indicates the pseudo-inverse matrix of  $D$ . Here, we can obtain all coefficients of the  $(k+1)$ -th trajectory segment  ${}^{k+1}f(t), \{t \in [(k+1)\Delta t, (k+m+1)\Delta t]\}$  by these calculations.

### 4 Trajectory generation sequence

In this section, practical sequence for iterative trajectory generation is discussed. Algorithm 1 is for main routine and Algorithm 2 is a subroutine for calculating the next trajectory segment. In Algorithm 1, firstly initialization process are done by setting an initial trajectory segment  ${}^0 f(t)$ , to which  $k = 0$  is assigned.  $t_0$  indicates the given start

### Algorithm 1 Iterative trajectory extension algorithm

```

Main( $t_0$ )
{
  Initialization  $t \leftarrow t_0, k \leftarrow 0, state \leftarrow EXECUTE$ 
  Set initial trajectory segment  ${}^0 f(t)$  with  $n, m$ 
  Sample  $m$  waypoints at every  $\Delta t$  on the initial trajectory segment
  while  $state \neq STOP$  do
    if  $k == 0$  then
       $T \leftarrow t_0$ 
    else
       $T \leftarrow GetCurrentTime()$ 
    end if
     $j \leftarrow 0$ 
    while  $j < m$  do
       ${}^{k+1}f((k+j)\Delta t) \leftarrow {}^k f((k+j)\Delta t)$ 
       $j \leftarrow j + 1$ 
    end while
     ${}^{k+1}f((k+1)\Delta t) \leftarrow {}^k f((k+1)\Delta t)$ 
     ${}^{k+1}f((k+m+1)\Delta t) \leftarrow O_2$ 
    if DesignateWaypoint( $\cdot$ ) == NONE then
      state  $\leftarrow STOP$ 
      break
    else
       ${}^{k+1}f((k+m+1)\Delta t) \leftarrow DesignateWaypoint()$ 
       ${}^{k+1}f(t) = NextTrajectorySegment({}^k f(t), n, m)$ 
      check_result = CheckRestrictions( ${}^{k+1}f(t)$ )
       $\Delta\tau \leftarrow GetCurrentTime() - T$ 
    end if
    if check_result == NG then
      while  $\Delta\tau \leq \Delta t$  do
         ${}^{k+1}f((k+m+1)\Delta t) \leftarrow DesignateWaypoint()$ 
         ${}^{k+1}f(t) = NextTrajectorySegment({}^k f(t), n, m)$ 
        check_result = CheckRestrictions( ${}^{k+1}f(t)$ )
        if check_result == OK then
          break
        end if
      end while
       $\Delta\tau \leftarrow GetCurrentTime() - T$ 
    end while
    if check_result == OK then
       $k \leftarrow k + 1$ 
    else
      state  $\leftarrow STOP$ 
    end if
    while  $\Delta\tau \leq \Delta t$  do
       $\Delta\tau \leftarrow GetCurrentTime() - T$ 
    end while
  end while
}

```

### Algorithm 2 Calculation of next trajectory segment

```

NextTrajectorySegment( ${}^k f(t), n, m$ )
{
   $[{}^{k+1}A_1, {}^{k+1}A_2, \dots, {}^{k+1}A_n]^T \leftarrow \begin{cases} D^{-1} \cdot H & \text{for } n = m + 3 \\ D^+ \cdot H & \text{for } n > m + 3 \end{cases}$ 
   ${}^{k+1}A_0 \leftarrow {}^k f((k+1)\Delta t)$ 
   ${}^{k+1}f(t) \leftarrow \sum_{i=0}^n {}^{k+1}A_i(t)^i$ 
  return  ${}^{k+1}f(t)$ 
}

```

time of the calculation and  $t_0 = 0$  is set for first start. The trajectory extension calculation is done every time period  $\Delta t$ . After getting  $m$  waypoints from the initial trajectory segment, the extension process starts. It is checked whether the process is just started utilizing  $k$  and the base time is set.  $GetCurrentTime()$  is a function for getting current time. The sampled waypoints on  ${}^k f(t)$  are diverted as the waypoints on  ${}^{k+1}f(t)$ . Additionally, the velocity

constraints at  $t = (k + 1)\Delta t$  and  $t = (k + m + 1)\Delta t$  are also given. For providing a designated waypoint for trajectory extension, a function *DesignateWaypoint*( ) is called. In this paper, it is not the main topic how to determine the waypoint and we assume that a waypoint is provided by *DesignateWaypoint*( ). *DesignateWaypoint*( ) returns the value of “NONE” when more trajectory extension is not required. In such case, the trajectory ends at the terminal point of  $k$ -th trajectory segment. When *DesignateWaypoint*( ) returned the waypoint, it is set as  ${}^{k+1}f((k + m + 1)\Delta t)$ . Here, all constraints for deriving  ${}^{k+1}f(t)$  ( $t \in [(k + 1)\Delta t, (k + m + 1)\Delta t]$ ) are prepared and a function *NextTrajectorySegment*( ${}^k f(t), n, m$ ) which is described in Algorithm 2 is called for calculating the coefficient vectors  ${}^{k+1}A_0, {}^{k+1}A_1, {}^{k+1}A_2, \dots, {}^{k+1}A_n$  of  ${}^{k+1}f(t)$ . Here, it is done to generate  ${}^{k+1}f(t)$  with the coefficient vectors. Mobile robot generally has the kinematic restrictions for its practicable motion. *DesignateWaypoint*( ) provides a waypoint however generated trajectory segment must meet such restrictions. *CheckRestrictions*( ${}^{k+1}f(t)$ ) is called for checking to meet the criteria. Restrictions for the mobile robot are expressed as follows.

$$\arg \max_{(k+1)\Delta t \leq t \leq (k+m+1)\Delta t} |{}^{k+1}\dot{f}(t)| \leq V_{\max} \tag{22}$$

$$\arg \max_{(k+1)\Delta t \leq t \leq (k+m+1)\Delta t} \left| \frac{d {}^{k+1}\theta(t)}{ds} \right| \leq C_{\max} \tag{23}$$

where,

$$|{}^{k+1}\dot{f}(t)| = \left\{ \{ {}^{k+1}\dot{x}(t) \}^2 + \{ {}^{k+1}\dot{y}(t) \}^2 \right\}^{\frac{1}{2}} \tag{24}$$

$$\left| \frac{d {}^{k+1}\theta(t)}{ds} \right| = \left| \frac{{}^{k+1}\dot{x}(t){}^{k+1}\ddot{y}(t) - {}^{k+1}\ddot{x}(t){}^{k+1}\dot{y}(t)}{\{ {}^{k+1}\dot{x}(t) \}^2 + \{ {}^{k+1}\dot{y}(t) \}^2} \right|^{\frac{3}{2}} \tag{25}$$

$s$  is the distance along trajectory segment from  $(k + 1)\Delta t$  to  $(k + m + 1)\Delta t$ .  ${}^{k+1}\theta(t)$  is the posture along  $(k + 1)$ -th trajectory segment and  $\frac{d {}^{k+1}\theta(t)}{ds}$  is the curvature value.  $V_{\max}$  and  $C_{\max}$  are the maximum values for the motion velocity and the curvature. When these are not satisfied, derived  $(k + 1)$ -th trajectory segment is not qualified. *check\_result* that is the return value from *CheckRestrictions*( ${}^{k+1}f(t)$ ) gets “OK” when  ${}^{k+1}f(t)$  was qualified, otherwise it returns “NG” to *check\_result*. In the case of *check\_result* == “OK”,  $k$  is increased by 1 and then the calculation proceeds to the next turn. If  $\Delta t$  is enough larger than calculation time of the trajectory segment, there is the possibility that several re-calculations with the other waypoints to find the qualified one can be acceptable. Therefore, in this sequence, processing

time  $\Delta \tau$  for finding  $(k + 1)$ -th trajectory segment are measured to check whether there is still the margin for time to calculate with the other designated waypoints. When  ${}^{k+1}f(t)$  was found after the trials within the time period  $\Delta \tau$ , the value *check\_result* is set to “OK”. This means to leave this seeking process and proceed to the next turn by incrementing  $k$ . In case of that selected candidates of the waypoints can not qualify the criteria during  $\Delta t$ , it means that the trajectory extension process is once suspended by *state*  $\leftarrow$  STOP. This means that the robot stops at the end of the current trajectory segment. After stopping, it can take much time to find the way to go and it is possible to restart to generate the trajectory when the way was found. When *state* becomes “STOP”, this algorithm is once stopped. For restarting the extension process after that,  $t_0$  is set to the current time and Algorithm 1 is called. By this sequence, the trajectory including temporal stoppage can be designed by utilizing proposed trajectory segment calculation.

### 5 Simulation experiments

Here, we performed computer simulations. The waypoints are provided manually and the criteria  $V_{\max} = 0.4$  [m/sec] and  $C_{\max} = 5.0$  were set in following cases.

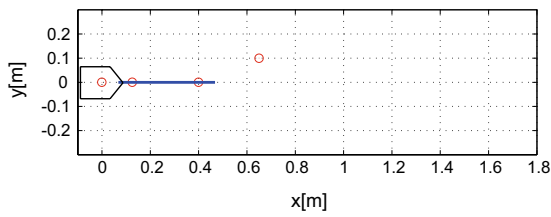
#### 5.1 Forward motion

For Case 1,  $\Delta t = 2.0$ [s] and  $m = 2$  and also below initial trajectory segment ( $n = 5, k = 0$ ) were set.

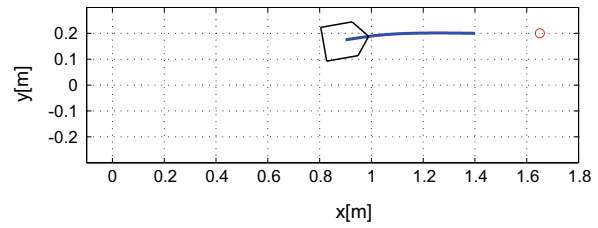
$${}^0f(t) = \sum_{i=0}^5 {}^0A_i t^i \quad \{ t \in [0, 4] \} \tag{26}$$

$$= \begin{bmatrix} 1/40 \\ 0 \end{bmatrix} t^3 + \begin{bmatrix} -3/640 \\ 0 \end{bmatrix} t^4 \tag{27}$$

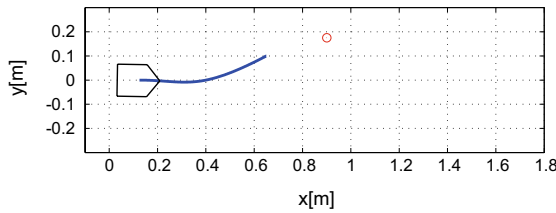
The waypoints on the this initial trajectory segment are selected as  $[0.0, 0.0]^T$  at  $0.0$ [s],  $[0.125, 0.0]^T$  at  $2.0$ [s] and  $[0.4, 0.0]^T$  at  $4.0$ [s]. The velocity vector at the end points of the trajectory segment become  $[0.0, 0.0]^T$  To execute a trajectory extension based on this initial trajectory, we assigned a designated waypoint to each trajectory segment. For Case 1, we prepared designated waypoints  $[0.65, 0.1]^T, [0.9, 0.175]^T, [1.15, 0.2]^T, [1.4, 0.2]^T$  and  $[1.65, 0.2]^T$  for the 1st, 2nd, 3rd, 4th, and 5th trajectory segments, respectively. Figure 2a shows the above-mentioned initial trajectory. Figure 2b, c, d, e, and f also show extension results for each designated waypoint. Circles indicate the locations of designated waypoints. of the trajectory segment had overlapped duration with the next trajectory segment. For example, initial trajectory segment during



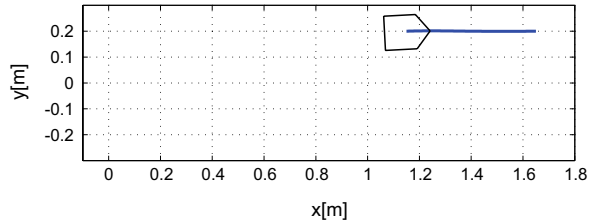
(a) Initial trajectory ( $n = 5, \Delta t = 2.0, m=2$ ) : designated waypoint  $[0.65, 0.1]^T$



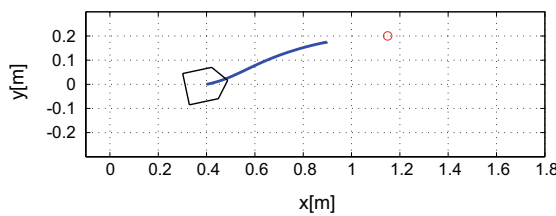
(e) 4th trajectory segment,  $t \in [8.0, 12.0]$  : designated waypoint  $[1.65, 0.2]^T$



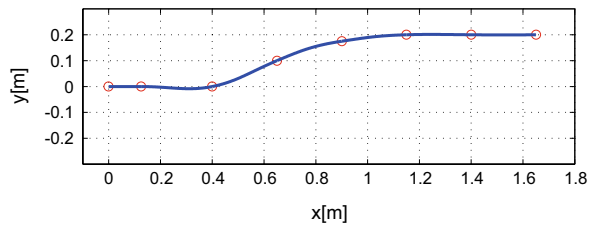
(b) 1st trajectory segment,  $t \in [2.0, 6.0]$  : designated waypoint  $[0.9, 0.175]^T$



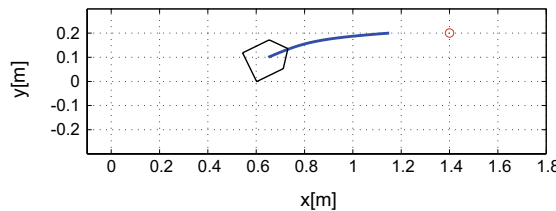
(f) 5th trajectory segment,  $t \in [10.0, 14.0]$



(c) 2nd trajectory segment,  $t \in [4.0, 8.0]$  : designated waypoint  $[1.15, 0.2]^T$



(g) Resultant trajectory,  $t \in [0.0, 14.0]$ , in Case 1



(d) 3rd trajectory segment,  $t \in [6.0, 10.0]$  : designated waypoint  $[1.4, 0.2]^T$

**Fig. 2** Simulation results: case 1

$t \in [0, 2.0]$  was used to organize the trajectory, and the one during  $t \in [2.0, 4.0]$  was later renewed by 1-st trajectory segment; the 1-st trajectory segment during  $t \in [2.0, 4.0]$  was also used to organize the trajectory. The following trajectory segments were also derived in a similar manner. By this procedure, the resultant trajectory ( $t \in [0.0, 14.0]$ ) shown in Fig. 2g was provided and pass through all designated waypoints.

For Case 2 (Fig. 3), the parameters for extension and the initial trajectory was the same with Case 1 ( $\Delta t = 2.0[s], m = 2$  and Eq. (26)). Another set of designated waypoints  $[0.6, -0.12]^T, [0.775, -0.25]^T, [1.0, -0.225]^T, [1.15, -0.04]^T, [1.125, 0.175]^T, [1.0, 0.32]^T$  and  $[0.78, 0.35]^T$  was provided. Figure 3 indicates generated trajectory segments and its development procedure. Figure 3g shows the resultant trajectory ( $t \in [0.0, 18.0]$ ).

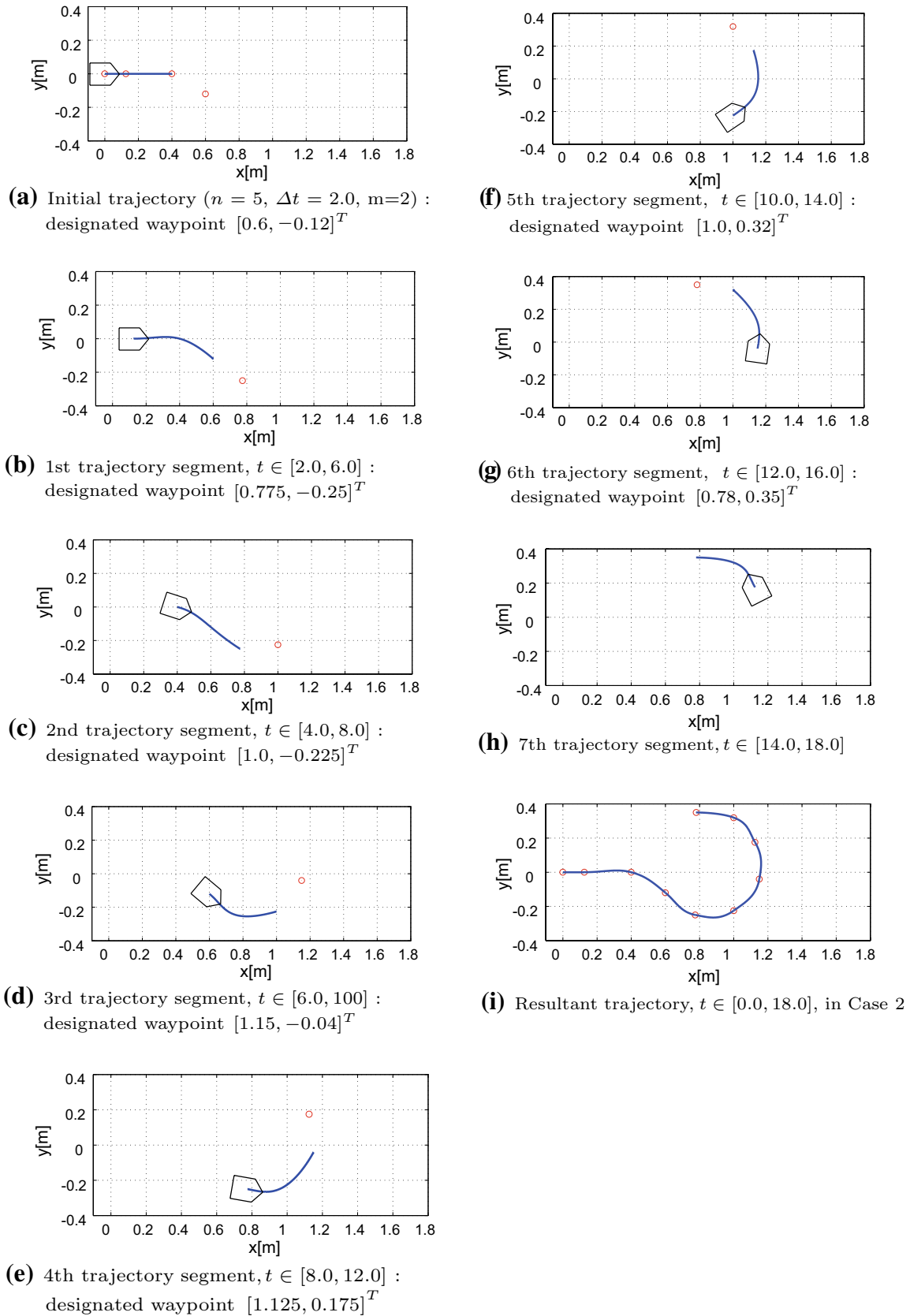


Fig. 3 Simulation results: case 2

The set of designated waypoints was only different from Case 1, but the different smooth trajectory was generated.

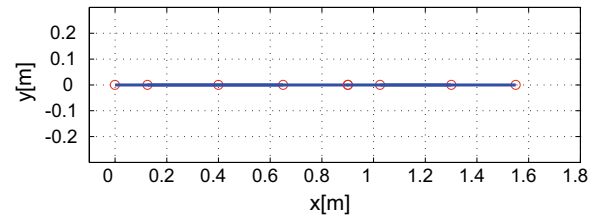
### 5.2 Stop-and-go motion and back-up motion

As a trial for the other type of the trajectory generation, it was done to verify to generate the “stop-and-go” trajectory by proposed method. By proposed method, it is possible to stop once at the end of the current trajectory segment when the robot must be stop intentionally by some reasons or the qualified trajectory segment can not be found during  $\Delta t$ . Case 3 (Fig. 4) shows the result of a trajectory for “stop-and-go” motion. Figure 4a shows the resultant trajectory and Fig. 4b indicates time development of the trajectory and the velocity. Solid line in the graphs of Fig. 4b indicates the position or velocity in the  $x$  axis and dotted line indicates the position or velocity in the  $y$  axis. Initial trajectory was the same with Eq. (26) again. We also set the waypoints  $[0.65, 0.0]^T$ ,  $[0.9, 0.0]^T$  for first phase of the extension. After designating the waypoint  $[0.9, 0.0]^T$  for 2nd one, the waypoint was not provided. Therefore, after reaching to  $[0.9, 0.0]^T$  at  $t = 8.0[s]$ , the extension process was interrupted. Thus, the trajectory stayed at  $[0.9, 0.0]^T$  temporary for  $2.0[s]$ . During  $t \in [8.0, 10.0]$ , the trajectory was not evolved and the velocity became zero. As second phase of the extension, when  $t = 10.0$ , below initial trajectory segment for restart was set.

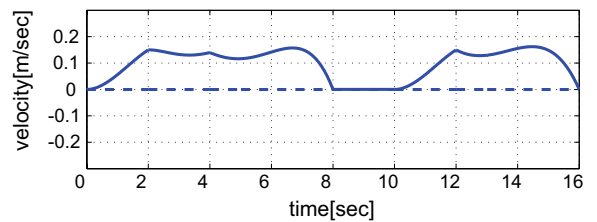
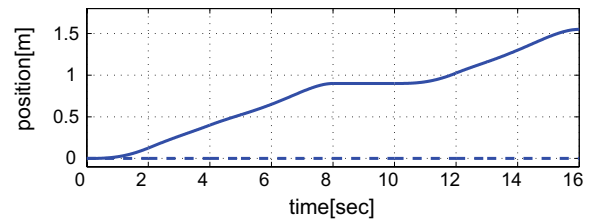
$${}^4f(t) = \sum_{i=0}^5 {}^4A_i(t - 10)^i \quad \{t \in [10, 14]\} \quad (28)$$

$$= \begin{bmatrix} 0.9 \\ 0 \end{bmatrix} + \begin{bmatrix} 1/40 \\ 0 \end{bmatrix} (t - 10)^3 + \begin{bmatrix} -3/640 \\ 0 \end{bmatrix} (t - 10)^4 \quad (29)$$

This is 4th trajectory segment and has the same curve shape with the initial trajectory segment. The waypoint  $[1.05, 0.0]^T$  was designated for restarting the second phase of the trajectory. After that, the trajectory extension process was done with designated waypoints and it was terminated at  $[1.55, 0.0]^T$  again. This case shows that the trajectory extension is ended spontaneously by interrupting to designate new waypoint. It is because the condition of Eq. (17) was introduced. By the condition, the velocity vector at the end point of every trajectory segment becomes  $O_d$ . When the trajectory extension proceeds, the velocity vector at  $(k + 1)\Delta t$  is forwarded to generate the next trajectory segment and the velocity profile is renewed. From Fig. 4b, the behavior of the velocity was emerged by only designating the waypoints and it could be confirmed that the velocity was transferred between  $k$ -th and  $(k + 1)$ -th trajectory segment at every  $\Delta t$ . As the result of Case 3, we verified that the “stop-and-go” trajectory could be generated.



(a) Resultant trajectory,  $t \in [0.0, 16.0]$ , in Case 3



(b) Time development of the trajectory and the velocity

Fig. 4 Simulation results: case 3

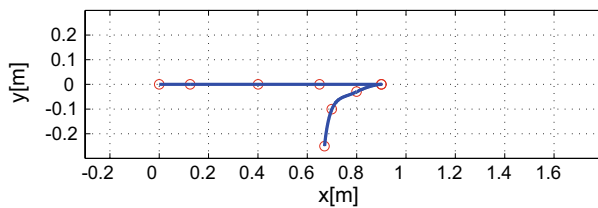
One more important behavior of the mobile robot is back-up motion. Here, we also attempt to verify to generate the trajectory of back-up motion. The same initial conditions with Case 3 are utilized. The waypoints  $[0.65, 0.0]^T$  and  $[0.9, 0.0]^T$  are designated for first phase of the trajectory extension. After designating the waypoint  $[0.9, 0.0]^T$ , the next waypoint was not provided and the trajectory was suspended at  $[0.9, 0.0]^T$  from  $t = 8.0[s]$  to  $t = 10.0[s]$ . After that, second phase of trajectory extension restarts by setting the below trajectory segment at  $t = 10.0[s]$ .

$${}^4f(t) = \sum_{i=0}^5 {}^4A_i(t - 10)^i \quad \{t \in [10, 14]\} \quad (30)$$

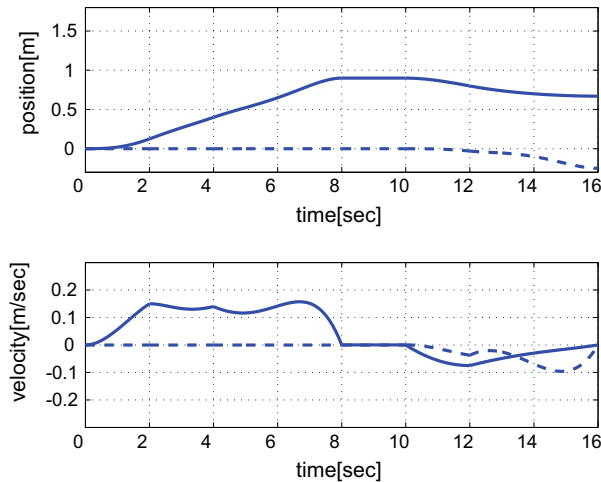
$$= \begin{bmatrix} 0.9 \\ 0 \end{bmatrix} + \begin{bmatrix} -3/80 \\ 1/800 \end{bmatrix} (t - 10)^2 + \begin{bmatrix} 1/160 \\ -11/1600 \end{bmatrix} (t - 10)^3 + \begin{bmatrix} 0 \\ 1/800 \end{bmatrix} (t - 10)^4 \quad (31)$$

Then, the waypoint  $[0.67, -0.25]^T$  is designated and the trajectory extension was proceeded until  $t = 16.0[s]$ .





(a) Resultant trajectory,  $t \in [0.0, 16.0]$ , in Case 4



(b) Time development of the trajectory and the velocity

**Fig. 5** Simulation results: case 4

Figure 5a shows the resultant trajectory for a back-up motion and Fig. 5b shows the behavior of the position and the velocity in  $x$  and  $y$ .

Moreover, from these simulation results, it was also confirmed that the velocity at the end point of each trajectory segment becomes  $[0.0, 0.0]^T$ . This characteristic provided by proposed method is useful that the mobile robot can stop traveling at the end point of the current trajectory segment without adding any special conditions and calculations. Some of these trajectory are difficult to express with a single polynomial and it is an advantage of proposed method based on iterative extension concept.

## 6 Conclusions

In this paper, we described an online trajectory extension method for mobile robots. Our approach iteratively generates the next trajectory segment based on constraints from the existing trajectory segments and additional designated waypoints. It utilizes a polynomial curve as the basis of trajectory segments and derives the coefficients of the curve iteratively. The whole trajectory is resultantly organized by

combining these generated trajectory segments. The results of computer simulations show that the proposed method accomplishes trajectory extension by depending on the constraints of the existing trajectory segment and designated waypoint.

## References

1. Sakamoto N, Okugawa M (2012) Human following control of porter robot with velocity vectors. In: Proceedings of ASME 2012 5th annual dynamic systems and control conference joint with the JSME 2012 11th motion and vibration conference, pp 817–821
2. Tachi S, Tanie K, Komoriya K, Abe M (1985) MELDOG: the guide dog robot. *IEEE Trans Biomed Eng* 32(7):256–270
3. Karreman DE, van Dijk EMAG, Evers V (2012) Using the visitor experiences for mapping the possibilities of implementing a robotic guide in outdoor sites. In: Proceedings of the 21st IEEE international symposium on robot and human interactive communication, pp 1059–1065
4. Zacharie M (2010) Security guard robot detecting human using Gaussian distribution histogram method. *J Comput Sci* 6(10):1144–1150
5. Lee H-T, Lin W-C, Huang C-H (2011) Indoor surveillance security robot with a self-propelled patrolling vehicle. *J Robot Vol* 2011. Article ID 197105. doi:10.1155/2011/197105
6. Kelly A, Nagy B (2003) Reactive nonholonomic trajectory generation via parametric optimal control. *Int J Robot Res* 22(7–8):583–601
7. Maekawa T, Noda T, Tamura Ozaki T, Machida K (2010) Curvature continuous path generation for autonomous vehicle using B-spline curves. *Comput Aided Des* 42:350–359
8. Bezier P (1986) *Courbes et surfaces*. Mathmatiques et CAO, Hermes, Paris
9. Jolly KG, Kumar RS, Vijayakumar R (2009) A Bezier curve based path planning in a multi-agent robot soccer system without violating the acceleration limits. *Robot Auton Syst* 57:23–33
10. Ma L, Yang J, Zhang M (2012) A two-level path planning method for on-road autonomous driving. In: Proceedings of international conference on intelligent system design and engineering application, pp 661–664
11. Kawabata K, Ma L, Xue J, Zheng N (2015) A path generation for automated vehicle based on bezier curve and via-points. *Robot Auton Syst* 74:243–252
12. Choi JW, Curry RE, Elkaim GH (2010) Continuous curvature path generation based on bezier curves for autonomous vehicles. *IAENG J Appl Math* 40(2):IJAM\_40\_2\_07
13. Zhou F, Song B, Tian G (2011) Bezier curve based smooth path planning for mobile robot. *J Inf Comput Sci* 8(12):2441–2450
14. LaValle SM, Kuffner JJ (2001) Randomized kinodynamic planning. *Int J Robot Res* 20(5):378–400
15. Melchior NA, Simmons R (2007) Particle RRT for path planning with uncertainty. In: Proceedings of 2007 IEEE international conference on robotics and automation, pp 1617–1624
16. Kuwata Y, Teo J, Fiore G, Karaman S, Frazzoli E, How JP (2009) Real-time motion planning with applications to autonomous urban driving. *IEEE Trans Control Syst Technol* 17(5):1105–1118
17. Yang K, Moon S, Yoo S, Kang J, Doh NL, Kim NB, Joo S (2014) Spline-based RRT path planner for non-holonomic robots. *J Intell Robot Syst* 73:763–782
18. Karaman S, Walter MR, Perez A, Frazzoli E, Teller S (2011) Anytime motion planning using the RRT. In: Proceedings of

- IEEE international conference on robotics and automation, pp 1478–1483
19. Paromtchik I, Asama H (2000) A motion generation approach for an omnidirectional vehicle. In: Proceedings of international conference on robotics and automation, pp 1213–1218
  20. Kawabata K, Xue J, Ma L, Zheng N (2014) A sequential path extension method for mobile robot. Proc IEEE TENCON 2014:075
  21. Kawabata K, Xue J, Ma L, Yokota S, Mitsukura Y, Zheng N (2015) Iterative polynomial-based trajectory extension for mobile robot. In: Proceedings of 2015 IEEE international conference on advanced intelligent mechatronics, pp 255–260