Motion Planning for Multiple Mobile Robots Using Time-Scaling

István Komlósi and Bálint Kiss
Department of Control Engineering and Information Technology
Budapest University of Technology and Economics
Hungary

1. Introduction

Recent developments in robotics have raised the interest in mobile robots due to their wide range of applications requiring high level of autonomy of individual mobile agents. This increased level of autonomy needs more sophisticated path-planning and motion control methods, which have been studied extensively in the recent years [10].

This paper focuses on a typical industrial application in which several mobile robots share the same workspace. These units may carry materials or be capable of carrying out various repair jobs at some locations, etc. Due to the tight schedules to achieve high productivity in such applications, it is imperative to synchronize the motion of the units so that they arrive at the required location on time without collision with obstacles or with other units. Coordinated planning of such multi-robot systems is addressed in the literature based on some search methodology [7] or using neural network based techniques [8].

Since a high number of units can be involved, it may not be possible to design collision free trajectories for all robots, i.e. avoiding each other and all static obstacles cannot be guaranteed only by the path geometry. Instead of constructing trajectories with complex geometries so that they have only a few or no intersections at all, one can design simple paths which disregard moving obstacles (i.e. other robots) and take only the static obstacles into account, so the path geometries are supposed to be fixed for each robot. Such motion planning algorithms are readily available in the literature [1,3,4] for simple polygonal obstacle representations. This initial path geometry designed and assigned to each unit ensures the avoidance of static obstacles.

The avoidance of the moving obstacles can be achieved by choosing a suitable velocity profile along the fixed-geometry trajectories. In order to obtain the proper velocity input, we assign first a default velocity profile which the unit would use without the dynamic obstacles. Such a profile can be designed to be optimal in some sense (minimal time or cost) and may satisfy additional constraints, if necessary [2]. The default velocity profile consists of intervals with constant or zero acceleration. Acceleration parameters change at the boundaries of these intervals. This default velocity profile is then modified such that a unit decelerates or accelerates its motion to let other units pass through potential collision areas (identified from the path geometries).

We suppose that a priority level is assigned to each robot so that a unit with a given priority regards all units with higher priority level as dynamic obstacles. The ranking of the units is

arbitrary, e.g. faster units may receive higher priority in order not to slow them down. Other considerations can be also incorporated in the choice of priority order. The velocity distributions of the robots are thus determined in a serial way according to the priority ordering of the robots. Recall that one may get a globally better solution in terms of the overall time used to complete all trajectories if the priority order of the robots is relaxed but such a relaxation would imply the need of a high level search algorithm. The study of such a high level search algorithm is beyond the scope of this paper.
The velocity tuning method presented in this paper for an individual robot is based on time-scaling. The default time-scale determined by the default velocity profile is parameterized by a *virtual time parameter* which has to be then mapped to the real or global time such that no collision occurs. The mapping between the virtual and global time parameters is referred to as the time-scaling in the sequel. The time-plane is the space spanned by the virtual and global time parameters.
If a collision is possible between two units according to the path geometries (so that they intersect each other), the collision to be avoided can be converted into a so called static time-obstacle in the time-plane [5,6]. A mapping avoiding the static time-obstacles shall be used to define the time-scaling of the path of the current unit. Once the path is expressed by the global time parameter, this procedure can be repeated for all units down to the one with the lowest priority.
The remaining part of the paper is organized as follows. Section 2 presents an overview on time-scaling, Section 3 presents our path-planning method in the time-plane, Implementation results are presented in Section 4 and Conclusions are drawn in Section 5.

2. An overview on time-scaling

In this section some mathematical background is given for the time-scaling. The default velocity that is assigned to a robot and so the time distribution of its path is expressed using a virtual time parameter τ. We denote this reference path by $\Gamma(\tau)$ and define the default velocity as

$$\frac{d}{d\tau}\Gamma(\tau) = v(\tau) \tag{1}$$

The time distribution of the reference trajectory is constructed regardless to any moving obstacles. Motion of dynamic obstacles (i.e. other robots) is expressed with the global time parameter t, thus if $O_{dyn}(t)$ represents the trajectory of a dynamic obstacle and

$$\Gamma(\tau) \cap O_{dyn}(t) \neq \emptyset \tag{2}$$

then the current unit collides with the dynamic obstacle. To avoid this collision we must find a function $\tau = \theta(t)$ that alters the time distribution of the reference path such that

$$\Gamma(\theta(t)) \cap O_{dyn}(t) = \emptyset \tag{3}$$

The function $\theta(t)$ maps the virtual time values to global or real time. A mapping that ensures avoidance of all dynamic obstacles will be the time-scaling function. Such a time-scaling function may be constructed from several functions over different time intervals.

Methods for finding an adequate time-scaling function in our application will be presented in the next section. Based on the real time and virtual time parameters, we can define the $t-\tau$ time-plane. If a collision is possible between two units according to the path geometries, the collision to be avoided can be converted into a time-obstacle O_{time} which appears as a static obstacle in the time-plane. A time-obstacle is assigned to a collision area in the workspace and it is the set of those virtual and real time pairs where any parts of the unit and the moving obstacle are located at the same place within the collision area corresponding to the time-scaled units default velocity. In most of the cases they will appear as rectangular shaped time-obstacles; in special cases, however, they might be enclosed by nonlinear curves.

Avoiding dynamic obstacles in the workspace is analogous to avoiding the corresponding time-obstacles on the time-plane. Based on this analogy, the time-scaling function is the path on the time-plane that avoids all time-obstacles. After constructing this path, the scaled reference path can be generated. Since $\tau = \theta(t)$ we have

$$\frac{d}{dt}\Gamma(\tau) = \frac{d\Gamma(\tau)}{d\tau}\frac{d\tau}{dt} \tag{4}$$

which, using (1), yields to

$$\frac{d}{dt}\Gamma(\tau) = v(\tau)\dot{\theta}(t) \tag{5}$$

The velocity according to the scaled reference path can be generated by multiplying the original speed by the derivative of the time-scaling function w.r.t. the real time

$$\tilde{v}(t) = v(\theta(t))\dot{\theta}(t) \tag{6}$$

where $\tilde{v}(t)$ is the scaled velocity.

2.1 Criteria for the time-scaling function

A proper time-scaling function must not only avoid the time-obstacles, but must also satisfy criteria according to the kinematic constraints prescribed for the control (velocity) inputs. A collision-free course requires a time-scaling function not mapping any time value into a time-obstacle, which means that $(t,\theta(t)) \notin O_{time}$ must be satisfied for all time values and all time-obstacles.

Since time cannot rewind and we allow the robot to stop only at its final position, the scaling function $\theta(t)$ must be strictly monotonically increasing. It might also be required that the robot's velocity does not decrease under a specified minimal value except at the start and goal positions, where the velocity is assigned to be zero. We assume that the default velocity is the maximal possible speed along the trajectory and its values cannot be exceeded at any time. This implies that the scaled velocity must not be greater at any real time instance than the velocity at the corresponding virtual time instance. It is also assumed that the acceleration and the deceleration are also bounded both in negative and positive directions. The acceleration (deceleration) limit is considered to be independent of time. These aforementioned constraints define the inequalities

$$v_{\min} \le \tilde{v}(t) \le v(\tau) \tag{7}$$

which, together with (6), results in

$$v_{\min} \le v(\tau)\dot{\theta}(t) \le v(\tau) \tag{8}$$

for the scaled velocity where $v_{\min}$ is the specified minimum speed value. After dividing with $v(\tau)$ it follows that

$$\frac{v_{\min}}{v(\tau)} \le \dot{\theta}(t) \le 1 \tag{9}$$

which defines a global upper bound for the time-scaling function meaning that the $\tau = t$ curve in the time-plane must not be intersected. The lower bound will be the solution of the differential equation

$$\dot{\theta}(t) = \frac{v_{\min}}{v(\tau)} \tag{10}$$

The constraint on the acceleration reads

$$a_{\min} \le \frac{d}{dt}\tilde{v}(t) \le a_{\max} \tag{11}$$

where $a_{\max}$ and $a_{\min}$ are the admissible maximal positive and negative acceleration values. Since

$$\frac{d}{dt}\tilde{v}(t) = \frac{d^2\tau}{dt^2}v(\tau) + a(\tau)\left(\frac{d\tau}{dt}\right)^2 \tag{12}$$

one gets

$$a_{\min} \le v(\tau)\ddot{\tau} + a(\tau)\dot{\tau}^2 \le a_{\max} \tag{13}$$

The solutions for the maximal values will result in two curves which will be used by the path planning algorithms in the time-plane.

2.2 Construction of the time-obstacles

A time-obstacle is always assigned to a collision area around the intersection of two geometric paths. Each collision area along the path of one unit generates a time-obstacle. The time-obstacle assigned to a collision area is the set of those virtual-time and real-time pairs where any parts of two units are located at the same place. We have to take into account the real physical dimensions of the units, which means that the entering time to a collision area is when the front of the unit arrives at the boundary of the collision area, and the exit time is when its rear point leaves it. If the time-scaled unit cannot enter the area until the higher priority unit has passed through, the time-obstacle will be rectangle shaped with vertexes $(t_{enter}, \tau_{enter})$, (t_{enter}, τ_{exit}), (t_{exit}, τ_{exit}), (t_{exit}, τ_{enter}) where τ_{enter} and τ_{exit} are the time values of entering and exiting the collision area of the time-scaled unit, while t_{enter} and t_{exit}

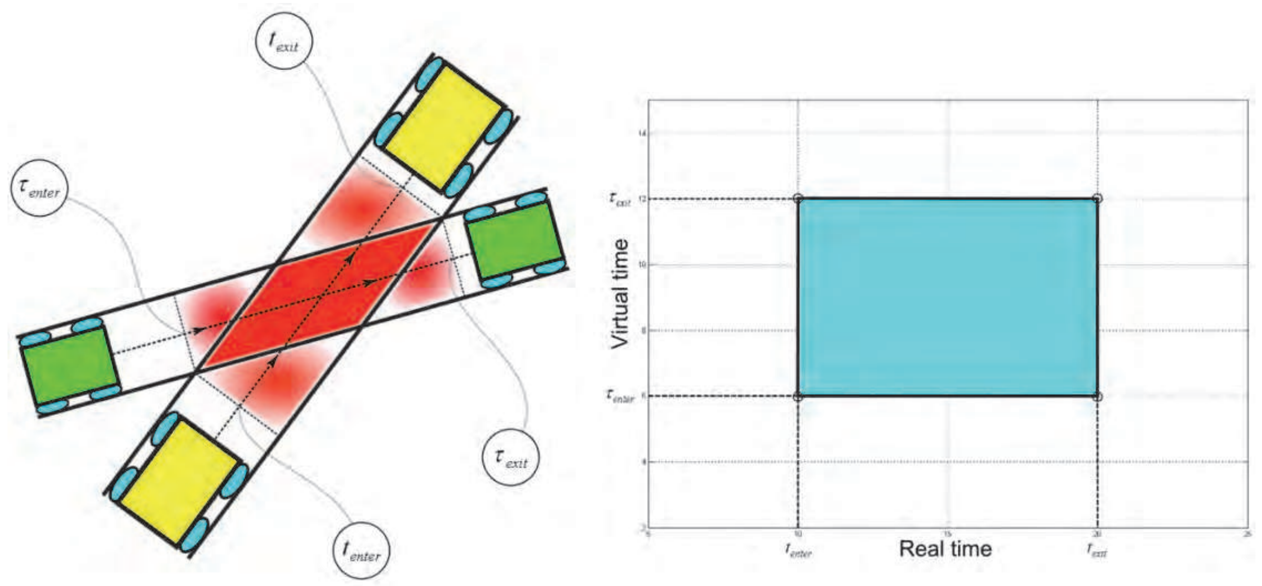


Fig. 1. Colliding units in the workspace (on the left) and the time-obstacle of the colliding units (on the right)

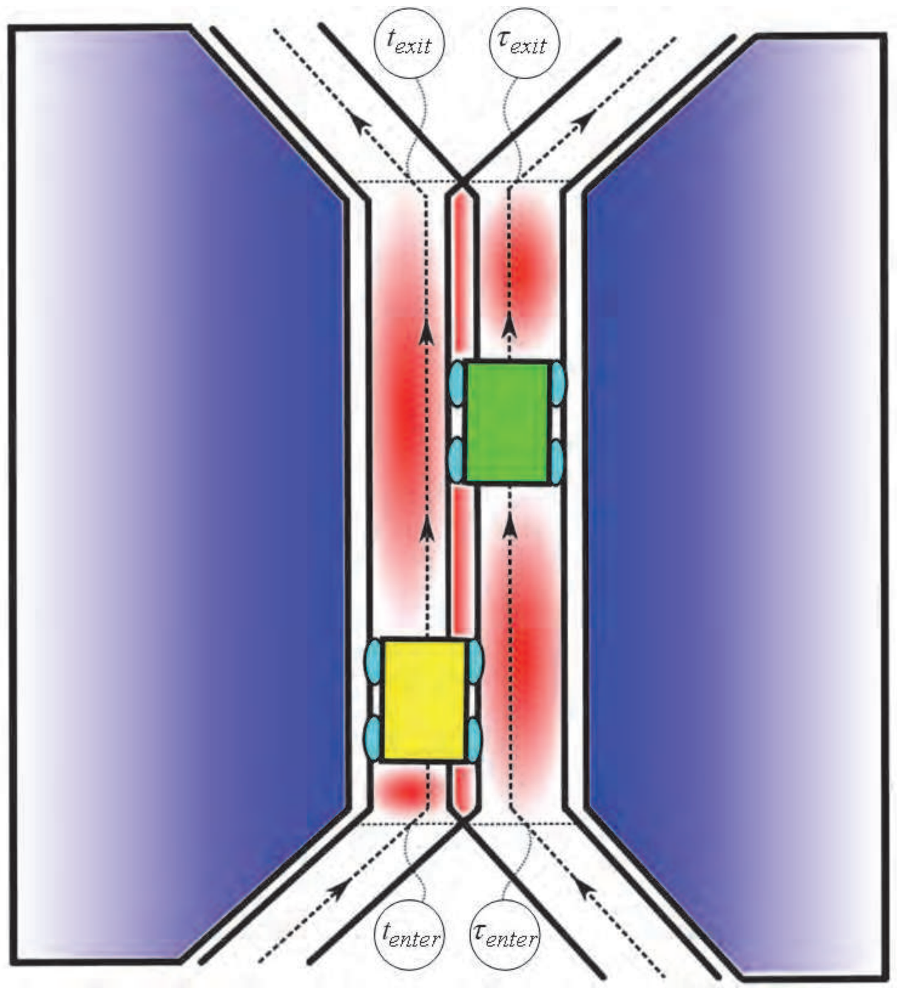


Fig. 2. Colliding units in a corridor (in the workspace)

are the similar values for the given dynamic obstacle. Fig. 1 shows a collision situation in the workspace and on the time plane.
In some special cases the path geometries may have common sections (e.g. the units follow each other in a narrow corridor). In such cases, when the motion directions of the colliding units are identical, one has to examine the time-values when certain parts of the collision area are free for the scaled unit. In this case the time-obstacle is enclosed by nonlinear curves. Fig. 2 shows an example of two robots proceeding along the same way in a narrow corridor and Fig. 3 shows the time-obstacle representation of the collision area.
Suppose that we have accelerating units in the corridor. Equation (14) defines the area which is occupied by both units at the same time instance:

$$\frac{1}{2}a_{unit}(\tau-\tau_{enter})^2+v_{unit}(\tau-\tau_{enter})+\Gamma(\tau_{enter})=\frac{1}{2}a_{obs}(t-t_{enter})^2+v_{obs}(t-t_{enter})+O_{dyn}(t_{enter}) \quad (14)$$

where $\Gamma(\tau_{enter})\in O_{dyn}(t_{enter})$. From (14) one can derive the equations of the curves that enclose the time-obstacles in case of accelerating motions (narrow corridor case).

$$\tau(t)=\tau_2-\frac{v_{unit}}{a_{unit}}+\frac{1}{a_{unit}}\sqrt{v_{unit}^2+2v_{obs}a_{unit}(t-t_1)+a_{obs}a_{unit}(t-t_1)^2} \quad (15a)$$

$$\tau(t)=\tau_1-\frac{v_{unit}}{a_{unit}}+\frac{1}{a_{unit}}\sqrt{v_{unit}^2+2v_{obs}a_{unit}(t-t_2)+a_{obs}a_{unit}(t-t_2)^2} \quad (15b)$$

In (15a) the fronts of the units arrive at the border of the corridor at τ_1 and t_1 , respectively, while their rear sides arrive to the same location at τ_2 and t_2.
In the simpler case, when the time-scaled unit moves with constant speed, Equation (16) defines the points of the time-obstacle which is enclosed by the curves defined by Equations (17a) and (17b).

$$v_{unit}(\tau-\tau_{enter})+\Gamma(\tau_{enter})=\frac{1}{2}a_{obs}(t-t_{enter})^2+v_{obs}(t-t_{enter})+O_{dyn}(t_{enter}) \quad (16)$$

$$\tau(t)=\tau_1+\frac{v_{obs}}{v_{unit}}(t-t_2)+\frac{1}{2}\frac{a_{obs}}{v_{unit}}(t-t_2)^2 \quad (17a)$$

$$\tau(t)=\tau_2+\frac{v_{obs}}{v_{unit}}(t-t_1)+\frac{1}{2}\frac{a_{obs}}{v_{unit}}(t-t_1)^2 \quad (17b)$$

In (15a), (15b), (17a) and (17b) the fronts of the units arrive at the border of the corridor at τ_1 and t_1 , respectively, while their rear sides arrive at the same location at τ_2 and t_2. Similarly at τ_3 and t_3 the front of the units arrive at the exit of the corridor, respectively, while their rear sides arrive at the same location at τ_4 and t_4. A nonlinear time-obstacle has thus six vertices (see Fig. 3).
In the sequel we focus mainly on rectangle shaped time-obstacles but the results can be extended for obstacles presented in Fig. 3.

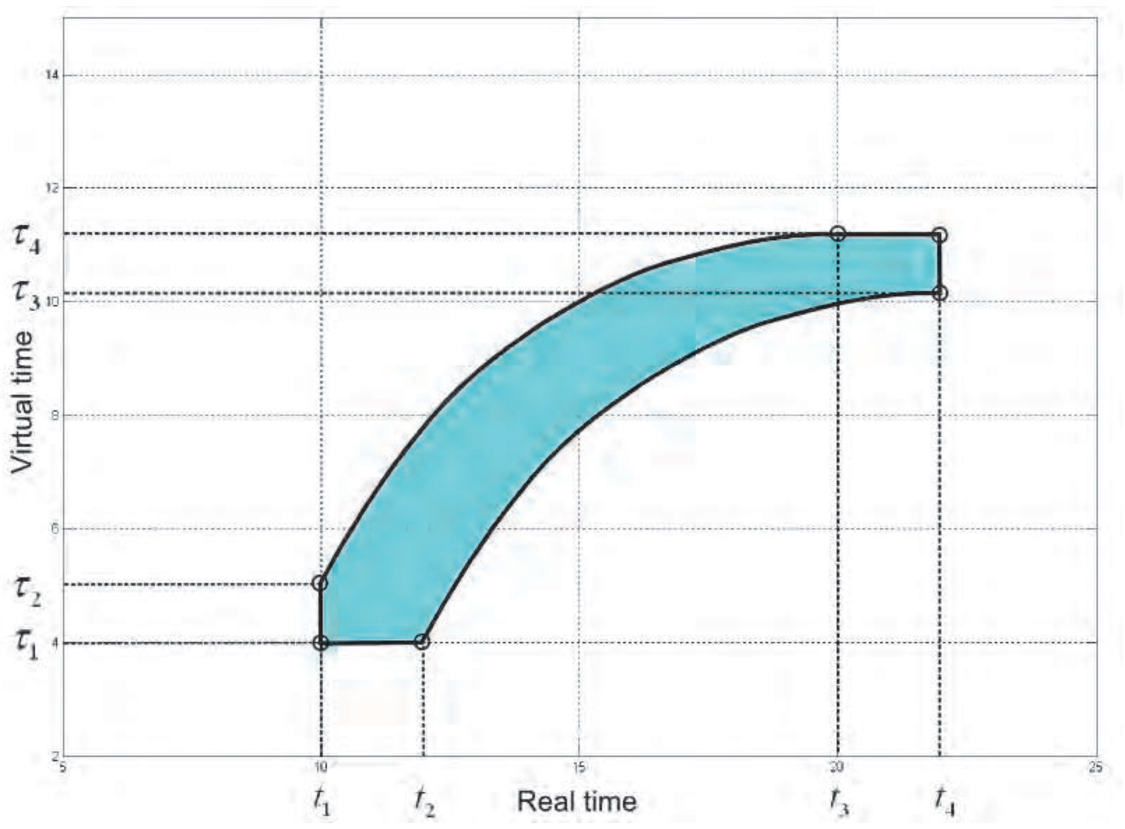


Fig. 3. Nonlinear time-obstacle

3. Path-planning in the time-plane

In this section the method of trajectory-planning in the time-plane will be presented, involving the solution of differential equations derived from the kinematic constraints prescribed for the robots motion, connecting dedicated points on the time-plane and avoiding time-obstacles.

Let us emphasize again that the default velocity profile consists of intervals with constant or zero accelerations. Acceleration parameters change at the boundaries of these intervals. We denote these points by τ_i where i refers to the index of the time instance where the acceleration change occurs. At a certain time τ where its value is assumed to be

$$\tau_i \le \tau < \tau_{i+1} \tag{18}$$

the expression of the default velocity is

$$v(\tau) = a_i(\tau - \tau_i) + v_i \tag{19}$$

where

$$a_i = a(\tau_i)\,, \qquad v_i = v(\tau_i) \tag{20}$$

are the acceleration and velocity parameters at the border point τ_i . Fig. 4 shows an example for such a velocity function.

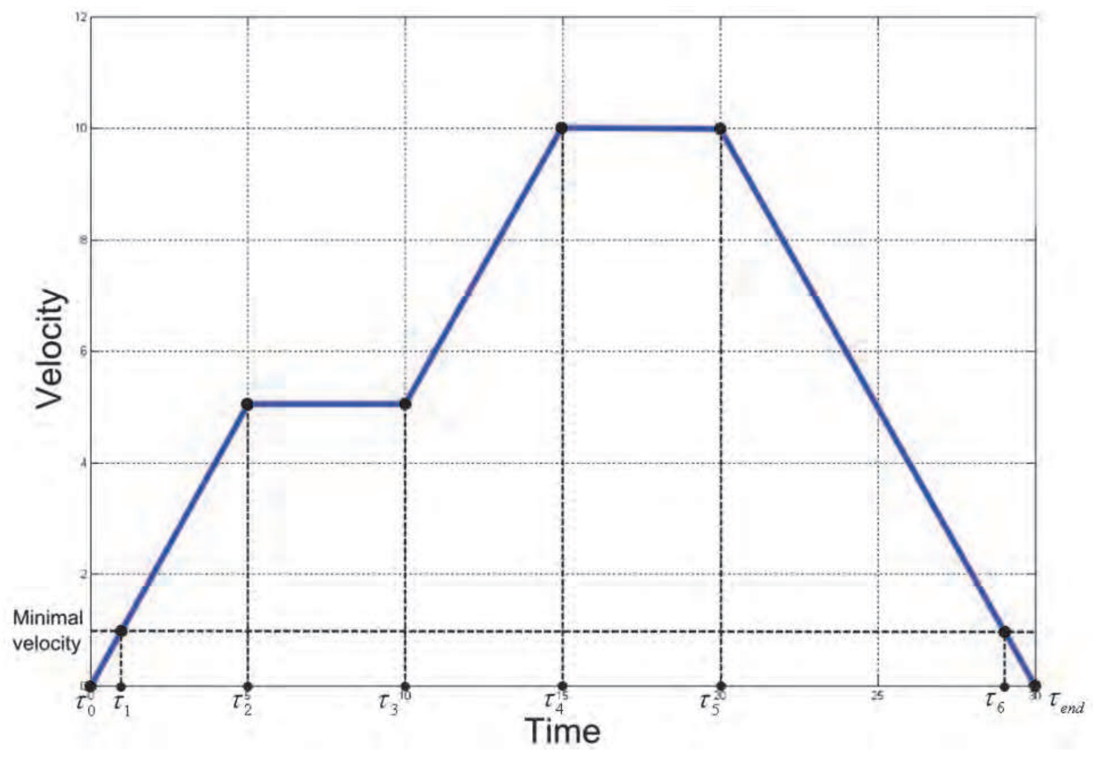


Fig. 4. Default velocity profile

3.1 Solving the differential equations

Regarding the admissible values of the accelerations specified by (13) and (19) the following equations can be obtained for the motion of the time-scaled unit:

$$a_i\tau\ddot{\tau} + a_i\dot{\tau}^2 + v_i\ddot{\tau} = a_{\max} \tag{21}$$

$$a_i\tau\ddot{\tau} + a_i\dot{\tau}^2 + v_i\ddot{\tau} = a_{\min} \tag{22}$$

$$a_i\tau\ddot{\tau} + a_i\dot{\tau}^2 + v_i\ddot{\tau} = 0 \tag{23}$$

which means that solutions of (21) and (22) result maximal and minimal accelerations in the scaled velocity, while the solution of (23) results zero acceleration. The solutions of (21) and (22) span an area on the time-plane. From a certain starting point on the time-plane only the points within this area can be reached. Let us denote such a starting point with $T_s = (t_s, \tau_s)$ and assume that the value of the default velocity at the corresponding instant is $v_s = v(\tau_s)$, while its scaled value at the corresponding t_s time instance is $\tilde{v}(t_s) = \delta_s v_s$. The $\tau = \theta(t)$ function that satisfies (21) is denoted by ${}^a\theta_{a\max}(t)$ in case of non-zero acceleration or ${}^v\theta_{a\max}(t)$ in case of zero acceleration. The subscript refers to the effect of the function on the velocity, which is $a_{\max}$ in this case.

The exact formula for ${}^a\theta_{a\max}(t)$ reads

$${}^a\theta_{a\max}(t) = \tau_s - \frac{v_s}{a_i} + \frac{1}{a_i}\sqrt{v_s^2 + 2a_i v_s \delta_s (t - t_s) + a_i a_{\max}(t - t_s)^2} \tag{24}$$

and its derivative is

$$ {}^{a}\dot{\theta}_{a\max}(t) = \frac{v_s\delta_s + a_{\max}(t-t_s)}{\sqrt{v_s^2 + 2a_i v_s \delta_s (t-t_s) + a_i a_{\max}(t-t_s)^2}} \tag{25} $$

while its inverse is

$$ {}^{a}\theta_{a\max}^{-1}(\tau) = t_s - \frac{v_s\delta_s}{a_{\max}} + \frac{1}{a_i a_{\max}}\sqrt{a_i^2 v_s^2 \delta_s^2 + a_i a_{\max}(v^2(\tau) - v_s^2)} \tag{26} $$

Applying the function ${}^{a}\theta_{a\max}(t)$ to the default velocity $v(\tau)$, the resulting scaled velocity will be $\tilde{v}(t) = {}^{a}\dot{\theta}_{a\max}(t) v({}^{a}\theta_{a\max}(t))$ that has a simple form

$$ \tilde{v}(t) = a_{\max}(t-t_s) + \delta_s v(\tau_s) \tag{27} $$

These expressions are valid for those time values where $t \in (t_s, {}^{a}\theta_{a\max}^{-1}(\tau_{i+1}))$, i.e. it is only valid over the interval for which the equations were solved. In order to span the solution for several intervals, the initial conditions τ_s and δ_s must be updated. When reaching any of the border points, the following changes have to be made regarding the parameters: $\tau_s = \tau_{i+1}$, $t_s = {}^{a}\theta_{a\max}^{-1}(\tau_{i+1})$ and $\delta_s = {}^{a}\dot{\theta}_{a\max}(\tau_{i+1})$ in the formula of ${}^{a}\theta_{a\max}(t)$. The formula of ${}^{v}\theta_{a\max}(t)$ reads

$$ {}^{v}\theta_{a\max}(t) = \frac{1}{2}\frac{a_{\max}}{v_i}(t-t_s)^2 + \delta_s(t-t_s) + \tau_s \tag{28} $$

its derivative is

$$ {}^{v}\dot{\theta}_{a\max}(t) = \frac{a_{\max}}{v_i}(t-t_s) + \delta_s \tag{29} $$

while its inverse is

$$ {}^{v}\theta_{a\max}^{-1}(\tau) = t_s - \frac{\delta_s v_i}{a_{\max}} + \frac{v_i}{a_{\max}}\sqrt{\delta_s^2 + 2\frac{a_{\max}}{v_i}(\tau - \tau_s)} \tag{30} $$

The function ${}^{v}\theta_{a\max}(t)$ results the same scaled velocity as ${}^{a}\theta_{a\max}(t)$ does, so

$$ \tilde{v}(t) = {}^{v}\dot{\theta}_{a\max}(t) v({}^{v}\theta_{a\max}(t)) = a_{\max}(t-t_s) + \delta_s v(\tau_s) \tag{31} $$

Similarly these are valid for the time values where $t \in (t_s, {}^{v}\theta_{a\max}^{-1}(\tau_{i+1}))$. The solutions of (22) which result maximal negative acceleration, are ${}^{a}\theta_{a\min}(t)$ and ${}^{v}\theta_{a\min}(t)$. Expressions of these functions, the derivatives and the inverses read

$$ {}^{a}\theta_{a\min}(t) = \tau_s - \frac{v_s}{a_i} + \frac{1}{a_i}\sqrt{v_s^2 + 2a_i v_s \delta_s (t-t_s) + a_i a_{\min}(t-t_s)^2} \tag{32} $$

$$^{a}\dot{\theta}_{a\min}(t) = \frac{v_s + a_{\min}(t-t_s)}{\sqrt{v_s^2 + 2a_i v_s \delta_s (t-t_s) + a_i a_{\max}(t-t_s)^2}} \tag{33}$$

$$^{a}\theta_{a\min}^{-1}(\tau) = t_s - \frac{v_s\delta_s}{a_{\min}} + \frac{1}{a_i a_{\min}}\sqrt{a_i^2 v_s^2 \delta_s^2 + a_i a_{\min}(v^2(\tau) - v_s^2)} \tag{34}$$

where $t \in (t_s, {}^{a}\theta_{a\min}^{-1}(\tau_{i+1}))$ and

$$^{v}\theta_{a\min}(t) = \frac{1}{2}\frac{a_{\min}}{v_i}(t-t_s)^2 + \delta_s(t-t_s) + \tau_s \tag{35}$$

$$^{v}\dot{\theta}_{a\min}(t) = \frac{a_{\min}}{v_i}(t-t_s) + \delta_s \tag{36}$$

$$^{v}\theta_{a\min}^{-1}(\tau) = t_s - \frac{\delta_s v_i}{a_{\min}} + \frac{v_i}{a_{\min}}\sqrt{\delta_s^2 + 2\frac{a_{\min}}{v_i}(\tau - \tau_s)} \tag{37}$$

where $t \in (t_s, {}^{v}\theta_{a\min}^{-1}(\tau_{i+1}))$. Both $^{a}\theta_{a\min}(t)$ and $^{v}\theta_{a\min}(t)$ result the scaled velocity function

$$\tilde{v}(t) = a_{\min}(t-t_s) + \delta_s v(\tau_s) \tag{38}$$

The solutions of (23) are $^{a}\theta_v(t)$ and $^{v}\theta_v(t)$, which can be used to keep the velocity constant once a desired value is reached. It can be any velocity within the robots' reach, but these functions are used for not letting the velocity decrease under a certain minimal value. The expressions of the functions, their derivatives and their inverses are as follows:

$$^{a}\theta_v(t) = \tau_s - \frac{v_s}{a_i} + \frac{1}{a_i}\sqrt{v_s^2 + 2a_i v_s \delta_s (t-t_s)} \tag{39}$$

$$^{a}\dot{\theta}_v(t) = \frac{v_s\delta_s}{\sqrt{v_s^2 + 2a_i v_s \delta_s (t-t_s)}} \tag{40}$$

$$^{a}\theta_v^{-1}(\tau) = t_s + \frac{v^2(\tau) - v_s^2}{2a_i v_s \delta_s} \tag{41}$$

where $t \in (t_s, {}^{a}\theta_v^{-1}(\tau_{i+1}))$. Similarly,

$$^{v}\theta_v(t) = \frac{v_s}{v_i}(t-t_s) + \tau_s \tag{42}$$

$$^{v}\dot{\theta}_v(t) = \frac{v_s}{v_i} \tag{43}$$

$$ {}^{v}\theta_v^{-1}(\tau) = \frac{v_i}{v_s}(\tau - \tau_s) + t_s \tag{44} $$

where $t \in (t_s, {}^{v}\theta_v^{-1}(\tau_{i+1}))$. These solutions enforce the scaled velocity to be

$$ \tilde{v}(t) = v_s \tag{45} $$

Since the constraint $\dot{\theta}(t) \le 1$ must be respected (see (9)), it has to be checked whether the value of the derivative of an applied $\theta(t)$ function reaches the unity at a certain point. We denote this point by $T_{ch} = (t_{ch}, \tau_{ch})$ where the control has to be changed to constant time-scaling and remain on the curve

$$ \theta_{const}(t) = m(t - t_{ch}) + \tau_{ch} \tag{46} $$

where in this case $m = 1$. Fig. 5 shows an example of applying different time scaling functions.

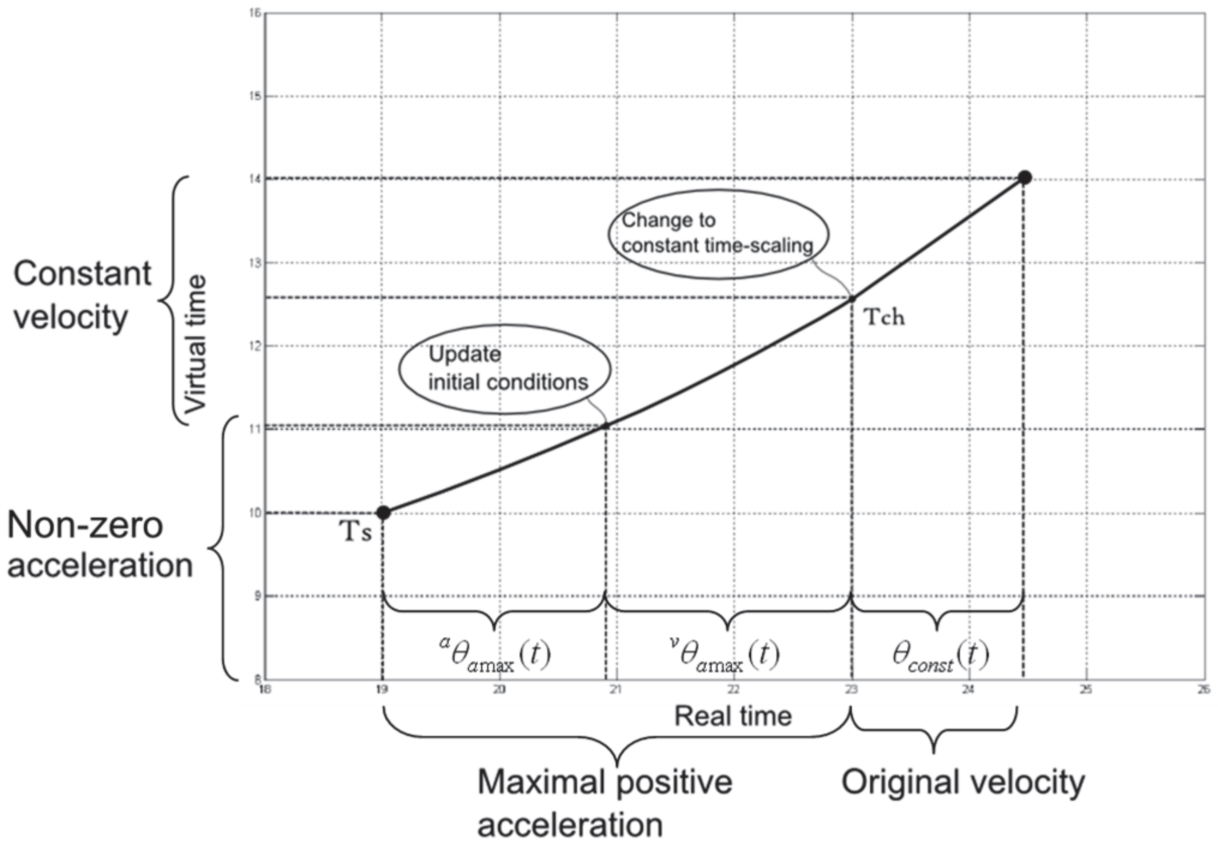


Fig. 5. Finding the changing point

To find the point T_{ch} the equation

$$ {}^{a}\dot{\theta}_{a\max}(t_{ch}) = 1 \tag{47} $$

or

$$ {}^{v}\dot{\theta}_{a\max}(t_{ch}) = 1 \tag{48} $$

should be solved, depending on the actual interval. Solving (47) results a quadratic equation

$$a_{\max}(a_{\max}-a_i)(t_{ch}-t_s)^2+2v_s\delta_s(a_{\max}-a_i)(t_{ch}-t_s)+v_s^2(\delta_s^2-1)=0 \tag{49}$$

The one satisfying $t_{ch}\in(t_s,{}^a\theta_{a\max}^{-1}(\tau_{i+1}))$ of the two roots of (49) should be selected, which results $\tau_{ch}={}^a\theta_{a\max}(t_{ch})$.
In case of (48), the solution for t_{ch} is much simpler

$$t_{ch}=(1-\delta_s)\frac{v_i}{a_{\max}}+t_s \tag{50}$$

and $\tau_{ch}={}^v\theta_{a\max}(t_{ch})$. Note that the formulae (51) and (52) only give accurate value if ${}^a\dot{\theta}_{a\max}({}^a\theta_{a\max}^{-1}(\tau_{i+1}))\geq 1$. The corresponding condition for (50) is ${}^v\dot{\theta}_{a\max}({}^v\theta_{a\max}^{-1}(\tau_{i+1}))\geq 1$, i.e. it has to be checked whether the value of the derivative at the next point determined by the function corresponding to the next border point is greater than one.
For negative accelerations, we must guarantee that the scaled velocity does not drop below $v_{\min}$. Therefore, the point $T_{ch}=(t_{ch},\tau_{ch})$ where the time scaling is changed to ${}^a\theta_v(t)$ or ${}^v\theta_v(t)$ (with $v_s=v_{\min}$ in their formulae) can be determined easily:

$$t_{ch}=\frac{\delta_s v(\tau_s)-v_{\min}}{a_{\min}}+t_s \tag{51}$$

and $\tau_{ch}={}^a\theta_{a\min}(t_{ch})$ or $\tau_{ch}={}^v\theta_{a\min}(t_{ch})$ respectively. Fig. 6 shows an example of finding this point.

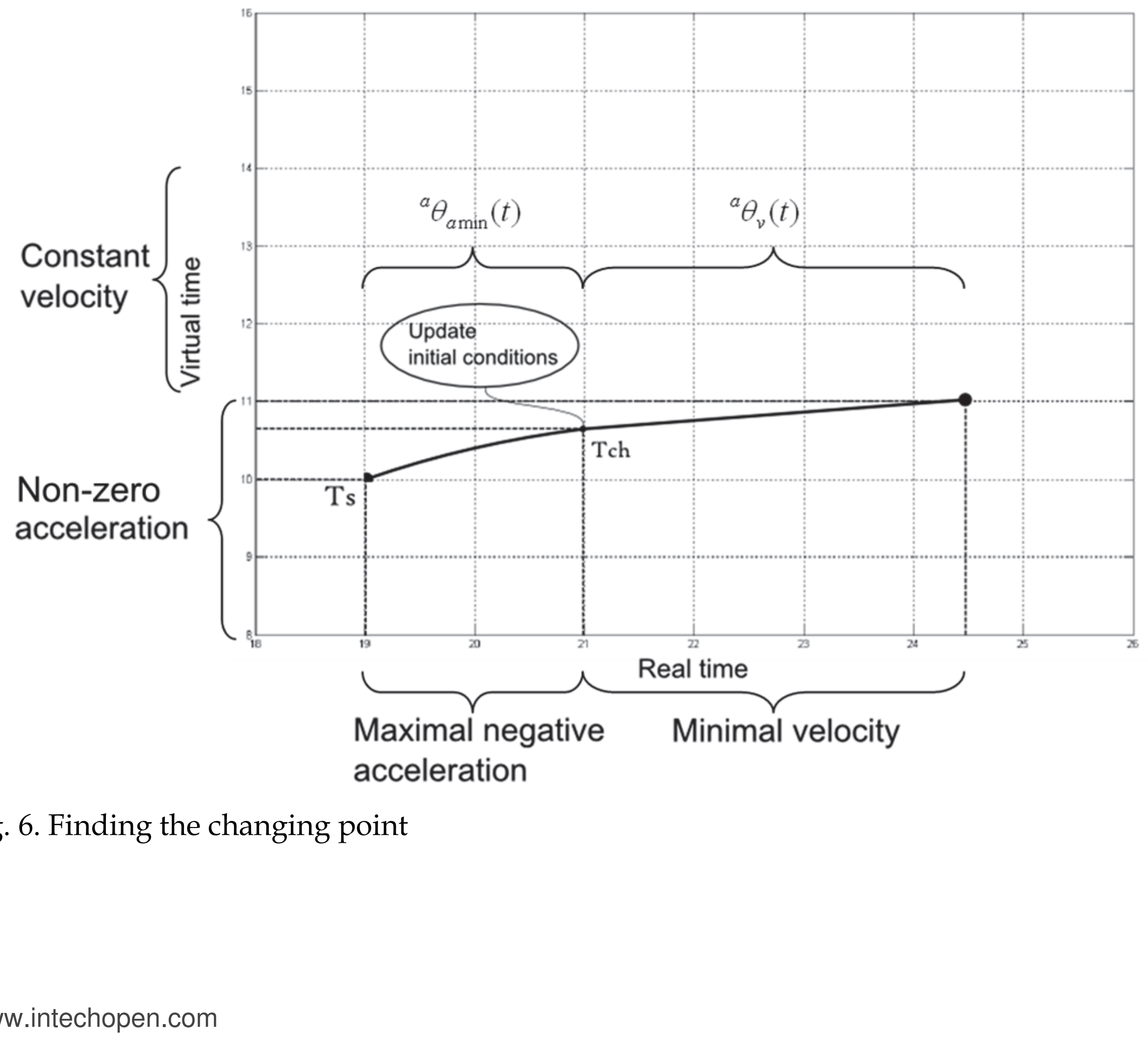


Fig. 6. Finding the changing point

When moving from a given point $T_s = (t_s, \tau_s)$ to another point $T_g = (t_g, \tau_g)$ on the time-plane, it has to be checked first whether T_g is within the area of reachable points from T_s. In order to do so, the boundaries of the reachable area have to be determined. The upper boundary will be the union of curves resulting maximal acceleration and then maximal speed, and the lower boundary will be the union of ones resulting maximal deceleration and then minimal velocity. Let us denote the upper boundary that origins in T_s and ends at t_g by $\Theta_{[Ts \to Tg],MAX}$ and the lower boundary by $\Theta_{[Ts \to Tg],MIN}$. T_g is reachable from T_s if and only if the inequalities below are both satisfied

$$\tau_g \le \Theta_{[Ts \to Tg],MAX}(t_g), \qquad \tau_g \ge \Theta_{[Ts \to Tg],MIN}(t_g) \tag{52}$$

Fig. 7 shows the area enclosed by these boundaries.

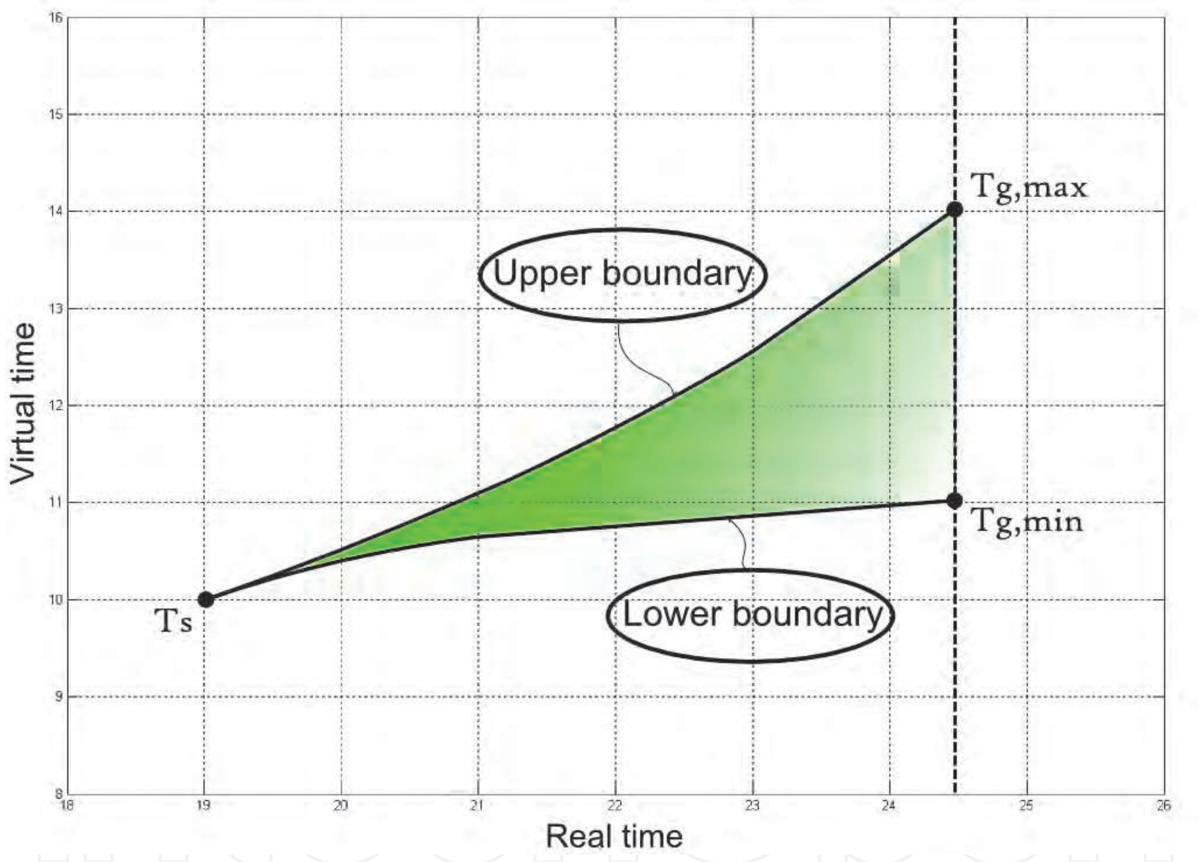


Fig. 7. The area of reachable points

If (52) is satisfied, T_g is within the reach of T_s and a trajectory connecting these two points can be planned. The main idea of trajectory planning is to travel along the upper or the lower boundary depending on the initial conditions in T_s and then switch to a straight line i.e. constant time-scaling. The switching point $T_{sw} = (t_{sw}, \tau_{sw})$ can be determined by solving the equation

$$\frac{\tau_g - \tau_{sw}}{t_g - t_{sw}} = \dot{\Theta}_{[Ts \to Tg],MAX}(t_{sw}) \tag{53}$$

or

$$\frac{\tau_g - \tau_{sw}}{t_g - t_{sw}} = \dot{\Theta}_{[Ts \to Tg],MIN}(t_{sw}) \tag{54}$$

depending on the selected boundary. In case of (53), the switching point can be on a ${}^{a}\theta_{a\max}(t)$ curve or on a ${}^{v}\theta_{a\max}(t)$ curve, so (53) transforms to

$$\frac{\tau_g - \tau_{sw}}{t_g - t_{sw}} = {}^{a}\dot{\theta}_{a\max}(t_{sw}) \tag{55}$$

or

$$\frac{\tau_g - \tau_{sw}}{t_g - t_{sw}} = {}^{v}\dot{\theta}_{a\max}(t_{sw}) \tag{56}$$

Solving (55) results in a quadratic equation, introducing the notation $\gamma = a_i v_s \delta_s$ and $v_g = a_i(\tau_g - \tau_s) + v_s$

$$\begin{aligned}&\left((\gamma + a_i a_{\max}(t_g - t_s))^2 - v_g^2 a_i a_{\max}\right)(t_{sw} - t_s)^2 + \\ &+\left(2(a_i a_{\max}(t_g - t_s)^2 + \gamma)(\gamma(t_g - t_s) + v_s^2) - 2v_g^2\gamma\right)(t_{sw} - t_s) + \\ &+\left(\gamma(t_g - t_s) + v_s^2\right)^2 - v_g^2 v_s^2 = 0\end{aligned} \tag{57}$$

Of the roots of (57) one has to select the solution that satisfies $t_{sw} \in ({}^{a}\theta_{a\max}^{-1}(\tau_i), {}^{a}\theta_{a\max}^{-1}(\tau_{i+1}))$, after that $\tau_{sw} = {}^{a}\theta_{a\max}(t_{sw})$. Solving (56) also leads to a quadratic equation

$$\frac{1}{2}\frac{a_{\max}}{v_i}(t_{sw} - t_s)^2 - \left(\frac{a_{\max}}{v_i}(t_g - t_s)\right)(t_{sw} - t_s) + \tau_g - \tau_s - \delta_s(t_g - t_s) = 0 \tag{58}$$

One has to select the solution that satisfies $t_{sw} \in ({}^{a}\theta_{a\max}^{-1}(\tau_i), {}^{a}\theta_{a\max}^{-1}(\tau_{i+1}))$, after that $\tau_{sw} = {}^{v}\theta_{a\max}(t_{sw})$.
Determining the switching point in case of travelling on the lower boundary the following equations should be solved

$$\frac{\tau_g - \tau_{sw}}{t_g - t_{sw}} = {}^{a}\dot{\theta}_{a\min}(t_{sw}) \tag{59}$$

$$\frac{\tau_g - \tau_{sw}}{t_g - t_{sw}} = {}^{v}\dot{\theta}_{a\min}(t_{sw}) \tag{60}$$

$$\frac{\tau_g - \tau_{sw}}{t_g - t_{sw}} = {}^{a}\dot{\theta}_{v}(t_{sw}) \quad \text{with} \qquad v_s = v_{\min} \tag{61}$$

depending on the possible location of the switching point. The solution of (59) and (60) are similar to (55) and (56) except that at the expressions $a_{\max}$ is replaced by $a_{\min}$. Solving (61) leads to a quadratic equation

$$\gamma^2(t_{sw}-t_s)^2+\left(2\gamma^2(t_g-t_s)+2\gamma(v_s^2-v_g^2)\right)(t_{sw}-t_s)+\left(\gamma(t_g-t_s)+v_s^2\right)^2-v_g^2v_s^2=0 \tag{62}$$

One has to select the solution that satisfies $t_{sw}\in({}^a\theta_v^{-1}(\tau_i),{}^a\theta_v^{-1}(\tau_{i+1}))$, after that $\tau_{sw}={}^a\theta_v(t_{sw})$. After determining the switching point T_{sw} the following equation can be used to travel along a straight line until reaching T_g:

$$\theta_{const}(t)=\frac{\tau_g-\tau_{sw}}{t_g-t_{sw}}(t-t_{sw})+\tau_{sw} \tag{63}$$

Here $t\in(t_{sw},t_g)$. Fig. 8 shows an example of finding such a switching point.
T_{g0} can be reached directly from T_s. It is possible to reach T_{g1} by travelling along the upper boundary and then switching to a straight line at T_{sw1}. Similarly, T_{g2} can be reached by travelling along the lower boundary and then switching to a straight line at T_{sw2}.

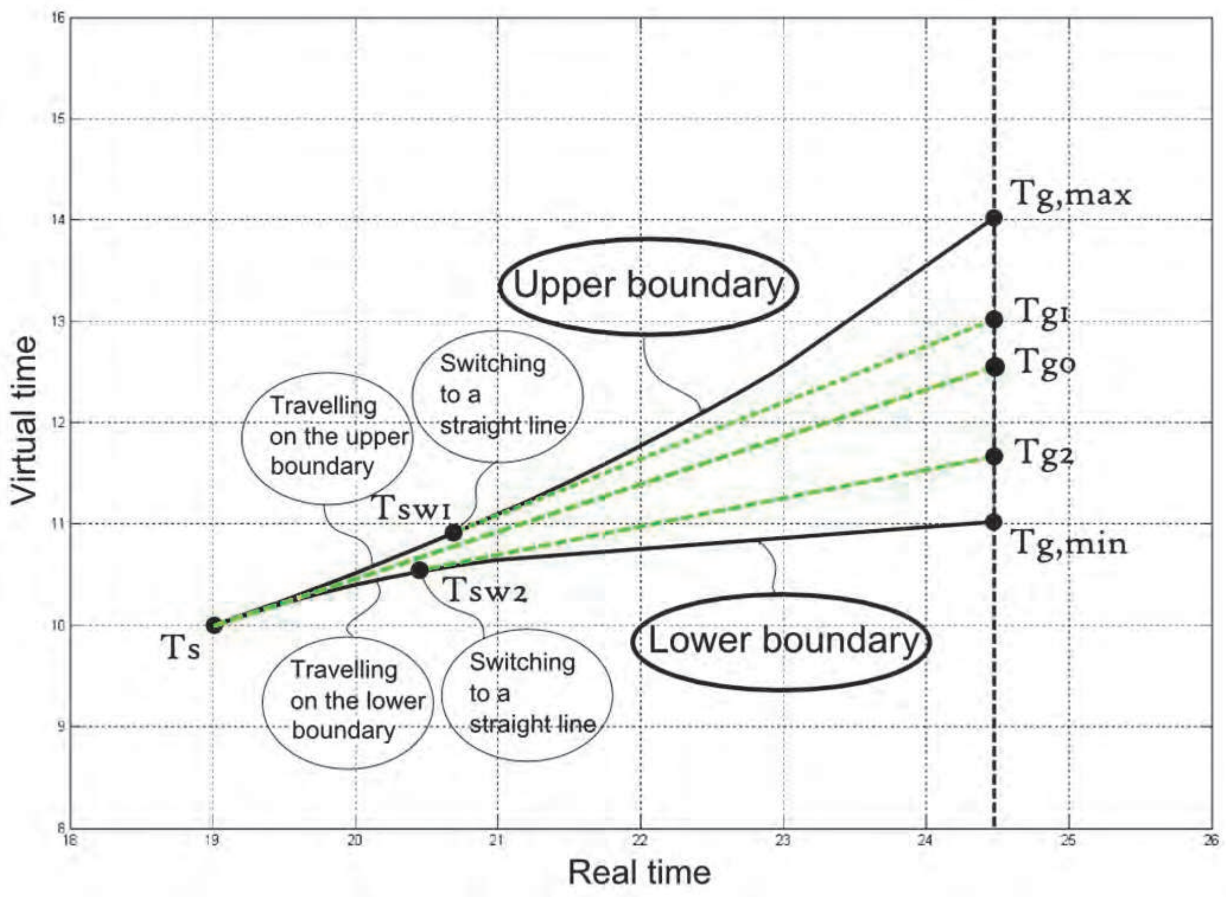


Fig. 8. Travelling on a border than switching to a straight line

Note that constant time-scaling with the general formula

$$\theta_{const}(t) = m(t - t_s) + \tau_s \tag{64}$$

results the scaled velocity function

$$\tilde{v}(t) = m^2 a_i (t - t_s) + m v_s \tag{65}$$

Denoting the curve that origins in T_s and ends in T_{sw} by $\Theta_{[Ts \to Tsw]}$ and the one that origins in T_{sw} and ends in T_g by $\Theta_{[Tsw \to Tg]}$, the path from T_s to T_g is

$$\Theta_{[Ts \to Tg]} = \Theta_{[Ts \to Tsw]} \cup \Theta_{[Tsw \to Tg]} \tag{66}$$

One should follow the following steps when constructing this path:

1. From a dedicated starting point construct the boundaries of the area of the reachable points. Update initial conditions at the border points and apply the appropriate type of curve for every interval between the starting point and the one desired to reach. When reaching a point where the scaled velocity would overrun the limits change to constant time-scaling or to the solution that results minimal speed, respectively.
2. Examine whether the desired point to reach is within the area of the reachable points.
3. Depending on the initial conditions at the starting point travel along the upper or the lower boundary until switching to a straight line is possible. Concatenate this line to the curve sequence between the starting and the switching point.

3.2 Avoiding time-obstacles

Avoiding time-obstacles includes collision detection with time-obstacles and a specific path planning method based on the algorithms mentioned above. Collision detection is simple in case of a rectangular time-obstacle. Since the solutions of the differential equations are given in analytic form, the exact time-values have to be substituted into those formulae in order to check whether a certain point of the curve belongs to a time-obstacle. Let us denote the vertices of a rectangular time-obstacle by its four corner points on the time-plane:

$$\begin{matrix} T_{enter,enter} = (t_{enter}, \tau_{enter}) & T_{enter,exit} = (t_{enter}, \tau_{exit}) \\ T_{exit,enter} = (t_{exit}, \tau_{enter}) & T_{exit,exit} = (t_{exit}, \tau_{exit}) \end{matrix} \tag{67}$$

The edges of this time-obstacle are the intervals between these points. In case of the horizontal edges, if one of

$$\tau_{enter} \le \Theta_{[Ts \to Tg]}(t_{enter}) \le \tau_{exit}, \qquad \tau_{enter} \le \Theta_{[Ts \to Tg]}(t_{exit}) \le \tau_{exit} \tag{68}$$

is satisfied, then a collision occurs. A similar formula can be applied for the vertical edges, thus a collision occurs when one of

$$t_{enter} \le \Theta^{-1}_{[Ts \to Tg]}(\tau_{enter}) \le t_{exit}, \ t_{enter} \le \Theta^{-1}_{[Ts \to Tg]}(\tau_{exit}) \le t_{exit} \tag{69}$$

is satisfied.

Note that one has to substitute the exact time-values into the appropriate element (i.e. curve sequence) of $\Theta_{[Ts \to Tg]}$ and $\Theta^{-1}_{[Ts \to Tg]}$ corresponding to the given time-value.

In case of nonlinear time-obstacles, the simplest way to determine a collision is to regard the scaled velocities. If

$$\left(\int_{tenter}^{texit} \tilde{v}(t)dt + \Gamma(t_{enter}) \right) \cap O_{dyn}(t_{enter}, t_{exit}) \neq \varnothing \tag{70}$$

then the unit collides with a dynamic obstacle.

In the sequel, a path planning method based on the algorithm of connecting points on the time-plane and determining collisions with time-obstacles will be presented. Because of the presence of time-obstacles, an exact path between dedicated points might not exist, even if the desired point of reach is within the reachable area of the starting point T_s. Fig. 9 shows an example of several time-obstacles lying between the starting and the goal positions, while Fig. 10 shows that T_s cannot be connected with T_g directly because of a collision with a time-obstacle.

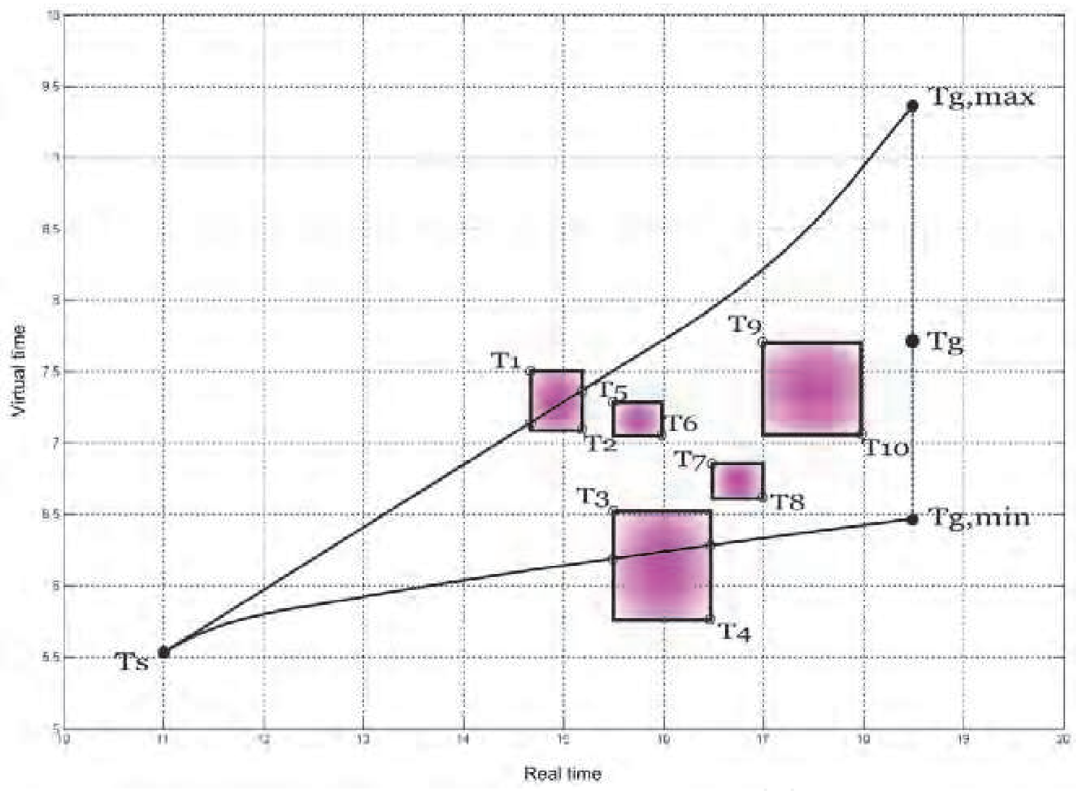


Fig. 9. Time-obstacles between two points

A path from the starting point avoiding the time-obstacles might only guarantee an endpoint with the same real-time coordinate t_g as the desired T_g, but the resulting virtual-time coordinate may differ from the desired τ_g. It is advised to examine which points with the real-time coordinate t_g can be reached from this starting point T_s. At first the corner points of the time-obstacles which can be reached without colliding with any other time-obstacles have to be determined. To do so, a graph (i.e. tree) building approach is proposed.

At first the parent node is selected to be the actual point, desired to be reached. Child nodes are the upper left and lower right vertices of the time obstacle that the specific route hits first

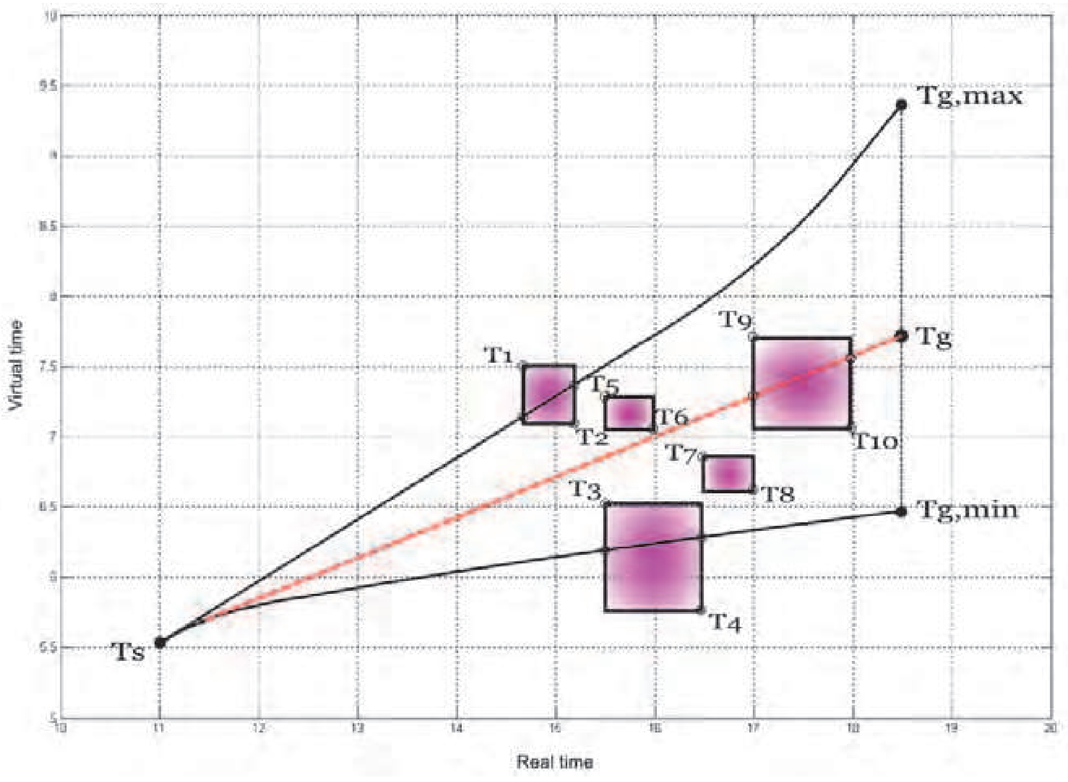


Fig. 10. The goal point cannot be reached without collision

(i.e. the one that is the nearest to the starting point and has an intersection with the actual path). The graph building can be done sequentially. Fig. 11 shows steps of a sequence of determining the child nodes and connecting them to their parent nodes.

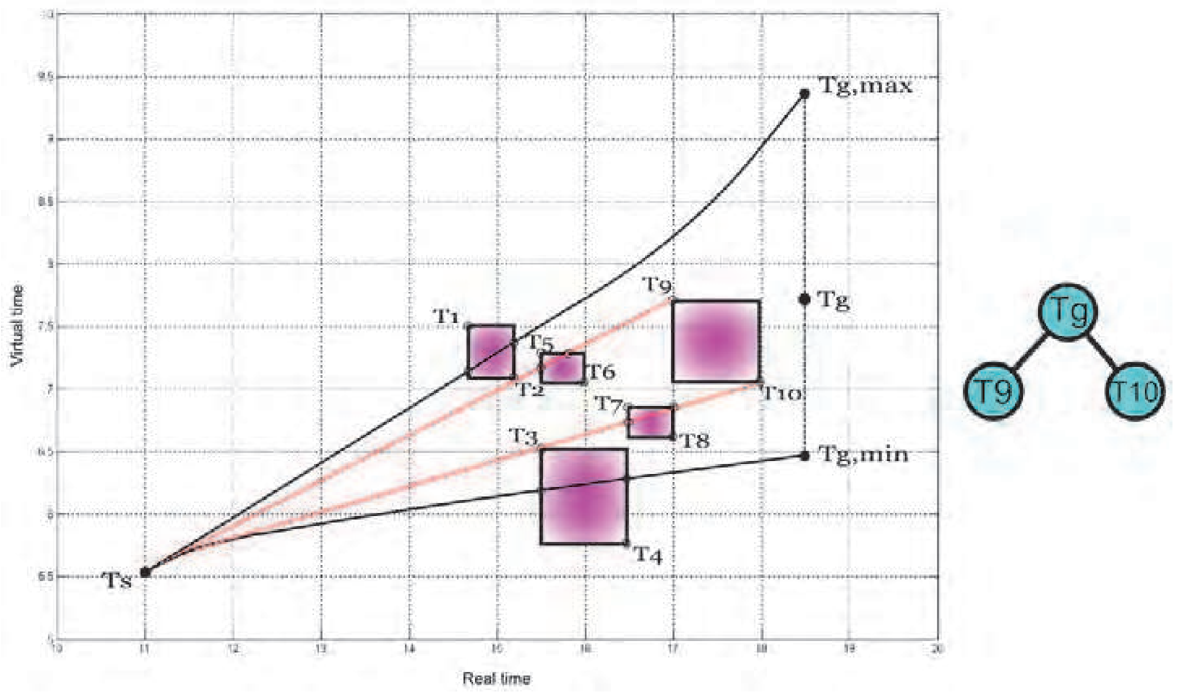


Fig. 11. Selecting child nodes by determining collisions on the time-plane (on the left) and building the graph (on the right)

The path from T_s to T_g intersects with the time-obstacle with corner points T_9 and T_{10}, thus in the graph these become the children of T_g. Now the desired point of reach will be T_9 or T_{10}, respectively. Fig. 12 shows that none of them can be reached without collision, thus they will also have child nodes in the graph.

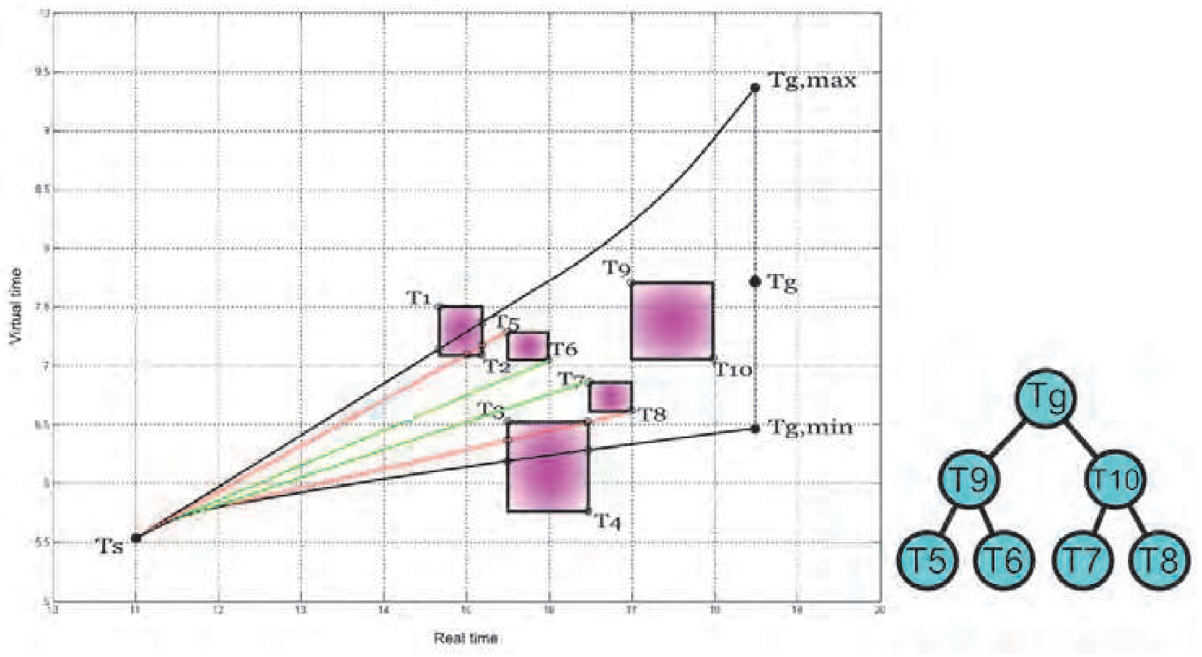


Fig. 12. Selecting child nodes by determining collisions (on the left) and building the graph (on the right)

Final steps of graph building is presented by Fig. 13:

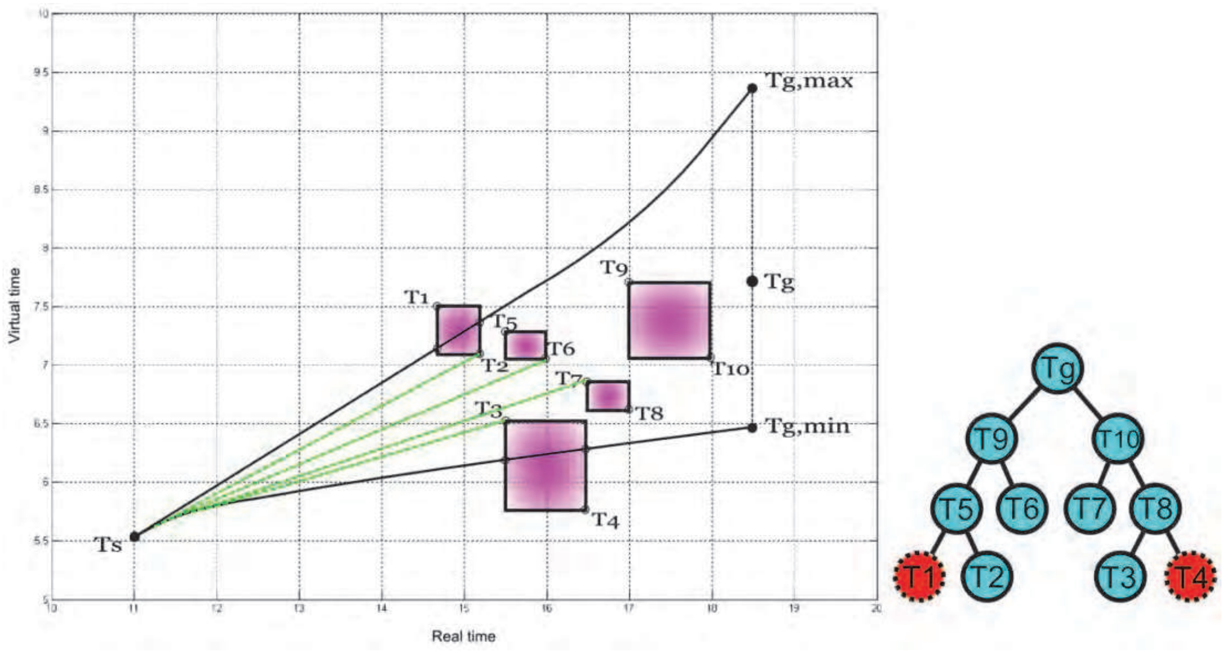


Fig. 13. The corner points that can be reached with no collision (on the left), building the graph (on the right)

On Fig. 13 the dotted contour denotes that T_1 and T_4 are located outside of the area of reachable points. Fig. 14 represents the final graph.

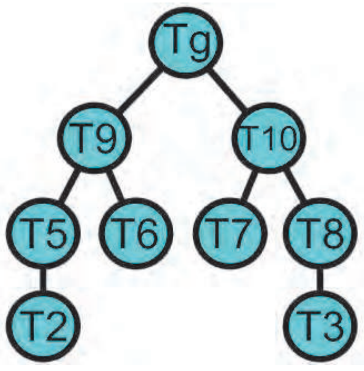


Fig. 14. The final graph

Nodes with no children provide the vertices that can be reached from the starting point without collision. After determining these points, a new starting point must be assigned from the ones determined afore, and the algorithm must be carried on until finding all possible routes from the original starting point to the ones with real-time coordinate t_g . Fig. 15 shows all possible routes from the original starting point to the points with real-time coordinate t_g while Fig. 16 shows its graph representation. Dot-contoured nodes denote that no route exists from their parent node that reaches them without collision or exceeding the kinematic constraints.

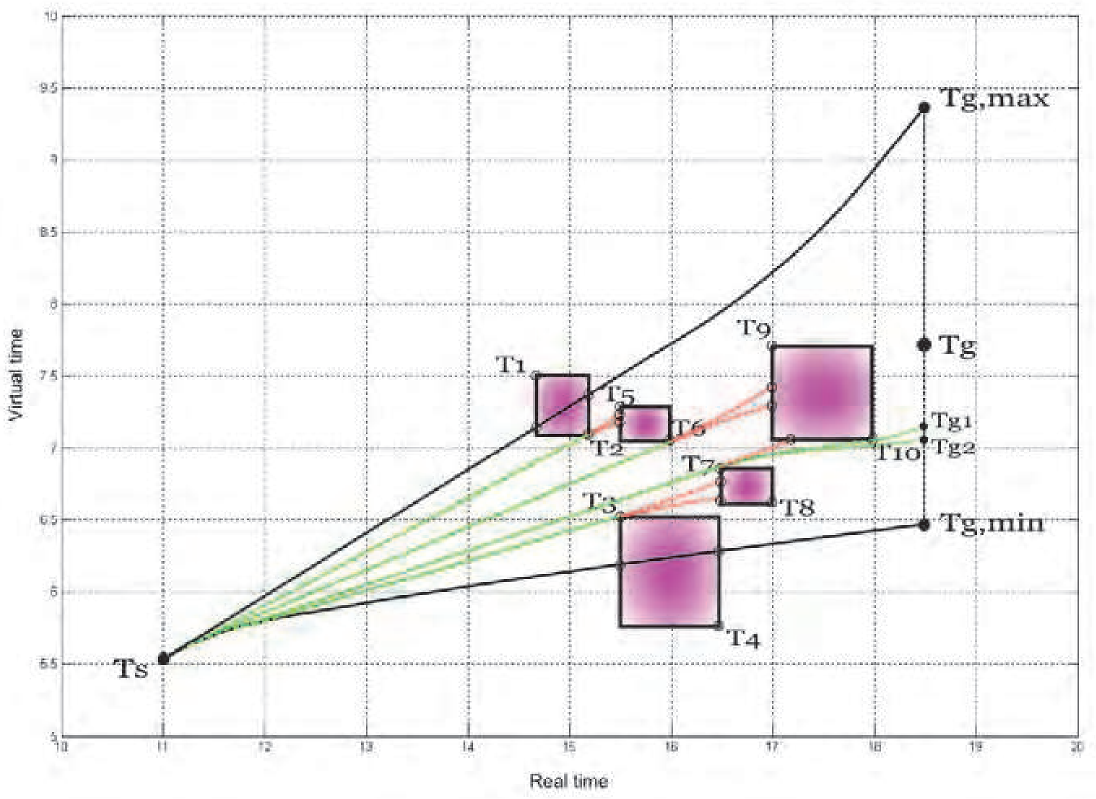


Fig. 15. Possible routes from Ts

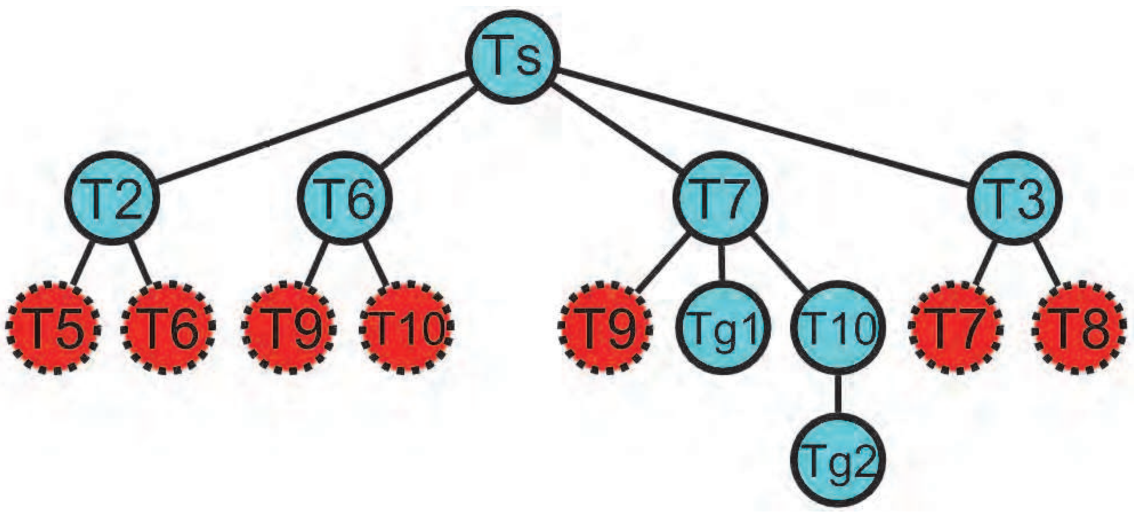


Fig. 16. Graph representation of the possible routes

3.3 The global path-planning method

Path-planning on the time-plane starts from the point at which the original velocity reaches a desired minimal value that was referred to as $v_{\min}$. The curve $\tau = t$ and the curve that spans through all intervals and results the minimal velocity enclose the area that the robot can reach during the time of its motion. Any other points outside this area are unreachable due to the kinematic constraints. Fig. 17 shows an example for such an area.

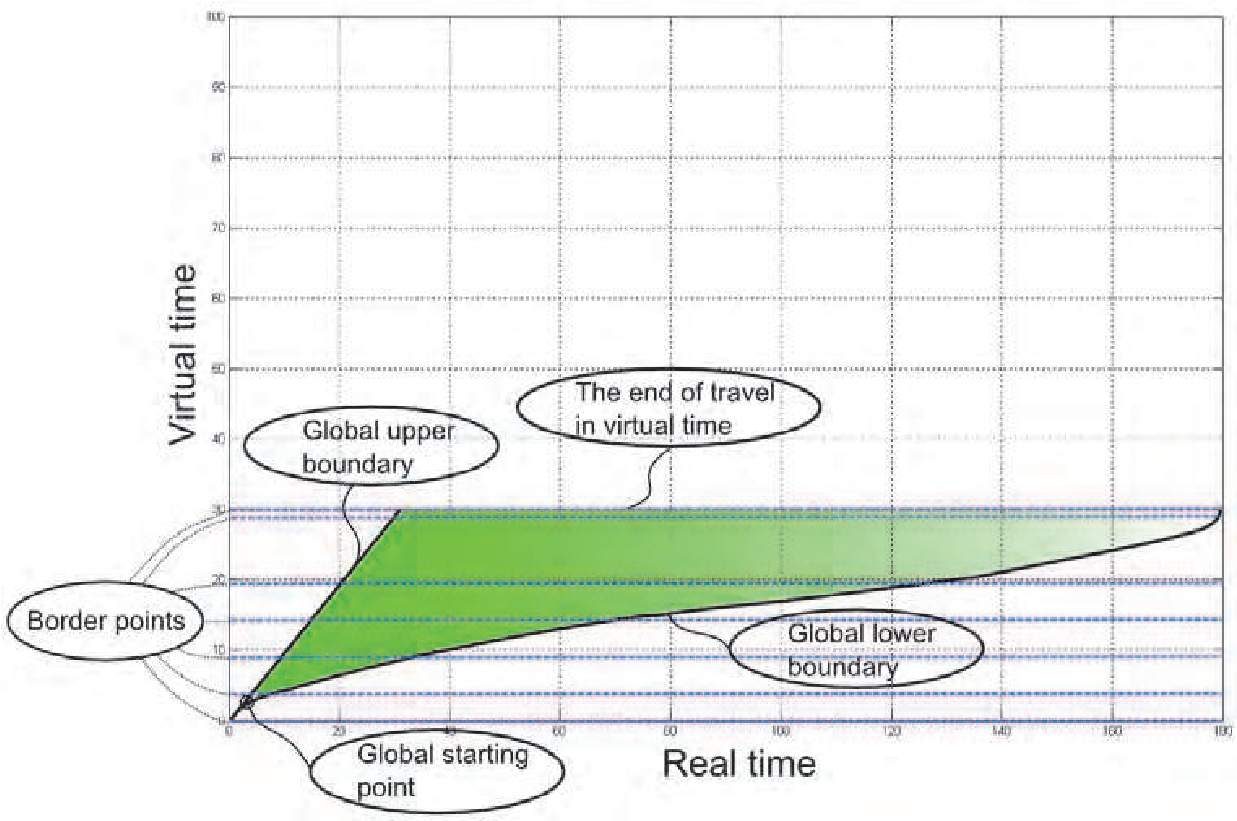


Fig. 17. Global area of reachable points

The goal of path-planning is to reach a point on the time-plane with virtual time coordinate τ_{end} (the end of travel in virtual time) and with a minimal value of real-time coordinate, i.e. to minimize the duration of travel in real time. In order to reach such a point, the following considerations should be followed:

1. When starting a path from a point, apply maximal control inputs and maintain this course until reaching a point with virtual time coordinate τ_{end}.
2. In case of collision, determine all time-obstacle corner points that can be reached from the starting point with no collision.
3. Assign a new starting point from these corner points and continue from step 1. If no further movement can be made, go back to a previous starting point and select another corner point to reach.

Navigating through all time-obstacles may not give an optimal solution. The possibility of delaying the departure of a certain robot should also be considered, which allows time-obstacles to disappear from its time-plane. Depending on the certain area of application, it can cause difficulties due to tight schedules. Fig. 18 shows an example of delaying a unit's departure so it can use its default velocity and arrive at its desired destination earlier then by using time scaling-based obstacle avoidance.

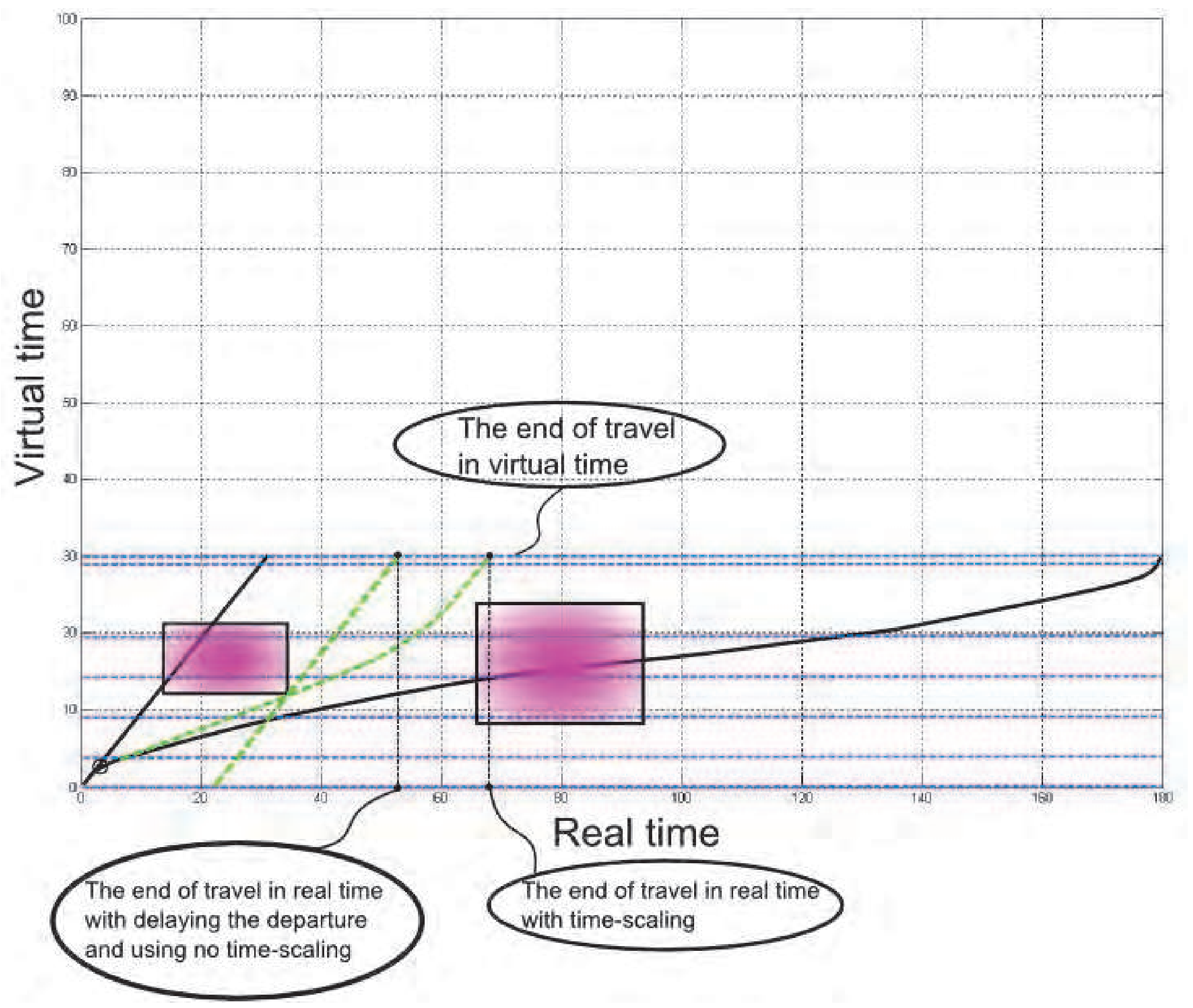


Fig. 18. Arrival times differ when delaying departure and using no time-scaling

3.4 Generation of the scaled velocity

The scaled velocity can be generated by multiplying the original velocity function and the derivative of the time-scaling function (see (6)). It is only necessary to calculate the values of the new velocity in those real-time instants where parameters in the original velocity have changed and where the time-scaling function changes its nature. The latter occurs typically at switching points where constant time-scaling is applied after maximal inputs and changing points where maximal or minimal value of velocity is reached. After these calculations, these points have to be connected by linear segments.

3.5 Synchronizing the motion of the robots

As it has been already mentioned, it is imperative to synchronize the units' motion in an area crowded by robots. A straightforward solution is to define a hierarchy between the robots so that higher priority units disregard ones with lower priority level. Such a priority order can be defined regarding several factors, and priority levels may also be redistributed regularly in order to meet the actual requirements. The general algorithm managing the multi-unit fleet is the following:

1. Consider the highest priority level.
2. Take a robot with a given priority level and design an optimal path avoiding static obstacles.
3. Assign an optimal velocity to this robot.
4. If the robot collides with any other robots with higher priority, determine the possible time-obstacles.
5. Construct the time-scaling function and generate the scaled velocity.
6. Take the next robot with lower priority and continue from step 1.

4. Implementation results

We present a robot system with four robots, the workspace and the trajectories are shown in Fig. 19.

The length of the robots are assumed to be 1 meter while their width are assumed to be 0.8 meters. (These values may be also different for each robot.) The paths consist of linear segments and they are extended with the widths of the robots. The coordinates of the points where the robots enter or leave the collision areas are calculated using these geometric parameters. On Fig. 19 the points S1, S2, S3 and S4 denote to the initial locations of the robots while G1, G2, G3 and G4 determines the goal locations respectively. The highest priority robot is Robot#1, the next one is Robot#2 then Robot#3 and Robot#4 has the lowest priority level. Table 1 contains information about the boundaries of the path elements (x and y coordinates respectively). We designed simple default velocites which are shown in Table 2 together with the scaled velocity. The admissible values of the maximal positive and negative accelerations for Robot#1 and Robot#3 are 0.2 m/s^2 and -0.2 m/s^2 while for Robot#2 and Robot#4 the same values are 0.5 m/s^2 and -0.5 m/s^2 respectively. The minimal velocity is 0.2 m/s for all robots. Since Robot#1 has the highest priority it does not need to be scaled. The time-scaling process for Robot#2, Robot#3 and Robot#4 are shown on Fig. 20, 22 and 24. Datatips on the time-scaling function indicate the relevant time instances where the scaled velocity values have to be calculated. The original and the resulted scaled velocity functions (with blue and green colors respectively) are drawn on Fig. 21, 23 and 25.

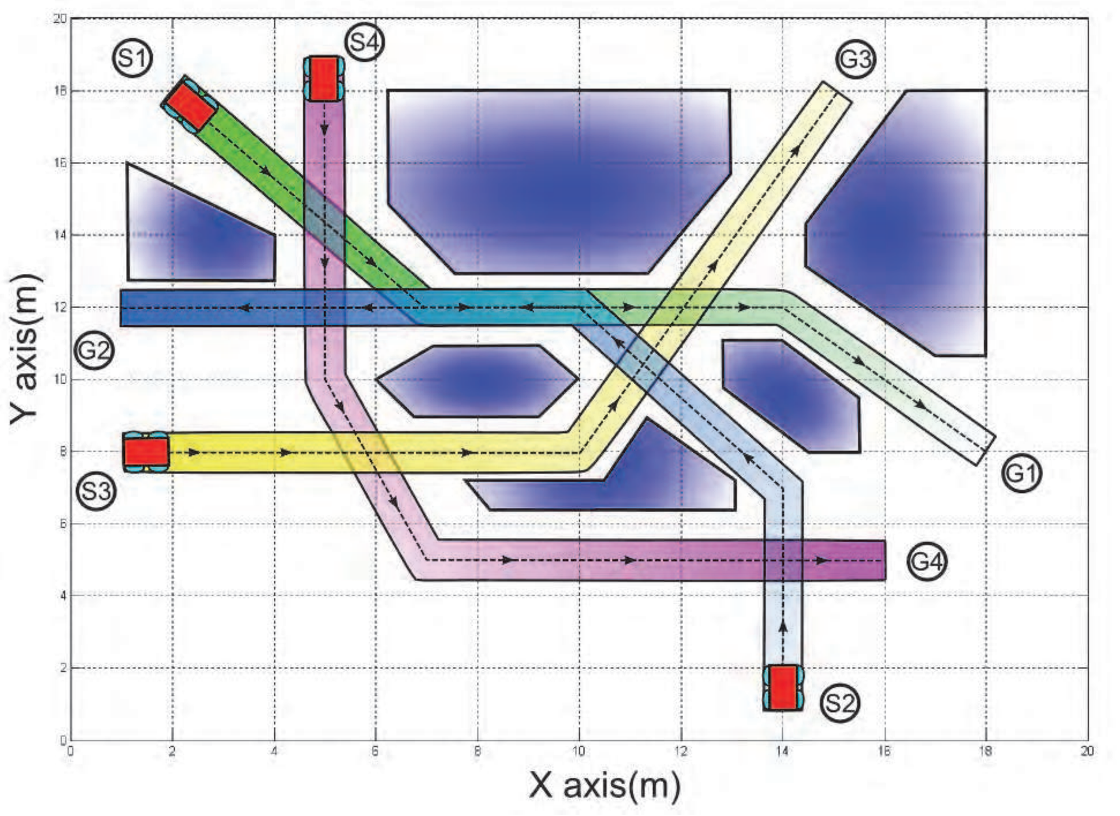


Fig. 19. Workspace

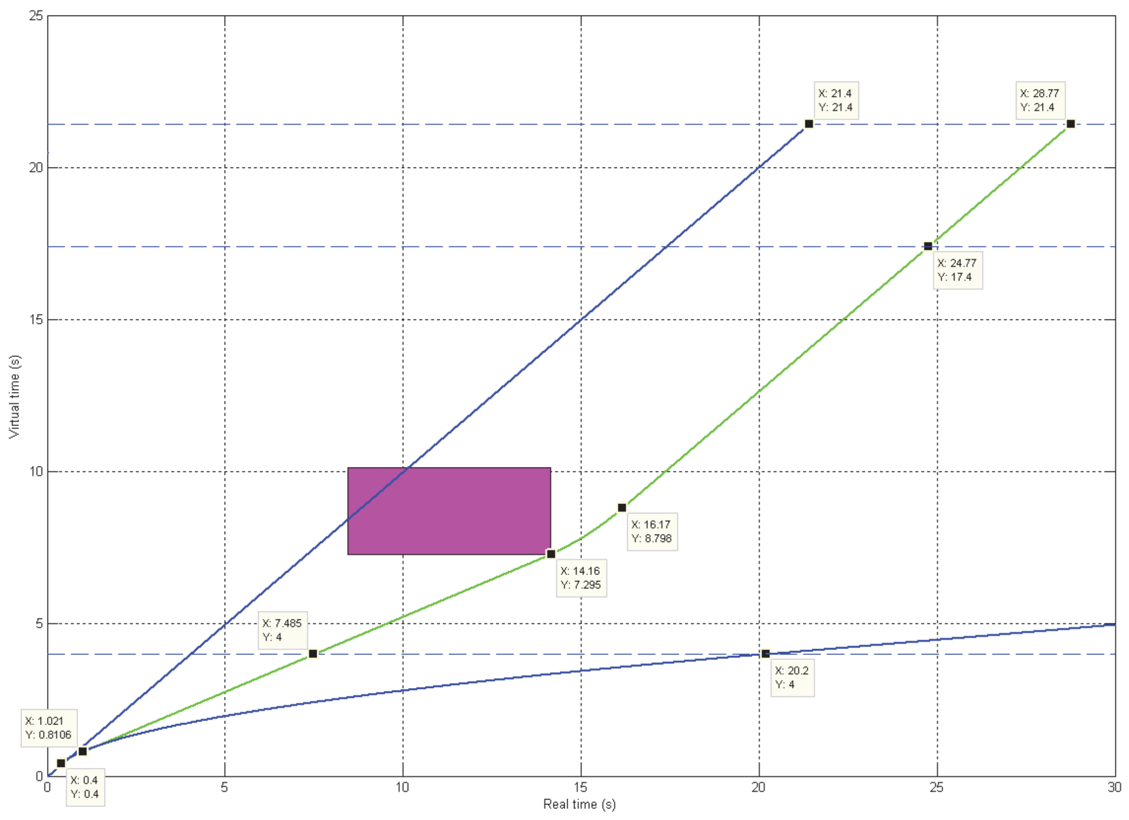


Fig. 20. Time-scaling for Robot#2

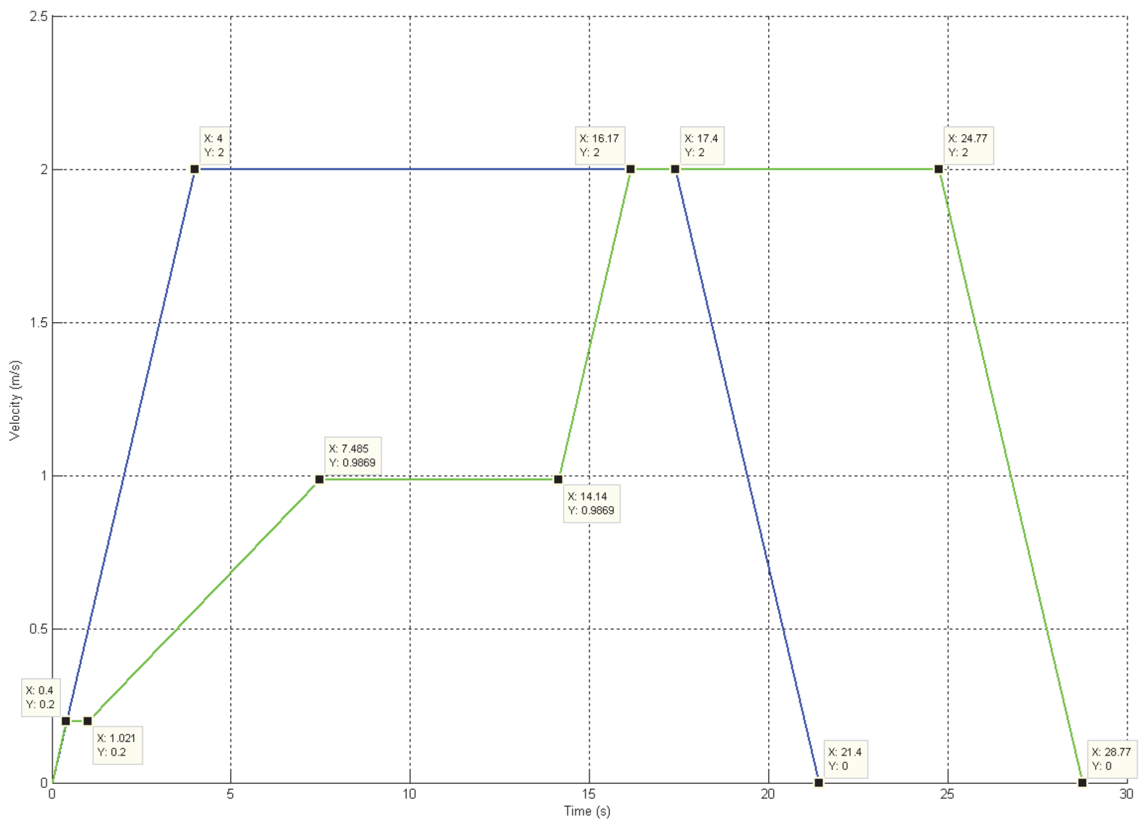


Fig. 21. Scaled velocity for Robot#2

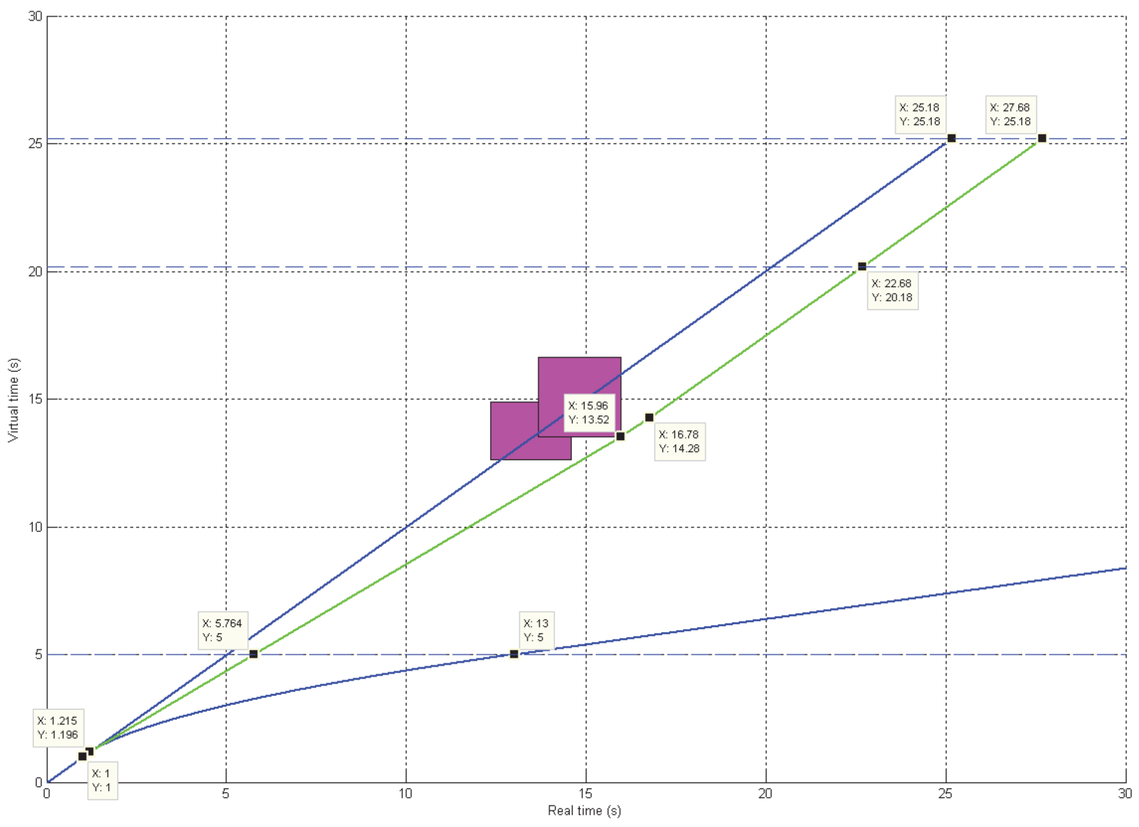


Fig. 22. Time-scaling for Robot#3

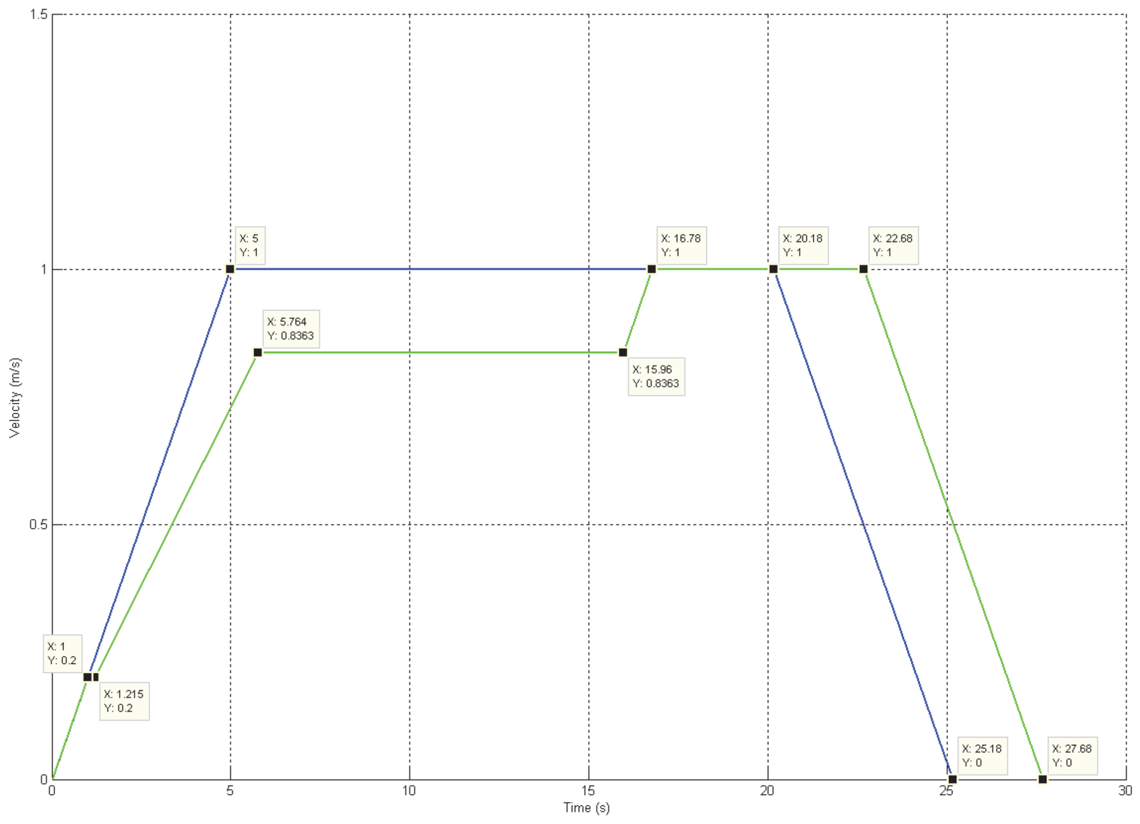


Fig. 23. Scaled velocity for Robot#3

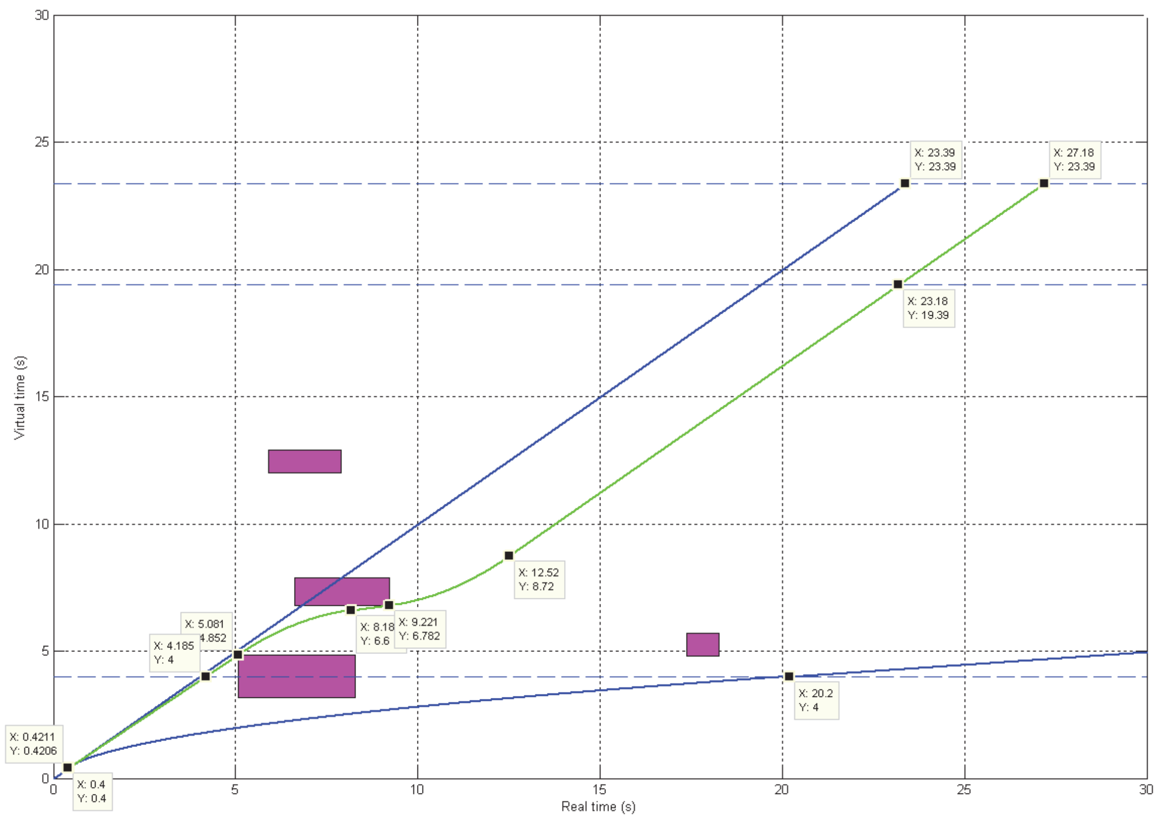


Fig. 24. Time-scaling for Robot#4

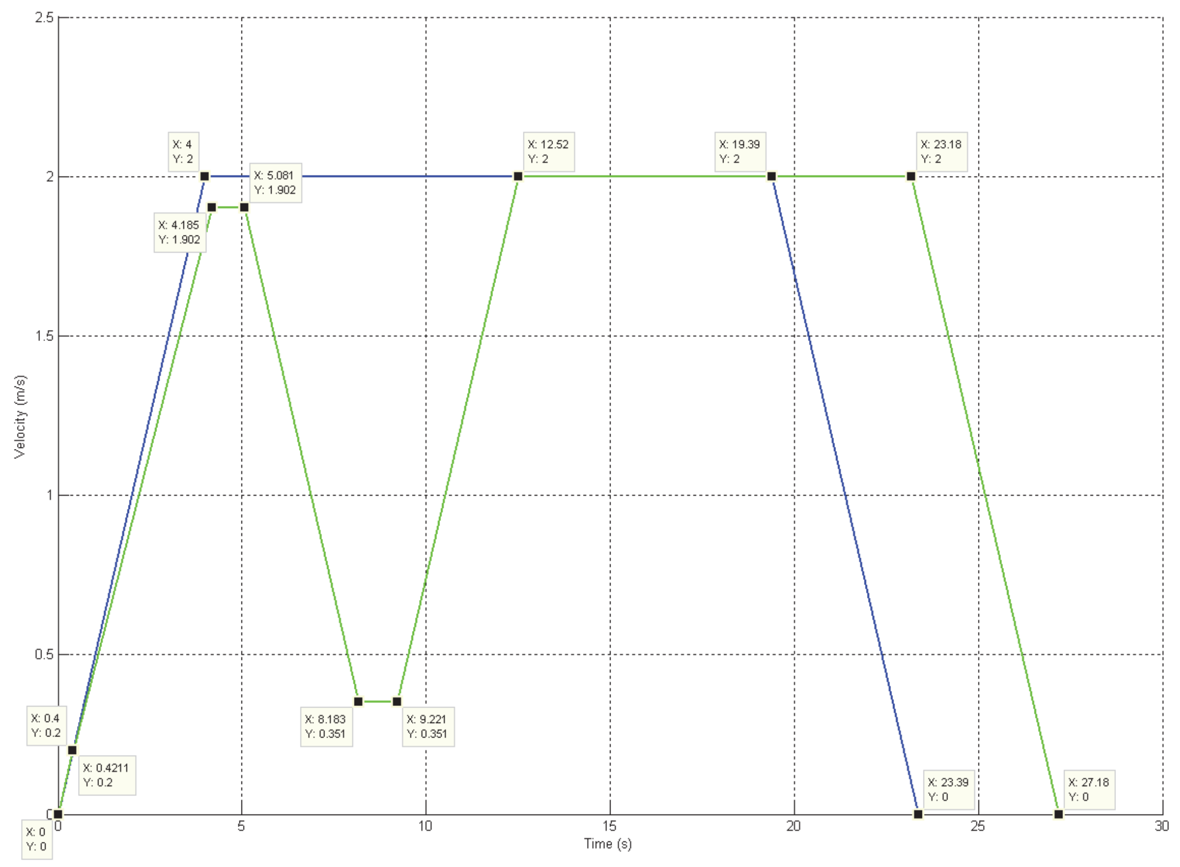


Fig. 25. Scaled velocity for Robot#4

Information on the relevant time instances, the derivate of the time-scaling function at those instances and the calculated scaled velocity values are shown in Table 2.

Robot#	x coordinate(m)	y coordinate(m)
Robot#1	2	18
	7	12
	14	12
	18	8
Robot#2	14	1
	14	7
	10	12
	1	12
Robot#3	1	8
	10	8
	15	18
Robot#4	5	19
	5	10
	7	5
	16	5

Table 1. Path of the robots

Robot#	Virtual time(s)	Real time(s)	$\dot{\theta}(t)$	Default velocity (m/s)	Scaled velocity(m/s)
Robot#2	0	0	1.0000	0	0
	0.4000	0.4000	1.0000	0.2000	0.2000
	0.8106	1.0213	0.4935	0.4053	0.2000
	4.0000	7.4846	0.4935	2.0000	0.9869
	7.2854	14.1425	0.4935	2.0000	0.9869
	8.7984	16.1686	1.0000	2.0000	2.0000
	17.4031	24.7733	1.0000	2.0000	2.0000
	21.4031	28.7733	1.0000	0	0
Robot#3	0	0	1.0000	0	0
	1.0000	1.0000	1.0000	0.2000	0.2000
	1.1957	1.2149	0.8363	0.2391	0.2000
	5.0000	5.7638	0.8363	1.0000	0.8363
	13.5249	15.9575	0.8363	1.0000	0.8363
	14.2764	16.7760	1.0000	1.0000	1.0000
	20.1803	22.6799	1.0000	1.0000	1.0000
	25.1803	27.6799	1.0000	0	0
Robot#4	0	0	1.0000	0	0
	0.4000	0.4000	1.0000	0.2000	0.2000
	0.4206	0.4211	0.9510	0.2103	0.2000
	4.0000	4.1850	0.9510	2.0000	1.9020
	4.8524	5.0813	0.9510	2.0000	1.9020
	6.5996	8.1832	0.1755	2.0000	0.3510
	6.7816	9.2205	0.1755	2.0000	0.3510
	8.7200	12.5184	1.0000	2.0000	2.0000
	19.3852	23.1836	1.0000	2.0000	2.0000
	23.3852	27.1836	1.0000	0	0

Table 2. Results of time-scaling for the robots

5. Conclusions

In this chapter a time-scaling based obstacle avoidance method was presented that is able to avoid dynamic obstacles as well as static obstacles. We designed and described an algorithm being able to find a time-scaling function that avoids all time-obstacles in the time-plane and thus ensures a collision free path in the robots' workspace.

The determined time-scaling function also satisfies the criteria according to the kinematic constraints prescribed for the control (velocity) inputs. The solutions of the differential equations describing the motion of the time-scaled units were presented with a method that makes finding a path between dedicated points in the time-plane possible.

Since the solutions of the differential equations are given in analytic form there is no need for time-consuming computations and determining the scaled velocity function is simple.

Planning on the time-plane is based on a novel and fast tree-building method. The path-planning algorithm is considered to be complete since it gives a definite answer in a finite time. All components of the algorithm are well suited for real time implementation.

Should time-scaling fail to work we still have a method to avoid collision with delaying the robots. Fig. 24 shows a scenario where a better solution is achieved by using time-scaling than simply delaying the robot.

6. Acknowledgments

This work is connected to the scientific program of the "Development of Quality-Oriented and Harmonized R+D+I Strategy and Functional Model at BME" project. This project is supported by the New Hungary Development Plan (Project ID: TÁMOP-4.2.1/B-09/1/KMR-2010-0002). The work is partially supported by the Hungarian Scientific Research Fund under grant OTKA K71762.

7. References

[1] J.C. Latombe. *Robot Motion Planning*. Kluwer Academic Publishers, Boston, MA, 1991.

[2] Yongguo Mei; Yung-Hsiang Lu; Hu, Y.C.; Lee, C.S.G. Energy-efficient motion planning for mobile robots. In: Proceedings of the IEEE Int. Conf. on Robotics and Automation, pages: 4344-4349. 2004.

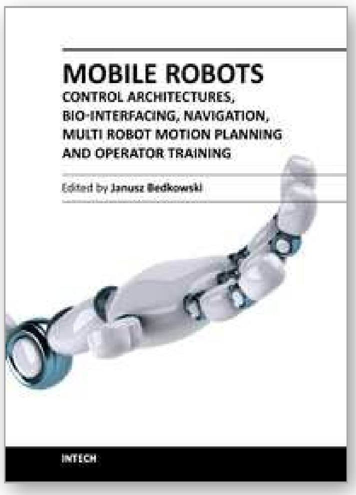
[3] V. Kunchev, L. Jain, V. Ivancevic, and A Finn. Path planning and obstacle avoidance for autonomous mobile robots: A review. Volume 4252 of Lecture Notes in Artificial Intelligence, pages 537–544. SpringerVerlag, 2006.

[4] F. Cuesta and A. Ollero, Intelligent Mobile Robot Navigation, ser. Springer Tracts in Advanced Robotics. Heidelberg: Springer, 2005, vol. 16.

[5] J. Reif and M. Sharir, "Motion planning in the presence of moving obstacles", Journal of the Association for Computing Machinery, vol. 41, no. 4, pp. 764–790, 1994.

[6] S. B. Moon and S. Ahmad, "Time scaling of cooperative multirobot trajectories", IEEE Transactions on Systems, Man, and Cybernetics, vol. 21, no. 4, pp. 900–908, 1991.

[7] M. Peasgood, C.M. Clark and J. McPhee, "A Complete and Scalable Strategy for Coordinating Multiple Robots Within Roadmaps", IEEE Transactions on Robotics, vol. 24, no 2, pp. 283-292, 2008.
[8] H. Li, S. X. Yang and M. L. Seto, "Neural-Network-Based Path Planning for a Multirobot System With Moving Obstacles", IEEE Transactions on Systems, Man and Cybernetics-Part C: Applications and Reviews, vol. 39, no. 4, pp. 410-419, 2009.
[9] K. G. Jolly, R. Sreerama Kumar, and R. Vijayakumar, "A Bezier curve based path planning in a multi-agent robot soccer system without violating the acceleration limits", Robotics and Autonomous Systems, vol. 57, pp. 23-33, 2009.
[10] Steven M. LaValle, "Planning algorithms" , Cambridge University Press, 2006


MOBILE ROBOTS
CONTROL ARCHITECTURES, BIO-INTERFACING, NAVIGATION, MULTI ROBOT MOTION PLANNING AND OPERATOR TRAINING
Edited by Janusz Bedkowski
INTECH

Mobile Robots - Control Architectures, Bio-Interfacing, Navigation, Multi Robot Motion Planning and Operator Training
Edited by Dr. Janusz Będkowski

ISBN 978-953-307-842-7
Hard cover, 390 pages
Publisher InTech
Published online 02, December, 2011
Published in print edition December, 2011

The objective of this book is to cover advances of mobile robotics and related technologies applied for multi robot systems' design and development. Design of control system is a complex issue, requiring the application of information technologies to link the robots into a single network. Human robot interface becomes a demanding task, especially when we try to use sophisticated methods for brain signal processing. Generated electrophysiological signals can be used to command different devices, such as cars, wheelchair or even video games. A number of developments in navigation and path planning, including parallel programming, can be observed. Cooperative path planning, formation control of multi robotic agents, communication and distance measurement between agents are shown. Training of the mobile robot operators is very difficult task also because of several factors related to different task execution. The presented improvement is related to environment model generation based on autonomous mobile robot observations.