12-20-2004

A Performance and Schedulability Analysis of an Autonomous Mobile Robot

Ala' Adel Qadi
University of Nebraska at Lincoln, aqadi@cse.unl.edu

Steve Goddard
University of Nebraska – Lincoln, goddard@cse.unl.edu

Jiangyang Huang
University of Nebraska - Lincoln, jyhuang@unl.edu

Shane Farritor
University of Nebraska - Lincoln, sfarritor@unl.edu

University of Nebraska–Lincoln, Computer Science and Engineering
Technical Report TR-UNL-CSE-2004-0015
Issued 12/20/2004

A Performance and Schedulability Analysis of an Autonomous Mobile Robot

Ala′ Qadi Steve Goddard
Computer Science & Engineering
University of Nebraska–Lincoln
Lincoln, NE 68588-0115
{*aqadi, goddard*}*@cse.unl.edu*

Jiangyang Huang Shane Farritor
Mechanical Engineering
University of Nebraska–Lincoln
Lincoln, NE 68588-0656
{*jyhuang, sfarritor*}*@unl.edu*

Technical Report TR-UNL-CSE-2004-0015
December 2004

Keywords: real-time systems, robotics, highway safety marker, applied real-time scheduling theory

Abstract

We present an autonomous, mobile, robotics application that requires dynamic adjustments of task execution rates to meet the demands of an unpredictable environment. The Robotic Safety Marker (RSM) system consists of one lead robot, the foreman, and a group of guided robots, called robotic safety markers (a.k.a., barrels). An extensive analysis is conducted of two applications running on the foreman. Both applications require adjusting task periods to achieve desired performance metrics with respect to the speed at which a system task is completed, the accuracy of RSM placement, or the number of RSMs controlled by the foreman. A static priority scheduling solution is proposed that takes into consideration the strict deadline requirements of some of the tasks and their dynamic periods. Finally, a schedulability analysis is developed that can be executed online to accommodate the dynamic performance requirements and to distinguish between safe operating points and potentially unsafe operating points.

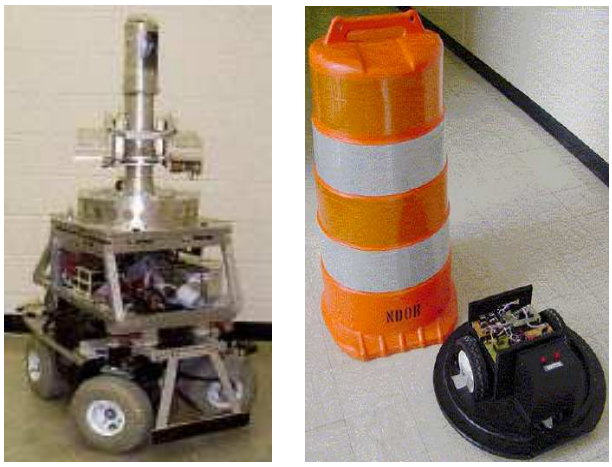
1. Introduction

Applying traditional real time systems scheduling theory to robotic applications is not a new concept. Examples of applying the classic periodic task model to robotics can be found in [9, 2, 11, 12, 5]. The Robotic Safety Marker (RSM) system [4, 14, 13], however, introduces a different real-time scheduling problem. The RSM system is a mobile, autonomous, robotic, real-time system that automates the placement of highway safety markers in hazardous areas, thereby eliminating risk to human workers. The RSMs operate in mobile groups that consist of a single lead robot—called the foreman—and worker robots—called RSMs—that carry a highway safety marker, commonly called a barrel.

Control of the RSM group is hierarchical and broken into two levels—global and local control—to reduce the per-robot cost. The foreman robot performs global control. To move the robots, the foreman locates each RSM,

plans its path, communicates destinations points (global waypoints), and monitors performance. Local control is distributed to individual RSMs, which do not have knowledge of other robots and only perform local tasks. Figure 1 shows a picture of the current prototypes. While the RSM is in near final form, the foreman prototype is based on a robot that was used for a prior project, and serves as a proof-of-concept platform for the global control of the RSM system.

Our previous work considered the problem of detecting the RSMs and sending them waypoints [14]. However at that early stage, we did not allow the foreman to move and plan its own path while controlling the RSMs. We have now implemented the foreman motion and path control in addition to a path prediction/correction algorithm for controlling the RSMs. Combining these activities introduced a whole different scheduling problem than previously considered in [14].



(a) The prototype foreman.

(b) A prototype RSM (a.k.a., barrel robot).

Figure 1. Prototype RSM foreman and worker robots.

In this work we consider performance with respect to the speed at which the foreman (and therefore the group) can move as well as the accuracy of RSM path prediction and correction. In fact, the execution rate of many of the tasks are directly related to these performance criteria. The dynamic parameters of the new combined system and desired performance, however, can lead to overload conditions. That is, the system is not schedulable unless the performance of one of the system activities is reduced.

This paper presents a detailed analysis of the robotics application, the relationship between task periods and performance, and an online schedulability test that can be used to distinguish between safe operating points and potentially unsafe operating points. The analysis and online schedulability test provide a framework for a future application-level control algorithm that can make dynamic performance/schedulability tradeoffs. For example, using the analysis framework presented here, one option to resolving an overload condition might be for the foreman to move slower, which gives it more time to compute a safe path for itself or to compute more accurate paths for the RSMs. Another option to eliminating overload might be to reduce the accuracy of RSM placement.

A third option is to reduce the number of RSMs under the foreman's control. This work provides the framework required to identify overload conditions and evaluate the impact of various corrective actions.

The rest of this paper is organized as follows. Section 2 presents an overview of the foreman's main control units. Section 3 describes the path planning and speed control task set, related challenges, and the relationship between task periods and the speed at which the robot can safely move, which depends on sensors and obstacles in the environment. Section 4 describes a tradeoff that must be made between the accuracy of RSM placement and the rate at which their movement must be monitored by the foreman. Section 5 presents a real-time scheduling analysis for the system that can be executed online. Finally, Section 6 presents a short conclusion.

2. The Foreman Design

It is the foreman's job to locate each RSM, plan their individual paths, communicate destination points (global waypoints), and monitor the performance of each RSM. The foreman is also autonomous. Thus, it has to plan its own path and motion while performing the tasks related to RSM control. These activities are modeled as real tasks; Sections 3, 4 and 5 provide more detail on these these activities and their real-time attributes.

The foreman consists of seven units, as shown in Figure 2: main unit, power unit, communication unit, localization unit, sonar unit, sensor unit and motor unit. The main unit, which is the central processing unit of the foreman, consists of a PC/104-*Plus* embedded processor system with RS232 and RS485 serial ports and a parallel port interface. The operating system for the main unit is Windows CE. The power unit consists of two 12V batteries and DC converters. It supplies $\pm$12V and $\pm$5V voltages for the system. A standard RS232 serial port is used to interface with the communication unit—a 9XStream^{TM} 900 MHz OEM RF module.

The foreman uses a SICK laser scanner, with an effective range of 32 meters, to scan the horizon for the RSMs. The main unit receives the laser data through a high speed RS485 serial port and processes the data using a modified Hough algorithm, as described in [14], to determine the RSM positions. The foreman then plans their paths accordingly.

The sonar unit consists of a ring of 24 active sonar sensors, with 15° separation, that provides 360° coverage around the foreman. Each sonar sensor has a maximum effective range of 7.25 meters. The main unit pings the 24-sonar ring in the sonar unit through the parallel port and relies on the collected data to safely navigate the foreman. The sonar sensors are used instead of the laser scanner to collect environment information to plan the foreman's own path because it is much faster to process the sonar signal with high accuracy than to process the laser scanner data, which is computationally intensive (Sections 3 and 4 present task execution time details).

The sensor unit consists of a Rabbit 3000 microprocessor, four encoders and a gyro sensor. The Rabbit 3000

microprocessor reads sensors and localizes the foreman. At the same time, the main unit communicates with the Rabbit 3000 microprocessor via winsock networking. The motor unit connects to the main unit via a DM5406, an analog I/O data module with two DAC outputs and 16 digital I/O lines. In the motor unit, a PIC16F84 microcontroller controls the steering and driving motors.

3. Foreman Path Planning and Speed Control

The foreman depends on the sonar sensors to plan its path by processing the sonar signals to determine the presence of obstacles and their distance. The maximum speed at which the foreman can travel is related to the rate the sonar signals can be gathered and processed. If the foreman moves faster than the sonar signals can be processed, then the motion will be unsafe because there might be an obstacle in the path that will be undetected at that rate.

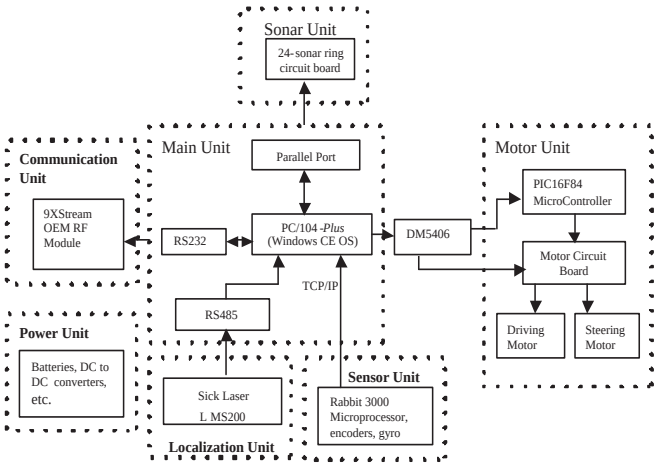


Figure 2. Main control units of the foreman.

3.1 The Motion Control Task Set

The 24 sonar sensor signals are pinged in sequence with a delay of $2ms$ between consecutive sensors to eliminate crosstalk. The motion control for the foreman code can be modeled as a set of periodic tasks: 24 tasks for sending sonar signals (one for each sensor), similarly another 24 tasks for receiving sonar signals and a path-plan/speed-control task. These tasks all execute with a common period p_s, which is called the scan period. Each sonar send task sends a command to its corresponding sonar sensor to transmit its signal. Each sonar receive task reads the corresponding sonar sensor after the signal is echoed back to the sensor. The parameters for the motion control task set are shown in Table 1, where e, p, d, ϕ and *max* J are the execution time, period, relative deadline, phase, and maximum jitter respectively. The phase represents the earliest possible release time for a task and maximum jitter is the maximum delay between the phase and the actual release time of the task. In this task set, jitter is caused by delays in receiving sonar signals, which are primarily dependent on the location of objects in the environment.

The sonar send tasks are released with a delay between them to eliminate crosstalk. The phase of these tasks, ϕ_{send_i}, is given by Equation (1), where i is the task index, τ is the delay used to eliminate crosstalk between

Task	e	p	d	ϕ	*max J*
Sonar-Send$_i$	$e_{send} = .085ms$	p_s	e_{send}	ϕ_{send_i}	0
Sonar-Receive$_i$	$e_{recv} = .03ms$	p_s	$e_{send} + e_{recv}$+*max* Δt	ϕ_{recv_i}	*max* Δt
Path-Plan/Speed-Control	$e_{plan} = 1.32ms$	p_s	e_{plan}	ϕ_{plan}	0

Table 1. Motion control task set. Phase parameters ϕ_{send_i}, ϕ_{recv_i}, and ϕ_{plan} are defined by Equations (1), (2), and (5) respectively. The maximum jitter parameter $max\ \Delta t$ is defined by Equation (4).

consecutive sonar send tasks and e_{send} is the execution time of a sonar send task. These tasks have zero jitter and are required to execute as soon as they are released; hence a relative deadline equal to its execution time.

$$\phi_{send_i} = (i-1) \cdot (\tau + e_{send}) \qquad 1 \leq i \leq 24 \tag{1}$$

$$\phi_{recv_i} = (i-1) \cdot \tau + i \cdot e_{send} \qquad 1 \leq i \leq 24 \tag{2}$$

$$\Delta t = \frac{2 \cdot D_{obstacle}}{340m/s} \tag{3}$$

$$max\ \Delta t = \frac{2 \cdot D}{340m/s} \tag{4}$$

$$\phi_{plan} = p_s - e_{plan} \tag{5}$$

A sonar receive task is not released until its corresponding sonar send task has been executed and the signal is reflected back, which is called an echo. Equation (2) gives the phase for any sonar receive task i. The jitter of a sonar receive task, however, is dependent on the time delay between the transmission of a sonar signal and the reception of its echo, denoted as Δt. If an object is $D_{obstacle}$ meters away, the echo time delay can be computed using Equation (3) where the speed of sound is assumed to be 340 meters/second.[1] ($D_{obstacle}$ is multiplied by 2 in Equation (3) because the signal has to travel $D_{obstacle}$ meters before it is reflected back). Since we do not know the distance to objects *a priori*, a minimum distance, D, at which an object must be detected for the path-plan/speed-control task to safely control the robot's motion is defined. The maximum echo time delay—and hence maximum jitter—is then computed using D in Equation (4). If an object is farther than D meters away, the path-plan/speed-control task does not need to know about it because it will not provide any additional useful data in this scan period. Thus, receipt of an echo after *max* Δt time units is ignored.

The path-plan/speed-control task computes the path of the foreman and controls its speed based on the data collected from the sonar signals. The design of the control system is based on the assumption that this task executes at the end of the scan period, but after all of the useful sonar signals have been received. This is an artifact

[1]The actual speed of sound varies slightly depending on environmental conditions.

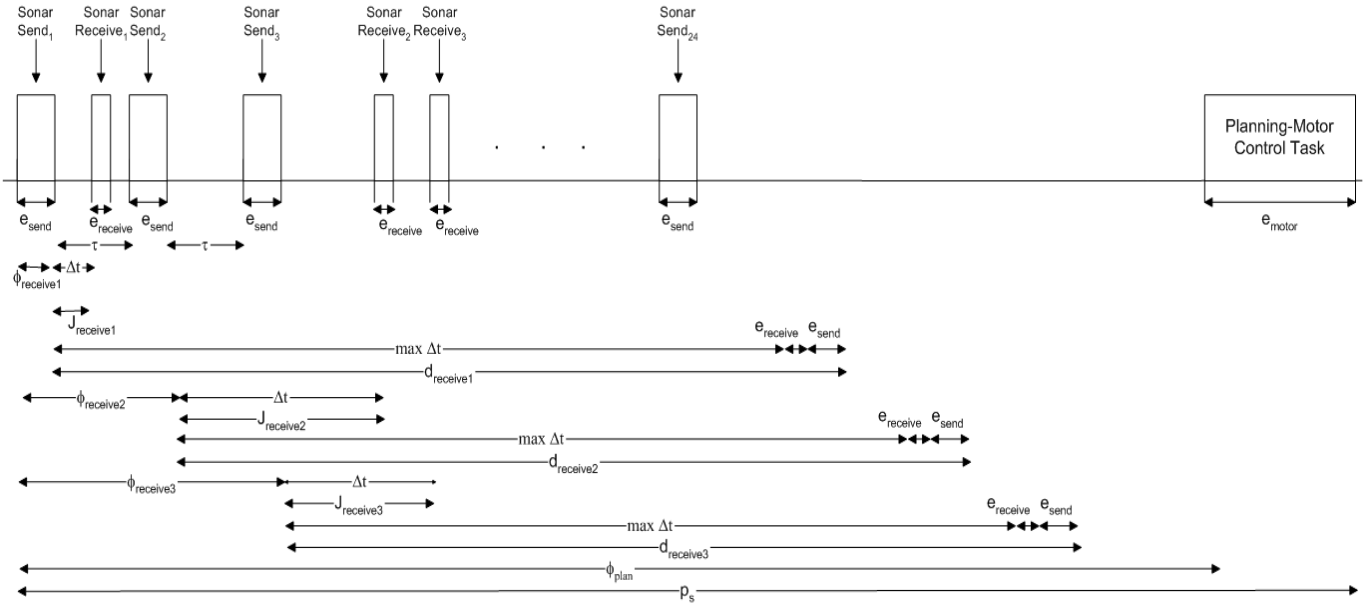


Figure 3. Example motion control task set release times.

of a previous control-loop design approach in which the environment was scanned as fast as possible. However, the application of real-time scheduling theory to this system lets us reduce the scan rate so that processor capacity can be safely allocated to other system tasks. Thus, to simplify the application of real-time technology to this system, the phase of the path-plan/speed-control task is set to coincide with the end of the scan period, as shown by Equation (5). To ensure the task completes by the end of the period, its relatively deadline must then be set equal to its execution time.

Figure 3 demonstrates possible release times and task executions for the motion control task set in a period of p_s time units.

3.2 Continuous Motion Planning

The goal of this task set is to achieve a continuous safe movement of the foreman at the maximum safe speed while still being able to meet all task deadlines. To achieve continuous movement, the path is divided into a number of segments. Each segment is delineated by a scan point that marks the beginning of a scan period. We must, however, allow enough time for all motion control processing to complete within the scan period (i.e., before arriving at the next scan point). Thus, the length of the scan period, p_s, is dependant on the traveling speed of the foreman and the desired minimum object detection distance D. To simplify control, it is desirable for the foreman to have a constant speed between any two scan points. Under these constraints and assumptions,

Equation (6) defines a lower bound on p_s that is required to safely control the foreman's motion.

$$\begin{aligned} p_s &\geq \phi_{recv_{24}} + d_{recv_{24}} + e_{plan} \\ &= 23 \cdot \tau + 24 \cdot e_{send} + max\ \Delta t + e_{recv} + e_{send} + e_{plan} \\ &= 23 \cdot \tau + 24 \cdot e_{send} + \frac{2 \cdot D}{340m/s} + e_{recv} + e_{send} + e_{plan} \\ &= 23 \cdot \tau + 25 \cdot e_{send} + \frac{2 \cdot D}{340m/s} + e_{recv} + e_{plan} \end{aligned} \tag{6}$$

We now quantify the relationship between p_s, the foreman's speed, D, and objects in the environment. Let each scan point in the foreman's path be denoted S_i. At least p_s time units must elapse before the foreman leaves point S_i and arrives at point S_{i+1}. A Scanning Zone, or simply $Zone\ i$, is defined as the area we can travel safely in without the need for another sonar scan. Scanning $Zone\ i$ is the area between point S_i and point S_{i+1}. The foreman achieves continuous motion by scanning $Zone\ i+1$ while traveling through $Zone\ i$. Of course, this requires that the foreman scan $Zone\ 0$, the first zone, before starting its movement. Figure 4 shows the distribution of scan points in time and distance from the moment the system starts (*note: v_{max} is the foreman's maximum speed*).

Let v_{maxi} denote the *maximum safe speed* at which the foreman can move through $Zone\ i$ while guaranteeing a continuous, crash-less motion. Obstacles in the environment, p_s and τ, all constrain v_{maxi}. To illustrate these constraints on v_{maxi}, we consider two different cases: an obstacle free environment, and an environment in which obstacles exists.

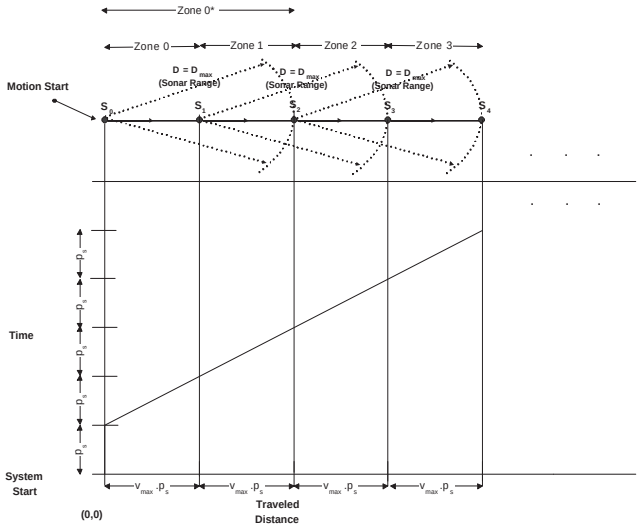


Figure 4. Scanning point distribution in time and space with no obstacles.

Case 1: Obstacle Free Environment.

Let M_{safe} represent the maximum distance the robot can move safely. In this case, M_{safe} is the minimum distance scanned by the sonar sensors: D. As we can see in the top part of Figure 4, at time $t = 0$ the foreman is initially at scan point S_0. We start our initial scan but do not start the motion until the end of the first scan period. At this time $M_{safe} = D$ because the foreman does not start the motion until the end of p_s time

units. Therefore $Zone\ 0^*$ extends to a distance of D. We can keep moving safely in $Zone\ 0^*$, but this will imply the need to stop at point S_2 at the end of $Zone\ 0^*$ to scan $Zone\ 2$. We can avoid the stop if we divide $Zone\ 0^*$ into two smaller zones—$Zone\ 0$ and $Zone\ 1$—and scan $Zone\ 2$ while moving in $Zone\ 1$.

However, since we start gathering the sonar data (for $Zone\ i$) p_s time units before we reach $Zone\ i$, the size of the newly scanned region will be reduced, as illustrated in Figure 4. The worst case is if the oldest data gathered by a sonar sensor is in the direction of motion. Therefore, $M_{safe} = D - v \cdot p_s$ where v is the foreman's speed. The path in this case should be divided into a number of path scan points with a distance of M_{safe} between any two scan points (Figure 4).

Since at least p_s time units must separate each scanning point, we can calculate v_{max} using Equation (7), assuming constant speed between any two scan points.

$$v_{max} = \frac{M_{safe}}{\textit{Avilable Time to Complete the Motion}} = \frac{D - v_{max} \cdot p_s}{p_s} = \frac{D}{p_s} - v_{max} = \frac{D}{2 \cdot p_s} \tag{7}$$

Let D_{max} denote the maximum effective distance of the sonar sensors. In this application $D_{max} = 7.25m$. To calculate the maximum speed associated with the maximum detectable sonar range D_{max}, we need first to use Equation (6) to calculate the minimum value of p_s for D_{max}. $D = D_{max} = 7.25m$. The minimum value for p_s occurs when the minimum value for τ, which is $2ms$ is used. Substituting $2ms$ in Equation (6), we get $p_s = 92.1ms \simeq 93ms$. Therefore $v_{max=}7.25m/(2 \times 93ms) = 38.98m/s$. The final version of the foreman should be able to travel at speeds in the range of 10 to $20m/s$. This analysis shows that, at any speed less than or equal to $38.98m/s$, the processor will be able to process the data and plan the path using the minimum period of p_s if no obstacles exist.

If at any scan point S_i we change the sonar period p_s or change the sonar detection range D, then Equation (7) becomes

$$v_{maxi+1} = \frac{D_i - v_i \cdot p_{si}}{p_{si+1}}, \tag{8}$$

where v_{maxi+1} is the next maximum speed, p_{si+1} is the next sonar period, D_i is the current sonar detection distance, v_i and p_{si} are the current speed and sonar period respectively.

Case 2: Obstacles Exist. The existence of an obstacle in the foreman's path introduces constraints on motion planing:

- The maximum distance the robot can safely move is not the maximum distance that can be measured with the sonar, but rather the distance between the obstacle and the foreman, $D_{obstacle}$. This distance is not related

to the length of the period as it was in the previous case.

- The second difference is that because of obstacles in the path we may have to change the foreman's speed at the path scan point when we discover the obstacle. This means that the maximum speed for the zone after the obstacle is also dependent on the speed before reaching the obstacle. Note that this is similar to treating the obstacle as a scan point because we cannot see behind the obstacle.

As in Case 1, the system starts at scan point S_0 with $D = D_{max}$ and $p_s =$ ***DefaultValue***. The initial speed of the robot is then set using Equation (7) with $D = D_{max}$ if no object was detected and $D = D_{obstacle}$ otherwise. After the robot starts moving it can detect obstacles at any distance less than or equal to $D - v \cdot p_s$ from scan point S_i, where v is the foreman's speed. Figures 5(a) and 5(b) show what happens when the foreman encounters an obstacle under various conditions. Figure 5(a) shows the case where we need to adjust the speed to keep both p_s and D the same. Figure 5(b) demonstrates the case where we reduce p_s and D.

First, consider the scenario shown in Figure 5(a). At path scan point S_i the robot was moving at speed v_i and using its maximum sonar signal range $D = D_{max}$. At point S_i the foreman starts scanning for obstacles in $Zone\ i+1$ while it is moving in $Zone\ i$. At point S_{i+1} (after p_s time units) the path-plan/speed-control task released in period p_{si} has just finished executing. At that point, the robot has detected that the obstacle exists in $Zone\ i+1$. Because of the obstacle, the next path point, which was planned to be at $S_{(i+2)org}$, will now move closer to point S_{i+2}, converting $Zone\ i+1$ to $Zone\ i+1^*$. The speed for $Zone\ i+1$ between S_{i+1} and $S_{(i+2)org}$ was originally set at point S_i, but because of the obstacle we have to readjust the speed at point S_{i+1}.

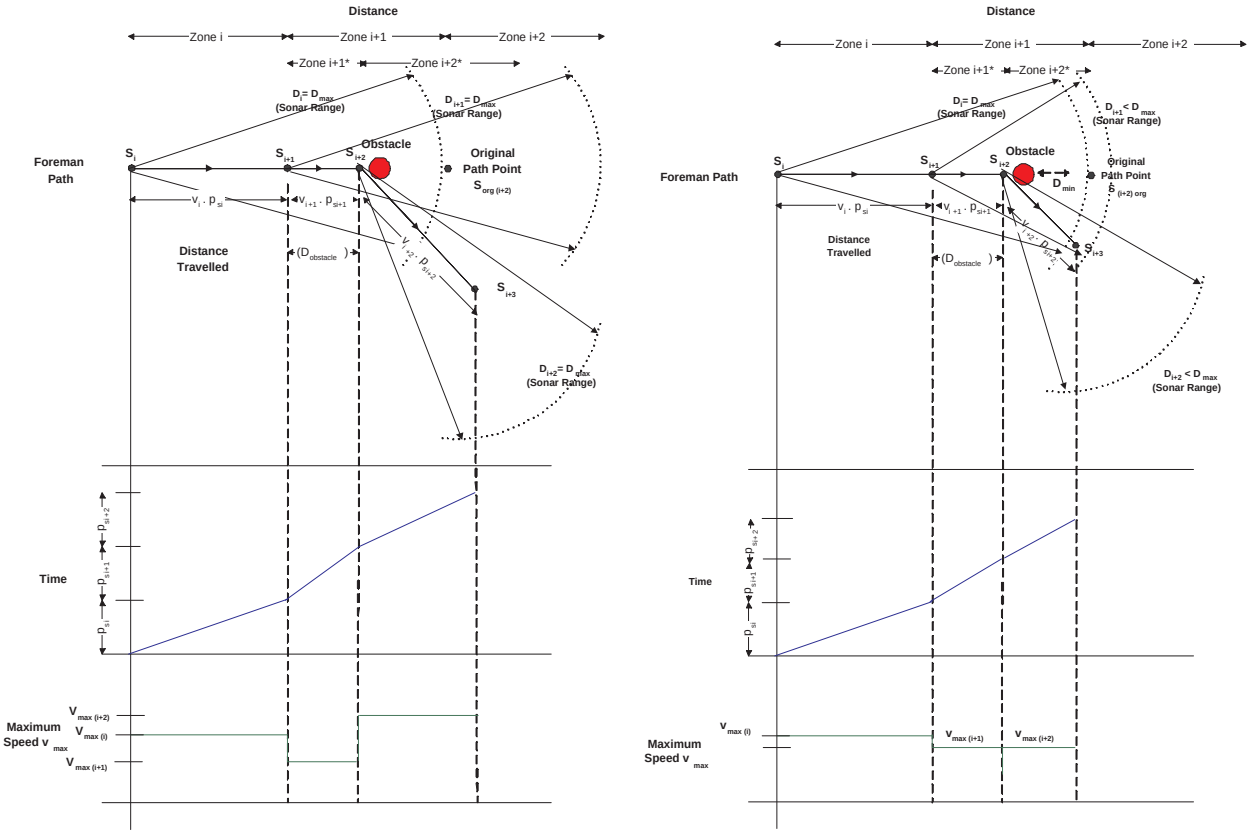
Since the scan must complete before reaching the obstacle, we need either to adjust the maximum speed or p_s to meet the new constraint. If the obstacle is at a distance $D_{obstacle} \leq v_i \cdot p_{si}$ the new constraint on the speed becomes

$$v_{maxi+1} = \frac{D_{obstacle}}{p_s}. \tag{9}$$

As shown in Figure 5(a), if we decide to keep $p_{si+1} = p_{si}$, $D = D_{max}$, we need to adjust v_{maxi+1} (downward) according to Equation (9) for $Zone\ i+1^*$ to meet the obstacle constraint. The maximum speed for $Zone\ i+2^*$ is then determined at point S_{i+2} using Equation (8). The maximum speed for this zone will increase since the robot will be able to travel a greater distance in the same time period p_s.

Now consider the scenario demonstrated in Figure 5(b). In this scenario, we have flexibility in p_s but we wish to maintain a constant maximum speed in $Zone\ i+1^*$ and $Zone\ i+2^*$ while still meeting the obstacle constraint. This is accomplished by adjusting D, p_s, or both.

The adjusted sonar detection distance should allow detection for a minimum safe distance beyond the obstacle.



(a) Constant p_s and D with speed adjustments at scan points.

(b) Constant speed (after the obstacle) with adjustments to p_s and D at scan points.

Figure 5. Adjusting the robot's operating points as obstacles are encountered.

Therefore the new constraint on D will be $D_{obstacle} + D_{min} \leq D = D_{i+1} \leq D_{max}$, where D_{min} is the minimum safe distance beyond the obstacle to be scanned. Once we calculate D_{i+1}, we can then calculate p_{si+1} from Equation (6), but we need to ensure that the robot does not reach the obstacle before the end of the next period. Therefore, v_{maxi+1} will be

$$v_{maxi+1} = \frac{D_{obstacle}}{p_{si+1}} \tag{10}$$

However, the goal is to derive a constant maximum speed in the next *two* zones, $Zone\ i+1^*$ and $Zone\ i+2^*$. The maximum speed for $Zone\ i+2^*$, determined at point S_{i+1}, can be calculated using Equation (8). Therefore,

$$\begin{aligned} & v_{maxi+1} = v_{maxi+2} \\ \Longrightarrow \quad & v_{maxi+1} = \frac{D_{i+1} - v_{maxi+1} \cdot p_{si+1}}{p_{si+2}} \\ \Longrightarrow \quad & \frac{D_{obstacle}}{p_{si+1}} = \frac{D_{i+1} - \frac{D_{obstacle}}{p_{si+1}} \cdot p_{si+1}}{p_{si+2}} \end{aligned}$$

$$\implies \qquad p_{si+2} = p_{si+1} \cdot \frac{D_{i+1} - D_{obstacle}}{D_{obstacle}} \tag{11}$$

Substituting p_{si+1} with its minimum value from Equation (6) in Equation (11) we get

$$p_{si+2} = \gamma \cdot \frac{D_{i+1}}{D_{obstacle}} + \frac{2 \cdot D_{i+1} \cdot (D_{i+1} - D_{obstacle})}{340 \cdot D_{obstacle}} \tag{12}$$

Where $\gamma = 23 \cdot \tau + 25 \cdot e_{send} + e_{recv} + e_{plan}$. However, we need to make sure that p_{si+2} is still bounded by Equation (6), that is

$$p_{si+2} \geq \gamma + \frac{2D_{i+2}}{340}. \tag{13}$$

Equating Equation (12) with the minimum value of p_{si+2} in Equation (13), yields Equation (14). This equation can be used to calculate the value of D_{min} that produces a constant maximum speed through $Zone\ i+1^*$ and $Zone\ i+2^*$.

$$\frac{2D_{i+1}^2}{340 D_{obstacle}} + \left(\frac{\gamma}{D_{obstacle}} - \frac{2}{340}\right) \cdot D_{i+1} - \frac{2D_{i+2}}{340} - 2\gamma = 0,\ D_{obstacle} + D_{min} \leq D_{i+1}, D_{i+2} \leq D_{max} \tag{14}$$

Since both D_{i+1} and D_{i+2} are unknown, a feasible solution to Equation (14) can be found if we set the value for one of the variables and calculate the other. After finding the value for D_{i+1}, values for p_{si+1} and p_{si+2} can be computed using Equation (6).

The two scenarios shown in Figure 5 and the equations presented in this section illustrate that the length of the scan period, the maximum traveling speed of the foreman, and the object detection distance are all interrelated. The sonar sensors provide a physical maximum object detection limitation of $7.25m$. Mission requirements have established a minimum safe scanning distance of $1m$ and a minimum traveling speed of $1m/s$ for the robot. These boundary values for D can be combined with Equations (6) and (7) and parameter values in Table 1 to derive a range for p_s: $55.34ms \leq p_s \leq 3650ms$.

The processor utilization for this task set is inversely proportional to the length of the scan period since the same (maximum) amount of work must be done in each scan period. Thus, one way to reduce processor load in an overload condition is to increase the scan period. This, in turn, may reduce the speed at which the robot can complete its mission, resulting in lower system performance. Another option to reducing processor load is to reduce the performance of the system with respect to control of the RSMs. This tradeoff is described next.

Task	e	p	d	ϕ	*max* J
Scanning	$12ms$	p_l	p_l	0	0
Detecting	$.0172 \cdot n^2 + .1695 \cdot n + 12.69$	p_l	p_l	0	0
Predicting	$e_{predict} = 3.8 \cdot n$	p_l	p_l	0	0
Planning	$16ms$	$1500ms$	$1500ms$	0	0
Way Point$_i$	$8.33ms$	$1500ms$	$1500ms$	0	0
Window Resizing	$2ms$	p_l	p_l	0	0

Table 2. RSM motion planning and tracking task set. The variable n represents the number of RSMs being controlled by the foreman.

4. RSM Motion Planning and Tracking

Recall that, in addition to planning its own path, the foreman is responsible for sending the RSMs path waypoints they need. The foreman is also responsible for guaranteeing that the RSMs stay on their respective paths. To achieve these goals, a laser scanner (with an effective range of 32 meters) is used to determine the position of each RSM. The foreman then computes and sends global waypoints to each RSM, which then computes a local path from the current waypoint to the next waypoint.

The RSM motion planning and tracking performed by the foreman can be modeled as a set of tasks and precedence constraints shown in the processing graph of Figure 6. The task attributes e, p, d, ϕ, and *max* J are listed in Table 2. A phase and jitter of 0 is assumed for the precedence-constrained tasks in this set. It would also be reasonable to set the phase of a task equal to the sum of the execution times of its predecessor tasks in the task graph. However, since all of the tasks execute on the same processor and each task is released as soon as its predecessor completes, the task can be modeled as though they were all released at the same instant with priority given to predecessor tasks in the task graph [6].

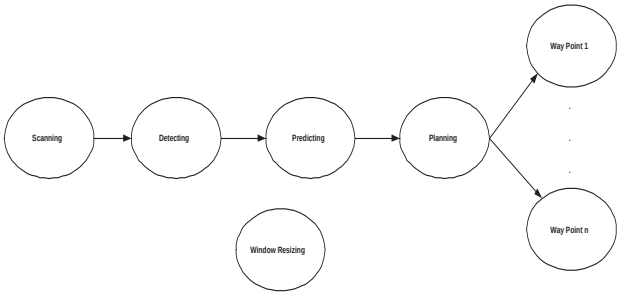


Figure 6. RSM motion planning and tracking processing graph.

This RSM motion planning and tracking task set is nearly the same task set that was analyzed in [14]. However, a path prediction task has been added to identify and correct deviations from the planned path in the actual path taken by RSMs. The rest of this section briefly describes the path prediction task and its impact on system performance. ([14] provides a description of the other tasks in this task set.)

4.1 The Prediction Algorithm

A prediction algorithm is used because the RSMs use dead reckoning to determine their position from a relative starting point; the RSMs have neither a compass nor a global position system (GPS) system to keep per unit costs low. Thus, to accurately correct error in the path taken by the RSMs, the foreman needs to know where each RSM will be at the next sampling period and how far it will have traveled from its current point. Moreover, in addition to being an essential part of an error correction mechanism, the prediction algorithm is useful for determining the current orientation angle for the RSMs since the they do not have a compass.

The RSMs move in a parabolic path determined by dead reckoning [4, 13]. Therefore the RSM path can be modeled by Equation (15).

$$y = a_2 x^2 + a_1 x + a_0 \tag{15}$$

At each scan we can approximate each RSM path by an equation of the form in Equation (15). The RSMs calculate different paths between each two way points. Therefore it is necessary to recalculate the path equation of each robot at each scan. The coefficients $a_2, a_{1,} a_0$ are calculated using the least square polynomial approximation described briefly using Equations (16), (17) and (18).

In general, in order to approximate a polynomial $p(x)$ of degree n, we will need $n + 1$ data points to get a unique solution. Let $(x_0, y_0), (x_1, y_1) \ldots (x_n, y_n)$ be the set of data points for which we want to approximate the polynomial function $p(x)$. Let $p(x)$ be given by the form

$$p(x) = \sum_{j=0}^{n} a_j x^j. \tag{16}$$

Then we can find the coefficients $a_0, a_{1 \ldots} a_n$ by solving the following system of linear equations

$$\begin{aligned}
&s_0 a_0 + s_1 a_2 + \cdots + s_n a_n = \sum_{k=0}^{n} y_k \\
&s_1 a_0 + s_2 a_2 + \cdots + s_{n+1} a_n = \sum_{k=0}^{n} x_k y_k \\
&.. \\
&.. \\
&.. \\
&s_n a_0 + s_{n+1} a_2 + \cdots + s_{2n} a_n = \sum_{k=0}^{n} x_k^n y_k,
\end{aligned} \tag{17}$$

where

$$s_j = \sum_{k=0}^{n} x_k^j. \tag{18}$$

The linear system in Equation (18) can be solved using various methods depending on the size of the system. Here we have a system of three equations since $p(x)$ is of degree 2. Since the system is very small, we used Cramer's rule to generate a solution.

In order to predict the next point on the curve $y(x)$ where the robot is going to be after Δt time units, we need to know the vector velocity of the robot. The direction angle of the velocity at point p_i can be calculated from the tangent of $y(x)$ at p_i. The velocity direction will be the same as the robot orientation at point p_i. The direction angle of the velocity θ at point p_i can be calculated from the tangent of $y(x)$ at $p_{i.}$

$$\tan\theta = \left.\frac{dy}{dx}\right|_{x=x_i} = 2a_2x_i + a_1$$

Therefore,

$$\theta = \tan^{-1}(2a_2x_i + a_1) \tag{19}$$

It is hard do determine exactly the magnitude of the velocity at point p_i, but if Δt is small enough so that Δx is small, then a good approximation is to use the velocity for the previous Δt for the next Δt. The magnitude of the velocity $|v|$ can be calculated as

$$|v| = \frac{\Delta l}{\Delta t} = \frac{l_i - l_{i-1}}{t_i - t_{i-1}} \tag{20}$$

Where Δl is the total distance traveled by the robot in time Δt. However, since the robot is moving in a parabolic path, we need to use integration to get the length of the curve between x_i and x_{i-1}.

$$|v| = \frac{\int_{x_{i-1}}^{x_i} \sqrt{1 + y'(x)^2}dx}{t_i - t_{i-1}} = \frac{\int_{x_{i-1}}^{x_i} \sqrt{1 + (2a_2x + a_1)^2}dx}{t_i - t_{i-1}} \tag{21}$$

If we use the shifting property of the definite integral to simplify the integral in Equation (21), we will get

$$\begin{aligned} |v| &= \frac{\int_{x_{i-1}-\frac{a_1}{2a_2}}^{x_i-\frac{a_1}{2a_2}} \sqrt{1 + (2a_2x)^2}dx}{t_i - t_{i-1}} \\ &= \frac{\frac{1}{2}|x|\sqrt{1+(2a_2x)^2} + \frac{1}{4|a_2|}\ln(2|a_2x| + \sqrt{1+(2a_2x)^2})\Big]_{x_{i-1}-\frac{a_1}{2a_2}}^{x_i-\frac{a_1}{2a_2}}}{t_i - t_{i-1}} \end{aligned} \tag{22}$$

Now we calculate Δx as

$$\Delta x = |v| \cos\theta \cdot \Delta t$$

The predicted value for the x coordinate of the next point will be

$$x_{i+1} = x_i + \Delta x$$

The y_{i+1} coordinate can be calculate by substituting x_{i+1} in the path equation, Equation (15).

$$y_{i+1} = a_2 x_{i+1}^2 + a_1 x_{i+1} + a_0$$

We need to use a minimum of three points for the prediction; we used four points in our implementation.

We now have two special cases that we need to consider.

Case 1: Coefficient a_2= 0. This case will happen if the robot was moving in a straight line for the last number of points considered. For the approximation in this case:

$$\theta = \tan^{-1} a_1$$

$$\Delta l = \sqrt{1 + a_1^2}(x_i - x_{i-1}).$$

Case 2: All coefficients are 0. This will happen only if the robot is idle and not moving. In this case:

$$\theta_{i+1} = \theta_i,\ x_{i+1} = x_i, y_{i+1} = y_i.$$

The previous analysis assumes that the foreman is static, but actually it is moving. The prediction algorithm can be extended to handle the dynamic case easily if we know the position of the foreman relative to a global coordinate system at each laser scan. We have added a Rabbit 3000 processor as an auxiliary processer, which only does dead reckoning to get the foreman's position, and a compass is used to determine the orientation. The coordinates for the RSMs, taken from laser scanner and the detecting task, need to be transformed into a global coordinate system before they can be used to predict the next point because they are relative to the current companion position. Equations (23) and (24) show how the coordinates are translated into a global coordinate

system

$$x_i = (x_i - X_c) \cdot \cos\psi + (y_i - Y_C) \cdot \sin\psi \tag{23}$$

$$y_i = -(x_i - X_c) \cdot \sin\psi + (y_i - Y_C) \cdot \cos\psi \tag{24}$$

where X_C, Y_C, ψ are companion coordinates and orientation angle in the global coordinate system respectively.

4.2 Error Analysis

The path prediction is based on a differential, predictor–corrector algorithm. Thus, the accuracy of the algorithm increases if the points used in the prediction are closer together. Scanning the horizon at a faster rate yields points closer to each other. Hence the error in the prediction is related to the rate at which RSM path data is collected from the laser scanner. Figures 7(a) and 7(b) show the average and maximum distance error between the predicted value and the actual value of the next point in an RSM's path. The graphs are based on data gathered from five RSMs moving in different paths and demonstrate an approximately linear relation between the prediction error ε and the laser sampling period p_l. Applying linear regression to the data we get the approximate relation between the average error $\varepsilon_{average}$ and p_l in Equation (25), and between the maximum error ε_{max} and p_l in Equation (26).

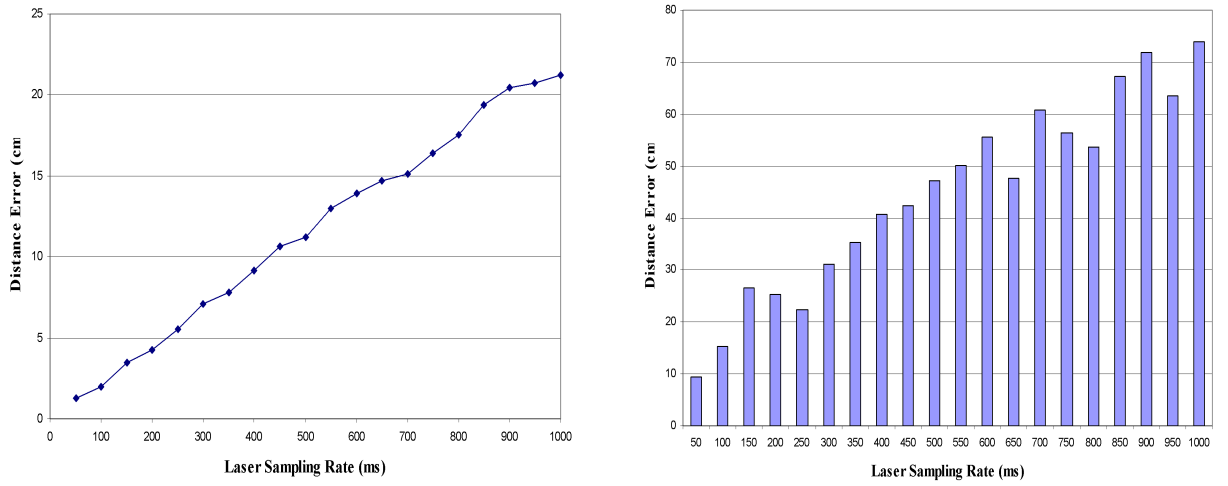
$$\varepsilon_{average} = 0.0219 \cdot p_l + 0.2517 \tag{25}$$

$$\varepsilon_{max} = 0.616 \cdot p_l + 12.467 \tag{26}$$

The relations in Equations (25) and (26) show that we can achieve a desired accuracy of path prediction by adjusting the laser scanner sampling period p_l. Normally, we want minimal error. However, if the system becomes overloaded due to too many obstacles in the environment, it may be desirable to increase the laser sampling period p_l rather than increasing the sonar sampling period p_s or reducing the speed of the foreman.

The laser sampling period p_l can be adjusted in multiples of $50ms$ because of the SICK Laser hardware and the design of the detecting tasks [14]. The minimum value for p_l is $50ms$. From Figure 7(b) we note that a period of $1000ms$ might generate an error of more than $75cm$. An error value higher than this will not be acceptable in most situations. Therefore an upper bound of $1000ms$ on p_l is a reasonable bound.

The execution time for the prediction algorithm is deterministic because a finite number of closed-form calculations are performed. These calculations have to be repeated, however, for each RSM. The worst case execu-



(a) Average error in distance v.s. laser sampling Rate.

(b) Maximum error in distance v.s. laser sampling rate.

Figure 7. Average and maximum distance error plotted for various laser sampling rates.

tion time, $e_{predict}$, for the prediction algorithm task controlling one RSM was determined experimentally to be $e_{predict} = 3.8ms$. We have also verified experimentally that Equation (27) computes $e_{predict}$ for n RSMs.

$$e_{predict} = 3.8 \cdot n \tag{27}$$

From Equations (25), (26) and (27) we conclude that the prediction task utilization, $\frac{e_{predict}}{p_s}$, is related to the number of RSMs and the accuracy of RSM path prediction. Higher accuracy requires a shorter p_l and, therefore, higher processor utilization. Increasing the number of RSMs tracked and controlled by the foreman also increases the utilization because it increases $e_{predict}$ as well as the number of waypoint tasks.

5. Real Time Scheduling

In the previous two sections we showed how the two task sets running on the foreman require period adjustments to attain a certain performance level. In the first case it was the speed at which the foreman could safely move. In the second case it was the accuracy of RSM path prediction. In this section we analyze the schedulability of the system and derive an online schedulability test. An affirmative result from the schedulability test ensures that all relatively deadlines will be met. A negative result indicates a possible overload condition in which performance guarantees cannot be made.

From an application point of view, it is preferable that the motion control task set execute with strictly greater priority than the RSM motion planning and tracking task set. Combining this desire with the optimality of dead-

Task Index	Task	p	Priority
1	Sonar Send$_i$	p_s	1
2	Plan/Speed	p_s	2
3	Sonar Receive$_i$	p_s	3
4	Scanning	p_l	4
5	Detecting	p_l	5
6	Predicting	p_l	6
7	Window Resizing	p_l	6
8	Planning	1500	7
9	Way Point$_i$	1500	8

Table 3. Task priority assignments.

line monotonic scheduling [8] results in the task priority assignment shown in Table 3. The priority assignment is not strictly deadline monotonic since the range for p_s is $55.34ms \leq p_s \leq 3650ms$, while the range for p_l is $50ms \leq p_l \leq 1000ms$. Under most operating conditions, however, the chosen priority assignment is deadline monotonic.

Note that Tasks 1, 3, and 9 in Table 3 are not single tasks but actually groups of tasks with common characteristics. For brevity, we assign them a single task index and priority. This is reasonable as long as priority ties are assumed to be broken in favor of the task with the smaller index i subscript (and hence earlier phase for the send and receive tasks).

5.1 Schedulability Analysis

The task set has predefined static priorities with phases and deadlines less than or equal to periods. The task set also has two dynamic periods. The goal is to find an efficient schedulability test for the task set that can be executed online (because of the dynamic work load). Our approach is based on the principles of time demand analysis presented in [7, 1].

For each task T_i, let R_i be the task response time, d_i be the relative deadline, J_i be the jitter, e_i be the worst case execution time, p_i be the period. According to the time demand analysis method presented in [1], if response time R_i, $1 \leq i \leq n$, computed using Equation (28) also satisfies $J_i + R_i \leq d_i$, then the task set is schedulable. In Equation (28), B_i is the blocking factor for task T_i, $hp(i)$ is the set of tasks with higher priority than task T_i. Equation (28) is solved by iteration because R_i appears on both sides of Equation (28). In general the equation is solved by setting ${R_i}^0 = 0$ for the first iteration, the iteration terminates when ${R_i}^{m+1} = {R_i}^m$.

$$ {R_i}^{m+1} = e_i + B_i + \sum_{\forall j \in hp(i)} \left\lceil \frac{{R_i}^m + J_j}{p_j} \right\rceil \cdot e_j \tag{28} $$

A schedulability test using the time demand analysis requires finding a solution for Equation (28) for every task in the task set. This is inefficient for two reasons. First, it assumes worst-case alignment of periods for all tasks, which over states the response time for most of the tasks in this task set. Second, dynamic periods in the task set require this test to be done online and the computation time to find a solution for Equation (28) is not deterministic. Therefore we present a more efficient schedulability test for this task set based on time demand analysis principles and proprieties of the task set.

Theorem 5.1. *All Sonar Send tasks (Task 1) will always meet their deadlines if* $p_s \geq 23 \cdot \tau + 24 \cdot e_{send} + max\ \Delta t + e_{recv} + e_{send} + e_{plan}$.

Proof: The sonar send tasks have the highest priority in the system (see Table 3) and p_s is defined such that all 24 sonar send tasks are released and must complete execution before the end of the period. Therefore there is no demand (interference) from higher priority tasks. The sonar send tasks have a strict deadline requirement equal to their execution time, but task *Sonar Send*$_{i+1}$ is not released until τ time units after task *Sonar Send*$_i$ has finished execution. Therefore only one sonar send task is ever released and ready to execute at any time, and there can be no interference from any previous sonar send task. Since each sonar task requires at most e_{send} time units to execute (which is equal to its relative deadline), all sonar send tasks will always meet their deadline. □

Theorem 5.2. *The Path-Plan/Speed-Control task (Task 2) will always meet its deadline if* $p_s \geq 23 \cdot \tau + 24 \cdot e_{send} + max\ \Delta t + e_{recv} + e_{send} + e_{plan}$.

Proof: The sonar send tasks and path-plan/speed-control task have the same period, fixed phase(s), and zero jitter. Thus, it is sufficient to prove that if the path-plan/speed-control task meets its deadline in the first p_s time interval it will always meet its deadline.

The latest absolute deadline for a sonar send task is $\phi_{send_{24}} + d_{send_{24}} = 23 \cdot (\tau + e_{send}) + e_{send} = 23 \cdot \tau + 24 \cdot e_{send}$. However, this is less than the phase of the path-plan/speed-control task:

$$\begin{aligned} \phi_{plan} = p_s - e_{plan} &\geq 23 \cdot \tau + 25 \cdot e_{send} + \frac{2 \cdot D}{340 m/s} + e_{recv} + e_{plan} - e_{plan} \\ &> 23(\tau + e_{send}) = \phi_{send_{24}} + d_{send_{24}} \end{aligned}$$

By Theorem 5.1, all send sonar tasks meet their deadlines. Therefore there can be no processor demand (interference) created by higher priority tasks in the interval between when the path-plan/speed-control task is released and the end of the period. Since the path-plan/speed-control task requires at most e_{plan} time units to execute

(which is equal to its relative deadline), it will always meet its deadline and complete at or before the end of the period. □

Theorem 5.3. *All Sonar Receive tasks (Task 3) will always meet their deadlines if* $p_s \geq 23 \cdot \tau + 24 \cdot e_{send} +$ *max* $\Delta t + e_{recv} + e_{send} + e_{plan}$.

Proof: Let T_{recv_i} represent sonar receive task i. We prove by contradiction that task T_{recv_i} will never miss a deadline. Assume that task T_{recv_i} misses its deadline at time t_d. It must be the case that higher priority tasks executed and prevented task T_{recv_i} from executing for e_{recv} time units before its deadline. Since all higher priority receive tasks have a deadline at least $\tau + e_{send}$ time units earlier and $\tau + e_{send} > e_{recv}$, none of those tasks could have kept the processor busy enough to make task T_{recv_i} miss its deadline.

Thus, it must be the case that sonar send tasks and/or the path-plan/speed-control task created the interference. However, the phase of the path-plan/speed-control task is defined such that it cannot execute until after all of the sonar send and receive tasks have completed since

$$\begin{aligned}\phi_{plan} = p_s - e_{plan} &\geq 23 \cdot \tau + 25 \cdot e_{send} + \frac{2 \cdot D}{340m/s} + e_{recv} + e_{plan} - e_{plan} \\ &> 23 \cdot \tau + 24 \cdot e_{recv} = \phi_{recv_{24}} + d_{recv_{24}} > \phi_{send_{24}} + d_{send_{24}}\end{aligned}$$

Therefore only send tasks can create interference. However, each deadline of a sonar send task is separated by at least $\tau + e_{send}$ time units, and by Theorem 5.1 no sonar send task misses its deadline. This means that in the interval $(t_d - (\tau + e_{send}), t_d]$ at most one sonar send task will be released and executed. Since $e_{recv} < \tau$ and the maximum processor demand in the interval is $e_{send} + e_{recv}$, $t_d - (\tau + e_{send}) + e_{send} + e_{recv} < t_d$and task T_{recv_i} will not miss its deadline at time t_d. Thus, all sonar receive tasks will always meet their deadlines. □

For the rest of the tasks, offline analysis is not enough to determine the schedulability of the tasks because some of the tasks have dynamic periods that are independent of p_s. Dynamic periods introduce complexity in determining the scheduling condition because of the need to do the time demand analysis online. Applying the time demand analysis method presented in [1] to Tasks 4 to 9 requires finding a solution to Equation (28) for each of the task iteratively. A careful analysis of the task set, however, reveals that even though several tasks have dynamic periods there are only three distinct periods in the task set at any one time. Theorem 5.4 states that schedulability can be established online with an affirmative result from two conditions.

Before we present the schedulability condition, we define the following notation:

- $DEM_{T_i,\ i=j,\ldots,k}(L)$: Demand (interference) by tasks j through k in any interval of length L.

- $\sum_{p_i=L} e_i$: Total execution time of tasks with period L.

Theorem 5.4. *All tasks will meet their deadlines if Equations* (29) *and* (30) *hold.*

$$\sum_{p_i=p_l} e_i + DEM_{T_i,\ i=1,\ldots,3}(p_l) \leq p_l \tag{29}$$

$$\sum_{p_i=1500} e_i + DEM_{T_i,\ i=1,\ldots,7}(1500) \leq 1500 \tag{30}$$

Proof Sketch: Tasks 1-3 will meet their deadlines by Theorems 5.1, 5.2, and 5.3. Equations (29) and (30) represent an instance of time demand analysis applied to the remainder of this task set.

Tasks 4-7 all have a phase of zero, a period of p_l, and deadline equal to p_l. Therefore each of these tasks will meet is deadline if the sum of its execution time plus all demand (interference) created by higher priority tasks is less than or equal to its deadline. Moreover, since all of the tasks execute on the same processor and each task is released as soon as its predecessor completes, the task can be modeled as though they were all released at the same time with priority given to predecessor tasks in the task graph [6]. Therefore, if Equation (31) holds, tasks 4-7 are schedulable.

$$\sum_{i=4}^{7} e_i + DEM_{T_i,\ i=1,\ldots,3}(p_l) \leq p_l \tag{31}$$

Tasks 8 and 9 both have a period of $1500ms$ with zero phase and deadlines equal to their period. Once again, each of these tasks will meet is deadline if the sum of its execution time plus all demand (interference) created by higher priority tasks is less than or equal to its deadline. Therefore, if Equation (32) holds, tasks 8 and 9 are schedulable.

$$\sum_{i=8}^{9} e_i + DEM_{T_i,\ i=1,\ldots,7}(1500) \leq 1500 \tag{32}$$

In conclusion, tasks 1-3 always meet their deadlines by Theorems 5.1, 5.2 and 5.3. Tasks 4-7 are schedulable if Equation (31) holds. Tasks 8 and 9 are schedulable if Equation (32) holds. Therefore the task set is schedulable if both Equations (31) and (32) hold. □

The proof sketch of Theorem 5.4 does not show how to compute the demand created by higher priority tasks. However, doing so is a straightforward application of Equations (2) and (11) in [1] by Audsley et al. Equation (11) is a generalization of Equation (2) and computes the demand of "sporadically repeating tasks" with inner and outer periods [1]. The sonar send and receive tasks fit this definition. Thus, $DEM_{T_i,\ i=1,\ldots,3}(p_l)$ can be calculated as follows, where L is the demand (interference) interval:

$$DEM_{T_i,\ i=1,\ldots,3}(L) = \begin{cases} \min\left(24, \left\lceil \frac{L}{\tau + e_{send}} \right\rceil\right) \cdot e_{send} + \min\left(24, \left\lceil \frac{e_{send}+e_{recv}+max\ \Delta t+L}{\tau+e_{send}} \right\rceil\right) \cdot e_{recv} + e_{plan} & \text{if } L < p_s \\ \left\lfloor \frac{L}{p_s} \right\rfloor \cdot (24 \cdot (e_{send} + e_{recv}) + e_{plan}) + \min\left(24, \left\lceil \frac{L - p_s \cdot \left\lfloor \frac{L}{p_s} \right\rfloor}{\tau + e_{send}} \right\rceil\right) \cdot e_{send} & \\ \qquad + \min\left(24, \left\lceil \frac{e_{send}+e_{recv}+max\ \Delta t+L-p_s \cdot \left\lfloor \frac{L}{p_s} \right\rfloor}{\tau+e_{send}} \right\rceil\right) \cdot e_{recv} + e_{plan} & \text{if } L \geq p_s \end{cases}$$

Similarly, $DEM_{T_i,\ i=1,\ldots,7}(1500)$ can be calculated as follows:

$$DEM_{T_i,\ i=1,\ldots,7}(1500) = DEM_{T_i,\ i=1,\ldots,3}(1500) + \left\lceil \frac{1500}{p_l} \right\rceil \cdot \sum_{p_i = p_l} e_i$$

5.2 Period Adjustments

From the analysis in the previous section, it should be clear that system schedulability is directly dependent on p_s, p_l, and the execution time of the tasks. However, these parameters are dependent on the selected level of system performance. The value of p_s is dependent on the speed of the foreman and the distance to obstacles (if any). The value of p_l is dependent on the accuracy of RSM placement and the number of RSMs being controlled. Finally the execution times of the detecting, predicting, and way point tasks are directly dependent on the number of RSMs being controlled by the foreman.

Thus, task periods p_l and/or p_s may need to be adjusted to achieve desired performance metrics in the following cases:

- Adjusting the speed of the foreman—either because we want to move faster from one position to the other or because there is an obstacle in the path.
- Increasing the accuracy of RSM path predictions.
- Increasing the number of RSMs being controlled.

Although period adjustments are only allowed at the end of the period being changed, the adjustment might lead to an unsafe state. If any of the system parameters change then we need check if the system is schedulable using Equations (29) and (30). If the system is schedulable then we proceed to execute the tasks at the new rates. If not, the decision to increase either p_l or p_s depends on what system goal is considered more important at the moment. For example, if moving the foreman to its target at a faster speed is the most important, p_l would be increased. This priority decision is dynamic, changing with the environment and tasks required by the foreman at any given moment. Determining such priorities is a research problem of its own and beyond the scope of this paper. However, the performance and schedulability analysis presented here provides a framework for quantifying the tradeoffs to be made.

6. Conclusion

We presented a mobile robotic application that requires adjusting sensor sampling rates to produce desired performance levels. A static priority scheduling solution is proposed that takes into consideration the strict deadline requirements of some of the tasks and their dynamic periods. We have shown how system parameters and environment changes can create overload conditions on the system processor and how system schedulability can be evaluated online.

The online schedulability test can be used to distinguish between safe operating points and potentially unsafe operating points. Moreover, the analysis and online schedulability test provides a framework for a future application-level control algorithm that can make dynamic performance/schedulability tradeoffs. Future work will also include generalizing the modeling and schedulability analysis presented here so that it can be applied more easily to tasks of other real time mobile autonomous systems.

References

[1] N. Audsley, A. Burns, M. Richardson, and K. T. A. Wellings. Applying new scheduling theory to static priority pre-emptive scheduling. *Software Engineering Journal*, 8(5):284–292, September 1993.

[2] F. Baccelli, B. Gaujal, and D. Simon. Analysis of preeptive periodic real-time systems using the (max,plus) algebra with applications in robotics. *IEEE Transactions On Control Systems Technology*, 10(3):368–380, May 2002.

[3] A. Burns, D. Prasad, A. Bondavalli, F. D. Giandomenico, K. Ramamritham, J. Stankovic, and L. Stringini. The meaning and role of value in scheduling flexible real-time systems. *Journal of Systems Architecture*, 46:305–325, 2000.

[4] S. Farritor and M. Rentschier. Robotic highaway saftey marker. In C. Mellish, editor, *ASME International Mechanical Engineering Congress and Exposition*, Montreal, May 2002.

[5] R. George and Y. Kanayama. A rate monotonic schedular for the real-time control of autonomous robots. In *Proceedings of the 1996 IEEE International Confernce on Robotics and Automation*, Minneapolis, Minnesota, April 1996.

[6] S. Goddard and K. Jeffay. Analyzing the real-time properties of a dataflow execution paradigm using a synthetic aperture radar application. In *Proceedings of the 3rd IEEE Real-Time Technology and Application Symposium*, pages 60–71, Montreal,CA, June 1997.

[7] J. Lehoczky, L. Sha, and Y. Ding. The rate monotonic scheduling algorithm - exact characterization and average case behaviour. In *Proceedings of 10th IEEE Real-Time Systems Symposium*, pages 166–171, December 1989.

[8] J.-T. Leung and J. Whitehead. On the complexity of fixedpriority scheduling of periodic real-time tasks. *Performance Evaluation*, 2(4):237250, December 1982.

[9] H. Li, J. Sweeney, K. Ramamritham, R. Grupen, and P. Shenoy. Real time support for mobile robotics. In *Proceedings of the 9th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, pages 10–18, May 2003.

[10] C. McElhone and A. Burns. Scheduling optional computations for adaptive real-time systems. *Journal of Systems Architectures*, 46:46–77, 2000.

[11] T. A. N. Miyata, J. Ota and H. Asama. Cooperative transport by multiple mobile robots in unknown static environments associated with real-time task assignment. *IEEE Transactions On Robotics and Automation*, 18(5):769–780, October 2002.

[12] D. Prasad and A. Burns. A value-based scheduling approach for real-time autonomous vehicle control. *Robotica*, 18:273–279, 2000.

[13] X. Shen. Control of robotic highway saftey markers. Master's thesis, Mechanical Engineering, University Of Nebraska-Lincoln, 2003.

[14] J. Shi, S. Goddard, A. Lal, and S.Farritor. A real-time model for the robotic highway safety marker system. In *Proceedings of the 10th IEEE Real-Time and Embedded Technology and Application Symposium*, pages 331–440, Toronto, CA, May 2004.