2008 IEEE International Conference on
Robotics and Automation
Pasadena, CA, USA, May 19-23, 2008

Using Dynamic Processing Windows for Robot Group Control

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

Jiangyang Huang
Itron, Inc
West Union, SC 29696
jiangyang.huang@itron.com

Shane Farritor
Mechanical Engineering
University of Nebraska–Lincoln
Lincoln, NE 68588-0656
sfarritor@unl.edu

Abstract

This paper presents a method for determining the feasible processing window for a leader robot that controls a group of follower robots. A processing window is defined as the time interval from the instant the platform starts collecting data to the moment the robot leader finishes planning and communicates with follower robots. The method determines the minimum processing window after considering the different bounds affecting the control of the robot, such as of robot's processing power, robot sensors and the motion of the robots in the group. We present a derivation of the bounds that affect the processing window of the leader robot and an algorithm for determining the feasible window or adjusting the parameters that affect the processing window in case a feasible processing window cannot be achieved because of an overload condition. The application considered in this paper is a group of mobile robots that self deploy, retrieve, and reconfigure barrels to improve the safety of highway construction/maintenance workers.

1 Introduction

The processing window is defined as the time interval from the instant the robot starts collecting data to the moment the robot finishes leader planning and communicates with follower robots. The processing window concept was introduced in our earlier work for a different robotic application [18]. In this paper we apply the concept to a different application with different requirements. We analyze a robotic application of a group of robots moving in a leader-follower combination and derive the bounds affecting the processing window for the leader robot. We also propose an algorithm for resolving the overload condition case and calculating slack processing time that can be used for running any non mission-critical software that might need to be executed on the leader robot.

The application considered in this paper is a group of robots that self deploy, retrieve, and reconfigure barrels to improve the safety of highway construction/maintenance workers. Safety barrels guide traffic and serve as a visible barrier between traffic and work crews. These barrels consist of a brightly colored plastic drum (approximately 130 cm high and 50 cm in diameter) that is attached to a heavy base. The robotic safety barrel replaces the heavy base with a mobile robot that transports the safety barrel. The robots work in teams to provide traffic control.

The system is designed with a distributed planning and control approach where the goal is to reduce the per-robot cost (several robots are needed and they are often struck by traffic). This is accomplished by eliminating expensive sensors and computation on the individual barrel robots (followers) by centralizing the intelligence and sensing (leader).

The leader robot uses a laser scanner to detect the follower robots and localizes the follower robots using image processing. However, local control is distributed to each individual barrel robot to reduce the required communication bandwidth-again reducing cost [7]. Relative position is calculated by Hough transformation using a single laser rangefinder mounted in the lead robot. The relative orientation is accomplished with an Extended Kalman Filter (EKF) [12].

Maintaining the system localization, desired separation between the leader and the followers and reducing error in path diversion imposes upper bounds on the processing window for the system. The real-time scheduling algorithm and robot's processing capacity imposes a lower bound on the processing window. An overload condition occurs at higher velocities and sharper turns in the robots' path. When an overload condition occurs an undesired or predictable behavior of the system might occur. This paper presents a solution to this problem.

2 Background and Related Work

The system that we consider in this case fits the category of real-time systems. A real-time system is a system that is required to complete its work and deliver its services on a timely basis. The main difference between a real time system and a normal system is that a real-time system is not just required to produce the correct output, but to produce the correct output on time [15]. The main essential terms used in in real-time systems are

- Task: A sequential piece of code that is executed repeatedly with some pattern.
- Period of a task: Time interval between the release of two consecutive instances of a task.
- Deadline of a task: The time instant by which the job must complete execution.

Real-time scheduling theory has been extensively applied to robotic systems (e.g. [10, 22, 17, 20, 23, 17, 5, 4, 11]).

Of these papers the most closely related work is by Shi, et al. [20] who proposed a rate monotonic scheduling method to analyze the system performance in deploying a group of barrel robots with smooth trajectories in highway work zones. But they did not consider the motion of the leader robot. Their system also did not consider the orientation estimation task for each barrel robot. None of the previous models exactly fits the nature and requirements of our current application. In [18] we provided a schedulability analysis for a different application were a robot is moving in an unpredictable environment. In this paper we adapt and extend the same concepts in [18] to a leader-follower robotic application.

Other related work in terms of robot group control can be found in [1, 3, 6, 8, 9, 14, 16, 21]. Fredslund et al. [9] proposed another local sensing method for robot formations. Each follower uses a laser rangefinder to determine the distance to its leader and then pans the camera to center the leader in the cameras' field of view. Blach et al. [1] proposed a behavior-based approach for maintaining the formation of a team of military unmanned ground vehicles as a scout unit. Each powerful vehicle is equipped with GPS, vision and hazard sensors. Lewis et al. [14] applied the concept of virtual rigid structures for formation maintenance. Each robot is controlled to maintain a rigid geometric relationship to another robot and to a frame of reference. But their application range is constrained in experimental environments because the localization is computed by a fixed and independent vision-based camera system.

Das et al. [6] presented a paradigm for switching between decentralized controllers that allows for changes in formation. A single omni-directional camera is used in all robots and a host computer is used as a centralized processing unit.

Parker et al. [16] presented a control approach for heterogeneous robots in which a more capable leader assists simpler follower robots to navigate using a chaining formation method. There is other work that focuses on issues such as motion planning such as, motion planning [3], hybrid control, [8], separate visual servoing task for each follower [21]. Due to the limited scope, those results are not discussed here. However what is common in the previously mentioned related work is that the followers are provided with considerable sensing capabilities. In contrast, in our application the followers have very limited sensing capabilities in order to cut down on the cost of the system.

3. Deriving Bounds on Processing Windows

All the tasks that have to be executed on the leader robot are shown in the task processing graph in Figure 3. For space limitation we will describe these tasks very briefly:

- Dead Reckoning Task: uses dead reckoning to calculate local coordinates of the leader robot.
- Scan Task: collects the data from the laser range finder.
- Localization Task: localizes the follower robots using the hough transform method for finding circles in images [2]. We developed a modified version of the hough transform in order to reduce the execution time. Instead of searching the whole image space, each follower has a smaller hough transform window assigned to it. The execution time of the localization task is given by Equation (1)

$$e_{hough} = \pi \cdot r_H^2 \cdot n \cdot e_{cal} + \min(361, k \cdot n) \cdot e_{trans} \quad (1)$$

e_{cal} represents the time to initialize the accumulator matrix for one follower and find the maximum value after the transform is executed for one element of the matrix. e_{trans} is the execution time for the Hough transform for one scan point. n is the number of followers. k is the maximum number of scan points for one follower. Here it is assumed that the distance between the leader and the follower is at least 150 cm, which is quite reasonable considering the mechanical dimensions of the two robots. Thus, in the worst case, the maximum number of scan points for one follower is $k = 2 \cdot \arctan(25/150)/0.5 \approx 34$. 361 is the maximum number of scan points obtained from the laser rangefinder. Thus, for all n followers the maximum number of scan points is 361. r_H is the radius of the Hough Transform window. Substituting $k = 34$ in Equation (1) we get Equation (2).

$$e_{hough} = \pi \cdot r_H^2 \cdot n \cdot e_{cal} + \min(361, 34 \cdot n) \cdot e_{trans} \quad (2)$$

Figure 1 shows an example of how the Hough Transform is used to localize the position of the follower robots.

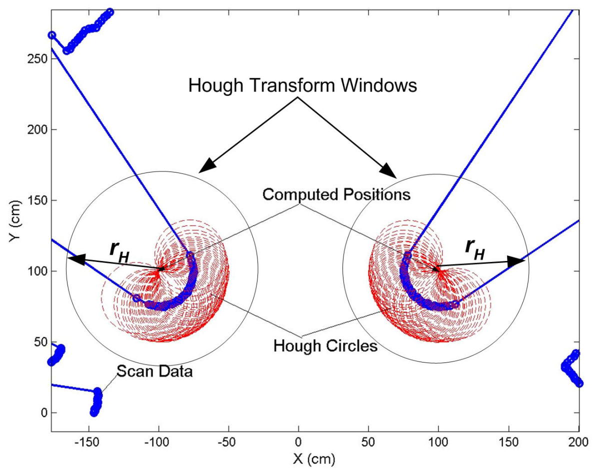

Figure 1. Follower position localization by Hough Transform

- Orientation Estimation Task: uses an extended Kalman filter to estimate the orientation of the follower robots.
- Communication Send Task: sends command and data from the leader to the followers.
- Communication Receive Task: Receives data from the followers to the leader.

- Command Task: receives external commands from a remote control station to the leader.

We will use a modified real time model that is based on processing windows that best fits our application. The model was proposed in our earlier work in [18]. The model divides the tasks into three sets of tasks, uses fixed priority scheduling and calculates lower and upper bounds on the processing window. The three task sets are

- $\mathbf{T_w}$ The task that includes all the tasks associated with the processing window: Scan Task, Localization Task, Orientation Estimation, Communication Send and Communication Receive.
- $\mathbf{T_{hp}}$ The task set that includes all the tasks with higher priority than $\mathbf{T_w}$: Dead Reckoning Task.
- $\mathbf{T_{lp}}$ The task set that includes all the tasks with lower priority than $\mathbf{T_w}$: Command Task.

System schedulibilty imposes a lower bound on the processing window while motion and sensing imposes upper bounds on the processing window. In order for the system (Leader robot) to execute all the tasks within their required deadlines and deliver the required results, the processing window must always satisfy Equation (3),

$$w_{sch} \leq w \leq \min(MSB) \tag{3}$$

where w_{sch} is the scheduilibity bound computed from Equation (11) and MSB is the array of upper bounds imposed on the processing window by sensor and motion requirements. Because we have n followers, there is an MSB array for each follower. For n followers, Equation (3) becomes Equation (4).

$$w_{sch} \leq w \leq \min(MSB_1, ..., MSB_n) \tag{4}$$

Figure 2 shows how w_{sch} and MSB define the operating space for the processing window.

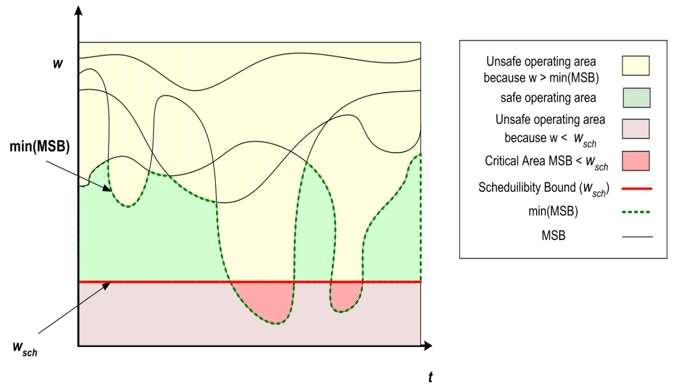
w
min(MSB)
w_sch
t
Unsafe operating area because w > min(MSB)
safe operating area
Unsafe operating area because w < w_sch
Critical Area MSB < w_sch
Scheduilibity Bound (w_sch)
min(MSB)
MSB

Figure 2. Operating Area Illustration

Section 3.1 presents the derivation of the lower bound on the processing window. Section 3.2 presents the derivation of the upper bounds on the processing window and Section 3.3 presents an algorithm for dealing with an overload condition.

3.1 Deriving the Lower Bound on Processing Windows

The function $g(n, E, \Delta I, R)$ [18] calculates the amount of time necessary to finish processing the all the tasks in Figure 3. The function $g(n, E, \Delta I, R)$ can be derived from the task graph and careful examination of the nature of the tasks. The final value of $g(n, E, \Delta I, R)$ is given by Equation (5), due to space limitations we will not discuss the derivation of $g(n, E, \Delta I, R)$ in detail in this paper. The periods for the task set are shown in Table 1.

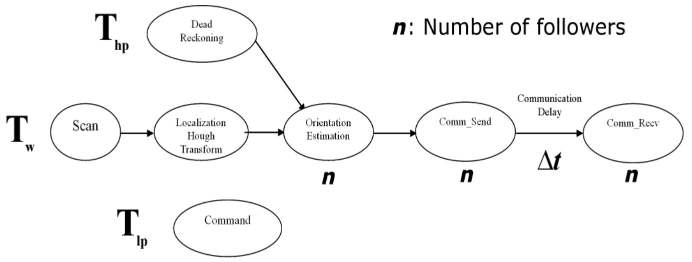

Figure 3. Leader Task Processing Graph

n is the number of robots and Δt is the time interval between sending a message to the followers and receiving the reply. Δt was determined experimentally to be 67ms.

$$\begin{aligned} g(n, E, \Delta I, R) &= e_{scan} + e_{hough} \\ &+ n \cdot (e_{recv} + e_{send} + e_{estimate}) + \Delta t + e_{sched} \end{aligned} \tag{5}$$

Substituting for e_{hough} from Equation (1) we get Equation (6)

$$\begin{aligned} g(n, E, \Delta I, R) &= e_{scan} + \pi \cdot r_H^2 \cdot n \cdot e_{cal} \\ &+ \min(361, k \cdot n) \cdot e_{trans} \\ &+ n \cdot (e_{recv} + e_{send} + e_{estimate}) + \Delta t + e_{sched} \end{aligned} \tag{6}$$

The processing window and the period for the Command task are derived using Equation 14 in [19]

$$w \geq \frac{g(n, E, \Delta I, R) + \sum_{T_j \in \mathbf{T_{hp}}} e_j}{1 - \sum_{T_j \in \mathbf{T_{hp}}} \frac{e_j}{p_j}} \tag{7}$$

Substituting for $g(n, E, \Delta I, R)$ from Equation (6) and for e_j and p_j from Table 1 in Equation (7) we can calculate the lower bound on w. Using the method described in [19] to assign periods for tasks in $\mathbf{T_{lp}}$ we can use $p_{cmd} = 2w$ and solve Equation in [19]. Thus

$$2 \cdot w \geq \sum_{T_j \in \mathbf{T_{lp}}} e_j + \left\lceil \frac{2 \cdot w}{w} \right\rceil \cdot g(n, E, \Delta I, R) + \sum_{T_j \in \mathbf{T_{hp}}} \left\lceil \frac{2 \cdot w}{p_j} \right\rceil \cdot e_j \tag{8}$$

$$w \geq \frac{1}{2} \left(\sum_{T_j \in \mathbf{T_{lp}}} e_j + 2 \cdot g(n, E, \Delta I, R) + \sum_{T_j \in \mathbf{T_{hp}}} \left\lceil \frac{2 \cdot w}{p_j} \right\rceil \cdot e_j \right) \tag{9}$$

Substituting $\frac{2 \cdot w}{p_j} + 1$ for $\left\lceil \frac{2 \cdot w}{p_j} \right\rceil$ we get

$$w \geq \frac{1}{2} \sum_{T_j \in \mathbf{T_{lp}}} e_j + g(n, E, \Delta I, R) + \frac{1}{2} \sum_{T_j \in \mathbf{T_{hp}}} \left(\frac{2 \cdot w}{p_j} + 1 \right) \cdot e_j \tag{10}$$

Task	Task Set	Execution Time e (ms)	Period p (ms)	Deadline d (ms)
Dead Reckoning	$\mathbf{T_{hp}}$	$e_{dr} = 5$	$p_{dr} = 40$	40
Scan	$\mathbf{T_w}$	$e_{scan} = 106$	w	w
Localization	$\mathbf{T_w}$	$e_{hough} = \pi \cdot r_H^2 \cdot n \cdot e_{cal} + \min(361, k \cdot n) \cdot e_{trans}$	w	w
Orientation Estimation	$\mathbf{T_w}$	$e_{estimate} = 1$	w	w
Communication Send	$\mathbf{T_w}$	$e_{send} = 2$	w	w
Communication Receive	$\mathbf{T_w}$	$e_{recv} = 1$	w	w
Command	$\mathbf{T_{lp}}$	$e_{cmd} = 25$	$p_{cmd} = 2w$	$2w$

Table 1. Task parameters

Solving for w we get Equation (11)

$$w \geq \frac{\frac{1}{2}\sum_{T_j \in \mathbf{T_{lp}}, \mathbf{T_{hp}}} e_j + g(n, E, \Delta I, R)}{1 - \sum_{T_j \in \mathbf{T_{hp}}} \frac{e_j}{p_j}} \tag{11}$$

3.2 Deriving Upper Bounds on Processing Windows

Robot group motion, localization and error correction all impose upper bounds on the processing window for the leader. The motion of the robot group imposes two bounds on the processing window, bearing angle bound and separation bound. The localization of the followers by the leader robot imposes the localization bound. And the error correction imposes the path diversion bound on the processing window. Next we drive these bounds.

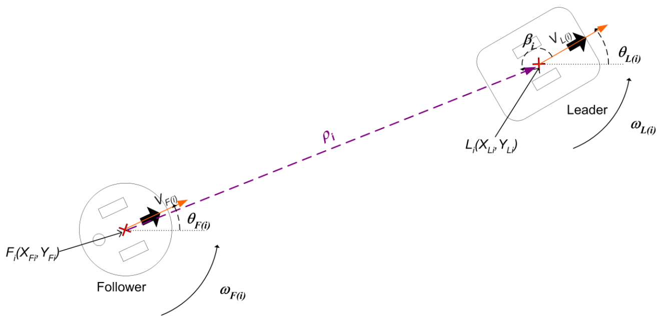

Figure 4. Leader-Follower Kinematics

Localization bound: The localization bound comes from the use of the Hough transform to localize followers as explained in the task descriptions. In order to track the followers and distinguish them from each other the next follower position should be within the Hough Transform window. This is because the previous follower position is used as the center for the Hough Transform window. Therefore to be able to localize the followers, the relative separation $\Delta\rho$ from the leader must be less or equal to Hough Transform window radius r_H. r_H is set to the follower radius r_F. This is because the followers are circular, and therefore it is not possible for another follower to be within a distance less than r_F. Thus,

$$\parallel \Delta\rho \parallel < r_H \tag{12}$$

$$\parallel \vec{\rho_i} - \vec{\rho_{i-1}} \parallel < r_H \tag{13}$$

$\parallel \Delta\rho \parallel$ can be calculated from kinematic analysis [12] according to Equation (14)

$$\parallel \Delta\rho \parallel = \sqrt{\rho_i^2 + \rho_{i-1}^2 - 2\rho_i\rho_{i-1}\cos(\Delta\beta)} \tag{14}$$

To simplify the problem, it is assumed that the separation $\rho \geq 150cm$, as in the worst case computation time of the localization task. Under this assumption, $\Delta\beta$ is small (.25 /150=0.167) and $\cos(\Delta\beta) = 0.9861$, which is approximately equal to one. Thus the dominant factor affecting the distance $\parallel \Delta\rho \parallel$ is the separation ρ . $\parallel \Delta\rho \parallel$ is approximated by Equation (15).

$$\parallel \Delta\rho \parallel \approx \sqrt{\rho_i^2 + \rho_{i-1}^2 - 2\rho_i\rho_{i-1}} = |\vec{\rho_i} - \vec{\rho_{i-1}}| \tag{15}$$

$|\vec{\rho_i} - \vec{\rho_{i-1}}|$ can be calculated from Equation (16) where $\dot{\rho}$ is the separation velocity.

$$|\vec{\rho_i} - \vec{\rho_{i-1}}| = |\dot{\rho}|\Delta t \tag{16}$$

The separation can be calculated from Equation (17) and the change in time Δt is equal to the processing window. Thus Equation (18) is our final desired equation that we can use to substitute in Equation (12), to calculate the bound on w. Substituting for both $\dot{\rho}$ and Δt we get Equation (19)

$$\dot{\rho} = V_F\cos(\theta + \beta) - V_L\cos(\beta) \tag{17}$$

$$|\vec{\rho_i} - \vec{\rho_{i-1}}| = |V_F\cos(\theta + \beta) - V_L\cos(\beta)|w \tag{18}$$

$$w \leq \frac{r_F}{|V_F\cos(\theta + \beta) - V_L\cos(\beta)|} \tag{19}$$

Figure 5 shows and the leader at point X_{Li}, Y_{Li} and the follower at point X_{Fi}, Y_{Fi}, after one processing window has elapsed, the leader is at point X_{Li+1}, Y_{Li+1} and the follower is at point X_{Fi+1}, Y_{Fi+1}. At point X_{Fi+1}, Y_{Fi+1} we can see the relative separation $\Delta\rho$, the position of the follower at point X_{Fi}, Y_{Fi} relative to the follower position at point X_{Fi+1}, Y_{Fi+1} denoted by $X^F{}_{Li}, Y^F{}_{Li}$. Recall from Section 3 that the Hough transform window center is at the relative previous follower position. Therefore the Hough Transform window is centered at $X^F{}_{Li}, Y^F{}_{Li}$ as shown in Figure 5.

Bearing Angle Bound: This bound is related to the robot kinematics as shown in Figure 4. This bound keeps this bearing angle β within a desired threshold. As long as $\Delta\beta \leq T_\beta$. Where $\Delta\beta$ is the change in the bearing angle during one processing window.

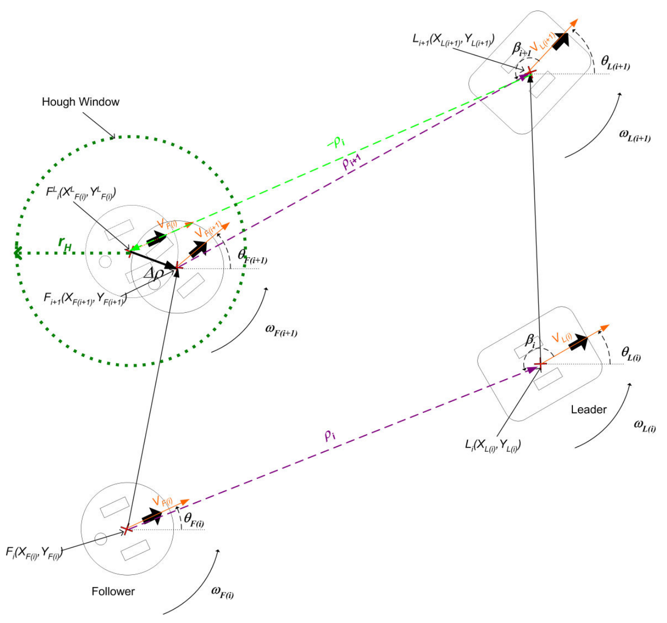

Figure 5. Illustration of Localization bound parameters

We found experimentally that this bound is much higher than other bounds and does not cause the system to enter into a critical region of the operating space.

Separation Bound: This bound keeps the separation between the leader and any follower ρ within a desired threshold T_ρ. The threshold is a percentage of the desired separation ρ_d. For different situations during the robot group motion, the desired separation might be different. Therefore if the change in the actual separation $\Delta\rho$ is less or equal to the percentage of the desired separation. Therefore $\Delta\rho \le T_\rho \cdot \rho_d$. Approximating $\Delta\rho$ using the same method we used for the localization bound we get Equation (20).

$$w \le \frac{T_\rho \cdot \rho_d}{|V_F \cos(\theta+\beta) - V_L \cos(\beta)|} \tag{20}$$

Follower path diversion bound: This bound allows us to control the smoothness follower path is. This bound also helps prevent followers from diverting from the original path due to error and noise. Therefore we always want to keep the follower within a threshold T_ϵ of the desired separation ρ_d on its desired path within one processing window. The follower at most moves a distance of $V_F \cdot w$.

Therefore in case of an error the follower would still be in the circle with radius $V_F \cdot w$ as shown in Figure 6. Figure 6 shows how by using the path diversion bound we can limit the diversion of follower of the original path due to an error between any two consecutive follower path points within a radius of $V_F \cdot w$. Thus Equation (21) gives the path diversion bound.

$$w \le \frac{T_\epsilon \cdot \rho_d}{V_F} \tag{21}$$

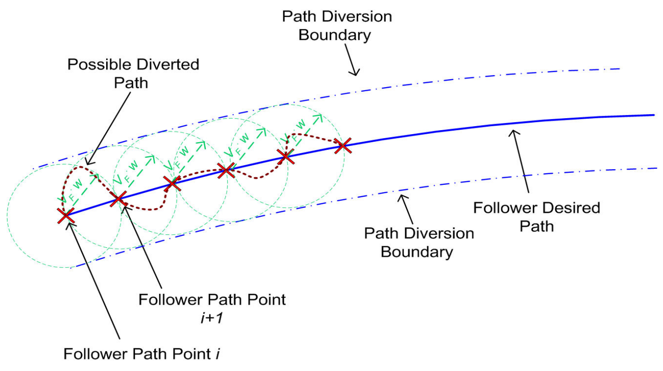

Figure 6. path diversion Illustration

3.3 Adjustment Algorithm

Because the system might likely enter the critical region of the operating space we developed an algorithm for maintaining the system state in the safe operating area. The algorithm chooses $\min(MSB_1, ..., MSB_n)$ as the value for processing window as long as Equation (4) is valid. We define the slack time as the difference between $\min(MSB_1, ..., MSB_n)$ and w_{sch}. Thus

$$t_{slack} = \min(MSB_1, ..., MSB_n) - w_{sch} \tag{22}$$

The slack time can be used for executing any soft deadline or extra tasks running on the system that do not belong to $\mathbf{T_w}$, $\mathbf{T_{hp}}$ or $\mathbf{T_w}$.

The case of $\min(MSB_1, ..., MSB_n) < w_{sch}$ leads to an unpredictable and undesired behavior of the system. We call this condition an overload condition, in an overload condition the system enters the critical region of the operating space as shown in Figure 2. Action must be taken to restore the system to the safe operating space. This action must be in one of two categories:

- Decrease w_{sch} by changing one or more of the parameters of $g(n, E, \Delta I, R)$ such that $w_{sch} \le w \le \min(MSB_1, ..., MSB_n)$.
- Increase $\min(MSB_1, ..., MSB_n)$ by changing one of the parameters of $\min(MSB_1, ..., MSB_n)$ such that $w_{sch} \le w \le \min(MSB_1, ..., MSB_n)$.

In our case $g(n, E, \Delta I, R)$ is dependent on the number of followers n and the radius of the Hough Transform window. Reducing the number of followers is not an option after the system is deployed and reducing the radius of the Hough Transform window beyond r_H will affect the localization bound too. Therefore if the system enters the critical region, the recovery is done by increasing $\min(MSB_1, ..., MSB_n)$. First the type of MSB that caused the system to enter the critical region is identified. Because all of the MSBs are dependent on either V_L or V_F or both and because we have control on these parameters, we can adjust these parameters to move the system out of the critical area and into a safe operating area again.

We always adjust V_L because the EKF algorithm [12] calculates the correct parameters for the follower once the leader velocity is determined to maintain the follower separation behind the leader.

If the system enters the critical area then after determining the type of MSB that caused the condition, V_L is calculated as follows:

- If the MSB that caused the overload condition is the bearing angle bound then V_L is calculated from Equation (23).

$$V_L = \frac{1}{\sin(\beta)}\left(\rho\left(\frac{T_\beta}{w_{sch}} + \omega_L\right) + V_F \sin(\beta+\theta)\right) \tag{23}$$

- If the MSB that caused the overload condition is the separation bound then V_L is calculated from Equation (24).

$$V_L = \frac{1}{\cos(\beta)}\left(V_F \sin(\beta+\theta) - \frac{T_\rho \rho_d}{w_{sch}}\right) \tag{24}$$

- If the MSB that caused the overload condition is the localization bound then V_L is calculated from Equation (25).

$$V_L = \frac{1}{\cos(\beta)}\left(V_F \sin(\beta+\theta) - \frac{r_H}{w_{sch}}\right) \tag{25}$$

- If the MSB that caused the overload condition is the path diversion bound then V_L is calculated from Equation (26).

$$V_L = \frac{T_\epsilon \rho_d}{w_{sch}} \tag{26}$$

  Even though the path diversion bound is calculated using V_F, we calculate V_L and feed it to the EKF algorithm to calculate V_F.

After V_L is calculated the processing window is set to w_{sch} and t_{slack} to zero.

4 Experimental Results

Figure 7 shows the system setup with one leader and two followers. Commands are sent to the leader through a remote control station. The remote control station commands are: move in a straight line, turn, turn direction, turn radius and desired speed.

We demonstrate the need for the adjustment algorithm through two cases that show the occurrence of an overload condition when the system enters a critical area in the operating space. The localization and separation bounds can cause the system to enter a critical area when $|V_F \cos(\theta + \beta) - V_L \cos(\beta)|$ becomes maximum. Because all the follower robots follow behind the leader robot, the range of β is $[90°, 270°]$. The range of θ is $[90°, 270°]$. If V_L and V_F are constants then using two variable calculus [13] we can find that the maximum of $|V_F \cos(\theta + \beta) - V_L \cos(\beta)|$ is $\sqrt{V_F^2 + V_L^2}$ which occurs at $\theta = \pm 90°$ and $\beta = 180° \arctan(V_F/V_L)$. The situation might arise when the separation between the leader and the followers is large and

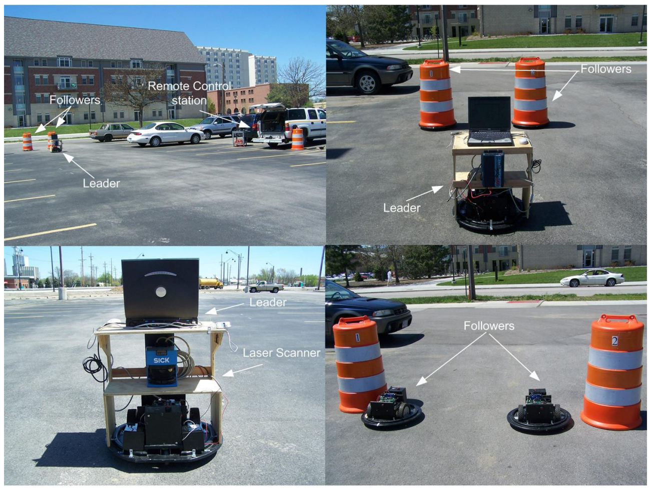

Figure 7. Robots and Setup

the path involves short or sharp turns for the leader. However if V_F and V_L are variable, the overload condition might occur even without sharp turns.

We demonstrate how the overload situation arises and is corrected with two experimental tests with one leader and two followers.

4.1 Test 1: Closed Path, Smooth Turns

In this test the robot group travels in a closed path with turn parameters, desired leader speed $V_{Ld} = 25cm/s$ and turn radius $r_t = 2m$ and desired separation $\rho = 390cm$. Figure 8 shows the leader's and followers' actual path. Figure 9 shows the processing and slack time. Figure 10 shows the effective motion and sensor bounds on the processing window that cause the system to enter critical regions. We can see that processing window overlaps either the $\min(MSB)$ and w_{sch} depending on which region of the operating space the system is in. We can see that w_{sch} is constant because none of the paraments for w_{sch} changes through the experiment. Figure 11 shows the desired and actual leader velocity. We can see that the path diversion bound and separation bound become less than the schedulibilty bound at the sharper parts of the turns.

4.2 Test 2: Closed Path, Sharper Turns

In this test the robot group travels in a closed path with turn parameters, desired leader speed $V_{Ld} = 30cm/s$ and turn radius $r_t = 1.2m$ and desired separation $\rho = 325cm$ [1]. Figure 12 shows the leader and followers actual path. Figure 13 shows the processing and slack time. Figure 14 shows the effective bounds that cause the system to enter critical regions. Figure 15 shows the desired and actual leader velocity. We can see that the critical regions in this test at the turns are larger than Test 1 because the turns are sharper.

[1]This paper is accompanied by a video of part of Test 2. The file name is Qadi_using_processing_windows_for_Robot_Group_Control.mpg

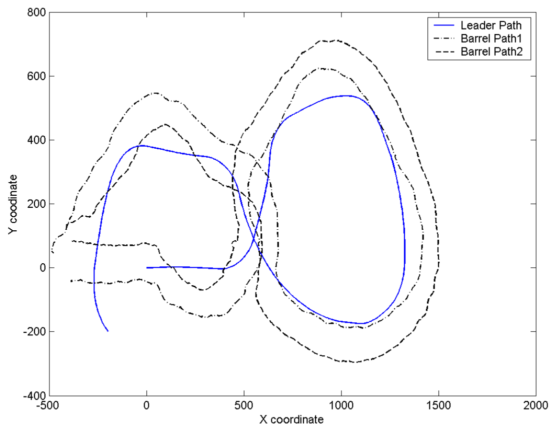

Figure 8. Test 1: Robots path

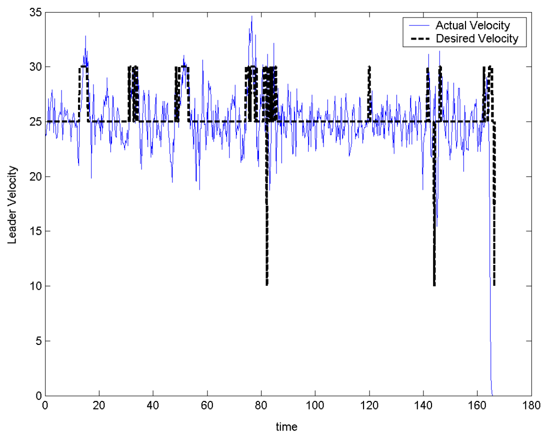

Figure 11. Test 1: Desired and Actual Leader Velocity

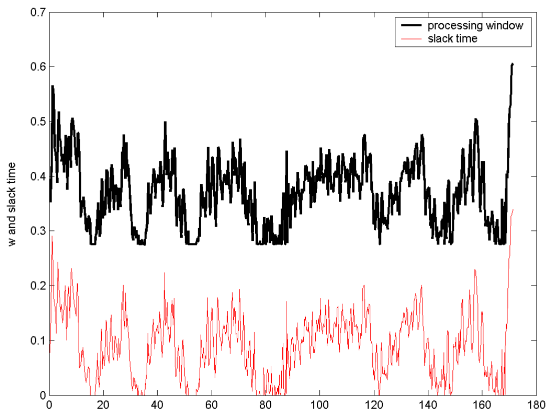

Figure 9. Test 1: Processing Window and Slack Time

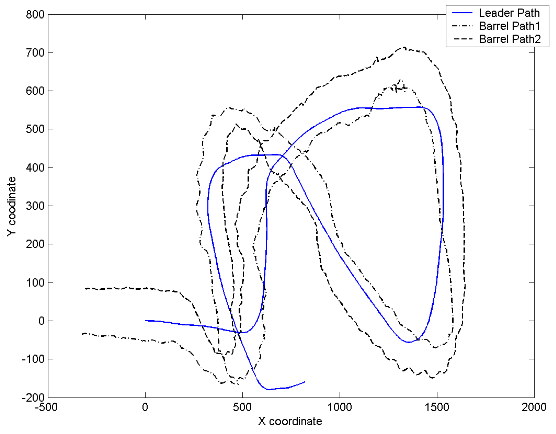

Figure 12. Test 2: Robots path

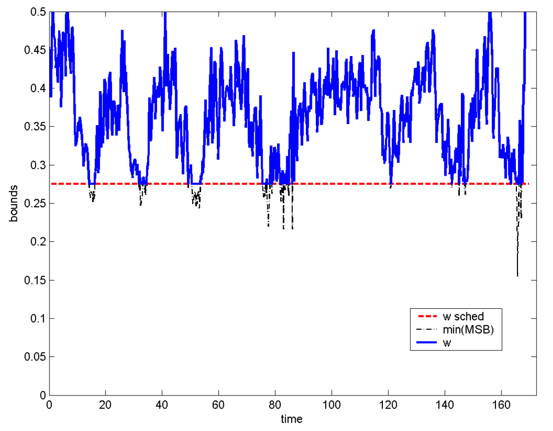

Figure 10. Test 1: Effective Processing Window Bounds

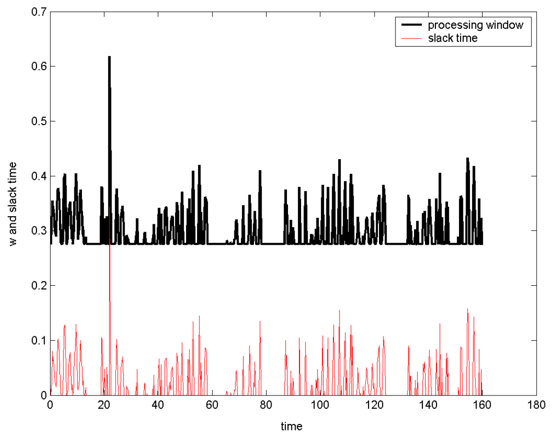

Figure 13. Test 2: Processing Window and Slack Time

5 Conclusion

We have adopted the analysis techniques provided in [18] to a leader-follower robot group combination application to calculate schedulibilty and motion and sensing bounds on the processing window for the leader robot. We have presented an algorithm for dealing with overload conditions that occur when the follower robots make sharp turns behind the leader robot. The algorithm guarantees a feasible processing window such that all tasks meet their deadlines and there is no unpredicted system behavior. The algorithm also calculates slack time that can be available to use for any non mission critical tasks. In case of an overload condition, the algorithm identifies which bound caused the overload

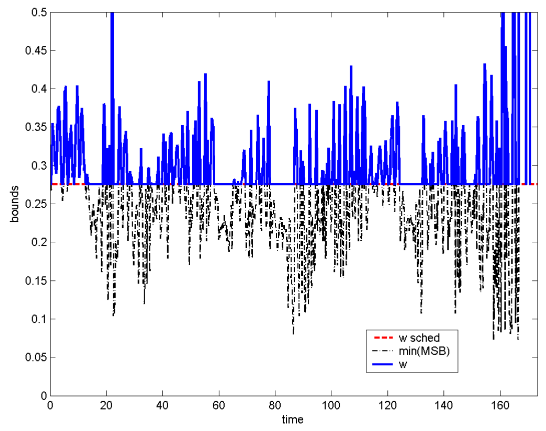

Figure 14. Test 2: Effective Processing Window Bounds

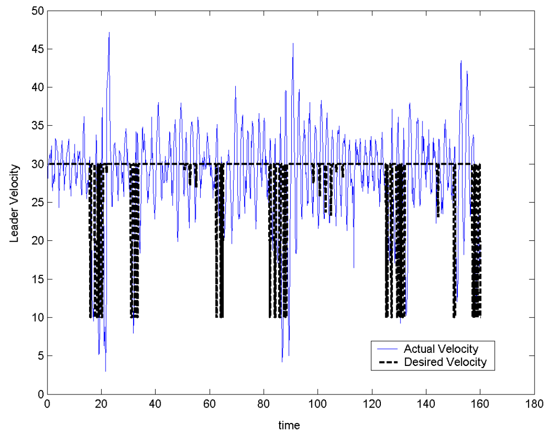

Figure 15. Test 2: Desired and Actual Leader Velocity

condition and adjusts the leader velocity in order to restore the system to a safe state. Our experimental results show that our method eliminated the critical operating regions and kept the system in a safe operating. The same method can be applied to different applications with different processing capability. The derivation of the bounds will follow the same guidelines and will depend on the application kinematics and requirements.

References

[1] T. Balch and R.C.Arkin. Real-time obstacle avoidance for fact mobile robots. *IEEE Transactions on Systems, Man and Cybernetics*, 14(6):926–939, Sept.-Oct 1998.

[2] D. Ballard. Generalizing the hough transform to detect arbitrary shapes.

[3] T. Barfoot and C. Clark. Motion planning for formations of mobile robots. *Robotics and Autonomous Systems*, 46(2):65–78, February 2004.

[4] G. Beccari, S. Caselli, M. Reggiani, and F. Zanichelli. Rate modulation of soft real-time tasks in autonomous robot control systems. In *Proceedings of the 11th Euromicro Conference on Real-Time Systems ECRTS*, pages 153–158, York, U.K., June 1999.

[5] L. Becker, E. Nett, S. Schemmer, and M. Gergeleit. Robust scheduling team-robotics. *Journal of Systems and Software*, 7(1):3–16, 2005.

[6] A. Das, R. Fierro, V. Kumar, J. Ostrowski, J. Spletzer, and C. Taylor. vision-based formation control framework. *IEEE Transactions on Robotics and Automation*, 18(5):813–825, 2002.

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

[8] R. Fierro, A. K. Das, V. Kumar, and J. Ostrowski. Hybrid control of formations of robots. In *Proceedings of IEEE International Conference on Robotics and Automation*, volume 1, 2001.

[9] J. Fredslund and M. Mataric. A general algorithm for robot formations using local sensing and minimal communication. *IEEE Transactions on Robotics and Automation*, 18:837–846, 2002.

[10] R. George and Y. Kanayama. A rate monotonic scheduler for the real-time control of autonomous robots. In *In Proceedings of the 1996 IEEE International Conference on Robotics and Automation*, pages 239– 248, Minneapolis, MN, April 1996.

[11] H. Hassan, J. Simo, and A. Crespo. Enhancing the flexibility and the quality of service of autonomous mobile robotic applications. In *Proceedings of the 14th Euromicro Conference on Real-Time Systems ECRTS*, 2002.

[12] J. Huang. *Localization and Follow-The-Leader Control of A Heterogeneous Group of Mobile Robots*. PhD thesis, University of Nebraska-Lincoln, May 2007.

[13] R. Larson, B. Edwards, and R. Hostetler. *Calculus*. McDougal Littell/Houghton Mifflin, June 2001.

[14] M. Lewis and K.-H. Tan. High precision formation control of mobile robots using virtual structures. *Autonomous Robots*, 4(4):387–403, 1997.

[15] J. Liu. *Real-time Systems*. Prentice-Hall, 2000.

[16] L. E. Parker, B. Kannan, F. Tang, and M. Bailey. Tightly-coupled navigation assistance in heterogeneous multi-robot teams. In *Proceedings of the IEEE International Conference on Intelligent Robots and Systems*, pages 157–162, Sendai, Japan, 2004.

[17] M. Piaggio, A. Sgorbissa, and R. Zaccaria. Preemptive versus non-preemptive real time scheduling in intelligent mobile robotics. *Journal of Experimental and Theoretical Artificial Intelligence*, 12(2):235–245, September-October 2000.

[18] A. Qadi, S. Goddard, J. Huang, and S. Farritor. Dynamic speed and sensor rate adjustement for mobile robotic systems. In *Proceedings of The 19th Euromicro Conference on Real-Time Systems*, pages 239–248, Pisa, Italy, July 2007.

[19] A. Qadi, S. Goddard, J. Huang, and S. Farritor. Modelling computational requirements of mobile robotic systems using zones and processing windows. Technical Report TR-UNL-CSE-2007-0016, Univeristy Of Nebraska-Lincoln, Depratement of Computer Science and Engineering, September 2007.

[20] 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.

[21] R. Vidal, O. Shakernia, and S. Sastry. Formation control of nonholonomic mobile robots with omnidirectional visual servoing and motion segmentation. In *IEEE International Conference on Robotics and Automation*, pages 584–589, Taipei, Taiwan, 2003.

[22] M. Wargui, M. Tadjine, and A. Rachid. A scheduling approach for decentralized mobile robot control. In *Proceedings of the 1997 IEEE/RSJ International Conference on system Intelligent Robots and Systems*, pages 1138–1143, September 1997.

[23] M. Zaera., M. Esteve, C. Palau, J. Guerri, F. Martineza, and P. de Cordoba. Real-time scheduling and guidance of mobile robots on factory floors using monte carlo methods under windows nt. In *Proceedings of 8th IEEE International Conference on Emerging Technologies and Factory Automation*, pages 67–74, 2001.