ASME 2012 5th Annual Dynamic Systems and Control Conference joint with the JSME 2012 11th Motion and Vibration Conference
DSCC2012-MOVIC2012
October 17-19, 2012, Fort Lauderdale, Florida, USA

DSCC2012-MOVIC2012-8544

VELOCITY OCCUPANCY SPACE FOR UNMANNED GROUND VEHICLES WITH ACTUATION ERROR

Rachael A. Bis*
Dept. of Mechanical Engineering
University of Michigan
Ann Arbor, MI 48109-2125
Email: rachbis@umich.edu

Huei Peng
Dept. of Mechanical Engineering
University of Michigan
Ann Arbor, MI 48109-2125

A. Galip Ulsoy
Dept. of Mechanical Engineering
University of Michigan
Ann Arbor, MI 48109-2125

ABSTRACT

The velocity occupancy space (VOS) algorithm was developed to allow an unmanned ground vehicle (UGV) to avoid moving and stationary obstacles and navigate efficiently to a goal using only uncertain sensor data. The original VOS concept was designed for an ideal, holonomic UGV that was capable of perfect, repeatable and instantaneous velocity changes. The method presented here adapts VOS through the use of extended velocity obstacles (EVOs) so that VOS can operate on experimental UGVs that suffer from actuation error. For this research, the EVOs have been designed based on the performance of a SuperDroid ATR, but they can be easily calibrated for other velocity controlled UGVs. The proposed method is validated through numerous simulations.

INTRODUCTION

Summary

The velocity obstacle concept has become a common choice for both autonomous vehicle navigation and as a way to model avoidance behavior for artificial intelligence applications [1]. In previous papers, [2, 3], we introduced Velocity Occupancy Space (VOS)—which is based in part on the velocity obstacle concept—as an algorithm that continuously selects velocities which will allow an unmanned ground vehicle (UGV) to avoid stationary and moving obstacles while navigating to a goal using only uncertain sensor data. VOS selects vehicle velocities under the assumption that the UGV is holonomic and is able to instantaneously accelerate to the desired velocity. For real-world UGVs, there will be acceleration errors, and the commanded velocities cannot be exactly achieved. Thus, the obstacle avoidance and goal finding assumptions made in the formation of the velocity search space are not valid and the vehicle cannot be assured of following a safe and efficient path.

Therefore, this paper will focus on a method by which VOS (and the use of velocity obstacles, in general) can be modified to accommodate velocity controlled UGVs that experience actuation error. Specifically, this paper focuses on how a velocity obstacle (as calculated for VOS) can be extended based on measured vehicle response in order to allow a simulated UGV to operate safely at relatively high speeds using VOS.

Literature Review

Velocity obstacles, as developed by Fiorini and Shiller [4] and Shiller *et al.* [5], are a first-order method of motion planning that use vehicle and obstacle velocities directly to avoid collisions in a time-varying environment. This method computes a velocity obstacle, which consists of all vehicle velocities that will lead to a collision with an obstacle, based on the obstacle's velocity and distance from the vehicle. In later papers, Large *et al.* adapt the velocity obstacle concept in order to deal with non-linear velocity obstacles (obstacles which change direction or speed) [6] and to account for long obstacles (such as hallway walls) [7].

The techniques mentioned above have limited applications because the robot must know, be able to accurately measure, or predict the position and velocity of the obstacles. Therefore this method is not practical in an unknown environment with a significant amount of sensor error. Fulgenzi *et al.* address this problem in [8, 9] by using Bayesian Occupancy Filters (BOF) to create Probabilistic

*Please address all correspondence to this author.

Velocity Obstacles (PVO). We have also developed an approach, VOS, which can be used to avoid moving obstacles using uncertain sensor data [10].

Even though VOS and BOF/PVO compensate for sensor error, they select desired vehicle velocities under the assumption that the vehicle can instantaneously accelerate to the selected velocity without any actuation error. While this assumption is acceptable in simulations, they are not realistic for experimental platforms.

In a previous paper, [11], we expanded the VOS concept to enable it to operate on a differential-drive, acceleration controlled UGV that was capable of instantaneous changes in acceleration but *not* instantaneous changes in velocity. This was done by translating a VOS produced UGV velocity command into a series of piece-wise continuous wheel acceleration commands which would cause the UGV to reach the same velocity and (in some cases, position) by the end of the time step as it would if it had been able to instantaneously change velocity. However, most available UGVs are velocity controlled and suffer from actuation error, therefore the piece-wise constant wheel acceleration model could not be used to safely control these experimental platforms without low-level access and a redesign of their servo control systems. As a result, we developed the velocity command, actuation error based method introduced in this paper to compensate for the inaccurate behavior observed in these UGVs. Using this method, achievable wheel velocity reference signals can be sent to each of the wheel velocity controllers.

Widyotriatmo and Hong integrate sensor and actuation uncertainty into a Partially Observable Markov Decision Process (POMDP) in order to obtain an optimal action policy for a robot at each time step [12]. While using a probabilistic framework to account for actuation error is appropriate for path planning, it is a dangerous choice for performing obstacle avoidance. Even if there is a low probability of a large (and collision causing) actuation error occurring, it is still necessary for the obstacle avoidance system to assume a worst case scenario in order to assure the robot's safety, instead of only compensating for the most probable scenario.

Purpose and Scope

The purpose of this paper is to develop a method by which VOS can be augmented to safely operate on a linear and rotational velocity controlled UGV that suffers from significant actuation error. This error may be caused by a delayed motor response, an ill tuned motor feedback system or uncertain terrain – any error which makes the velocity and position of the vehicle difficult to predict and control but that can still be confidently bounded. Once the error is measured and bounded, EVOs can be used to assure safe vehicle operation while using the VOS method.

In the second section of this paper, background on traditional velocity obstacles and VOS will be reviewed. Then, in the third section, extended velocity obstacles (EVOs) will be derived based on the measured error from a SuperDroid ATR robotic platform. In the fourth section, the results of obstacle avoidance from numerous simulation trials using the EVOs will be presented. Finally, conclusions and plans for future research will be given.

BACKGROUND

Velocity Obstacles

The concept of a velocity obstacle, as detailed by [4, 5, 7, 13], was used in the development of VOS. Both traditional velocity obstacles and VOS require a vehicle to be able to instantaneously change velocities in order for the velocity obstacles to remain valid.

Under the velocity obstacle concept, all robot velocities that will lead to a collision between the robot and an obstacle, *Obstacle A* in Figure 1, are considered to be part of the velocity obstacle, $\overrightarrow{VO_A}$, of that obstacle. As long as the tip of the robot's velocity vector, $\vec{v}_r$, does not fall within $\overrightarrow{VO_A}$ and the obstacle's velocity, $\vec{v}_A$, remains constant, then the robot will avoid a collision with *Obstacle A*. Individual velocity obstacles, such as $\overrightarrow{VO_A}$ and $\overrightarrow{VO_B}$, can be combined along with the set of reachable robot velocities in order to find a safe velocity for the robot. The subset of velocity space that has been determined to be reachable presumably takes the kinematic constraints of the vehicle into account; however it is still assumed that these velocities are instantaneously reachable.

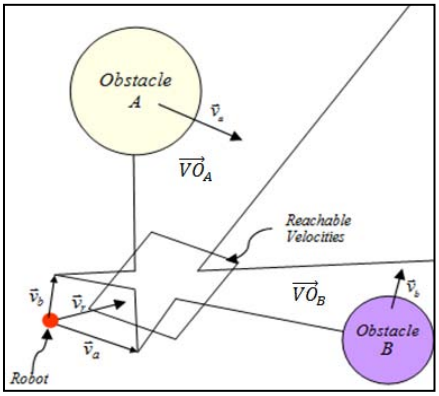

FIGURE 1. ROBOT AND VELOCITY OBSTACLES OF OBSTACLES A AND B IN CONFIGURATION SPACE. ADAPTED FROM [4]

Velocity Occupancy Space

Velocity Occupancy Space (VOS) was developed to allow for safe autonomous robot navigation in an unknown environment in the presence of moving obstacles using only uncertain sensor data [2, 3]. It is based on the paring of an occupancy grid [14], which has been used to avoid stationary obstacles, with the velocity obstacle concept [4, 5], which allows a robot to avoid precisely known moving obstacles. Initially, using this method, the uncertain sensor data is

represented by decomposing the robot's environment into an occupancy grid where each element of the grid is given a value that is representative of the algorithm's level of certainty that the element is actually occupied with an obstacle. The approximate location of each "obstacle" (aggregates of occupied grid elements) is found and the obstacle's movement is tracked. Basic differencing techniques are used to estimate each obstacle's velocity.

Using the position and velocity from the occupancy grid, the positions of the robot and obstacle(s) can be converted from configuration space, Figure 2, into velocity space, Figure 3. In velocity space, each element represents a robot velocity. The robot is located at its current velocity and each obstacle is located at the velocity that the vehicle would have to assume in order to collide with that obstacle in one time step; in this way, the obstacle's position in velocity space takes into account the obstacle's velocity as well as the obstacle's distance from the robot.

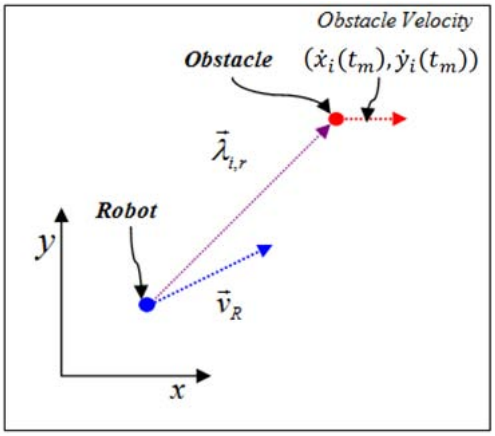

FIGURE 2. CONFIGURATION SPACE REPRESENTATION OF THE ROBOT, WITH VELOCITY $\vec{v}_R$, AND AN OBSTACLE, WHERE $\vec{\lambda}_{i,r}$ IS THE VECTOR BETWEEN THE ROBOT AND OBSTACLE

After the velocity obstacles and goal have been found in velocity space, it is necessary to populate VOS with values in order to select the best robot velocity. Velocity occupancy space consists of weighted elements that correspond to possible robot velocities. The weight of each element is based on two sets of factors. The first set forms a repulsive weight, *R*, based on the possibility that this velocity might lead the robot to a collision with an obstacle. The repulsive values are found according to the equation

$$R = W_R \left[D_R \cdot A_R(W_{AR}) \cdot \left(\frac{W_{TTC}}{TTC} + \frac{1}{CD} \right) \cdot E_{Oc} \right] \quad (1)$$

where the weights (W_R, W_{TTC}, and W_{AR}) are defined and optimized based on the robot's environment, and W_R is the overall repulsive weight; a measure of how important it is to avoid obstacles in comparison to reaching the goal. The term E_{Oc} is the occupancy value for the obstacle element with which each robot velocity will lead to a collision. The other terms are variables related to the robot's state and environment and include D_R, which is the repulsive direction term, *TTC*, which is a measure of the time to collision and *CD* which is the Cartesian distance between the robot and the obstacle. Please see [2] for more details on the weightings and terms that make up Eq. 1.

The repulsive angular term, A_R, is modified in this paper to account for actuation error. This term will be examined in more detail in the next section of this paper.

The second set of factors is based on how quickly and directly a velocity will lead the robot to its goal. All elements in velocity occupancy space are given distinct attractive weightings in order to prevent large portions of VOS from being equally weighted when the elements do not represent equally advantageous velocities. The attractive values for all of the elements in VOS are found from the equation

$$A = [W_{VD} \cdot VD + VC + W_A \cdot A_A] \quad (2)$$

where *VD* is a measure of the difference in velocity between the robot velocity and goal's velocity and *VC* is the magnitude of the change in velocity that would be required if the algorithm was to select the robot velocity in question. The attractive angle term, A_A, causes the algorithm to favor velocities that minimize the angle between the robot's velocity and the goal and the weights, W_{VD} and W_A, are optimized based on the robot's objectives. Again, further details on the weightings and terms that make up Eq. 2 can be found in [2].

After both the repulsive and attractive weights have been calculated, they are given opposite signs and summed to form the velocity occupancy space search matrix according to

$$VOS(\dot{x}_r(t_m), \dot{y}_r(t_m)) = R + A \quad (3)$$

where each element of $VOS(\dot{x}_r(t_m), \dot{y}_r(t_m))$ is the attractive /repulsive weight of the robot velocity $(\dot{x}_r(t_m), \dot{y}_r(t_m))$. From this matrix, the algorithm will select the most attractively (and least repulsively) weighted robot velocity for each time step. This velocity will allow the robot to avoid obstacles (due to the repulsive, *R*, term) and navigate to the goal (due to the attractive, *A*, term). See [2, 10] for a detailed description.

Angle Equation

A VOS velocity obstacle is formed for each filled element from Cartesian occupancy space using the observed obstacle location and estimated velocity, which are measured as previously described. By using the information from the

individual elements, the certainty that each element is occupied can be preserved and used to find the likelihood that a specific robot velocity will lead to a collision with the occupant of that element. Using the VOS method, an element (robot velocity) in velocity space is assumed to be part of the velocity obstacle if it fulfills the following criteria.

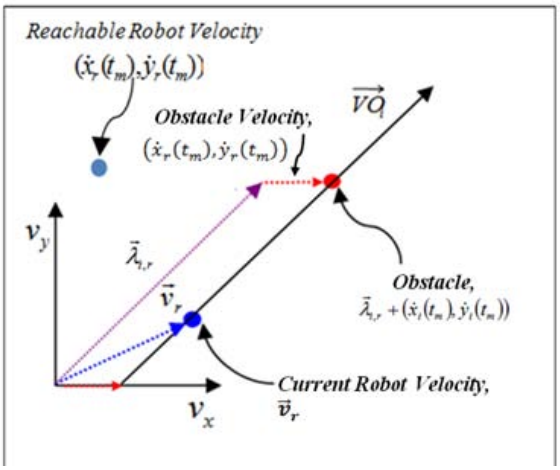

FIGURE 3. VELOCITY SPACE REPRESENTATION OF THE ROBOT VELOCITY, $\vec{v}_r$, THE OBSTACLE IN VELOCITY SPACE AND THE VELOCITY OBSTACLE, $\overrightarrow{VO_i}$

First, the velocity represented by the element must move the robot onto a collision course with the obstacle as defined by the relative angles between the robot's and obstacle's positions and velocities. The term, A_R, is a Boolean term used to denote if this circumstance has been achieved (i.e. if the relative angle between a prospective robot velocity and the velocity of an obstacle will lead to a collision based on their current relative positions). If the velocity will lead to a collision then A_R is defined as one, and the robot will be less likely to choose this velocity, but if the velocity is safe then A_R is zero and the robot can safely select this velocity. More information on this term and its influence on the formation of velocity occupancy space can be found in [2]. A_R is found from the equation

$$A_R = \begin{cases} 1 & if\ tan^{-1}\left(\frac{\vec{\lambda}_{i,r}(\dot{y}_i)}{\vec{\lambda}_{i,r}(\dot{x}_i)}\right) \ldots \\ \ldots = tan^{-1}\left(\frac{|\dot{y}_r - \dot{y}_i \cdot (1 \pm V_U)| \pm P_A}{|\dot{x}_r - \dot{x}_i \cdot (1 \pm V_U)| \pm P_A}\right) \cdot \left(1 \pm (W_{AR} - 1)\right) \\ 0 & otherwise \end{cases} \quad (4)$$

where all time dependent variables are from time t_m (i.e., t_m has been removed for brevity). In this equation $\vec{\lambda}_{i,r}(\dot{y}(t_m))$ and $\vec{\lambda}_{i,r}(\dot{x}(t_m))$ are the relative displacement vectors between the robot and the obstacle element i divide by one time step (see Figures 2 and 3), $\dot{y}_r(t_m)$ and $\dot{x}_r(t_m)$ are the x- and y-velocities that the velocity space element represents, and $\dot{x}_i(t_m)$ and $\dot{y}_i(t_m)$ are the obstacle element velocity. In other words, if the angle of the relative velocity vector between the robot and the obstacle, $tan^{-1}\left(\frac{|\dot{y}_r(t_m) - \dot{y}_i(t_m))|}{|\dot{x}_r(t_m) - \dot{x}_i(t_m))|}\right)$, is equivalent to the angle of the relative position between the robot and the obstacle, $tan^{-1}\left(\frac{\vec{\lambda}_{i,r}(\dot{y}_i(t_m))}{\vec{\lambda}_{i,r}(\dot{x}_i(t_m))}\right)$, then the element's velocity fulfills the angle criteria.

The $\pm$ symbol is used throughout this paper to represent 'within range'. For example, in Eq. (4), if there exists values that fall between $[(1 - V_U), (1 + V_U)]$, $[-P_A, P_A]$, and $\left[\left(1 - (W_{AR} - 1)\right), \left(1 + (W_{AR} - 1)\right)\right]$ which can be appropriately substituted for each the terms on the right side of Eq. (4) which contain a $\pm$ symbol and that will cause the two sides of Eq. (4) to be equal, then A_R will equal one. Please see [2] for more information.

The V_U, P_A, and W_{AR} terms in Eq. (4) are determined based on obstacle and environmental conditions. These terms help to ensure the robot's safety in an uncertain environment. As they are not specifically relevant to the material in this paper, the reader is directed to [2, 15] for a complete description.

DERIVATION OF EXTENDED VELOCITY OBSTACLES (EVOS)

Even though VOS is a velocity based navigation method, it is not possible to control a velocity commanded vehicle with VOS unless the vehicle is capable of instantaneous velocity changes (i.e., has negligible mass). To compensate for this shortcoming, VOS was adapted to apply to velocity-commanded UGVs that suffer from actuation error by extending the bounds of each velocity obstacle to account for actuation error. An EVO is a velocity obstacle that includes not only the desired or commanded UGVs velocity (see Figure 3), but also all other UGV velocities within the pre-computed error bounds that may occur due to actuation error.

While the actuation error that this extension is primarily designed for is delayed velocity change, the method described in this paper is capable of compensating for multiple types of actuation error, including slip (either due to the kinematics of the vehicle or to the terrain), poor motor control or a well-known (e.g. one motor has partially lost power) malfunction of the vehicle. The method presented can be used as long as the effects of the actuation error can be bounded.

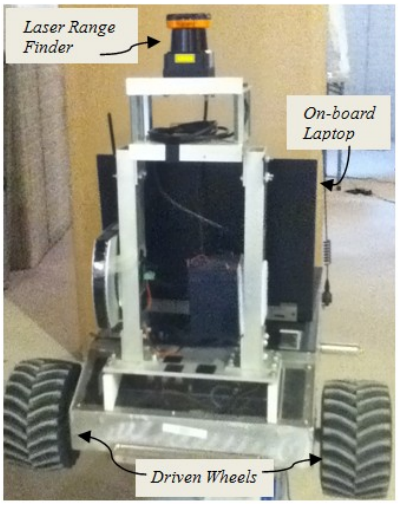

FIGURE 4. SUPERDROID ROBOT

The UGV used to calibrate the EVOs for the initial simulations and which was also used for experimental testing [2] was a SuperDroid ATR. The SuperDroid has two driven, heavily treaded front wheels and one omni-directional back wheel (see Figure 4).

Two specific types of error are accounted for with EVOs, positional and velocity error. Both of these errors must be addressed in order to ensuring a safe and effective robot velocity selection.

Positional Error

The first type of error was in the robot's position. The VOS algorithm operates in two parallel loops. The first loop acquires sensor data, builds configuration space and tracks the obstacles. The second loop uses the obstacle and robot information in order to build VOS. The second loop must project the locations of the obstacles and robot ahead by one motion time step as it is building VOS while the robot is in the process of carrying out the last command. Because the actuation error causes the robot's actual position at the end of that time step to be uncertain, the $\vec{\lambda}_{i,r}(\dot{x}(t_m), \dot{y}(t_m))$ term (see Figure 2) used to compute the velocity obstacles may be incorrect and therefore the velocity obstacle cannot be accurately constructed.

For example, in Figure 5, the algorithm will assume, based on the calculated $\vec{\lambda}_{i,r}$, that $\vec{v}_r$ will lead to a collision (and therefore that velocity will be avoided). However, if $\vec{\lambda}_{i,r}{}^{A}$ is the actual distance then $\vec{v}_r{}^{A}$, and not $\vec{v}_r$, will lead to a collision. In this scenario, the algorithm may mistakenly choose an unsafe robot velocity due to the error in the robot's position.

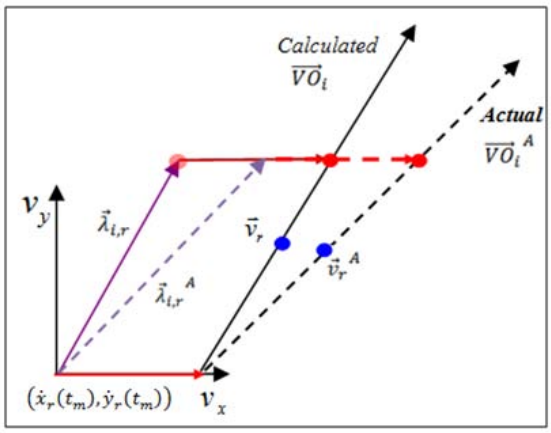

FIGURE 5. VELOCITY OBSTACLES BASED ON THE CALCULATED ROBOT POSITION AND THE ACTUAL (ERROR INFLUENCED) ROBOT POSITION

To compensate for the positional error, the maximum error in the average UGV linear and angular velocities were found over the course of one motion time step. The error for the SuperDroid was due primarily to the non-instantaneous velocity change and therefore the previous and current velocity commands where used as the basis for calculating the positional error. The dependence of the actual, measured velocity on the current and previously commanded velocities was found from the SuperDroid test data using the equation

$$\varepsilon_v(t_{m-1}) = \frac{\bar{v}_{m-1} - v_{m-2}}{v_{m-1} - v_{m-2}} \tag{5}$$

where v_{m-1} is the current linear velocity *command* and $\bar{v}_{m-1}$ is the average of the *measured* velocities over one motion time step. It should be remembered that the VOS algorithm is computing v_m while executing v_{m-1}, so v_{m-2} is the previously commanded linear velocity. The maximum linear velocity error for a specific motion time step, $e_v(t_{m-1})$, can therefore be calculated using the equation

$$e_v(t_{m-1}) = \left(1 - \varepsilon_v(t_{m-1})\right) \cdot v_{m-2} + \varepsilon_v(t_{m-1}) \cdot v_{m-1} \tag{6}$$

The same two equations can also be used to find, ε_ω and e_ω, the maximum error for the angular velocity, by substituting ω for v.

Based on the test data from the SuperDroid it was found that $\varepsilon_v = 0.51$ and $\varepsilon_\omega = 0.44$. Using these values, $e_v(t_{m-1})$ and $e_\omega(t_{m-1})$ will bound the maximum error for 98% of velocity changes (based on the experimental test data). For most of the other 2% of velocity changes, the actual change was typically so slight, that the error was due more to the

steady state variation in the average velocity than to error from acceleration and including these more extreme values would make ε_v and ε_ω unnecessarily conservative.

After the maximum velocity errors have been found they can be used to determine the maximum positional errors in the x- and y-directions ($e_x(t_m)$ and $e_y(t_m)$) using

$$e_x(t_m) = \int_0^{\Delta t_m} e_v(t_{m-1}) \cdot cos(e_\omega(t_{m-1}) \cdot \tau)\, d\tau - \int_0^{\Delta t_m} v(t_{m-1}) \cdot cos(\omega(t_{m-1}) \cdot \tau)\, d\tau \quad (7)$$

and

$$e_y(t_m) = \int_0^{\Delta t_m} e_v(t_{m-1}) \cdot sin(e_\omega(t_{m-1}) \cdot \tau)\ d\tau - \int_0^{\Delta t_m} v(t_{m-1}) \cdot sin(\omega(t_{m-1}) \cdot \tau) d\tau \quad (8)$$

where Δt_m is the length of one motion time step.

The maximum positional errors were then applied to the $\vec{\lambda}_{i,r}$ term in the angle equation (see Eq. (4)) in order to ensure that the actual robot position at the end of the current time step would be considered. The angle equation was augmented using the calculated $e_y(t_m)$ and $e_x(t_m)$ in the following manner

$$A_R = \begin{cases} 1 \ \ if\ tan^{-1}\left(\frac{\vec{\lambda}_{i,r}(\dot{y}_i) \pm \frac{e_y}{\Delta t_m}}{\vec{\lambda}_{i,r}(\dot{x}_i) \pm \frac{e_x}{\Delta t_m}}\right) \ldots \\ \ldots = tan^{-1}\left(\frac{|\dot{y}_r - \dot{y}_i \cdot (1 \pm V_U)| \pm P_A}{|\dot{x}_r - \dot{x}_i \cdot (1 \pm V_U)| \pm P_A}\right) \cdot \left(1 \pm (W_{AR} - 1)\right) \\ 0 \qquad otherwise \end{cases} \quad (9)$$

where all time dependent variables are from time t_m

Using the modified angle equation in Eq. (9), an expanded velocity obstacle was formed that took into account possible error in the robot's position at the end of the time step. As shown in Figure 6, the velocity obstacle has been expanded to include additional velocities that may lead to a collision, based on the uncertainty in the UGV's position.

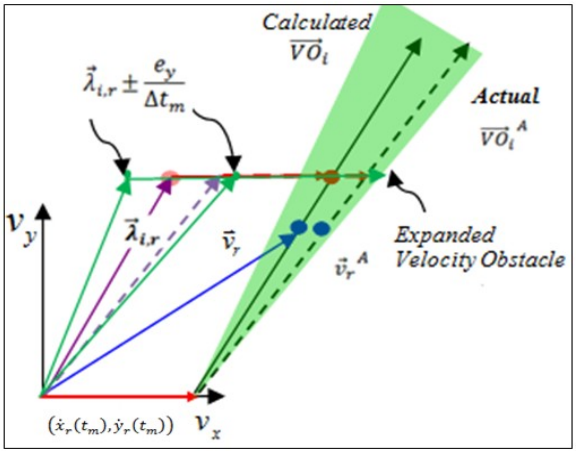

FIGURE 6. EXPANDED VELOCITY OBSTACLE USING ROBOT'S POSITIONAL ERROR BOUNDS

This method may seem overly conservative as it expands the velocity obstacle in both directions – instead of only in the direction corresponding to the change in the robot's velocity. However, it was found that the experimental robot would sporadically over- or undershoot the command velocity by accelerating or decelerating too rapidly. By the end of the time step, the robot's velocity had settled to the command velocity, but the positional error was now the opposite of what was expected (based on Eqs. (5)-(8)). This error occurred frequently enough that it was considered prudent to allow the velocity obstacle to be expanded in both directions as a conservative velocity choice around obstacles was considered preferable to a possible collision.

Velocity Error

The second type of error was in the robot's selected velocity. Each velocity obstacle is composed of all of the robot velocities that will lead to a collision between the robot and a specific obstacle – velocities which must be avoided. However, given the actuation error, the robot's actual velocity frequently differed from the command velocity selected by the algorithm. Therefore the robot might inadvertently move at a dangerous velocity. For example, in Figure 7, the algorithm has selected $\vec{v}_r$ as a safe velocity at which the robot may operate, however, the actuation error causes the robot to actually move at $\vec{v}_r{}^A$, which is along a velocity obstacle and places the robot at risk of a collision. In this scenario, the algorithm correctly selects a safe velocity, but the robot is unable to obey the commanded velocity.

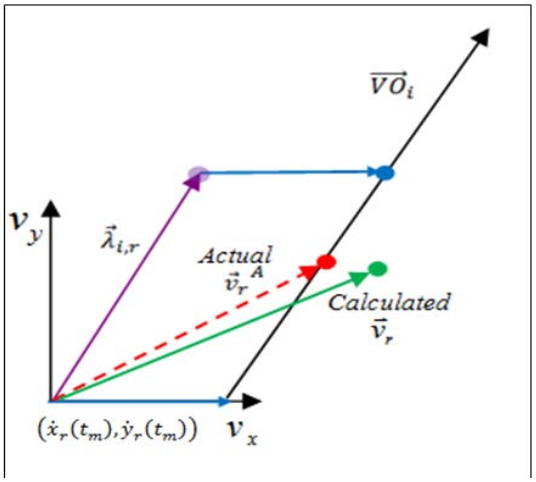

FIGURE 7. SCENARIO WHERE A POSSIBLE COLLISION MAY OCCUR DUE TO THE ROBOT'S CURRENT VELOCITY ERROR. PLEASE NOTE, FOR SIMPLICITY, THIS FIGURE DOES NOT SHOW THE POSITIONAL ERROR.

To compensate for the error in the UGV's current velocity, the maximum variations in the robot's linear and angular velocity, Eqs. (5 and 6), were again used. However, this time, the error terms were found using the currently commanded linear and angular velocities (v_{m-1} and ω_{m-1}) and the linear and angular velocity commands that the algorithm was currently selecting using VOS (v_m and ω_m). Therefore the error dependency equation is slightly altered from Eq. (5) to

$$\varepsilon_v(t_m) = \frac{\bar{v}_m - v_{m-1}}{v_m - v_{m-1}} \tag{10}$$

and the maximum linear velocity error, $e_v(t_m)$, is likewise slightly altered from Eq. (6) to be

$$e_v(t_m) = \left(1 - \varepsilon_v(t_m)\right) \cdot v_{m-1} + \varepsilon_v(t_m) \cdot v_m \tag{11}$$

Again, the angular velocity error terms, ε_ω and e_ω, can also be found by substituting ω for v.

The upper bounds on the directional velocity errors are

$$e_{vx}(t_m) = e_v(t_m) \cdot cos(e_\omega(t_m) \cdot \Delta t_m) \tag{12}$$

and

$$e_{vy}(t_m) = e_v(t_m) \cdot sin(e_\omega(t_m) \cdot \Delta t_m). \tag{13}$$

These terms were also used to augment the angle equation, but this time they increase the range of the robot's selected velocity. The angle term, with both the position error terms from the previous section and the velocity error terms from this section, is

$$A_R = \begin{cases} 1 \quad if\ tan^{-1}\left(\dfrac{\vec{\lambda}_{i,r}(\dot{y}_i) \pm \dfrac{e_y}{\Delta t_m}}{\vec{\lambda}_{i,r}(\dot{x}_i) \pm \dfrac{e_x}{\Delta t_m}}\right) \ldots \\ \ldots = tan^{-1}\left(\dfrac{\left|(\dot{y}_r \pm e_{vy}) - \dot{y}_i \cdot (1 \pm V_U)\right| \pm P_A}{\left|(\dot{x}_r \pm e_{vx}) - \dot{x}_i \cdot (1 \pm V_U)\right| \pm P_A}\right)\left(1 \pm (W_{AR} - 1)\right) \\ 0 \qquad otherwise \end{cases} \tag{14}$$

where all time dependent variables are from time t_m.

Using the expanded angle term, the velocity obstacles will now take into account both the error in the robot's position as well as the potential for error in the velocity that is selected for each motion time step.

An example of an actual EVO can be seen in Figures 8 and 9, where the velocity obstacles for a simple scenario with one moving obstacle are shown. The velocity obstacle is broader with EVOs, especially at velocities that are further away from the robot's current velocity (which may produce more actuation error). It should be noted that the magnitude of the repulsive values assigned to velocities in the obstacles do not increase, just the number of velocities that are considered repulsive.

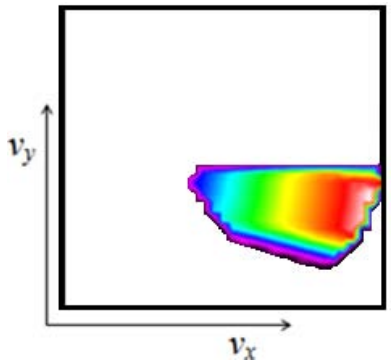

FIGURE 8. VELOCITY OBSTACLE WITHOUT EVOS

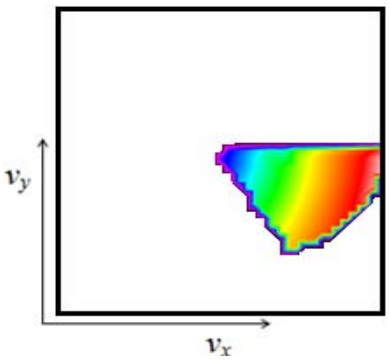

FIGURE 9. VELOCITY OBSTACLE WITH EVOS. THIS FIGURE USES THE SAME SCALE AS FIGURE 8.

RESULTS

An environment of approximately the same size and shape as the experimental testing area was simulated in order to exhaustively test the algorithm and analyze its performance with and without the actuation error extension. In order to cause the simulated robot to perform comparably to the experimental UGV, error was added to the simulated robot's position, following the same trend as was observed with the experimental robot, using the equations

$$x(t_m) = x(t_{m-1}) + \left(v(t_m) + e_v(t_m) \cdot RN(0-1.1)\right) \cdot cos\left(\left(\omega(t_m) + e_\omega(t_m) \cdot RN(0-1.1)\right) \cdot \Delta t_m\right) \cdot \Delta t_m \quad (15)$$

and

$$y(t_m) = y(t_{m-1}) + \left(v(t_m) + e_v(t_m) \cdot RN(0-1.1)\right) \cdot sin\left(\left(\omega(t_m) + e_\omega(t_m) \cdot RN(0-1.1)\right) \cdot \Delta t_m\right) \cdot \Delta t_m. \quad (16)$$

here the maximum linear and angular velocity errors, $e_v(t_m)$ and $e_\omega(t_m)$, were calculated according to Eq.(5) (using the current and previous velocity commands) and $RN(0-1.1)$ is a random number between 0 and 1.1. This random number was obtained using a uniform distribution and was used for two reasons. First, the observed velocity error was relatively evenly distributed between zero and the maximum velocity error bound so the uniform random number reproduced the actual robot behavior fairly accurately. Second, an upper bound of 1.1 was used instead of 1.0 both to represent additional sources of error that were not quantified (such as inexact timing of commands) and error from wheel slip that was not captured by the gyroscope and encoder data used to calculate the experimental UGV's velocity error.

A total of six sets of 100 simulations each were performed, with different velocity limits on the robot and the obstacles. Each simulation consisted of six randomly generated obstacles (in addition to the walls of the testing environment), with between one and four obstacles - moving at constant velocities randomly generated between the bounds shown in Table 1. The robot's maximum linear velocity and accelerations for the simulations are also shown in this table (the maximum angular velocity was always $1\frac{rad}{s}$).

TABLE 1. SIMULATION SPECIFICATIONS

Simulation Set #	EVOs?	Robot Maximums		Obstacles
		Linear Velocity	Linear Acceleration	Velocity Range
1	Yes	$0.5\frac{m}{s}$	$0.5\frac{m}{s^2}$	$-0.3\ to\ 0.3\frac{m}{s}$
2	No	$0.5\frac{m}{s}$	$0.5\frac{m}{s^2}$	$-0.3\ to\ 0.3\frac{m}{s}$
3	Yes	$0.7\frac{m}{s}$	$0.7\frac{m}{s^2}$	$-0.4\ to\ 0.4\frac{m}{s}$
4	No	$0.7\frac{m}{s}$	$0.7\frac{m}{s^2}$	$-0.4\ to\ 0.4\frac{m}{s}$
5	Yes	$1.0\frac{m}{s}$	$1.0\frac{m}{s^2}$	$-0.5\ to\ 0.5\frac{m}{s}$
6	No	$1.0\frac{m}{s}$	$1.0\frac{m}{s^2}$	$-0.5\ to\ 0.5\frac{m}{s}$

Figures 10and 11 show the algorithm's response with and without the actuation error extension to the same simulation.

In Figure 10, the algorithm has not increased the velocity obstacles based on the actuation error, but the robot's velocity response still suffers from the actuation error. The algorithm starts out moving the robot quickly towards the goal. At time step 7, the algorithm unsuccessfully attempts to avoid an obstacle by selecting a velocity that is up and to the left (relative to the image's orientation). Actuation error causes the robot not to respond as predicted and the robot collides with the obstacle at time step 8.

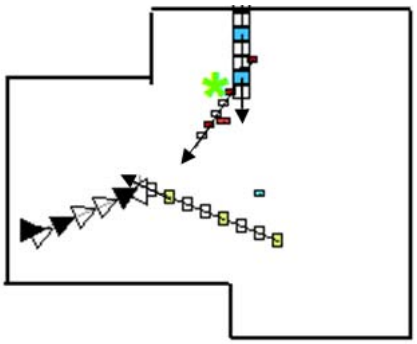

FIGURE 10. ALGORITHM'S RESPONSE TO A BASIC SIMULATION WITHOUT ACTUATION ERROR EXTENSION. THE ROBOT IS THE TRIANGLE, THE OBSTACLES ARE THE COLORED RECTANGLE AND THE GOAL IS THE GREEN ASTERISK.

In Figure 11 the actuation error extension was applied to the velocity obstacles. The robot does not initially move as quickly towards the goal as in the previous simulation, as the extended velocity obstacles react to the presence of approaching obstacles and cause more of the faster robot velocities to have a repulsive value. However, as an obstacle gets closer the algorithm selects more conservative velocities and waits for the obstacle to pass, instead of attempting to pass in front of the obstacle. The robot is then able to safely navigate to the goal.

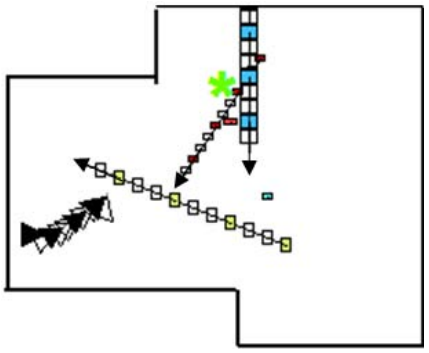

FIGURE 11. ALGORITHM'S RESPONSE WITH ACTUATION ERROR EXTENSION, TIME STEPS 1-11

In Table 2, the values of the evaluation metrics for the simulations are tabulated [2, 15]. However, the obstacle proximity and the acceleration were divided by the number of motion time steps in each simulation so that the results could be more accurately compared between simulation and experimental scenarios with different goal locations.

TABLE 2. SIMULATION EVALUATION METRIC VALUES FOR 100 TRIALS

Simulation Set #	Evaluation Metrics					
	Obstacle Proximity $\left(\frac{1}{m^2}\right)$	Distance Traveled (m)	Velocity Change $\left(\frac{m}{s}\right)$	Time (s)	# of Collisions	# of Timeouts
1	3.15	8.26	0.082	29.52	3 (3%)	3 (3%)
2	2.50	8.28	0.080	28.81	5 (5%)	2 (2%)
3	1.62	8.79	0.285	31.95	4 (4%)	0
4	1.80	8.93	0.345	27.46	7 (7%)	0
5	1.41	8.92	0.173	26.09	3 (3%)	0
6	2.13	9.13	0.292	15.31	11 (11%)	1 (1%)

As summarized in Table 2, the difference between the algorithm's performance with and without the actuation error extension diminished as the robot's and obstacle's maximum velocity decreased. For simulation sets 5 and 6 (where the obstacles and robot had the highest velocities), the actuation error extension made a statistically significant improvement in the number of collisions ($p < 0.011$, on a two-tailed, paired *t*-test) and in the robot's acceleration ($p < 0.005$) between the two sets. However, the actuation error extension also significantly increased the average amount of time that it took the robot to reach the goal ($p < 0.005$), due to more conservative velocity selection.

For simulation sets 3 and 4, there was a smaller difference in the number of collisions ($p < 0.181$) and the statistical difference between the sets for time and acceleration were the same as for sets 5 and 6. Finally, for simulation sets 1 and 2, there was not a statistically significant difference in the number of collisions. However, the simulations with the actuation error extension experienced more timeouts (being unable to reach the goal within 100 motion time steps) than the simulations without the extension. Two of these timeouts occurred in the simulation where the EVO-less simulation experienced a collision. While, from a safety perspective, timing out is preferable to a collision, it still means that the robot failed to reach the goal.

The actuation error extensions significantly improved the robot's ability to avoid collisions at higher speeds, but had less significant effect at lower speeds. At higher speeds the robot accumulated more positional error within one time step and the possible difference in velocities was also higher (leading to additional error). However, while the robot was more careful in its avoidance of obstacles it also selected more conservative velocities which resulted in the robot to taking a longer time to reach the goal.

For lower speeds, there was less possible position and velocity error and while Eq. (6) takes this into account, the normal precautions that the traditional VOS algorithm takes to compensate for sensor error appeared to be sufficient to keep the robot safe even with additional actuation error. The more conservative velocity selection that was produced using the actuation error extension was not necessary and served only to increase the time needed for the robot to reach the goal.

CONCLUSIONS AND FUTURE WORK

Conclusions

The EVO extension for VOS has been shown to significantly improve the algorithm's performance at higher speeds $\left(0.7\frac{m}{s^2} \text{ to } 1\ \frac{m}{s^2}\right)$ over the original VOS algorithm. However, this improvement comes at the cost of increasing the time that it will take the UGV to reach its goal. As has been the theme throughout the development of VOS, there are always tradeoffs between performance and safety for the algorithm and as safety must be the priority if the UGV is to interact with other vehicles, it has been necessary to accept slightly degraded performance in order to improve the algorithm's ability to avoid collisions.

Conversely, at lower speeds there is not a significant advantage (or disadvantage) to using the actuation error extension with VOS. This is due to the smaller amount of accumulated positional error that can occur within one time

step as well as the smaller possible change in the UGV's velocity. The original VOS algorithm is sufficient to compensate for the lesser amount of actuation errors that may occur under these circumstances. However, this change implies that at velocities higher than those tested for this research, the actuation error extension will be even more critical to ensuring the safety of a robot operating with VOS.

Future Work

EVOs have been applied to experimental vehicles in order to allow the UGV to avoid the moving and stationary obstacles and reach a goal location. Video results can be seen at https://sites.google.com/site/rachaelbis/thesis-appendix-c and a formal description of the experimental testing is available in [2].

The EVO extension can compensate for error induced from both the environment as well as from the vehicle – as long as the extent of the error can be bounded. In the future, the error caused by various types of unstable terrain (e.g. sandy or muddy soil) and possible vehicle malfunctions (such as partial loss of power) could be characterized and the compensation for these errors could be integrated into the VOS actuation error framework.

In addition, the work in this paper points clearly to the need to improve the motion control of UGVs for autonomous operation. Whereas current UGVs are adequately designed for their intended use as tele-operated vehicles, for reliable autonomous operation of UGVs, more stringent motion control specifications are necessary.

ACKNOWLEDGEMENTS

This research was supported in part by the Ground Robotics Reliability Center (GRRC) at the University of Michigan, with funding from government contract DoD-DoA W56H2V-04-2-0001 through the United States Army TARDEC. The authors would also like to thank William Westrick for his support with the SuperDroid robot.

REFERENCES

[1] Berg, J. v. d., Patil, S., Sewall, J., Manocha, D., and Lin, M., 2008, "Interactive navigation of multiple agents in crowded environments," Proceedings of the 2008 symposium on Interactive 3D graphics and games, ACM, Redwood City, California, pp. 139-147.

[2] Bis, R., 2012, "Velocity Occupancy Space: Autonomous Navigation and Dynamic Obstacle Avoidance with Sensor and Actuation Error," Ph.D., The University of Michigan, Ann Arbor.

[3] Bis, R., Peng, H., and Ulsoy, G., 2009, "Velocity Occupancy Space: Robot Navigation and Moving Obstacle Avoidance with Sensor Uncertainty," Dynamic Systems and Controls ConferenceHollywood, CA, pp. 363-370.

[4] Fiorini, P., and Shiller, Z., 1998, "Motion planning in dynamic environments using velocity obstacles," International Journal of Robotics Research, 17(7), pp. 760-772.

[5] Shiller, Z., Large, F., and Sekhavat, S., 2001, "Motion planning in dynamic environments: Obstacles moving along arbitrary trajectories," 2001 Ieee International Conference on Robotics and Automation, Vols I-Iv, Proceedings, pp. 3716-3721.

[6] Large, F., Sekhavat, S., Shiller, Z., and Laugier, C., 2002, "Towards real-time global motion planning in a dynamic environment using the NLVO concept," 2002 Ieee/Rsj International Conference on Intelligent Robots and Systems, Vols 1-3, Proceedings, pp. 607-612.

[7] Large, F., Laugier, C., and Shiller, Z., 2005, "Navigation among moving obstacles using the NLVO: Principles and applications to intelligent vehicles," Autonomous Robots, 19(2), pp. 159-171.

[8] Fulgenzi, C., Spalanzani, A., and Laugier, C., "Dynamic Obstacle Avoidance in uncertain environment combining PVOs and Occupancy Grid," Proc. Robotics and Automation, 2007 IEEE International Conference on, pp. 1610-1616.

[9] Fulgenzi, C., 2009, "Autonomous navigation in dynamic uncertain environment using probabilistic models of perception and collision risk prediction," Ph.D., Institut National Polytechnique de Grenoble.

[10] Bis, R., Peng, H., and Ulsoy, G., 2009, "Velocity Occupancy Space: Robot Navigation and Moving Obstacle Avoidance with Sensor Uncertainty," Dynamic Systems and Controls ConferenceHollywood, CA.

[11] Bis, R., Peng, H., and Ulsoy, A. G., 2010, "Velocity Occupancy Space for Differential Drive Vehicles," *Dynamic Systems and Controls Conference*Cambridge, MA, pp. 249-256

[12] Widyotriatmo, A., and Keum-Shik, H., "Decision making framework for autonomous vehicle navigation," Proc. SICE Annual Conference, 2008, pp. 1002-1007.

[13] Large, F., Vasquez, D., Fraichard, T., and Laugier, C., 2004, "Avoiding cars and pedestrians using velocity obstacles and motion prediction," IEEE Intelligent Vehicles Symposium, pp. 375-379.

[14] Moravec, H., and Elfes, A., "High resolution maps from wide angle sonar," Proc. Robotics and Automation. Proceedings. 1985 IEEE International Conference on, pp. 116-121.

[15] Bis, R., Peng, H., and Ulsoy, A. G., 2012, "Velocity Occupancy Space: Autonomous Navigation in an Uncertain, Dynamic Environment," *International Journal of Vehicle Autonomous Systems*.